S'initier à l'analyse des données avec le logiciel R

Vincent Richard

De omnibus dubitandum
(Il faut douter de toute chose)

René Descartes (1596-1650)

• Pourquoi consacrer une documentation à R ?

R est un logiciel libre, il est donc accessible au plus grand nombre. Il répond à un besoin des équipes des pays en développement qui ne peuvent pas consacrer chaque année des sommes importantes au renouvellement des licences des logiciels de statistiques proposés dans le commerce.

Mais R a son propre langage et il vous oblige donc à une période d'apprentissage pour vous familiariser avec sa programmation. Il utilise un langage orienté objet et ce sont ces objets que je vous propose de découvrir tout au long de ce livre. Vous allez avoir à dominer le langage de ce logiciel puis ensuite pour certains, moins aguerri, à vous frotter au domaine de la statistique.

Une fois à l'aise dans vos échanges avec R, vous allez pouvoir profiter de la puissance de cet outil dans le champ de la statistique où de nombreuses extensions vous attendent sur des domaines que vous n'avez pas encore osés explorer. Les capacités graphiques de R vont vous surprendre une fois passée l'étape de la découverte.

Il est donc maintenant temps de vous jeter à l'eau et de démarrer sans panique, comme pour le vélo il y aura probablement quelques chutes mais je vous conseille de ne pas perdre espoir et de vous remettre en selle très rapidement.

Table des matières

Chapitre 1

Premiers pas avec le logiciel R

Objectifs à atteindre à cette étape

Savoir installer la version du logiciel R correspondant au système d'exploitation de son ordinateur

Comprendre le mode de fonctionnement entre l'utilisateur et le logiciel

Savoir consulter l'aide proposée par le logiciel R

Créer des objets sous R

Utiliser un éditeur de commande

Sauvegarder les commandes

Définir un répertoire de travail

Quitter le logiciel R

1.1 - Installer R

R est disponible pour la plupart des environnements de travail de vos ordinateurs (Windows, MacOs ou les systèmes de type UNIX).
Le logiciel est distribué gratuitement et est disponible sur le net à l'adresse suivante: http://www.r-project.org/
Pour répondre à la grande demande de téléchargement et éviter les blocages, il existe des sites appelés miroirs à l'adresse suivante: http://cran.r-project.org/; ainsi tous les fichiers utiles à l'installation ou à la mise à jour sont accessibles à partir de différents endroits du monde.

Vous allez maintenant télécharger R, sur la page d'accueil de l'adresse: http://www.r-project.org/
Cliquer alors sur CRAN dans le menu "Download, Packages", choisir un site miroir, une nouvelle page s'ouvre et dans la fenêtre "Download and Install R" choisir la version qui correspond à l'interface de votre ordinateur.

Download R for Linux
Download R for MacOS X
Download R for Windows

Une fois l'installation terminée, il vous suffit de cliquer sur le programme dans le répertoire où il est installé ou à partir du raccourci qui s'est installé dans vos programmes, ou applications, et cela en fonction du système d'exploitation de votre ordinateur.

1.2 - R au démarrage

Au démarrage, R ouvre une fenêtre appelée "console" où s'affiche le texte suivant:

```
R version 2.15.0 (2012-03-30)
Copyright (C) 2012 The R Foundation for Statistical
Computing
ISBN 3-900051-07-0
Platform: i386-apple-darwin9.8.0/i386 (32-bit)
R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines
conditions.
Tapez 'license()' ou 'licence()' pour plus de
détails.
R est un projet collaboratif avec de nombreux
contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les
publications.
Tapez 'demo()' pour des démonstrations, 'help()'
pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au
format HTML.
Tapez 'q()' pour quitter R.
>
```

Le symbole '>' qui apparaît à la fin du texte est appelé prompt, il signifie que R attend vos questions.

En effet, R fonctionne sur le mode question / réponse.

Vous lui posez une question et si cette dernière est bien libellée alors il vous répond. La difficulté sera donc de bien rédiger les questions selon le format de programmation accepté par le logiciel, car il est têtu et peut ne pas vous répondre.
Tapez votre question, assurez-vous qu'elle est correctement rédigée, puis appuyez sur la touche "entrée" de votre clavier, si vous utilisez directement la console pour poser votre question.

R exécute alors votre question (ou votre commande) puis se repositionne sur le prompt '>'. Vous pouvez alors lui poser une nouvelle question.
Si votre question est incomplète, le symbole '+' apparaîtra. Il signifie que R attend des informations complémentaires, souvent il s'agit d'une parenthèse oubliée, mais si vous ne savez pas ce que R attend, il suffit d'appuyer sur la touche "echap" ou "esc" de votre clavier pour annuler votre question et faire réapparaitre le prompt '>'. Il vous faut alors reformuler votre question en y apportant les corrections nécessaires pour que R comprenne votre demande.
Pour faire réapparaître votre question sans avoir à la réécrire vous pouvez utiliser les flèches de déplacement (vers le haut ou vers le bas) de votre clavier. Ainsi vous n'aurez plus qu'à apporter les corrections nécessaires à votre précédente question pour qu'elle devienne compréhensible par R avant de valider avec la touche "entrée".

1.3 - Créer des objets sous R

1.3.1 - Les objets sous R

On retiendra deux grands types d'objets sous R:

- les données

Elles peuvent se présenter sous forme de vecteurs, de matrices, de listes, de tables de données. Elles peuvent être créées directement avec R ou être importées à partir de données au format Excel, Stata, CSV…
Les données créées ou importées dans R peuvent être utilisées avec une fonction, être modifiées ou décrites à l'aide de différentes fonctions selon qu'elles sont sous forme de variables quantitatives ou qualitatives.

Afin de permettre à R de reconnaître les objets de type 'données', il faut donc leur attribuer un nom, qui prendra la forme d'une chaîne alphanumérique c'est à dire permettant d'avoir des lettres en majuscule ou minuscule et des chiffres. Cependant le nom d'un objet de type "donnée" devra toujours débuter par une lettre. Le seul élément de ponctuation autorisé dans le nom d'un objet de type donnée est le point '.'.
Attention, le logiciel R est sensible à la casse ainsi il différenciera les mots : Test, test, TEST et les considérera comme 3 objets différents. Il sera donc important de reprendre la bonne écriture d'un objet quand on posera des questions à R.

- Les fonctions

Elles permettent de modifier ou de décrire les données. Elles sont également utilisées pour réaliser des calculs et des tests statistiques ainsi que des graphiques.

Une fonction sous R s'écrit généralement en minuscule et est suivie de parenthèses dans lesquelles sont insérés, si besoin, des paramètres en rapport avec la fonction utilisée.

Les paramètres inscrits entre les parenthèses seront séparés par des virgules. Donc pour les nombres décimaux, il faudra utiliser l'écriture anglo-saxonne c'est à dire le point (3.5 et non pas 3,5).

Exemple : la fonction **sum()** - on peut calculer la somme de 5 et 6 en écrivant à la suite du prompt
>sum(5,6)

Ce à quoi R devrait vous répondre
```
[1] 11
```

Conseil
Sachant que généralement les fonctions sont nommées par des lettres en minuscule (ex: sum()), il est recommandé pour les objets de type 'données' que vous créerez d'utiliser une écriture commençant par exemple par une majuscule afin de les identifier plus rapidement au sein de vos futurs programmes.
Exemple : Matable

1.3.2 - Manipuler des objets

- Assigner une valeur à un objet R

- Pourquoi assigner une valeur à un objet?
- Il est souvent plus facile de manipuler un objet auquel on aura affecté le résultat d'une fonction ou d'un calcul. Ainsi dans la session active de R les objets créés pourront être appelés à tout moment, sans avoir à relancer la procédure qui a permis de les définir.
- Il y a 3 façons différentes pour assigner une valeur à un objet R en utilisant les symboles suivants: <- ou = ou ->.

Exemple

```
>Chiffre.1<-5
>Chiffre.2=6
>7->Chiffre.3
```

Rien ne semble se passer, évidemment puisque vous avez juste attribué des valeurs à des objets, pour visualiser le contenu des objets, écrivez donc maintenant

```
>Chiffre.1; Chiffre.2; Chiffre.3
```

Cette dernière ligne permet de voir la valeur qui a été attribuée à chacun des objets R. Le 'point-virgule' permet d'écrire sur la même ligne plusieurs commandes qui seront

lues par R comme s'il s'agissait de lignes différentes. Ainsi la dernière commande aurait pu s'écrire:
>Chiffre.1
>Chiffre.2
>Chiffre.3

Maintenant vous êtes prêts pour manipuler des objets R.

Conseils
Le symbole '#' est utilisé pour mettre des commentaires dans vos programmes. Les informations qui suivent ce symbole ne sont pas considérées par R comme faisant partie des questions que vous lui adressez. Elles permettront d'expliquer ce que vous avez demandé à R. Ce symbole est utile pour ajouter des notes de commentaires dans un programme en cas de partage avec d'autres utilisateurs ou également pour revenir sur un programme que vous avez écrit.

- R comme calculatrice

R autorise l'utilisation des opérateurs classiques :
Addition "+"
Soustraction "-"
Multiplication "*"
Division "/"
Puissance "^"

Exemple

```
>10/3
[1] 3.3333333
# aucun objet n'a été ici assigné au résultat de la division

>20+100
[1] 120
```

```
>Chiffre.1+Chiffre.2
[1] 11

# vous auriez pu également utiliser la fonction sum()
>sum(Chiffre.1,Chiffre.2)
[1] 11
```

- Manipuler des chaînes de caractères avec R

Dans R, les chaînes de caractères doivent être mises entre guillemets : " " ou ' '.

Exemple

```
>A.1<-"Bonjour"
>A.1
[1] "Bonjour"
```

La fonction **paste()** permet de concaténer des chaines de caractères.

Exemple

```
>paste(A.1,"Utilisateurs de R",sep="-")
```

sep est un paramètre qui permet de définir le séparateur qui sera utilisé entre les chaînes de caractère. Ici, il vous a été proposé d'utiliser le trait d'union, mais vous auriez pu également utiliser juste un espace en écrivant sep=" ".

- Retrouver la liste des objets créés dans la console

Il est intéressant de pouvoir lister l'ensemble des objets que vous avez créés avec R, soit pour en rappeler un dont vous avez oublié la casse utilisée (majuscule ou minuscule), soit pour en effacer certains que vous ne souhaitez plus utiliser ou

que vous craignez de confondre avec un nouvel objet ayant un nom similaire.
Pour réaliser cette procédure, R vous propose la fonction **ls()**.

Exemple
>ls()
```
[1] "A.1"  "Chiffre.1"  "Chiffre.2"  "Chiffre.3"
```

- Effacer des objets de la liste

Pour effacer un objet vous utiliserez la fonction **rm()**, en mettant entre les parenthèses le nom de l'objet que vous voulez supprimer

Exemple
>rm(Chiffre.1)
# pour vérifier si votre commande a été prise en compte utiliser la fonction ls()
ls()
```
[1] "A.1"  "Chiffre.2"  "Chiffre.3"
```

Conseil
Si vous souhaitez supprimer l'ensemble des objets de la console, vous utiliserez la commande suivante:
>rm(list=ls())

1.4 - Retrouver la liste des commandes utilisées

Si vous souhaitez voir les dernières commandes que vous avez utilisé dans la console, vous utiliserez la fonction **history()**. Elle va ouvrir une fenêtre dans laquelle vous verrez apparaître les dernières commandes utilisées.
Pour avoir accès à toutes les commandes, il faudra ajouter le paramètre Inf (pour infini) entre les parenthèses de la fonction : history(Inf).

Vous pourrez ainsi sauvegarder vos commandes dans un fichier texte ou les copier/coller dans un script de programme.

1.5 - Obtenir de l'aide avec R

Vous pouvez avoir besoin d'aide notamment pour savoir comment utiliser une fonction et quels paramètres insérer entre les parenthèses. Vous utiliserez alors la fonction **help()** ou le '**?**' suivi du nom de la fonction pour laquelle vous souhaitez avoir de l'information.

Exemple
les deux écritures suivantes sont équivalentes
>help(sum)
>? sum

Si vous souhaitez avoir accès directement à l'ensemble de l'aide au format html à partir de votre navigateur (par exemple Mozilla ou Internet-explorer), tapez:

> help.start()

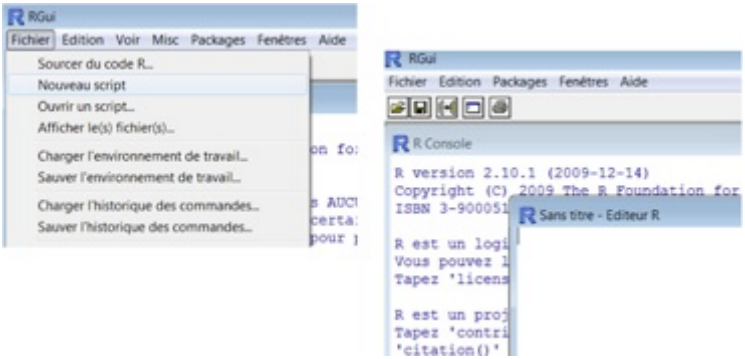
1.6 - Installer une extension (package)

R utilise un système d'extension, les packages, pour répondre à des besoins d'analyses spécifiques. Le logiciel R est d'ailleurs lui même une bibliothèque appelée "base" contenant un certain nombre d'extensions utiles à son fonctionnement de base.
Chaque extension est une collection regroupant des outils d'une même thématique. Certaines extensions sont automatiquement disponibles lorsque R est lancé. Il suffit de

taper la fonction **search()** pour connaître la liste des extensions lancées dans votre session active.
Vous devrez, en fonction de vos besoins et votre progression en matière d'analyse en statistique, installer d'autres extensions. Il faudra les télécharger sur le CRAN puis dans le menu 'software' de la page d'accueil vous devrez cette fois-ci cliquer sur 'packages'. La liste des extensions s'affiche et vous devrez alors choisir celle que vous souhaitez télécharger. Vous aurez le choix entre trois fichiers en fonction de votre interface (Windows, MacOs ou Linux). Sous MacOs ou Windows, vous utiliserez le menu Packages de votre logiciel pour installer le fichier téléchargé dans votre environnement R.
Une fois installées, les extensions seront alors appelées dans la console avec la fonction **library()**.

1.7 - Utiliser l'éditeur de R

Plutôt que de continuer à taper vos commandes dans la console, il est conseillé d'avoir recours à un éditeur qui vous permettra de sauvegarder vos commandes mais aussi de les modifier plus facilement face au refus de R de répondre à votre question.
Il existe un éditeur intégré dans votre environnement R, il suffit de l'appeler dans le menu fichier, sous Windows choisir l'option nouveau script.



Avec R dans MacOs, choisir l'option nouveau document dans le menu fichier.
A partir de maintenant, toutes les commandes seront saisie dans la fenêtre d'édition de script. Pour transférer vos questions vers la console de R, il suffira de taper Crtl+R pour les ordinateurs fonctionnant sous Windows et Cmd+touche "Entrée" sous Mas OS.

1.8 - Répertoire de Travail

Pour sauvegarder le fichier de commande et l'ensemble des fichiers créés lors d'une session de travail, il est nécessaire de définir un répertoire de travail.
A l'ouverture R assigne un répertoire de travail par défaut pour le connaître, il suffit d'utiliser la commande **getwd()** selon la procédure suivante:

```
>getwd()
# si vous utilisez la fenêtre d'édition de script, n'oubliez pas d'envoyer votre question dans la console Crtl+R (Windows) ou cmd+"Entrée"(MacOs)
```

(NB: w pour working, d pour directory)

Le répertoire est alors le répertoire par défaut utilisé par R.
Vous pouvez définir votre propre répertoire de travail afin que chacune de vos sessions de travail soit sauvegardée avec les fichiers de données que vous exploiterez avec R. Vous utiliserez alors la commande suivante **setwd()**:

```
>setwd(dirname(file.choose()))
```

Cependant il faut avant vous assurer que, dans le répertoire que vous souhaitez utiliser, il y ait au moins un fichier quelle que soit son extension. En effet cette commande ne vise pas à ouvrir un fichier mais juste à définir le répertoire de travail sans avoir à écrire l'ensemble du chemin qui peut être parfois fastidieux et source d'erreur.

NB:

file.choose() permet de choisir un fichier de façon interactive d'où la nécessité d'avoir au moins un fichier dans le répertoire.
dirname() extrait uniquement le chemin du fichier

1.9 - Quitter R

Pour quitter R, vous pouvez utiliser la fonction suivante **q()**:

```
>q()
```

ou vous pouvez quitter en fermant la fenêtre R.

NB: lorsque vous quitterez R, une boite de dialogue s'affichera pour vous demander si vous souhaitez sauvegarder votre espace de travail.

Si vous répondez positivement à cette question, R créera deux fichiers :

- Un fichier avec l'extension .RData contenant les objets créés lors de la session que vous venez de quitter. Il vous suffira d'ouvrir ce fichier dans R pour avoir accès à vos objets.
- Un fichier avec l'extension .Rhistory pouvant être ouvert à l'aide de l'éditeur de texte contenant les commandes.

Cependant, si vous avez utilisé un éditeur pour écrire vos commandes au fur et à mesure, vous n'aurez pas besoin de ces 2 fichiers. Il vous sera facile de recréer l'ensemble des objets en relançant votre fichier de commandes sauvegardé depuis l'éditeur. Fort de cette information, vous répondrez alors par la négative à la boite de dialogue précédente.

Pour relancer rapidement votre script, utiliser alors dans R la commande **source()**:

```
>source(file.choose())
```

indiquer le chemin de votre fichier contenant votre script et cliquer sur ce fichier, il sera lu du début à la fin par R et ainsi l'ensemble de vos objets sera recréé.

1.10 - Référence de R

La fonction **citation()** permet d'afficher les éléments nécessaires pour les références de R dans une publication scientifique.

Exercices d'application

Vous utiliserez l'éditeur de R pour ces exercices et vous vous appliquerez à bien annoter votre programme R pour qu'il soit compréhensible à toute personne.

Ex1 - Effacer tous les objets contenus dans la console R et vérifier que la console ne contient plus d'objets

Ex2 - Manipulation de chaînes de caractère

Créer un objet R que vous appellerez Prenom et qui contiendra votre prénom

Créer un objet R que vous appellerez Nom et qui contiendra votre prénom

A l'aide de ces deux objets, créer un objet R appelé Identite qui donnera votre nom suivi de votre prénom, séparé par une virgule.

Afficher le contenu de cet objet

Ex3 - Manipulation de données numériques

Créer un objet R appelé Femme contenant la valeur 257 836, correspondant à la population féminine d'une région d'une année.

Créer un objet R appelé Homme contenant la valeur 247 643, correspondant à la population masculine d'une région d'une année.

Créer un objet R appelé Population à partir des deux objets Femme et Homme. Afficher le total de la population de la région dans la console.

Créer un objet R appelé Sexratio qui sera le rapport Homme/Femme. Afficher la valeur de cet objet dans la console.

Si le nombre de décès dans la région au cours de l'année chez les femmes est de 236 et celui des hommes de 328, à l'aide d'objets R vous donnerez le taux de mortalité pour 1000 personnes pour chacun des 2 sexes et pour l'ensemble de la population. Afficher les valeurs des objets dans la console

Ex4 - Afficher dans la console la liste des objets créés dans R

Chapitre 2

Manipuler les données au travers des objets et des opérateurs de R

Objectifs à atteindre à cette étape
Connaître les différents types d'objets de R
Connaître les opérateurs utilisés dans R
Connaître les attributs spéciaux et leur utilisation
Générer des données avec R
Manipuler des données
Exporter de données au format csv depuis R
Importer des données vers la console R

2.1 - Les types d'objets sous R

Sous R, un objet permet de stocker différentes informations.

Un vecteur

C'est un objet à une seule dimension contenant des informations de même nature, même mode : soit toutes numériques, soit toutes alphanumériques (chaînes de caractères), soit logiques (TRUE/FALSE), soit vides. On utilise par exemple la fonction **c()** pour créer des vecteurs. Un vecteur peut donc contenir plusieurs valeurs.

Exemple
>Age<- c(2,4,5,10,8,7,12,11,3,5,6)

Nature des données contenues dans les objets

Nature		
null	Objet vide	NULL
logical	booléen	TRUE, FALSE ou T, F
numeric	Nombre réel	12, 2.56, pi, 1e-10
complex	Nombre complexe	2i
character	Chaîne de caractère	'Bonjour', "Féminin "

La fonction **mode()** permet de retrouver la nature des informations d'un objet.

Exemple

>mode(Age)

cette fonction permet de vérifier la nature des valeurs contenues dans le vecteur Age.

Une matrice

C'est un objet à deux dimensions contenant des informations de même nature : soit toutes numériques, soit toutes alphanumériques (chaînes de caractères), soit logiques, soit vides. On utilise la fonction **matrix()** avec le paramètre ncol qui indique le nombre de colonnes.

Exemple

Créer un vecteur contenant les nombres de 1 à 15.
>Chiffre<-c(1:15)

Créer une matrice à partir de ce vecteur.
>Matrice.1<-matrix(Chiffre,ncol=3)
>Matrice.1 # pour voir le contenu de l'objet Matrice.1

Par défaut, le remplissage des matrices se fait par colonne. Si vous souhaitez un remplissage par ligne, il faut ajouter le paramètre byrow=TRUE.

>Matrice.2<-matrix(Chiffre,ncol=3,byrow=TRUE)
>Matrice.2

Une liste

C'est un objet qui vous permettra de stocker des objets différents qui n'auront pas forcément le même mode ni la même longueur. On utilise la fonction **list()** pour le créer.

Exemple

```
>Maliste<-list(Age, Matrice.1,Matrice.2)
```

Un tableau de données (data frame)

Un tableau de données comprend des colonnes de même longueur mais de modes qui peuvent être différents. Les colonnes correspondent à des variables et les lignes à des enregistrements. Vous manipulez habituellement ces objets pour vos données collectées à partir de tableurs. On utilise la fonction **data.frame()** pour les créer sous R. Vous apprendrez par la suite comment importer des données déjà collectées dans un tableur.

Exemple

- Créer un vecteur Sexe contenant 30 valeurs 16 hommes et 14 femmes.

```
>Sexe<-rep(c("M","F"),c(16,14))
```
#La fonction **rep()** permet de créer un vecteur en répétant certaines informations.

- Créer un vecteur Age comprenant 30 valeurs.

```
>Age<-rep(seq(10,14.5,by=0.5),3)
```

La fonction **seq()** permet de créer un vecteur contenant les valeurs comprises entre 10 et 14.5 avec un pas de 0.5 et la fonction rep() associée de répéter ce vecteur 3 fois.

- Créer la table de données.

>Matable<-data.frame(Sexe,Age)

La fonction **View()**, qui a la particularité de commencer par une majuscule, permet d'afficher dans une fenêtre indépendante les données de Matable.

>View(Matable)

La fonction **str()** permet de voir la structure de la table avec les caractéristiques de chacune des colonnes constituant cette table.

>str(Matable)

2.2 - Les opérateurs

Le chapitre 1.3.2 vous a permis de découvrir les opérateurs arithmétiques classiques (+, -, *, /, ^) et les opérateurs d'assignation (<-, = ou ->). Il existe d'autres types d'opérateurs dans R pour permettre la manipulation des données.
Les opérateurs de comparaisons:

>	supérieur
<	inférieur
<=	inférieur ou égal
>=	supérieur ou égal
==	*égal*
!=	différent

NB: vous remarquerez l'utilisation du double signe égal "==" car le signe simple "=" est reconnu par R comme opérateur d'assignation (cf page 12).

Les opérateurs logiques

&	*et*
|	*ou*
!	*non*

Les opérateurs d'extraction

[ ]
$

Ils permettent d'extraire une partie des données. Vous apprendrez à les utiliser dans les exemples qui suivent.

Exemples:

Créer un vecteur appelé Age comme indiqué ci-dessous.

```
>Age<-c(2,4,5,10,8,7,12,11,3,5,6)
# la fonction c() permet de créer un vecteur.
```

Pour extraire de l'objet Age, les âges supérieurs ou égaux à 5 et inférieurs à 10, vous écrirez:

```
>Age[Age>=5 & Age<10]
#vous venez d'utiliser des opérateurs logiques, des opérateurs de comparaison et des opérateurs d'extraction.
```

Pour extraire les âges inférieurs à 5 ainsi que les âges supérieurs à 10, vous écrirez:

>Age[Age<5|Age>10]

Pour extraire les âges qui sont différents de 5, vous écrirez:

>Age[Age!=5]

Pour extraire la quatrième valeur du vecteur Age, vous écrirez:

>Age[4]

2.3 - Les attributs spéciaux

Certains attributs réalisent des fonctions et d'autres ne font rien.

NA	Pour les valeurs manquantes. NB: NA pour "not available"
TRUE (ou T)	Valeur logique pour Vrai.
FALSE (ou F)	Valeur logique pour Faux.
NULL	Créer un objet vide.
Inf (ou -Inf)	Valeur pour l'infini (ou moins l'infini).
letters	Pour obtenir les 26 lettres de l'alphabet en minuscule.
pi	π=3,141593.
ifelse()	Fonction permettant d'affecter une valeur à un objet en fonction de la valeur d'un autre objet.
if(){}else{}	Mêmes attributions que la fonction précédente.

for(){}
while() {}
repeat{}
	Pour les répétitions en boucle.

Exemples

```
>Rayon<-c(5,6,7,8,12,4,3,10,8,9)
>Parcelle<-letters[1:10] # les 10 premières lettres de l'alphabet
>Aire<-pi*Rayon^2
>Parcelle;Aire
>Limite.1<-ifelse(Aire<100,0,1)
>Limite.1
>Limite.2<-NULL #Créer un objet vide utilisé par la boucle
>for(i in 1:10) {
        if(Aire[i]<100){Limite.2[i]<-0}else{Limite.2[i]<-1}}
>Limite.2
# Avec Limite.1 et Limite.2, vous avez deux méthodes pour un même résultat.

>Aire[Parcelle=="e"]
#Cette commande permet de rechercher la valeur de l'aire de la parcelle "e"
```

2.4 - Manipuler des données

Les données qualitatives peuvent être :

- soit binaires ("Oui","Non"),
- soit nominales ("Cheval", "Vache", "Mouton"),
- soit ordinales ("CP","CE1","CE2","CM1","CM2").

La différence entre les variables nominales et ordinales tient au fait qu'il y a pour ces dernières un classement dont l'ordre de chacun des niveaux, qui la composent, a son importance.

fonctions		
factor	Factorisation	"Masculin", "Féminin".
ordered	Factorisation ordonnée	"CP","CE1","CE2","CM1", "CM2".

Exemple

Vous avez estimé l'intensité d'une douleur lors d'une étude.

```
>Niveau<-rep(c("Supportable","Faible","Intense","Forte",
"Absente"),c(5,16,18,7,22))
>summary(Niveau)

>Niveau.1<-ordered(rep(c("Supportable","Faible","Intense",
"Forte","Absente"),c(5,16,18,7,22)),levels=c("Absente",
"Faible", "Supportable", "Forte","Intense"))
 >summary(Niveau.1)
```

NB : La fonction **ordered()** vous permet de présenter vos résultats par ordre croissant d'importance.

Les données de format "character" sont factorisées automatiquement, mais pas les données numériques. Pour ces dernières, il est nécessaire d'utiliser la fonction **factor()** pour que R ne les considère pas comme des données quantitatives.

Les données quantitatives peuvent être:

- Soit continues (mesure avec une unité de mesure, ex: âge)
- Soit discrètes (ex: nombre d'enfants)

- Soit temporelles (ex: dates)

Pour qu'une variable quantitative soit considérée comme une classe, vous utiliserez la fonction factor().

Si vous décidez de coder votre estimation de la douleur de l'exemple précédent avec des chiffres

```
>Niveau.A<-rep(c(3,2,5,4,1),c(5,16,18,7,22))
>summary(Niveau.A)
```

la fonction summary considère votre variable comme une variable quantitative, il faut donc la factoriser pour qu'elle soit considérée comme une variable qualitative.

```
>Niveau.A1<-factor(rep(c(3,2,5,4,1),c(5,16,18,7,22)))
>summary(Niveau.A1)
#ou utiliser la fonction table() à la place de summary()
>table(Niveau.A)

>Niveau.A2<-ordered(rep(c(3,2,5,4,1),c(5,16,18,7,22)),
levels=c(5,4,3,2,1))
>summary(Niveau.A2)
```

2.5 - Manipuler des dates

Généralement, les champs « dates » des fichiers que vous importerez seront de type "character".
Pour convertir ces données en date reconnue par R, il faut utiliser l'une ou l'autre des fonctions **as.Date()** ou **strptime()**.

Exemples

```
>Dcalend<-c("17/02/92","27/02/92", "14/01/92","28/02/92",
"01/02/92","02/01/92")
```

```
>Dcalend
>Dcalend.2<-as.Date(Dcalend,"%d/%m/%y")
>Dcalend.2
```

Dans ce premier exemple, l'année était exprimée avec deux chiffres nous avons utilisé le paramètre 'y' en minuscule dans la fonction as.Date().
Quand l'année est exprimée avec quatre chiffres, on utilise alors le paramètre 'Y' en majuscule.

Exemples

```
>Dcalend.1<-c("17/02/1992","27/02/1992", "14/01/1992",
"28/02/1992", "01/02/1992", "02/01/1992 ")
>Dcalend.21<-as.Date(Dcalend.1,"%d/%m/%Y")
>Dcalend.21
```

Extraire des parties de date

- Le numéro de semaine d'une date:

```
>as.character(Dcalend.21, "%U")
```

- L'année d'une date

```
# pour l'année avec quatre chiffres.
>as.character(Dcalend.21, "%Y")
#autre possibilité la fonction substr()
>substr(Dcalend.21,1,4)

# pour l'année à deux chiffres.
>as.character(Dcalend.21, "%y")
#autre possibilité la fonction substr()
>substr(Dcalend.21,3,4)
```

- Le mois d'une date

```
>as.character(Dcalend.21,"%m")
#autre méthode
>substr(Dcalend.21,6,7)
```

- Le jour d'une date

```
>as.character(Dcalend.21,"%d")
#autre méthode
>substr(Dcalend.21,9,10)
```

- Le jour dans l'année

```
>as.character(Dcalend.21,"%j")
# pour le jour de l'année entre 1 et 365.
```

Faire une différence entre deux dates
Vous utiliserez la fonction **difftime()** pour faire la différence entre deux dates.

Exemple

```
>Dcalend.22<-c("23/02/1993","17/07/1994", "24/10/1995",
"06/03/1992", "11/11/1992", "02/01/1993 ")
>Dcalend.22<-as.Date(Dcalend.22,"%j/%m/%Y")

>Temps<-difftime(Dcalend.22,Dcalend.21, units="days")
```

NB: Le paramètre units de la fonction difftime peut aussi prendre les valeurs "weeks", "mins","secs","hours" ou "auto". Attention, l'objet 'Temps' n'est pas considéré comme une variable numérique. Si vous souhaitez l'utiliser pour des calculs, il faudra d'abord le transformer en variable numérique avec la fonction **as.numeric()**.

```
>Temps<-as.numeric(Temps)
```

2.6 - Créer une table de données

Vous pouvez créer une base de données avec R mais ce n'est pas la meilleure façon pour entrer des données d'une étude.

Exemple d'une table de données générées avec R
>set.seed(3)
>Sexe<-rbinom(31,size=1,prob=.48)
La fonction **rbinom()** permet de générer de façon aléatoire des données selon une distribution binomiale. La fonction set.seed() permet de fixer le tirage réaliser par la fonction rbinom() et de s'assurer ainsi que chacun d'entre vous aura un objet R, appelé Sexe, qui aura le même contenu.
>Sexe1<-ifelse(Sexe==0,"Femme","Homme")
>Tranchage<-rep(c(" 0-4 ", " 5-9 ", " >10 "),c(8,9,14))
>Poids<-seq(10,25,by=0.5)
>Matable<-data.frame(Sexe,Sexe1,Poids, Tranchage)
>Matable[1:5,]
Cette dernière commande permet de ne voir que les 5 premières lignes.

>summary(Matable)
La fonction **summary()** permet de décrire la distribution de l'ensemble des données selon leur nature.

NB :
rnorm() # Fonction pour générer des données selon une loi de distribution normale.
Exemple: >rnorm(10) ou >rnorm(10,mean=100).

runif() # Fonction pour générer des données selon une loi de distribution uniforme.
Exemple: >runif(10) ou >runif(10,min=20,max=21).

2.7 - Exporter des données

Pour exporter des données, vous utiliserez la fonction **write.csv2()** qui transformera votre table de données en format texte avec le point virgule comme séparateur des colonnes.

Exemple

Vous allez ici exporter la table de données que vous avez générée au chapitre 2.6. Elle contient les variables suivantes: Sexe, Sexe1, Poids, Tranchage.

```
>write.csv2(Matable,file="Matable2_6.csv", row.names=F)
```

Le premier paramètre de la fonction correspond au nom de l'objet R que vous voulez exporter.
Le second paramètre qui commence par file= vous permet de définir le nom que vous souhaitez donner au fichier de sortie sans oublier l'extension .csv et les guillemets.
Le troisième paramètre row.names qui prend ici la valeur F (FALSE) permet de ne pas prendre en compte l'identifiant des lignes attribué automatiquement par R. Vous pouvez souhaiter garder la colonne correspondant à l'identifiant des lignes en omettant alors ce paramètre dans la fonction.

NB: Le fichier va se créer dans votre répertoire de travail. Si vous n'avez pas mémorisé votre répertoire de travail, vous pouvez le retrouver avec la fonction getwd().
Il est conseillé d'identifier et de modifier si nécessaire son répertoire de travail à chaque ouverture d'une nouvelle session de travail sur R.

2.8 - Importer des données dans R

Si vos données ont été préalablement saisies dans un tableur, il suffit de les enregistrer dans un format texte avec le point-virgule ";" comme séparateur des colonnes, généralement il s'agit de fichiers avec l'extension ".csv".

Vous utiliserez alors la fonction **read.table()** pour importer vos données. Certains paramètres doivent être fournis à la fonction read.table() pour lui permettre d'importer les fichiers au format csv.

Pour éviter de transcrire l'ensemble du chemin où se trouve le fichier à importer et éviter ainsi des erreurs, on utilisera dans les paramètres la fonction **file.choose()**.

Il sera nécessaire d'indiquer à R que les colonnes sont séparées par un point-virgule. Vous utiliserez le paramètre sep=";".

S'il s'agit de fichiers de données françaises, les décimales sont séparées par une virgule, mais R ne reconnaît que le point comme séparateur des décimales. Vous devrez donc préciser à R que dans votre fichier le séparateur est une virgule et R la substituera par un point. Vous utiliserez le paramètre dec=",".

Si vous souhaitez préciser à R que la première ligne contient le nom de colonnes (ou des variables), alors vous utiliserez le paramètre header=TRUE ou header=T.

Vous écrirez alors dans la console de R ou dans l'éditeur:

```
>Matable<-read.table(file.choose(),header=T,sep=";",dec=",")
```

Pour cet exemple, vous pouvez réimporter le fichier Matable2_6.csv que vous avez créée par exportation au chapitre 2.7.

NB: Vous devez affecter à un objet R le résultat de la fonction read.table(). Ici, nous l'avons appelé Matable. Cet objet sera de type data frame (table de données).
L'importation d'une table de données dans R va permettre de réaliser des statistiques sur ces données.

Importer un fichier de format .dta (logiciel Stata®)

Pour importer un fichier de ce type, il est nécessaire de charger la bibliothèque "foreign" et de l'appeler ensuite dans la console avec la fonction library().

```
>library(foreign)
>read.dta(file.choose())
```

Importer un fichier de format.xls (logiciel Excel®)

Pour importer un fichier de ce type, il est nécessaire de charger la bibliothèque "xlsReadWrite" pour windows et de l'appeler ensuite dans la console avec la fonction library().

```
>library(xlsReadWrite)
>read.xls(file.choose(),colNames=TRUE,sheet=1,from=1,
rowNames=T)
```

NB: Il est déconseillé d'utiliser R comme outils de saisie pour des collections de données importantes.

Si vous ne connaissez pas le format du fichier texte que vous souhaitez importer, vous pouvez l'afficher avec la fonction **file.show()**. Ainsi il vous sera possible d'explorer le fichier à importer : Quel est le séparateur des colonnes? Est-il différent du point-virgule?

Quel est le séparateur des nombres décimaux? Est-ce une virgule ou un point?
La première ligne du fichier contient-elle le nom des colonnes (des variables)?

Exemple

```
>file.show(file.choose())
```

Pour cet exemple, vous choisirez le fichier Matable2_6.csv que vous avez créée par exportation au chapitre 2.7.

Exercices d'application

Ex5 - Manipulation de champs "dates"

Créer deux objets R de type vecteur que vous nommerez Datdeb et Datfin. Le premier contiendra les chaines de caractères suivantes: 12/01/90, 02/01/90, 22/04/95, 05/06/91; le second les valeurs suivantes: 25/02/1990, 15/06/1991, 20/05/1996, 14/04/1992.
Les convertir ensuite au format date.
Créer un objet appelé Duree qui contiendra la durée entre Datdeb et Datfin et l'afficher sur la console.
Créer un objet An qui ne contiendra que les années de l'objet Datdeb, un objet Mois qui ne contiendra que les mois de l'objet Datdeb et un objet Jour qui ne contiendra que les jours de l'objet Datdeb. Afficher sur la console chacun de ces objets.

Ex6 - Extraction de données

Créer un objet appelé Poids qui sera une séquence de nombres de 20 à 45 avec un pas de 0.5 (cf page 25)
Afficher le contenu de l'objet Poids
Extraire la 16° valeur de l'objet Poids
Extraire les 1°, 16° et 31° valeurs de l'objet Poids
Extraire les poids dont la valeur est supérieure à 38
Extraire les poids dont les valeurs sont à la fois supérieures à 25 et inférieures à 37.
Créer un objet R appelé Classpoids qui prendra la valeur 1, si le poids est inférieur à 25, la valeur 2 si le poids est compris entre 25 et 30 et 3 si le poids est supérieur à 30. Quel est l'effectif de chacun des groupes?

Chapitre 3

Explorer et nettoyer les tables de données avec R

Objectifs à atteindre à cette étape
Connaître le contenu d'une table de données
Connaître et modifier les noms de variables
Créer de nouvelles variables
Identifier les données aberrantes
Fusionner des tables de données

3.1 - S'assurer de la complétude de l'importation

Après avoir importer dans un objet R des données provenant d'une base de données ou d'un tableur, il est important de s'assurer que toutes les lignes (ou enregistrements) et toutes les colonnes (variables) ont bien été importées.

Vous utiliserez la fonction **dim()** qui vous permettra de connaître le nombre de lignes et le nombre de colonnes contenues dans l'objet R que vous avez créé.

Exemple

Pour cet exemple, vous pouvez réimporter le fichier Matable2_4.csv que vous avez créé par exportation au chapitre 2.6.

```
>Matable<-read.table(file.choose(),header=T,sep=";",dec=",")

>dim(Matable)
[1] 31  4

#pour avoir uniquement le nombre de lignes.
>dim(Matable)[1]
[1] 31

#pour avoir uniquement le nombre de colonnes.
>dim(Matable)[2]
[1]  4
```

On peut aussi dans ce cas-là utiliser la fonction **length()**.

```
>length(Matable)
[1]  4
```

3.2 - Retrouver le nom des variables

Après l'importation mais aussi en cours de programmation vous aurez besoin de voir comment les variables ont été nommées afin de les utiliser sans faire d'erreur dans leur retranscription. Vous avez à votre disposition la fonction **names()**.

Exemple
>names(Matable)

```
[1] "Sexe"      "Sexe1"     "Poids"     "Tranchage"
```

3.3 - Connaître la structure d'une table de données

Il est possible d'avoir rapidement l'ensemble des informations précédentes et une information supplémentaire sur la structure de la table de données avec la fonction **str()**.
Vous aurez alors accès aux dimensions de votre table de données avec le nombre d'observations et le nombre de variables, le nom des variables et le mode des variables.

>str(Mydata)

```
'data.frame':  31 obs. of  4 variables:
 $ Sexe     : num  0 1 1 1 1 1 1 0 1 1 ...
 $ Sexe1    : Factor w/ 2 levels "Femme","Homme": 1 2 2 2
       2 2 2 1 2 2 ...
 $ Poids    : num  10 10.5 11 11.5 12 12.5 13 13.5 14
       14.5 ...
 $ Tranchage: Factor w/ 3 levels " >10 "," 0-4 ",..: 2 2
       2 2 2 2 2 3 3 ...
```

3.4 - Voir les données importées

Pour voir si l'importation s'est bien déroulée vous pouvez choisir d'afficher les données.

Si vous tapez le nom de l'objet R dans la console, l'ensemble des données va s'afficher mais le résultat ne sera pas très facile à explorer, vous pouvez alors choisir de n'afficher que les premières lignes de la table de données avec la fonction **head()**.
Cette fonction n'affiche par défaut que les six premières lignes, il est possible de lui demander d'en afficher plus, ou moins, en ajoutant le nombre de lignes désirées.

Exemple
>head(Matable)#affiche par défaut les 6 premières lignes.
>head(Matable,2)#affiche alors les 2 premières lignes.
>head(Matable,10)#affiche alors les 10 premières lignes.

Vous pouvez aussi choisir d'afficher les données dans une nouvelle fenêtre en utilisant la fonction **View().**
NB: la fonction View() commence par une majuscule.

Exemple
>View(Matable)

3.5 - Connaître l'identifiant des observations d'une table de données

La fonction **row.names()** permet de connaître l'identifiant des observations à condition d'avoir attribuer ces informations à row.names(). Mais connaître l'identifiant des observations n'a pas vraiment d'intérêt. Cependant cette fonction peut être utilisée afin de modifier le nom d'une observation.

Exemple
#Attribuer des identifiants aux observations de l'objet >Matable.
>Ident<-paste("VR",c(1:31)) #la fonction paste() permet de concaténer des chaînes de caractères

>row.names(Matable)<-Ident
>View(Matable) # vous permet de voir le résultat sur les données de l'objet Matable.

Pour effacer l'identifiant des lignes, on attribuera alors le paramètre NULL en utilisant la fonction row.names().

Exemple
>row.names(Matable)<-NULL
>View(Matable)

NB: Pour redonner les valeurs des identifiants, il vous suffira de vous rapporter à l'exemple précédent. Dans un cas plus général, vous pourrez aussi, avant d'effacer les identifiants avec le paramètre NULL, les sauvegarder dans un objet vecteur de R. Vous écrirez alors:

>SavIdent<-row.names(Matable) # pour sauvegarder.
>row.names(Matable)<-NULL # pour effacer.
>row.names(Matable)<-SavIdent
>View(Matable)

3.6 - Appeler une variable d'une table de données

Pour appeler une variable dans une ligne de commande de R vous utiliserez les opérateurs d'extraction, soit **$**, soit **[ ]**.

Exemple

>Matable$Sexe1
cette commande vous permet d'afficher dans la console le contenu de la variable Sexe1 de l'objet R de type dataframe nommée Matable.

>Matable[3]
cette commande vous permet d'afficher dans la console le contenu de la 3° variable de l'objet R de type dataframe nommée Matable.

3.7 - Modifier le nom d'une variable

Vous pouvez également modifier le nom d'une variable avec la fonction **names()** vue au chapitre 3.2.

Exemple
>names(Matable)[Matable$Sexe]<-"CODSEX"
>names(Matable)[3]<-"POIDS"

#pour modifier plusieurs noms en même temps.
>names(Matable)[c(2,4)]<-c("SEXE","GROUPAGE")

Pour effacer le nom des colonnes (variables), on attribuera alors le paramètre NULL en utilisant la fonction names(). Comme pour les identifiants, avant d'effacer les noms de variables avec le paramètre NULL, vous devrez les sauvegarder dans un objet vecteur de R.

Vous écrirez alors:

>Nomvar<-names(Matable) #pour sauvegarder.
>names(Matable)<-NULL #pour effacer.
>View(Matable) #pour voir les modifications de l'objet.
>names(Matable)<-Nomvar #pour réattribuer les noms.
>View(Matable) #pour voir les modifications de l'objet.

3.8 - Afficher le contenu d'une ligne ou d'une colonne

Si vous souhaitez afficher les données collectées pour un enregistrement, c'est à dire le contenu d'une ligne, ou les données d'une variable c'est à dire d'une colonne, vous utiliserez l'opérateur d'extraction **[ ]**.

Exemple

>Matable[3,] # affiche les données de la 3ième ligne.
>Matable[,3] # affiche les données de la 3ième colonne

NB: Attention à la place de la virgule dans la commande d'extraction. Les informations avant la virgule correspondent aux lignes, les informations après la virgule aux colonnes.
Remarquez que Matable[,3] vous donne le même résultat que Matable[3] (cf Chapitre 3.6).

Ainsi, si vous souhaitez connaître la valeur qui correspond au croisement de la 3ième ligne et de la 4ième colonne, vous écrirez:

>Matable[3,4]

Si vous ne souhaitez afficher qu'une partie de la table de données, vous identifierez les lignes et les colonnes à sélectionner.

>Matable[20:25,1:3]# extrait les lignes de 20 à 25 et les 3 premières colonnes.
>Matable[c(20,25),c(1,3)] # extrait les lignes 2à et 25 et les colonnes 1 et 3.

3.9 - Extraire une partie des données

Vous pouvez vouloir faire vos analyses sur une partie des données uniquement et dans ce cas choisir d'attribuer le résultat de l'extraction précédente à un objet R.

Exemple
>Matable1<-Matable[1:20,2:4]
#cette commande extrait dans un objet R les enregistrements (lignes) 1 à 20 pour les variables (colonnes) 2 à 4.

>Matable2<-Matable[1:20,-1]
#autre façon de procéder avec toutes les variables sauf la première.

>Matable3<-Matable[1:20,c(1,3,4)]
Cette commande extrait dans un objet R les enregistrements 1 à 20 pour les variables 1,3 et 4.

>Matable4<-Matable[-20,c(-1,-3)]
Cette commande extrait dans un objet R tous les enregistrements sauf le $20^{ème}$, et toutes les variables sauf la $1^{ère}$ et la $3^{ème}$.

NB: Vous pouvez vérifier les résultats de vos extractions avec la fonction View(), ou comparer le nombre de lignes et de colonnes de chacun des nouveaux objets R avec la fonction dim().

Pour réaliser une extraction en fonction d'un critère, il existe une fonction **subset().**

Exemple
>Matable5<-subset(Matable,CODSEX==1)
cette commande extrait les données pour les sujets de sexe masculin codé 1. Attention au signe d'égalité utilisé ici c'est le double égal ==. Pour rappel le signe = peut être substitué à la commande d'attribution<-.

>Matable6<-subset(Matable,SEXE=="Femme")
cette commande extrait les données pour les femmes.

>Matable7<-subset(Matable,POIDS>=20)
cette commande extrait les données pour les personnes ayant un poids supérieur ou égal à 20kg.

On peut également faire une extraction selon un critère en utilisant l'opérateur d'extraction **[ ]**.

Exemple

>Matable8<-Matable[Matable$POIDS<20,]
#attention de ne pas omettre la virgule qui implique que le filtre de sélection s'applique aux lignes. Ici, vous allez extraire toutes les lignes pour lesquelles la variable POIDS est inférieure à 20.

Pour réaliser la même sélection, vous pourriez écrire également la commande suivante:
>Matable9<-Matable[Matable[3]<20,]

3.10 - Ajouter une variable calculée

Vous aurez besoin au cours de votre analyse d'ajouter des variables pour effectuer par exemple des regroupements en classe d'une variable quantitative.

Il est recommandé de ne pas écraser les variables de l'objet d'origine mais plutôt de créer une nouvelle variable à ajouter dans la table des données afin de parer à une mauvaise commande.

Exemple
>Matable$POIDS20<-ifelse(Matable$POIDS>=20,1,0)
#cette commande crée une variable binaire codée 0-1 qui prend la valeur 1 si le poids est supérieur ou égale à 20 et 0 dans les autres cas.

La fonction **transform()** vous permettra également d'ajouter des variables calculées à partir d'autres variables du tableau.

Exemple
>Matable<-transform(Matable, POIDSENG=POIDS*1000)
#cette commande ajoute une variable POIDSENG qui calcule le poids en gramme en fonction du poids en kg.

>Matable<-transform(Matable,POIDS15=ifelse(POIDS< 15,1,0))
#cette commande ajoute une variable POIDS15 qui prend la valeur 1 si le poids est inférieur à 15 et 0 dans les autres cas.

NB: Dans la fonction transform(), on utilise le signe = pour attribuer une valeur à une variable. Dans cette fonction, il ne peut pas être substitué par le signe <-.

La fonction **cut()** vous permettra d'effectuer des regroupement par classe de variables quantitatives. Attention cependant la constitution des classes sera sous le format]a,b] c'est à dire avec exclusion de la valeur de la borne gauche. Il faudra s'en souvenir pour ne pas éliminer les premières valeurs de vos données.

Exemple
>Matable$GROUPOIDS<-
cut(Matable$POIDS,breaks=c(9,15,20,25))
#La variable POIDS se disperse entre les valeurs 10 et 25. Pour créer votre première classe, vous devez donc choisir la valeur 9. Vous aurez 3 classes (9,15], (15,20], (20,25].

3.11 - Factoriser les variables qualitatives ou discrètes

Pour que certaines variables qualitatives ou discrètes codées en format numérique soient reconnues comme telles par le logiciel R, il est nécessaire de les factoriser avec la fonction **factor()**.
La fonction str() vous permettra d'identifier les variables à factoriser. Les variables qualitatives de type chaîne de caractères sont déjà reconnues par R comme des facteurs.

Exemple
>str(Matable)

```
data.frame':   31 obs. of  8 variables:
 $ CODSEX   : num  1 0 1 1 0 0 0 1 1 0 ...
 $ SEXE     : Factor w/ 2 levels "Femme","Hom.: 2 1
 $ POIDS    : num  10 10.5 11 11.5 12 12.5 13 13
 $ GROUPAGE : Factor w/ 3 levels " >10 ": 2 2
 $ POIDS20  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ POIDSENG : num  10000 10500 11000 11500 12000
 $ POIDS15  : num  1 1 1 1 1 1 1 1 1 1 ...
 $ GROUPOIDS: Factor w/ 3 levels "(9,15]",..: 1 1 1
```

#Les variables CODSEX, POIDS20 et POIDS15 nécessitent d'être factorisées.

Vous pouvez choisir de les factoriser une à une avec la fonction factor() appliquée à chacune d'elle, ou de le faire grâce à une boucle.

1° étape: définir un objet R contenant les numéros des variables.

>Indic<-c(1,5,7)

2° étape: créer la boucle avec l'opérateur for()

>for(i in 1:3) {Matable[,Indic[i]]<-factor(Matable[,Indic[i]])}

#Indic [i] sera remplacé par sa valeur à chaque boucle. Quand i=1 alors Indic[1]=1, quand i=2 alors indic[2]=5 …
L'opérateur d'extraction [ ] a été utilisé comme précédemment (cf 3.7) et la position de la virgule indique que l'on applique la fonction à la colonne.

3° étape vérifier les changements

>str(Matable)

```
data.frame':   31 obs. of  8 variables:
 $ CODSEX   : Factor w/ 2 levels "0","1": 2 1 2 2 1
 $ SEXE     : Factor w/ 2 levels "Femme","Homme": 2
 $ POIDS    : num  10 10.5 11 11.5 12 12.5 13 13.5
 $ GROUPAGE : Factor w/ 3 levels " >10 ",..: 2 2 2
 $ POIDS20  : Factor w/ 2 levels "0","1": 1 1 1 1 1
 $ POIDSENG : num  10000 10500 11000 11500 12000 …
 $ POIDS15  : Factor w/ 2 levels "0","1": 2 2 2 ….
 $ GROUPOIDS: Factor w/ 3 levels "(9,15],….:1 1….
```

3.12 - Rechercher les valeurs aberrantes

On utilisera ici la fonction **which()** pour identifier les coordonnées des points aberrants.

Exemple 1
Vous allez dans un premier temps créer une valeur aberrante dans votre table de donnée. Il peut par exemple s'agir d'un erreur de saisie avec un poids en kg exprimé finalement en gramme.

>Matable[20,3]<-Matable[20,6]
#par cette commande vous avez affecté la valeur du poids en gramme du 20ème enregistrement à la variable POIDS normalement exprimée en kg.

>Result<-boxplot(Matable$POIDS)
affecte le résultat de la fonction boxplot dans un objet R.

>attributes(Result)
```
$names
[1] "stats" "n"   "conf"   "out"    "group" "names"
```
cette commande permet de visualiser les différents composants de Result. On s'intéressera à "out".

>Aberrant<-Result$out
cette commande affecte les valeurs de Result$out à un objet R appelé Aberrant

>which(Matable[,"POIDS"]%in%Aberrant)
#cette commande permet de donner l'enregistrement pour lequel la variable POIDS présente une valeur aberrante.

Exemple 2
Vous allez donner une valeur 3 à la variable CODSEX de votre 2ème enregistrement

>Matable[2,1]<-3
>which(Matable==3,arr.ind=T)
```
     row col
VR 2   2   1
```

cette commande va chercher sur l'ensemble de la table de données les valeurs correspondant au chiffre 3 et affichera les coordonnées de ces valeurs

NB: d'une façon plus générale, la fonction which() permet de rechercher les coordonnées correspondant à des valeurs dans une table de donnée.

Exemple

```
>which(Matable>2400,arr.ind=T)
      row col
VR 30  30   6
VR 31  31   6
```

cette commande affiche les coordonnées des valeurs supérieures à 24000. Elles correspondent à la ligne 30 et 31 de la 6ème colonne.

3.13 - Rechercher les valeurs manquantes

Vous utiliserez là encore la fonction which(). La paramètre arr.ind (pour array indic) prend la valeur TRUE, ce qui permet d'afficher dans la console les coordonnées (ligne et colonne) des différentes valeurs manquantes sur l'ensemble de la table de données.

```
>which(is.na(Matable),arr.ind=TRUE)
```

3.14 - Corriger les données comme dans un tableur

Une fois les erreurs identifiées vous serez susceptible de vouloir les corriger, soit vous pourrez revenir dans les documents papiers de l'étude et vous pourrez corriger l'erreur soit il vous faudra accepter de la transformer en une valeur manquante.

La fonction **edit()** permet d'ouvrir les données dans une nouvelle fenêtre et de modifier les données mais si vous n'avez pas affecté la fonction edit() à une nouvel objet R les corrections ne seront pas sauvegardées dans votre table de données. Il est préférable d'utiliser la fonction **fix()** qui permet de faire des corrections qui seront prises en compte sans souci dans l'objet R.

Exemple

>fix(Matable)
#ceci ouvre une nouvelle fenêtre et vous pourrez modifier la valeur de la variable POIDS du 20ème enregistrement modifié au chapitre 3.12.

>Matable<-edit(Matable)
#cette commande permet d'utiliser la fonction edit() et d'enregistrer les modifications dans l'objet Matable.

3.15 - Lier deux tables de données entre elles

Il y a deux façons de vouloir lier des tables soit en y ajoutant des lignes avec la fonction **rbind()** soit en ajoutant des colonnes avec les fonctions **cbind()** et **merge()**.

Exemple 1

>Matable10<-rbind(Matable5,Matable6)
cette commande permet de réunir les tables créées précédemment (cf 3.9) avec les hommes (Matable5) et les femmes (Matable6).

NB: Il faut s'assurer avant d'utiliser la fonction rbind() que les tables de données comprennent le même nombre de

colonnes et que les noms des colonnes sont identiques. La deuxième table de données sera ajoutée sous la précédente.

Exemple 2

Vous allez créer un objet R, appelé Fievre contenant 31 enregistrements qui prendront la valeur 1 en cas de fièvre et 0 dans le cas contraire.

```
>Fievre<-rbinom(31,size=1,prob=0.25)
```

Vous insérer ensuite cet objet comme une nouvelle colonne de la table de données, appelée Matable.

```
>Matable<-cbind(Matable,Fievre)
```

NB: cette commande nécessite que le classement des enregistrements dans les deux objets soit identique.

Vous changerez également le nom de cette nouvelle variable pour respecter la casse en majuscule de vos noms de variable dans la table de donnée Matable.

```
>names(Matable)[10]<-"FIEVRE"
```

Si vous souhaitez lier les tables en relation avec l'identifiant des enregistrements, vous utiliserez alors la fonction merge().

Exemple 3

On va créer un objet R contenant 31 enregistrements qui prendront la valeur 1 en cas de toux et 0 dans le cas contraire.

>set.seed(6)
>Toux<-rbinom(31,size=1,prob=0.28)
>Matable11<-data.frame(Toux)
>row.names(Matable11)<-Ident[31:1]
cette commande affecte les valeurs de Ident aux noms des lignes de la table de données Matable11 de manière inversée, c'est à dire en commençant par la 31ième et en finissant par le 1iére.

>Matable<-merge(Matable,Matable11,by="row.names")
#cette commande crée une table de données Matable en associant les colonnes des tables Matable et Matable11 selon un identifiant commun.

NB: Vous utiliserez le paramètre all.x=T si vous souhaitez utiliser l'ensemble des enregistrements de la 1ère table et seulement ceux de la 2nde table qui sont joints (all.y=T pour garder tous les enregistrements de la 2nde table). Si vous ne précisez pas la fonction merge() ne conservera que les enregistrements communs aux deux tables. Recherchez dans l'aide de la fonction merge() pour avoir plus d'information.

Vous changerez également le nom de cette nouvelle variable pour respecter la casse en majuscule de vos noms de variable.

>names(Matable)[11]<-"TOUX"

La fonction View() vous permet de contrôler les modifications obtenues dans la table de données (Matable).

N'oubliez pas de sauvegardez votre table de données quand vous interrompez votre initiation à R (cf chapitre 2.6). Vous pourrez ainsi repartir de la dernière table créée.

Vous prendrez soin aussi de sauvegarder votre ficher de commande si vous avez utilisé l'éditeur de R comme recommandé (cf chapitre 1.7).

Exercice d'application

Ex7 – Table de données
Soit la matrice 6*4 à coefficients réels suivante:

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

Créer l'objet Mat.1 correspond à cette matrice.
Créer l'objet Matable de type table de données à partir de l'objet Mat.1.
Renommer les colonnes avec les valeurs suivantes : SEXE, AGEJOUR, INFECTION, FRATRIE.
Renommer les lignes avec des valeurs allant de EPI1 à EPI6.
Extraire la ligne correspondant à l'individu EPI4,
Afficher le contenu de la colonne nommée FRATRIE
Ajouter une colonne DECES ayant les valeurs: 0,0,1,1,0,0
Ajouter une colonne SEXE2 qui prendra la valeur M si SEXE=1, la Valeur F si SEXE =2 et la valeur NP si SEXE=0
Ajouter une colonne POIDS qui est égale à 2950 + (AGE*126)
Ajouter une colonne POIDSKG qui transforme la variable POIDS qui est exprimée en gramme.
Afficher le contenu de Matable dans une fenêtre différente de la console.

Chapitre 4

Statistiques descriptives avec R

Objectifs à atteindre à cette étape

Connaitre les différents types de variables

Connaitre les paramètres de dispersion des variables qualitatives et des variables quantitatives

Décrire les variables qualitatives et les variables quantitatives

Réaliser un tableau de fréquence

4.1 - Généralités sur les types de variables

La collecte d'information vous amène à utiliser différents types de données classées en deux grands types.

Quantitatives	Qualitatives
• Continues	• Ordinales
• Discrètes	• Nominales
• Temporelles	• Binaires

- Les variables quantitatives continues

Les valeurs de la variable varient de façon continue et sont exprimées avec une unité de mesure.

Exemple

Poids exprimé en kilogrammes ou en grammes
Taille exprimée en mètre ou centimètres

NB: La transformation d'une variable quantitative continue en classe s'appelle la discrétisation ou groupement par classe (ex: classes d'âge).

- Les variables quantitatives discrètes

Les variables ne prennent que quelques valeurs entières dénombrables, un nombre fini de valeurs.

Exemple

Parité, nombre d'enfants
Activité de soin, nombre de consultations

- Les variables temporelles

Ce sont des variables quantitatives qui sont exprimées en unité de mesure du temps.

Exemple

Age de grossesse, exprimé en semaine
Date de naissance, jj/mm/aaaa
Age, exprimé en année ou en mois
Durée de la maladie, exprimée en jours

- Les variables qualitatives ordinales

Ce sont des variables qui s'expriment en classes qui peuvent être ordonnées selon une échelle de valeurs.

Exemple

Niveau d'étude: primaire, secondaire, universitaire
Intensité d'une douleur: faible, moyenne, intense

NB: Le codage numérique pour le traitement informatique ne doit pas les faire passer pour des variables qualitatives discrètes et n'autorise donc pas de les manipuler de façon arithmétique, on utilisera la fonction factor() si nécessaire (cf 3.11).

- Les variables qualitatives nominales

Pour ces variables, il n'est pas nécessaire de hiérarchiser les classes, c'est à dire que leur ordre n'a pas d'importance.

Exemple

Groupe sanguin: A, B, AB, O
Statut marital: veuf marié, divorcé, célibataire
Type de toiture: paille, tôles, tuiles...

- Les variables binaires ou dichotomiques

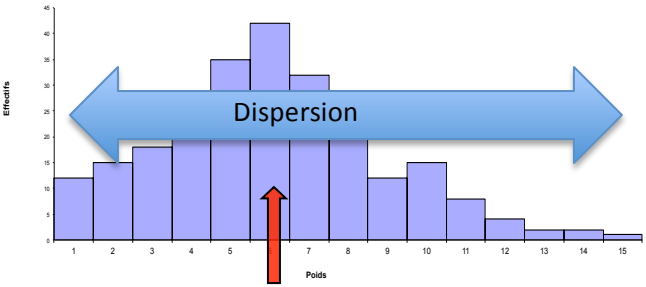
C'est un type particulier de variables nominales qui ne peuvent prendre que deux valeurs.

Exemple

Sexe : masculin, féminin
Présence ou absence d'un signe clinique, d'une maladie

4.2 - Généralités sur les statistiques descriptives

Pour décrire les données vous utiliserez des indicateurs de position et des indicateurs de dispersion.



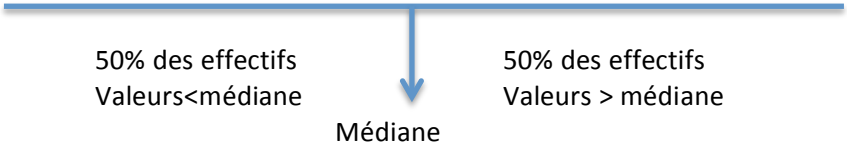
4.3 - Description des variables quantitatives continues

Paramètres de position	Fonctions R
Médiane	median(),summary()
Moyenne	mean(),summary()
Quartiles	quantile(),summary()

Paramètres de dispersion	Fonctions R
Minimum & étendue	min(),quantile(),summary()
Maximum & étendue	max(),quantile(),summary()
Variance	var()
Ecart-type	sd()
Intervalle de confiance	t.test()

- La médiane

C'est la valeur qui partage une série de données d'une variable quantitative en deux groupes d'effectifs égaux.

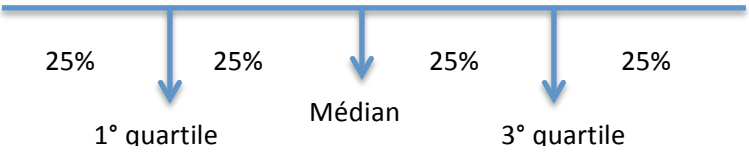


La médiane est un paramètre descriptif qui est présenté habituellement avec les extrêmes (Minimum et maximum) et l'étendue (différence entre maximum et minimum) qui est un paramètre de dispersion.
La médiane permet de résumer une distribution mais elle n'est pas utilisée pour les tests statistiques paramétriques.

Exemple
>median(Matable$POIDS,na.rm=T)

- Les quartiles

Il s'agit de 3 paramètres de position qui partagent une série de données en 4 groupes d'effectifs égaux.



Le 2nd quartile n'est pas différent de la médiane.

Exemple
>quantile(Matable$POIDS,na.rm=T)
le paramètre na.rm=T permet d'effectuer le calcul sans tenir compte des valeurs manquantes.

- La moyenne

C'est la somme algébrique des valeurs observées divisée par le nombre de sujets.

$$\mu = \frac{\sum_{i=1}^{n} Xi}{n}$$

C'est un paramètre de tendance centrale qui sert à résumer une série de données d'une variable quantitative.

Exemple
>mean(Matable$POIDS,na.rm=T)

- Les extrêmes

Ce sont les deux valeurs extrêmes de la distribution, maximum et minimum.

Exemple
>min(Matable$POIDS,na.rm=T)
>max(Matable$POIDS,na.rm=T)

- L'étendue

C'est la différence entre les deux valeurs extrêmes. Attention en cas de valeurs aberrantes, l'étendue donne une fausse image de la dispersion des valeurs d'une variable.

Exemple
>Etendue<-max(Matable$POIDS)-min(Matable$POIDS)
>Etendue
#La différence entre la valeur maximale et la valeur minimale est attribuée à un objet R appelé 'Etendue', que l'on appelle ensuite avec la deuxième ligne de la commande R.

- L'intervalle interquartile

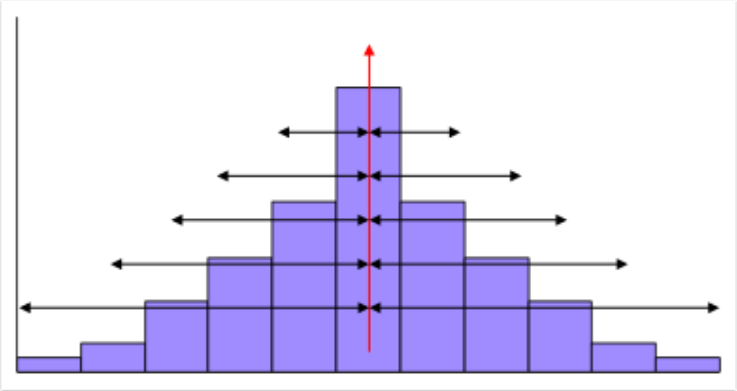
C'est la différence entre le 1^{er} quartile et le $3^{ième}$ quartile.

Exemple
>IntQuart<-quantile(Matable$POIDS)[4]-quantile(Matable $POIDS)[2]
>IntQuart

NB: Les extrêmes, l'étendue et l'intervalle interquartile sont des paramètres de dispersion associés à la médiane.

- La variance

C'est le paramètre de dispersion le plus utilisé. Il résume l'ensemble des écarts de chaque valeur d'une distribution par rapport à la moyenne



On la définit comme la moyenne des carrés des écarts à la moyenne.

$$\sigma^2 = \frac{\sum(xi - \mu)^2}{N}$$

La variance utilise toutes les valeurs de la distribution. C'est le meilleur indicateur de la dispersion autour de la moyenne.

Exemple

>var(Matable$POIDS, na.rm=T)

NB: Quand sa valeur est faible, alors la dispersion est minimale; plus sa valeur est élevée, plus la dispersion est importante.

- L'écart-type

C'est la racine-carré de la variance. Il est donc lié à la variance et traduit le fait d'avoir utilisé le carré des écarts pour la variance.

$$\sigma = \sqrt{\sigma^2}$$

Il mesure également l'écart par rapport à la moyenne, varie dans le même sens que la variance et s'exprime dans la même unité que la moyenne.

Exemple

>sd(Matable$POIDS, na.rm=T)

- L'intervalle de confiance de la moyenne

Cette mesure permet de savoir dans quel intervalle se trouve la moyenne d'une mesure d'une population à partir des données d'un échantillon avec risque d'erreur de 5%.

Exemple

>t.test(Matable$POIDS,conf.level=0.95)$conf.int

NB: Il y a donc 5 chances sur 100 que la moyenne de la population se trouve à l'extérieur de cet intervalle.

- Paramètres de dispersion d'une variable quantitative continue

La fonction **summary()** permet d'obtenir les extrêmes, les quartiles, la médiane et la moyenne d'une variable quantitative continue. Cette fonction ne tient pas compte des valeurs manquantes, il n'est donc pas nécessaire d'utiliser le paramètre na.rm=T avec cette fonction.

Exemple

>summary(Matable$POIDS)

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
10.00   13.75   17.50   17.50   21.25   25.00
```

4.4 - Description des variables qualitatives et quantitatives discrètes

Pour les données regroupées en classe, on utilise habituellement les tableaux de fréquence pour présenter les résultats. La construction de ces tableaux nécessite de connaître les effectifs dans chacune des classes de regroupement et également les pourcentages de ces effectifs rapportés à l'effectif global observé.

Paramètres	Fonction R
Effectifs	table()
Fréquence relative	prop.table()
Effectifs totaux	margin.table()
Somme cumulée	cumsum()

- Les effectifs

La description des données regroupées passe dans un premier temps par la description des effectifs par classe. Pour connaître ces effectifs, vous utiliserez la fonction **table()**.

Exemple

>table(Matable$GROUPAGE)

vous obtenez alors le résultat suivant

```
>10    0-4    5-9
 14     8      9
```

NB: Vous pourrez également utiliser la fonction summary() si vos variables sont reconnues par R comme des variables factorisées.

Exemple

>summary(Matable$SEXE)

#vous obtenez alors le résultat suivant

```
Femme Homme
16    15
```

NB: Si votre variable n'est pas factorisée, vous utiliserez la fonction table() mais vous pouvez également utiliser la fonction as.factor() dans les paramètres de la fonction summary().

Exemple

>summary(as.factor(Matable$SEXE))

NB: La fonction **as.factor()** permet à R de considérer une variable comme une variable factorisée uniquement au moment de l'utilisation de cette commande. La variable n'est pas alors factorisée de façon définitive (cf 3.11).

- La fréquence relative

Elle représente la proportion de la population observée dans chacune des classes de la variable. Vous utiliserez la fonction **prop.table()**. Cette fonction s'applique à un objet auquel vous affecterez le résultat de la fonction table().

Exemple

```
>Table1<-table(Matable$GROUPAGE)
>prop.table(Table1)
```

#vous obtenez alors le résultat suivant:

```
        >10       0-4       5-9
        0.4516129 0.2580645 0.2903226
```

NB: Généralement, la fréquence relative s'exprime sous la forme d'un pourcentage avec un chiffre après la virgule. Vous utiliserez alors la fonction **round()** pour obtenir des valeurs arrondies.

Exemple

```
>round(prop.table(Table1)*100,1))
```

#pour obtenir un pourcentage, vous multipliez par 100 les résultats de la fonction prop.table() puis vous arrondissez à un chiffre après la virgule avec la fonction round() et l'argument 1 associé.

#vous obtenez alors le résultat suivant:

```
        >10   0-4   5-9
        45.2  25.8  29.0
```

- Les effectifs totaux

Pour déterminer les effectifs totaux, qui peuvent être différents du nombre d'enregistrements du fait des valeurs manquantes, vous utiliserez la fonction **margin.table()**.

Exemple

```
>margin.table(Table1)
```

- Les effectifs cumulés

Le calcul des effectifs cumulés se fait avec la fonction **cumsum()** appliquée au résultat de la fonction table().

Exemple

```
>Table1<- table(Matable$GROUPAGE)
>cumsum(Table1[c(1,3,2)])
```

NB: Le calcul des sommes cumulées doit avoir une certaine cohérence. Pour l'exemple des groupes d'âge, il est nécessaire de classer les résultats dans un sens décroissant car les personnes les plus âgées sont passées par les tranches d'âge les plus jeunes. Vous utiliserez alors la sélection comme proposée dans l'exemple avec l'opérateur d'extraction **[ ]** et les colonnes de l'objet Table1 classées pour répondre à cette contrainte.

- Les proportions cumulées

Le calcul des proportions cumulées se fait avec la fonction cumsum() appliquée au résultat de la fonction prop.table().

Exemple

```
>Proportion<-round(prop.table(Table1)*100,1))
>cumsum(Proportion[c(1,3,2)])
```

- Construire un tableau de fréquence

Avec l'ensemble des commandes à notre disposition, vous pouvez construire un tableau de fréquence.

Exemple

```
>Variable<-Matable$GROUPAGE
>Effectifs<-table(Variable)
>Proportion<-round(prop.table(Effectifs)*100,1)
>E.cumul<-cumsum(Effectifs[c(1,3,2)])
>P.cumul<-cumsum(Proportion[c(1,3,2)])

>Age1<-c(Effectifs[2],Proportion[2],E.cumul[3],P.cumul[3])
>Age2<-c(Effectifs[3],Proportion[3],E.cumul[2],P.cumul[2])
>Age3<-c(Effectifs[1],Proportion[1],E.cumul[1],P.cumul[1])

>TablAge<-rbind(Age1,Age2,Age3)
>colnames(TablAge)<- c("Eff","(%)","Eff Cum","(% cum)")
>rownames(TablAge)<-c("0-4 ans","5-9 ans",">=10 ans")
>TablAge
# vous obtiendrez alors ce résultat
          Eff (%) Eff Cum (% cum)
0-4 ans     8 25.8      31   100.0
5-9 ans     9 29.0      23    74.2
>=10 ans   14 45.2      14    45.2
```

- Intervalle de confiance d'une proportion

L'intervalle de confiance d'un pourcentage sera calculé pour une variable qualitative binaire. Vous utiliserez la fonction **prop.test()**.

Exemple

```
>prop.test(table(Matable$SEXE)$conf.int

[1] 0.3339764 0.6944174
```

NB: Il s'agit de l'intervalle de confiance de la première valeur de la variable. La fonction prop.table() permet de visualiser les pourcentages de chacune de deux valeurs. L'intervalle de confiance de la seconde se fera par soustraction.

Exercices d'application

Ex8 – Variables quantitatives

Créer un objet appelé Mesure comprenant 30 mesures répondant à une loi de distribution uniforme avec un miminum à 20 et un maximum à 35 (cf page 35).
Calculer la médiane, la moyenne, l'écart-type de cet objet
Calculer l'intervalle de confiance de la moyenne

Créer un objet appelé Mesure comprenant 100 mesures répondant à une loi de distribution uniforme avec un minimum à 20 et un maximum à 35.
Calculer la médiane, la moyenne, l'écart-type de cet objet
Calculer l'intervalle de confiance de la moyenne

Ex9 – Variables qualitatives

Créer un objet appelé Caracter1 qui suit une loi binomiale de taille 1 et de 0.15, comprenant 110 valeurs (cf page 35)
Créer un objet appelé Caracter 2 qui suit une loi binomiale de taille 2 et de 0.40, comprenant 110 valeurs
Calculer les effectifs et les pourcentages par classe de chacun des 2 objets.
Dresser les tableaux présentant les effectifs associés aux pourcentages pour chacun des deux objets.

Chapitre 5

Présentation graphique des données avec R

Objectifs à atteindre à cette étape

Connaitre les différents types de graphiques.

Connaitre les graphiques à utiliser en fonction des variables qualitatives et des variables quantitatives.

Ajouter des informations complémentaires sur un graphique.

Insérer plusieurs graphiques dans une même fenêtre graphique.

Sauvegarder les graphiques sous différents formats.

5.1 - Diagramme en barre

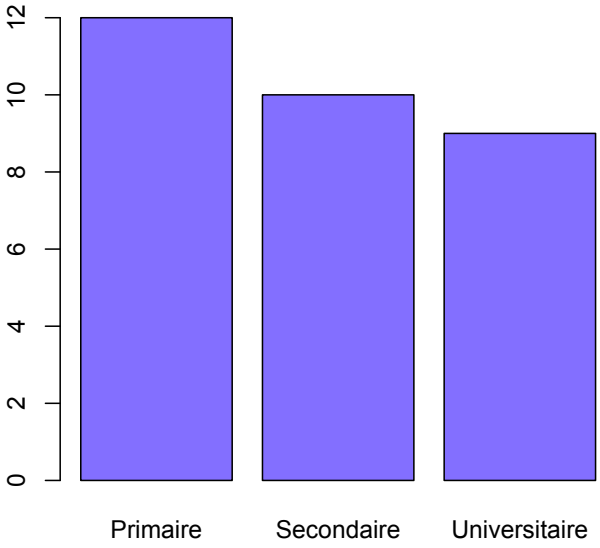
Ce type de graphique vous permettra de réaliser une présentation sur une seule coordonnée des variables qualitatives ordinales ou nominales. Vous utiliserez la fonction **plot().**

Exemple

```
>Matable$NIVSCOLMER<-factor(c(rep("Primaire",12),
rep("Secondaire",10),rep("Universitaire",9)))
```

#Cette commande permet de créer une variable niveau scolaire des mères de type facteur.

```
>plot(Matable$NIVSCOLMER,col='darkblue')
```



#Le paramètre 'col 'a permis de définir la couleur des barres.

Il est aussi possible d'ajouter des informations sur un graphique.

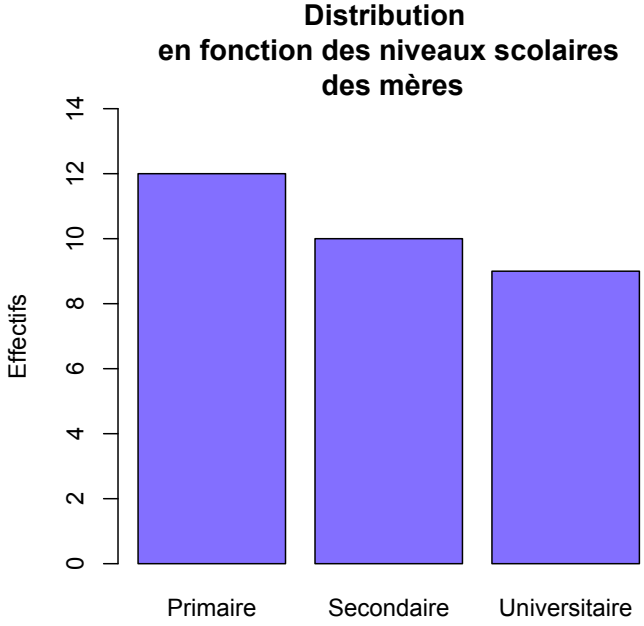
```
>plot(Matable$NIVSCOLMER,col='darkblue',ylab='effectifs')
```

Ce nouveau paramètre ylab va permettre d'ajouter un titre sur l'axe des ordonnées.

```
>plot(Matable$NIVSCOLMER,col='darkblue',ylab='effectifs',
ylim=c(0,14))
```

Le paramètre `'ylim'` permet de modifier les valeurs de l'axe des ordonnées.

```
>plot(Matable$NIVSCOLMER,col='darkblue',ylab='effectifs',
ylim=c(0,14), main='Distribution \n en fonction des niveaux
scolaires \n des mères')
```



Le paramètre `'main'` permet d'insérer un titre au dessus du graphique.

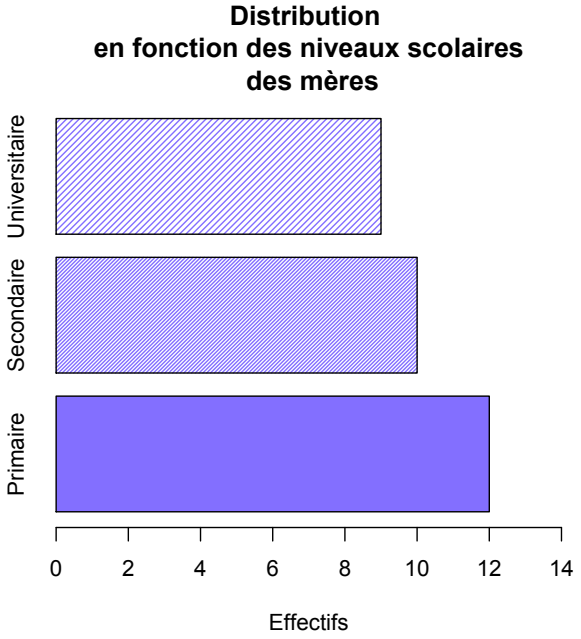
NB: l'expression `'\n'` exprimé dans la chaîne de caractère du titre permet d'identifier les niveaux de passage à la ligne.

5.2 - Diagramme en barre horizontale

Il répond à la présentation des variables de même type que le diagramme en barre.

Exemple

>plot(Matable$NIVSCOLMER,col='slateblue1', density=c(NA,50,30), xlab='Effectifs', xlim=c(0,14), main='distribution\n en fonction des niveaux scolaires \n des mères',horiz=T)



Les paramètres `horiz=T` et `density=c(NA,50,30)` ont permis pour le premier de présenter les barres à l'horizontale et le second de faire varier la densité de la couleur avec un format hachuré.

NB: Les paramètres `ylab` et `ylim` de la commande du diagramme en barre ont été remplacées dans cette nouvelle commande par `xlab` et `xlim`.

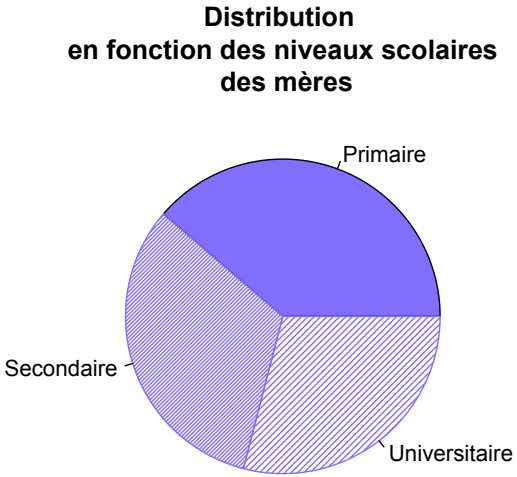
5.3 - Camembert

Il permet la présentation des variables de même type que les diagrammes en barre. Vous utiliserez la fonction**)**.

Exemple

>pie(table(Matable$NIVSCOLMER),col='slateblue1',density=c(NA,50,30),main='Distribution \n en fonction des niveaux scolaires \n des mères')

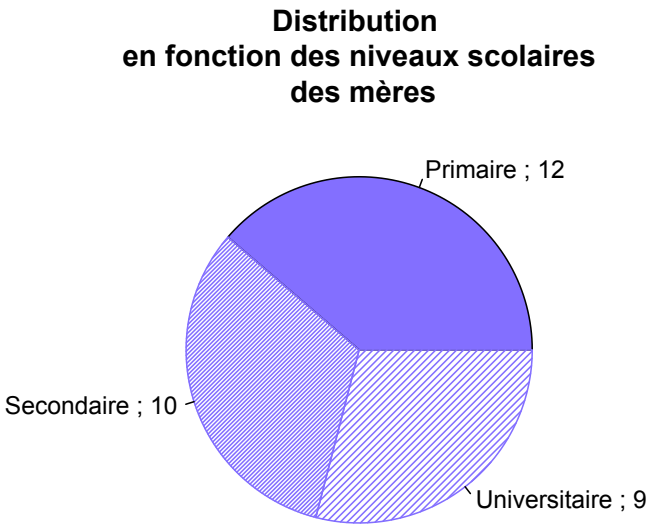
NB: La fonction pie() ne s'applique pas sur des données brutes.



Il est possible d'ajouter sur ce graphique les effectifs de chacune des classes avec le paramètre `labels`.

Exemple

```
>Nom<-levels(Matable$NIVSCOLMER)
>Donnee<- table(Matable$NIVSCOLMER)
>pie(Donnee,col='slateblue1',density=c(NA,50,30),main='Distr
ibution \n en fonction des niveaux scolaires \n des
mères',labels=c(paste(Nom[1],';',Donnee[1]),paste(Nom[2],";"
,Donnee[2]),paste(Nom[3],";",Donnee[3])))
```

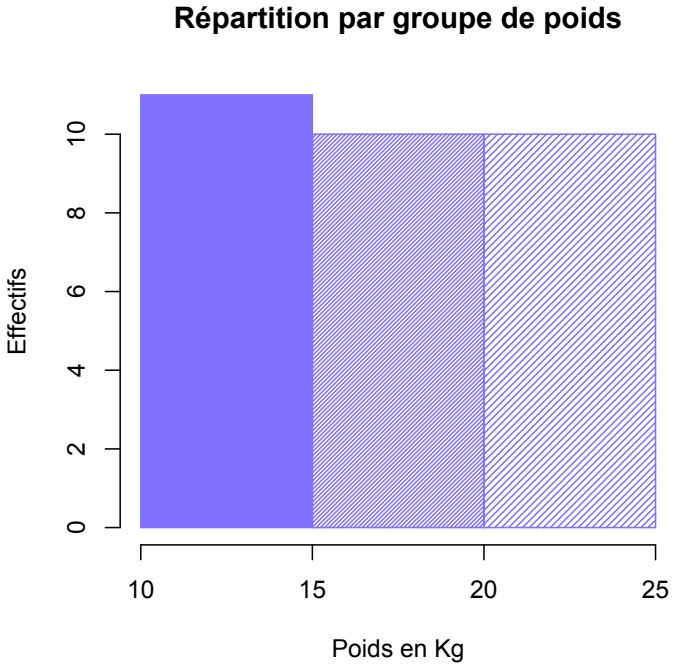


La fonction paste() permet de concaténer des chaînes de caractères alpha-numériques. Elle est utilisée ici avec le paramètre labels pour ajouter une information sur les effectifs séparés du nom de la classe par un point-virgule.

5.4 - Histogramme

Il est utilisé pour représenter des données quantitaitives discrétisées (en classes). Vous utiliserez la fonction **hist()** pour le réaliser.

Exemple
>Titre<-'Répartition par groupe de poids'
>hist(Matable$POIDS,breaks=c(10,15,20,25),col='slateblue1', density=c(NA,50,30),xlab='PoidsenKg',ylab='Effectifs',main=Titre)
le paramètre `breaks` permet de donner les limites de chaque classe



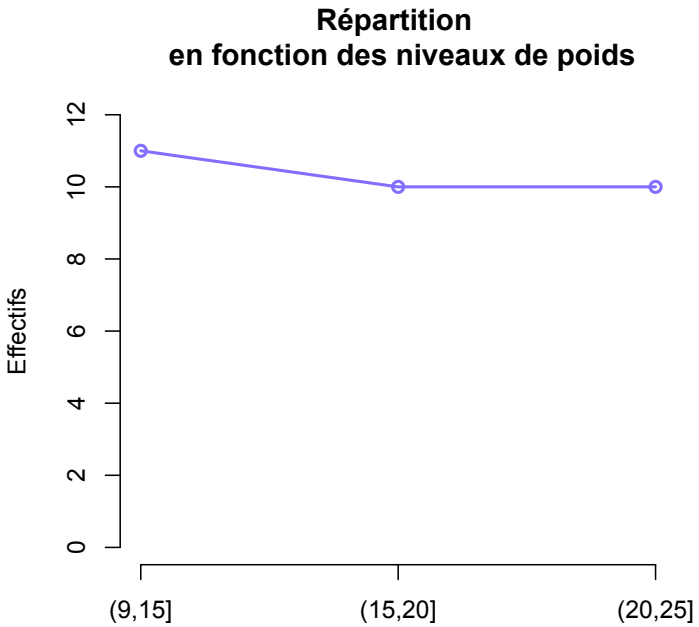
NB: Pour cette commande, vous utiliserez les données brutes de la variable POIDS.

5.5 - Polygone de fréquence

Il s'agit d'une courbe réunissant le sommet de chacune des classes d'une variable quantitative discrétisée. Vous utiliserez la fonction plot() avec des données agrégées (fonction table()) et l'argument `type` en fonction du type de courbe que vous souhaitez obtenir.

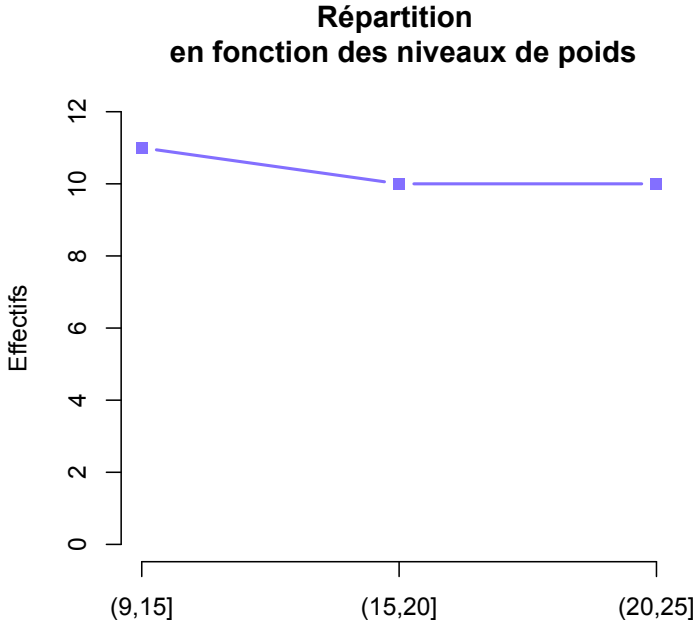
Exemple

```
>Donnee(table(Matable$GROUPOIDS))
>plot(Donnee,type='o',col='slateblue1',main='Répartition\n
en fonction des niveaux de poids', ylab='Effectifs',
ylim=c(0,12))
```




```
# Le paramètre type ='o' trace les points et la courbe les
réunissant.
```

>plot(Donnee,type='b',col='slateblue1',main='Répartition\n en fonction des niveaux de poids', ylab='Effectifs', ylim=c(0,12), pch=15)



Le paramètre `type='b'` permet d'avoir là aussi à la fois les points et la courbe les réunissant mais il est possible de choisir le type des points avec le paramètre `pch`.

5.6 - Différents types de points associés au paramètre pch

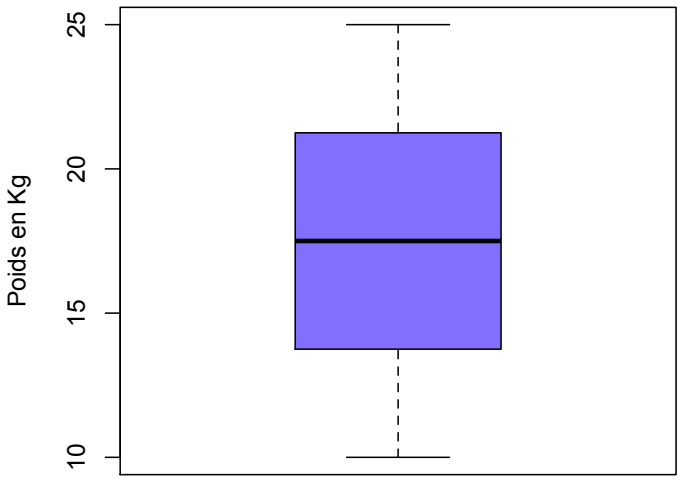
Il existe une grande variété de points appelés par le paramètre pch dans un graphique. Pour appeler ces points il faut utiliser les nombres correspondant à chacun d'entre eux.

5.7 - Boite à moustache

La boite à moustache permet de représenter sur un graphique les paramètres de position et de dispersion d'un variable quantitative continue : médiane, quartiles et étendue. Vous utiliserez la fonction **boxplot()** pour réaliser ce graphique.

Exemple

>boxplot(Matable$POIDS, col='slateblue1', xlab='Poids en Kg', main='Description de la variable POIDS')



NB1: La fonction boxplot.stats() vous permet de retrouver les valeurs associées à chacun des paramètres de ce graphique. Les moustaches inférieures et supérieures sont calculées par le retrait à la valeur du 1° quartile ou l'ajout à la valeur du 3° quartile de 1,5 fois l'intervalle interquartile.

NB2: Pour ajouter la valeur de la moyenne sur le graphique, vous utiliserez la fonction points().

```
>points(x=1,y=mean(Matable$POIDS),pch=4)
```

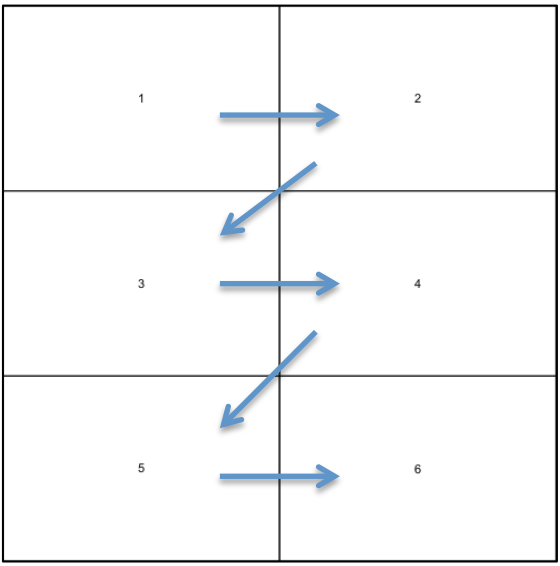
5.8 - Afficher plusieurs graphiques dans la même fenêtre

Pour afficher plusieurs graphiques dans une même fenêtre, vous pouvez utiliser la fonction **par()** avec le paramètre mfrow ou la fonction **layout()**.

Exemples

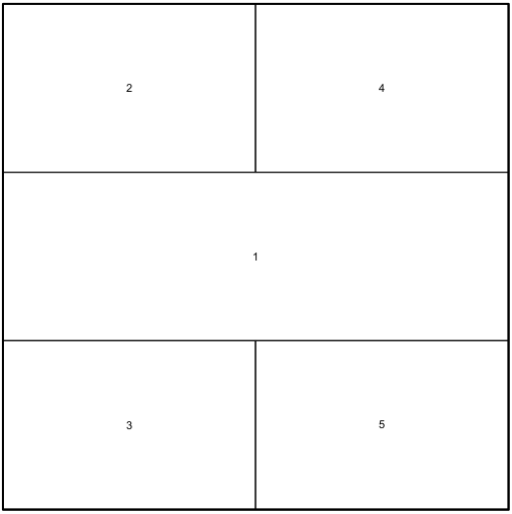
>par(mfrow=c(3,2))
cette commande vous permettra de diviser votre fenêtre graphique en 6 fenêtres: 3 lignes et 2 colonnes.

>layout.show(6)
cette commande affichera les six zones de la page graphique avec leur rang de remplissage de 1 à 6.



Le remplissage se fera donc ligne après ligne de gauche à droite.

```
>Zone<-matrix(c(2,4,1,1,3,5),byrow=T,ncol=2)
>layout(Zone)
>layout.show(max(Zone))
```



La fonction layout() permet de définir des zones graphiques de tailles différentes et également de choisir l'emplacement des graphiques en fonction de leur rang.

Pour revenir à la fenêtre graphique unique, vous appliquerez les commandes suivantes :

Si vous avez utilisé par(mfrow=…) pour créer une fenêtre graphique avec plusieurs éléments , alors vous écrirez:

```
>par(mfrow=c(1,1))
```

Si vous avez utilisé layout(), alors vous écrirez:

```
>layout(matrix(1))
```

5.9 - Afficher plusieurs fenêtres graphiques et les gérer

Vous remarquerez que le tracé d'un nouveau graphique dans une fenêtre unique efface le graphique précédent. Si vous souhaitez garder visibles les différents graphiques, avant le lancement des commandes permettant de les réaliser il faudra faire précéder cette commande de la fonction **windows()** (ou **quartz()** pour les utilisateurs sur Mac).
Pour gérer les différentes fenêtres graphiques vous ferez alors appel aux fonctions suivantes:

```
dev.cur()        # indique la fenêtre active
dev.list()       # liste les numéros de fenêtres créées et leur
                 nature
dev.set(x)       # fixe la fenêtre x comme active
dev.off()        # ferme la fenêtre active
dev.off(x)       # ferme la fenêtre numéro x
graphics.off()   # ferme toutes les fenêtres graphiques
```

5.10 - Sauvegarder un graphique sous différents formats

Vous pourrez sauvegarder les graphiques sous différents formats. Vous utiliserez alors les fonctions suivantes: pdf(), jpeg(),bmp(),png(),tiff()…
Pour connaitre toute la liste ?device

Exemples

```
>graphics.off()
# cette commande ferme toutes les fenêtres graphiques

>pdf(file="Figure.pdf")
# sauvegarde le graphique dans un fichier au format pdf
```

```
>Nom<-levels((Matable$NIVSCOLMER)
>Donnee<- table(Matable$NIVSCOLMER)
>pie(Donnee,col='slateblue1',density=c(NA,50,30),main='Distr
ibution \n en fonction des niveaux scolaires \n des
mères',labels=c(paste(Nom[1],';',Donnee1]),paste(Nom[2],";",
Donnee[2]),paste(Nom)[3],";",Donnee[3])))
# ces commandes permettent de réaliser le graphique

>dev.off()
# cette commande ferme la fenêtre virtuelle , ce qui permet
ensuite d'ouvrir le fichier Figure.pdf qui a été sauvegardé
dans le répertoire de travail. Si vous ne connaissez plus votre
répertoire de travail, utilisez la fonction getwd() (cf 1.8)
```

NB: si vous n'arrivez pas à ouvrir le fichier Figure.pdf, vérifiez que vous avez bien fermé la fenêtre graphique virtuelle en lançant la fonction dev.off().

Vous pouvez sauvegarder plusieurs graphiques dans ce fichier en insérant d'autres commandes de graphiques avant la fonction dev.off(). Vous aurez alors un fichier au format pdf contenant plusieurs pages avec un graphique sur chacune des pages.

Vous pouvez aussi faire le choix de sauvegarder au format pdf chaque graphique dans un fichier différent. Vous utiliserez alors les commandes suivantes.

Exemple

```
>graphics.off()
# cette commande ferme toutes les fenêtres graphiques
```

>pdf(file="Figure%03d.pdf", onefile=FALSE)
finir le nom du fichier par %03d pour incrémenter automatiquement le nom des fichiers

>Nom<-levels((Matable$NIVSCOLMER)
>Donnee<- table(Matable$NIVSCOLMER)
>pie(Donnee,col='slateblue1',density=c(NA,50,30),main='Distribution \n en fonction des niveaux scolaires \n des mères',labels=c(paste(Nom[1],';',Donnee1]),paste(Nom[2],";",Donnee[2]),paste(Nom)[3],";",Donnee[3])))
>hist(Matable$POIDS,breaks=c(10,15,20,25),col='slateblue1',density=c(NA,50,30),xlab='PoidsenKg',ylab='Effectifs',main='Répartition par groupe de poids')

>dev.off()

5.11 - Les paramètres d'une fonction graphique

Vous pouvez intervenir sur différents paramètres des fonctions réalisant des graphiques.

- Définir les limites inférieures et supérieures de l'échelle sur l'axe des abscisses.

xlim
Exemple
xlim=c(0,20) # pour des valeurs entre 0 et 20.

- Définir les limites inférieures et supérieures de l'échelle sur l'axe des ordonnées.

ylim
Exemple
ylim=c(0,80) # pour des valeurs entre 0 et 80.

- Définir un titre sur l'axe des abscisses.

xlab
Exemple
xlab="Semaines" # pour des données de l'axe des abscisses correspondant à des numéro de semaines).

- Définir un titre sur l'axe des ordonnées

ylab
Exemple
ylab="Nombre de cas" # pour des données de l'axe des ordonnées correspondant à des nombres de cas).

- Donner un titre au graphique

main
Exemple
main="Distribution des cas de grippe par semaine en 2013 au Sénégal" # pour des données regroupées du nombre de cas de grippe par semaine en 2013 au Sénégal

- Donner un sous titre au graphique

sub
Exemple
sub="Données de la surveillance sentinelle" # pour des données issue du réseau de surveillance sentinelle.

- Modifier la taille des valeurs affichées sur les axes. Vous noterez que la valeur par défaut est 1, donc pour diminuer de moitié vous choisirez 0.5)

cex.axis
Exemple
cex.axis=0.6 # pour des caractères réduits de 40%.

- Modifier la taille des caractères pour les légendes des axes (1 valeur par défaut)

cex.lab

Exemple

cex.lab=0.8 # pour des caractères réduits de 20%.

- Modifier la taille des caractères du titre (1 valeur par défaut)

cex .main

Exemple

cex.main=1.2 # pour des caractères de titre augmentés de 20%.

- Choisir la couleur des symboles et des lignes

col

Exemple

col="red" # pour un tracé de graphique de couleur rouge.

- Modifier le couleur des axes, titres et sous titres

col.main, col.axis, col.lab, col.sub

Exemple

col.main="blue" # pour un titre de couleur bleue.

- Effacer l'axe de abscisses ou des ordonnées

xaxt, yaxt

Exemple

xaxt='n' # pour effacer l'axe des abscisses.

- Tracer le cadre autour de la figure

bty # avec ("l", "o", "7 ",…)

NB: reportez-vous à l'aide pour avoir plus d'informations sur ces paramètres et les autres paramètres possibles afin d'améliorer vos graphiques.

5.12 - Ajouter des informations sur un graphique

Vous pouvez ajouter des informations complémentaires sur un graphique. Vous utiliserez alors des fonctions complémentaires dont voici une liste non exhaustive.

points()	#ajoute des points.
lines()	#ajoute des points reliés par une ligne.
abline()	#ajoute une ligne sans limite exemple: abline(h=3) trace une ligne horizontale d'ordonnée égale à 3.
segments()	#ajoute un segment de droite.
arrows()	#trace une flèche.
legend()	# ajoute une légende.
rug()	# ajoute des marques secondaires sur les axes
axis()	# ajoute une échelle sur un côté du graphique Cette fonction s'utilise par exemple lorsque vous avez utilisé le paramètre xaxt='n' ou yaxt='n' ou pour définir un deuxième axe des ordonnées à droite.
text()	#ajoute du texte.
mtext()	#ajoute du texte dans les marges.
title()	#ajoute des éléments de titre comme les paramètres déjà étudié au 5.10.

NB : Reportez vous à l'aide avec la fonction help() ou ? pour voir comment utiliser toutes ces fonctions.

5.13 - Modifier les paramètres de la fenêtre graphique

Vous pouvez modifier les paramètres de la fenêtre graphique avec la fonction **par()**. Vous l'avez déjà utilisée pour découper la fenêtre graphique afin d'y présenter plusieurs graphiques.

Exemple

```
>par(bg=" lightyellow")
# cette commande modifie la couleur de fond de la fenêtre graphique, avec pour l'exemple un fond jaune clair.

>par(adj=0)
# cette commande contrôle le positionnement du texte: 0 à gauche, 0.5 centré et 1 à droite.

>par(ann=F)
# cette command permet de tracer les figures sans les légendes des axes et leur titre, c'est à dire sans les annotations.

>par(new=T)
# cette commande permet de tracer un nouveau graphique sur un graphique existant en utilisant successivement deux fonctions graphiques de haut niveau.

>par(xpd=NA)
# avec cette commande NA tous les graphiques sont attachés à la fenêtre graphique. Si la valeur de xpd est TRUE, les graphiques sont attachés à la zone de figure, si la valeur est FALSE ils peuvent alors être hors de la zone de figure.
```

NB: les paramètres de ces exemples auraient pu être utilisés dans une seule et même commande. Ne pas hésiter à faire

appel à l'aide pour avoir l'exhaustivité des paramètres associés à la fonction par().

Exercices d'application

Ex 10

Définir une fenêtre graphique qui vous permettra d'insérer 3 graphiques sur deux colonnes 1 à gauche et 2 à droite et visualiser cette fenêtre graphique.
Insérer dans la première zone graphique, le graphique de type boite à moustache de l'objet Mesure de l'exercice Ex8. Vous intitulerez ce graphique "Boite à moustache de la variable Mesure" sur deux lignes.
Dans la deuxième zone graphique, insérer l'histogramme de cet objet Mesure selon des classes allant de 20 à 35 avec un pas de 2,5. Vous intitulerez ce graphique "Histogramme de la variable Mesure" pour qu'il s'affiche sur deux lignes. Vous nommerez l'axe des abscisses "Groupes" et l'axe des ordonnées "Effectifs".
Dans la troisième zone graphique, insérer le polygone de fréquence correspondant à la variable Mesure selon les groupes de l'histogramme. Vous intitulerez ce graphique "Polygone de fréquence de la variable Mesure" en l'affichant sur deux lignes. Vous nommerez l'axe des ordonnées "Effectifs" et réduirez la taille de la police des axes de moitié.

Ex 11

Faire, dans deux fenêtres distinctes, les graphiques correspondant aux objets Caracter1 et Caracter2 de l'exercice Ex9 qui sont des variables qualitatives. Vous représenterez la première variable sous forme d'un diagramme en barre où la première valeur de l'objet Caracter1 prendra la valeur "masculin" et la seconde la valeur "féminin". Vous donnerez

le titre "Sexe" à ce graphique et la couleur bleue pour la barre correspondant à la classe "masculin", la couleur rose pour la barre de la classe "féminin". Sur l'axe des x, vous veillerez à ce que le nom des barres "masculin" et "féminin" correspondant à chaque classe apparaisse bien.
Dans la seconde fenêtre graphique, vous représenterez les pourcentages des classes de la variable Caracter2 à l'aide d'un graphique de type camembert pour lequel vous nommerez les classes 0, 1, 2 par "primaire", "secondaire", "universitaire" en y insérant en plus les pourcentages correspondants. Vous donnerez comme titre à ce graphique "Niveaux scolaires".

Chapitre 6

Analyses univariées avec R

Objectifs à atteindre à cette étape

Décrire une variable quantitative en fonction d'une variable qualitative

Décrire une variable qualitative en fonction d'une variable qualitative

Réaliser des tableaux de contingence

Réaliser des graphiques par classe

6.1 - Analyse des variables quantitatives en fonction des classes d'une variable qualitative

Après avoir décrit la distribution générale d'une variable, l'étape suivante vous conduit à l'analyse univariée pour décrire des variables en fonction de sous-groupes. Vous utiliserez les fonctions by(), tapply() ou aggregate() pour réaliser ces analyses des variables quantitatives en fonction des classes d'une variable qualitative.

Exemples

```
>by(Matable$POIDS,Matable$SEXE,summary)
```

vous obtiendrez alors ce résultat

```
Matable$SEXE: Femme
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.00   13.25   18.00   17.13   20.75   25.00
------------------------------------------------
Matable$SEXE: Homme
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.50   14.38   17.25   17.84   22.62   24.50
```

```
>tapply(Matable$POIDS,list(Matable$SEXE),summary)
```

vous obtiendrez alors ce résultat

```
$Femme
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.00   13.25   18.00   17.13   20.75   25.00

$Homme
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.50   14.38   17.25   17.84   22.62   24.50
```

```
>aggregate(Matable$POIDS,list(Matable$SEXE),summary)
```

vous obtiendrez alors ce résultat

```
 Group.1 x.Min. x.1st Qu. x.Median x.Mean x.3rd Qu. x.Max.
1 Femme  10.00     13.25    18.00  17.13     20.75  25.00
2 Homme  10.50     14.38    17.25  17.84     22.62  24.50
```

Vous pouvez utiliser les mêmes commandes pour effectuer des analyses descriptives à partir de plusieurs sous groupes décomposés de plusieurs variables qualitatives.

Exemples

>by(Matable$POIDS,list(Matable$SEXE,Matable$ GROUPAGE),summary)

vous obtiendrez alors ce résultat

```
: Femme
:  >10
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  19.50   20.25   21.00   21.36   21.75   25.00
------------------------------------------------------------
: Homme
:  >10
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  18.50   20.75   23.00   22.14   23.75   24.50
------------------------------------------------------------
: Femme
:  0-4
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   10.0    11.0    11.5    11.8    13.0    13.5
------------------------------------------------------------
: Homme
:  0-4
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.50   11.25   12.00   11.67   12.25   12.50
------------------------------------------------------------
: Femme
:  5-9
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  15.00   15.25   15.50   16.17   16.75   18.00
------------------------------------------------------------
: Homme
:  5-9
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  14.00   14.88   16.25   15.92   16.88   17.50
```

>aggregate(Matable$POIDS,list(Matable$SEXE,Matable$ GROUPAGE),summary)

vous obtiendrez alors ce résultat

```
Group.1 Group.2 x.Min. 1st Qu. Median Mean 3rd Qu. Max.
1Femme    >10    19.50   20.25   21.00  21.36  21.75  25.00
2Homme    >10    18.50   20.75   23.00  22.14  23.75  24.50
3Femme    0-4    10.00   11.00   11.50  11.80  13.00  13.50
4Homme    0-4    10.50   11.25   12.00  11.67  12.25  12.50
5Femme    5-9    15.00   15.25   15.50  16.17  16.75  18.00
6Homme    5-9    14.00   14.88   16.25  15.92  16.88  17.50
```

NB: Avec la commande tapply vous ne pourrez pas utiliser le paramètre summary mais vous pourrez utiliser les autres fonctions telles que mean, median, var, sd, min, max.

Exemple

>tapply(Matable$POIDS,list(Matable$SEXE,Matable$ GROUPAGE),median,na.rm=T)

vous obtiendrez alors ce résultat

```
        >10   0-4  5-9
Femme    21  11.5 15.50
Homme    23  12.0 16.25
```

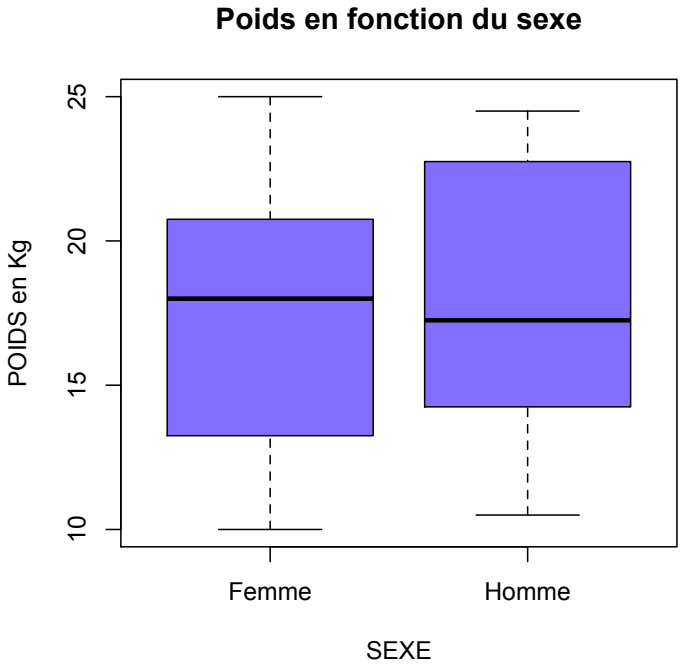
NB: N'oubliez pas d'ajouter na.rm=TRUE avec les fonctions autres que summary, si vous n'êtes pas sûr que votre fichier ne contient pas de données manquantes. Pour éviter toute surprise et erreur dans votre programmation, prenez l'habitude d'ajouter ce paramètre.

6.2 - Graphiques en classe de variables quantitatives

Vous pouvez présentez les données sous forme de graphiques en fonction des classes.

Exemple

>boxplot(Matable$POIDS~Matable$SEXE,col="slateblue1", xlab="SEXE",ylab="POIDS en Kg", main='Poids en fonction du sexe')



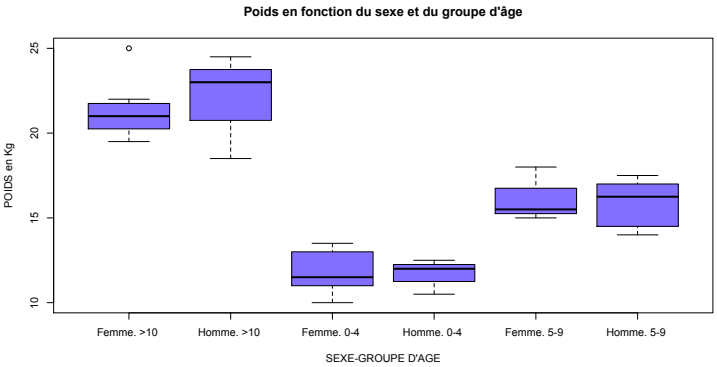
NB: vous pouvez ajouter la valeur de la moyenne sur le graphique avec la fonction points().

>points(1,mean(Matable$POIDS[Matable$SEXE=="Femme"]), pch=4)
>points(2,mean(Matable$POIDS[Matable$SEXE=="Homme"]), pch=4)

Mais aussi en fonction de plusieurs sous classes.

Exemple

>boxplot(Matable$POIDS~Matable$SEXE+Matable$GROUPAGE,col="slateblue1",xlab="SEXE",ylab="POIDS en Kg", main="Poids en fonction du sexe et du groupe d'âge")



6.3 - Analyse univariée d'une variable qualitative en fonction d'une autre variable qualitative

Comme pour les variables quantitatives, il est possible d'avoir une description d'une variable qualitative (effectifs et pourcentages) en fonction d'une autre variable qualitative.
Vous utiliserez les fonctions table () et prop.table() que vous aviez déjà utilisées pour décrire la distribution d'une variable qualitative (cf chapitre 4.4).

Exemple

>table(Matable$SEXE,Matable$NIVSCOLMER)

Il vous sera plus facile de passer par un objet R qui contiendra le résultat de cette commande pour ensuite l'appeler avec la fonction prop.table () et avoir ainsi les pourcentages.

Exemple

```
>Tab1<- table(Matable$SEXE,Matable$NIVSCOLMER)
```

pour les pourcentages en lignes

```
>prop.table(Tab1,1)
```

```
        Primaire   Secondaire   Universitaire
  Femme 0.2666667  0.4000000     0.3333333
  Homme 0.5000000  0.2500000     0.2500000
```

#la fonction round() vous permet de présenter vos résultats de façon plus lisible.

```
>round(prop.table(Tab1,1)*100,1)
```

```
       Primaire   Secondaire   Universitaire
  Femme     26.7         40.0           33.3
  Homme     50.0         25.0           25.0
```

pour les pourcentages en colonnes, remplacer le chiffre 1 par le chiffre 2 dans la fonction prop.table().

```
>round(prop.table(Tab1,2)*100,1)
```

```
      Primaire Secondaire Universitaire
  Femme     33.3         60.0           55.6
  Homme     66.7         40.0           44.4
```

Vous pouvez créer un tableau de contingence pour réunir l'ensemble de l'information concernant les effectifs et les pourcentages.

Exemple
affectation de valeurs à des objets R.

```
>Var.1<-Matable$SEXE
>Var.2<-Matable$NIVSCOLMER
```

```
>X1<-'Male'
>X2<-'Female'
>Z<-'%'
>Y1<-'Prim'
>Y2<-'Secon'
>Y3<-'Universit'

# utilisation de fonctions descriptives.

>a<-table(Var.1,Var.2)
>b<-round(prop.table(a,1)*100,digit=1)
>Total<-margin.table(a,2)

# construction de la table
>Conting<-rbind(a[1,],b[1,],a[2,],b[2,])
>dimnames(Conting)<-list(c(X1,Z,X2,Z),c(Y1,Y2,Y3))
>Conting<-rbind(Conting,Total)
>Total<-margin.table(Conting,1)
>Conting<-cbind(Conting,Total)
>Conting

# Vous obtiendrez alors ce résultat.

       Prim Secon Universit Total
Male    4.0     6        5.0    15
%      26.7    40       33.3   100
Female  8.0     4        4.0    16
%      50.0    25       25.0   100
Total  12.0    10        9.0    31
```

Pour avoir les pourcentages à côté des effectifs, vous pourrez utiliser le programme suivant à insérer après l'objet b du programme précédent :

```
>Total2<-margin.table(a,1)
>Conting2<-cbind(a[,1],b[,1],a[,2],b[,2],a[,3],b[,3])
>dimnames(Conting2)<-list(c(X1,X2),c(Y1,Z,Y2,Z,Y3,Z))
```

```
>Conting2<-cbind(Conting2,Total=Total2)
>Total2<-margin.table(Conting2,2)
>Total2[2]<-round((Total2[1]/Total2[7])*100,1)
>Total2[4]<-round((Total2[3]/Total2[7])*100,1)
>Total2[6]<-round((Total2[5]/Total2[7])*100,1)
>Conting2<-rbind(Conting2,Total=Total2)
>Conting2
```

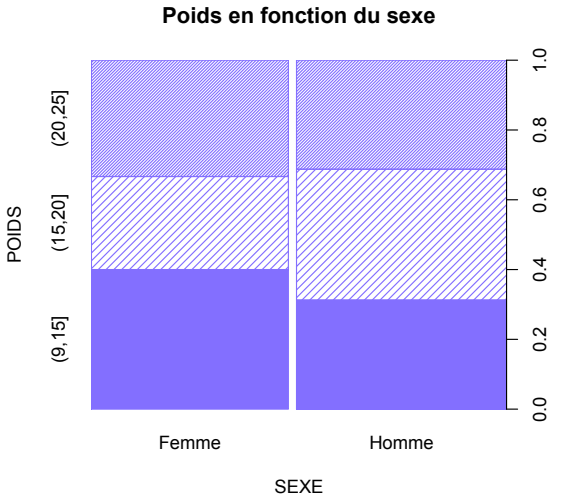
Vous obtiendrez alors ce résultat.

```
       Prim  %   Secon  %   Universit  % Total
Male      4 26.7     6 40.0         5 33.3    15
Female    8 50.0     4 25.0         4 25.0    16
Total    12 38.7    10 32.3         9 29.0    31
```

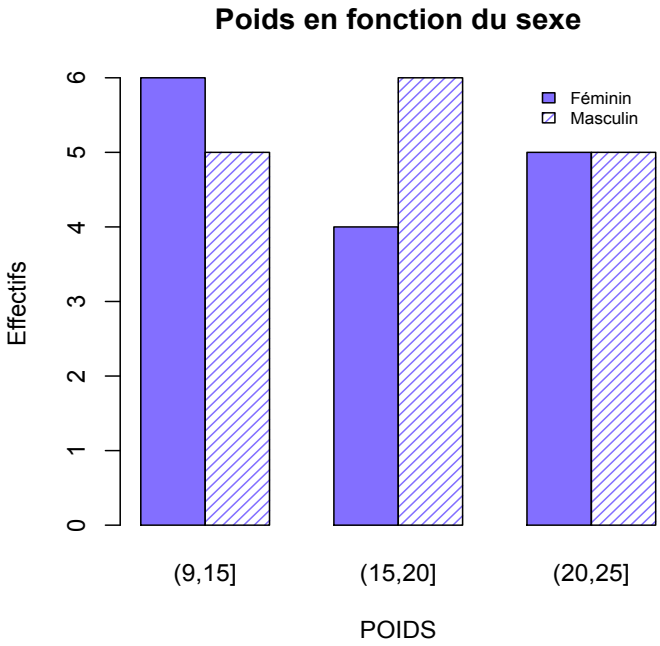
6.4 - Graphiques en classe de variables qualitatives

Vous utiliserez les fonctions plot() ou barplot() avec des arguments pour identifier la variable de factorisation.

>plot(Matable$GROUPOIDS~Matable$SEXE,col="slateblue1", density=c(NA,20,50),xlab="SEXE",ylab="POIDS", main ="Poids en fonction du sexe")



```
>barplot(table(Matable$SEXE,Matable$GROUPOIDS),
col="slateblue1",density=c(NA,20),xlab="POIDS",ylab="Effecti
fs", main='Poids en fonction du sexe',beside=T)
>legend(7,6,legend=c("Féminin","Masculin"),bty='n',fill="slat
eblue1",density=c(NA,20),cex=0.7)
```



NB: le paramètre density vous permet d'avoir des fonds rayés.

Exercices d'application

Ex 12

Créer un vecteur appelé Sexe qui suit une loi binomiale de taille 1 et de 0.55, comprenant 120 valeurs
Créer un vecteur appelé NivScol qui suit une loi binomiale de taille 2 et de 0.40, comprenant 120 valeurs
Dresser le tableau de contingence de croisement du niveau scolaire en fonction du sexe.

Ex 13

Créer un vecteur appelé Poids contenant 120 valeurs qui suit une loi normale de moyenne 2,8kg et d'écart type 0,08.
Créer un vecteur appelé Sexe qui suit une loi binomiale de taille 1 et de probabilité 0.55, comprenant 120 valeurs. Recodez la variable Sexe, qui prendra la valeur Masculin si la valeur est 0 et la valeur Féminin si la valeur est 1.
Donner la moyenne et la médiane du poids en fonction du sexe.
Faire un graphique de la variable poids en fonction du sexe. N'oubliez pas de donner un titre et des légendes au niveau des axes. Y insérer la valeur de la moyenne sous forme d'une étoile de couleur rouge.

Chapitre 7

Réalisation de tests statistiques avec R

Objectifs à atteindre à cette étape

Connaitre les différents tests statistiques et leurs conditions d'application.

Réaliser des tableaux de contingence et des tests sur les variables qualitatives.

Réaliser des tests de comparaison d'une variable quantitative en fonction d'une variable qualitative.

Réaliser des tests de liaisons entre deux variables quantitatives.

7.1 - Principes généraux des tests statistiques

Pour savoir si une différence observée entre deux variables est uniquement due au hasard (variabilité aléatoire) ou si elle reflète une «vraie» différence, il est nécessaire de s'appuyer sur des règles précises et objectives, c'est le rôle des tests statistiques.

Un test statistique est donc une procédure qui permettra de choisir entre deux hypothèses à partir de données observées, généralement issues d'un échantillon, tout en quantifiant le risque d'incertitude lié à la décision finale.

Toute la démarche d'un test est construite sur le fait que l'hypothèse de référence (hypothèse nulle, H_0) est considérée comme vraie. Il est donc important de bien définir cette hypothèse de référence et d'en déduire ensuite son hypothèse alternative.

Le choix de l'une ou l'autre des hypothèses conduit à un tableau à 4 cases résumant les situations possibles. Il permet d'approcher la notion de risque d'erreur en lien avec la décision finale.

H_0 (Pas de relation; pas de différence)	En réalité: **H_0 vraie** (Il n'existe pas de différence)	En réalité: **H_0 fausse** (Il existe une différence)
Décision: *H_0 fausse* (Rejet de l'hypothèse nulle)	Erreur de première espèce : **risque α** Conclure qu'il y a quelque chose alors qu'il n'y a rien (risque d'illusion)	**1- α**
Décision: *H_0 vraie* (Acceptation de l'hypothèse nulle)	**1-β** : puissance de test	Erreur de deuxième espèce : **risque β** Conclure qu'il n'y a rien alors qu'il y a quelque chose (risque de négligence)

7.2 - Tests statistiques pour comparer une variable qualitative en fonction d'une autre variable qualitative

7.2.1 - Tests du χ^2

Les tests du χ^2 (khi-deux) sont utilisés pour:

a- Comparer des distributions de variables qualitatives (nominales, ordinales, binaires) ou quantitatives discrètes

- χ^2 de conformité: pour la comparaison d'une distribution observée à une distribution dans une population
- χ^2 d'homogénéité: pour la comparaison de deux ou plusieurs distributions observées

b- Etudier la liaison entre les distributions de deux variables qualitatives (nominales, ordinales, binaires) ou quantitatives discrètes d'un même échantillon

- χ^2 d'indépendance

Echantillons

Variable	B1	B2	...	Bi	Total
A1	O11 C11	O12 C12		O1j C1j	t1
...					
Ai	Oi1 Ci1			Oij Cij	ti
Total	n1	n2	....	nj	N

Les effectifs théoriques sont les effectifs attendus et se calculent pour chaque case de la façon suivante:

$$C_{ij}=(t_i \times n_j)/N$$

Calcul de la statistique du test de χ^2

$$\chi^2 = \Sigma \left[(o_{ij} - c_{ij})^2 / c_{ij} \right]$$

Degré de liberté:

ddl = (r -1) * (k -1)

r=nombre de ligne et k =nombre de colonne

Les principes du test repose sur la comparaison des effectifs observés dans chacune des classes à la répartition théorique obtenue en supposant que les variables sont indépendantes.

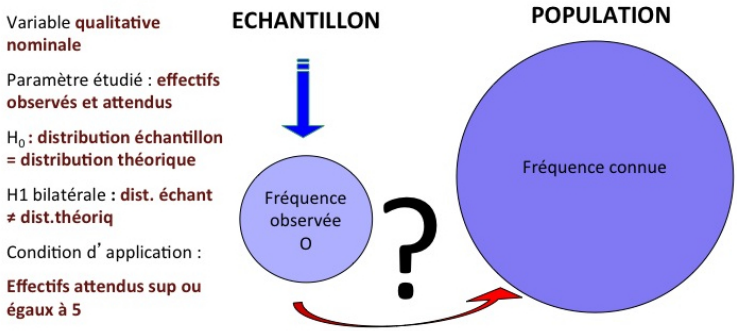
Les conditions d'application du test reposent sur un contrôle des effectifs théoriques qui doivent être supérieurs ou égaux à 5.

Les hypothèses nulles à tester sont:

- χ^2 conformité: l'échantillon observé provient de la population
- χ^2 d'homogénéité: les échantillons étudiés proviennent de la même population
- χ^2 d'indépendance: Il n'existe pas de liaison entre les deux variables qualitatives.

Les tables de la loi du χ^2, en fonction des degrés de liberté, permettent de connaître la valeur de p et de la comparer ensuite au seuil retenu, généralement $\leq 0,05$, pour conclure.
Les degrés de liberté sont calculés à partir du nombre de lignes et du nombre de colonnes.

A - Test de χ^2 de conformité ou d'ajustement



Pour comparer un pourcentage d'un échantillon à celui d'une population, vous utiliserez la fonction binom.test().

Situation bilatérale
binom.test(n1,n,pt)
n1= nombre de cas observé
n=effectif total
pt= proportion théorique

Situation unilatérale
binom.test(n1,n, pt, alternative = "greater")
binom.test(n1,n ,pt, alternative = "less")

Pour comparer plusieurs pourcentages, vous utiliserez la fonction **chisq.test()**.

Exemples:

Soit les effectifs de l'échantillon E1 à comparer aux pourcentages attendus dans la population Propop.

```
>E1<-c(89,37,30,28,2)
>Propop<-c(0.40,0.20,0.20,0.15,0.05)
>chisq.test(E1,p=Propop)
```

Vous obtenez alors ce résultat.

```
Chi-squared test for given probabilities

data:  E1
X-squared = 9.9901, df = 4, p-value = 0.04059
```

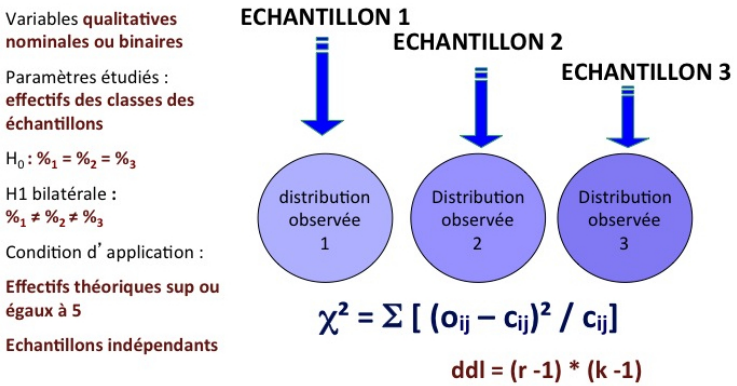
Soit un échantillon de 150 personnes contenant 73 hommes pour une proportion attendue dans la population de 53%.
>binom.test(73,150,0.53)

Vous obtenez alors ce résultat.

```
Exact binomial test

data:  73 and 150
number of successes = 73,
number of trials = 150, p-value = 0.2891
alternative hypothesis: true probability
of success is not equal to 0.53
95 percent confidence interval: 0.4043206 0.5695498
sample estimates:
probability of success : 0.4866667
```

B - Test de χ^2 d'homogénéité



Pour comparer les pourcentages de deux ou plusieurs échantillons, vous utiliserez la fonction **chisq.test()** qui est décrite au sous-chapitre suivant.

C - Test de χ^2 d'indépendance

Ce test de χ^2 est utilisé pour étudier la liaison entre deux variables qualitatives. Dans ce cas, l'hypothèse nulle à tester

est qu'il n'existe pas de relation entre les deux variables. Avec le logiciel R, vous utiliserez la fonction **chisq.test()**.
La fonction chisq.test() s'utilise avec l'argument "correct" qui prend la valeur TRUE ou FALSE en fonction des effectifs théoriques. Si les effectifs théoriques sont inférieurs à 5 alors le paramètre "correct" prend la valeur TRUE afin d'appliquer la correction de Yates au test de χ^2.
La fonction chisq.test() s'applique sur le résultat de la fonction table() ou peut-être utilisée directement avec les deux variables à comparer.
Le choix a été fait dans ce livre d'utiliser le résultat de la fonction table() puisque l'étape qui précède le test nécessite de réaliser un tableau de contingence (cf 6.2) qui utilise cette fonction afin d'identifier les pourcentages à comparer.
Avant de réaliser un test, il est nécessaire de vérifier également les conditions d'application du test.

Exemple

```
>chisq.test(table(Matable$GROUPOIDS,Matable$SEXE))$expe
cted
```

Vous obtiendrez alors ce résultat contenant les effectifs théoriques pour chacun des sous-groupes.

```
           Femme    Homme
(9,15]  5.322581 5.677419
(15,20] 4.838710 5.161290
(20,25] 4.838710 5.161290
```

Dans ce cas, il est nécessaire d'appliquer la correction de Yates puisque des effectifs théoriques sont inférieurs à 5.

```
>chisq.test(table(Matable$GROUPOIDS,Matable$SEXE),correc
t=TRUE)
```

Vous obtiendrez alors ce résultat.

```
Pearson's Chi-squared test
```

```
data:  table(Matable$GROUPOIDS, Matable$SEXE)
X-squared = 0.4591, df = 2, p-value = 0.7949
```

- X-squared représente la valeur du calcul du test de χ^2.
- df représente le nombre de degré de liberté, qui est la produit du (nombre de lignes – 1) et du (nombre de colonnes -1) ➔ ddl=(Nbligne-1)*(Nbcolonne-1)
- p-value représente la valeur de la probabilité en relation avec le test.

Dans cet exemple, vous conclurez alors qu'il n'a pas été montré de liaison statistiquement significative entre le sexe des enfants et leur groupe de poids.

Cependant, il y a deux façons de considérer cette liaison entre ces deux variables.
(i) Si la question avait été de s'intéresser à la répartition par sexe en fonction du groupe de poids (*pourcentages en ligne du tableau correspondant à la fonction suivante table(Matable$GROUPOIDS,Matable$SEXE)*), vous auriez conclu qu'il n' a pas été retrouvé de différence significative de la distribution des sexes par groupes de poids.

```
>round(prop.table(table(Matable$GROUPOIDS,Matable$SEXE
),1)*100,1)

# Le résultat donne les pourcentages suivants à comparer

        Femme  Homme
(9,15]   54.5  45.5
(15,20]  40.0  60.0
(20,25]  50.0  50.0
```

(ii) Si la question avait été de s'intéresser à la répartition des groupes de poids en fonction du sexe (*pourcentages en colonne du tableau correspondant à la fonction suivante table(Matable$GROUPOIDS,Matable$SEXE)*), vous auriez

conclu qu'il n'a pas été retrouvé de différence significative de la répartition des groupes de poids en fonction du sexe.

>round(prop.table(table(Matable$GROUPOIDS,Matable$SEXE),2)*100,1)

Le résultat donne les pourcentages suivants à comparer.

```
        Femme Homme
(9,15]   40.0  31.2
(15,20]  26.7  37.5
(20,25]  33.3  31.2
```

NB: C'est bien le même test de χ^2 qui permet de répondre à ces deux questions.

7.2.2 - Test exact de Fisher

Le test exact de Fisher permet de réaliser les mêmes comparaisons que le test de χ^2, et ce même avec des effectifs théroriques inférieurs à 5. Avec le logiciel R, vous utiliserez alors la fonction **fisher.test()**.
Comme précédemment, la fonction fisher.test() peut s'appliquer sur le résultat de la fonction table() ou elle peut être utilisée directement avec les deux variables à comparer.

>fisher.test(table(Matable$GROUPOIDS,Matable$SEXE))

Vous obtiendrez alors ce résultat:

```
Fisher's Exact Test for Count Data

data:  table(Matable$GROUPOIDS, Matable$SEXE)
p-value = 0.9024
alternative hypothesis: two.sided
```

NB: vous concluez comme dans l'exemple précédent avec les deux façons de considérer la question concernant la liaison entre ces deux variables.

7.2.3 - Test de Mac Nemar pour séries appariées

Le test de Mac Nemar est utilisé pour comparer des données appariées dans un tableau 2x2.
Il permet par exemple de comparer les résultats d'un nouveau test diagnostic à ceux d'un test de référence appliqué au même échantillon de population en s'intéressant aux paires discordantes.

Exemple

Soit un échantillon de 1600 personnes, comprenant 800 malades et 800 non malades en fonction du test de référence, auquel est appliqué le nouveau test. On obtient alors le tableau de contingence suivant.

		Test de référence		
		Malade	Non Malade	Total
Nouveau test	Malade	714	238	952
	Non Malade	86	562	648
	Total	800	800	1600

```
>Contingence<-matrix(c(714,86,238,562),ncol=2)
>mcnemar.test(Contingence)
```

Vous obtiendrez alors ce résultat:

```
McNemar's Chi-squared test

data:  Contingence
McNemar's chi-squared = 71.3086, df = 1,
p-value< 2.2e-16
```

L'hypothèse nulle étant basée sur l'équivalence des deux techniques, vous concluez alors à une différence significative entre les deux techniques.

7.3 - Tests statistiques pour comparer une variable quantitative en fonction d'une variable qualitative.

7.3.1 - Tests paramétriques

Pour comparer la moyenne d'une variable quantitative en fonction d'une variable catégorielle, vous utiliserez les analyses de variances (ANOVA). L'ANOVA determine la p-value à partir d'un test F de Fisher-Snedecor qui compare deux variances par leur rapport. Dans les principes de ce test, l'hypothèse nulle d'égalité des variances se traduit donc par un rapport égal à la valeur 1.
Les deux variances à comparer dans le cadre de l'ANOVA seront les variances inter-groupes (écart moyen entre chaque moyenne et la moyenne générale) et les variances intra-groupes (la variation moyenne des individus à l'intérieur des groupes).

Les conditions d'application du test reposent sur la normalité des distributions dans les différents groupes et l'égalité des variances dans les différents groupes.

Pour évaluer la normalité, vous utiliserez le test de Shapiro, et avec le logiciel R la fonction **shapiro.test()**. L'hypothèse nulle de ce test est que la distribution est normale.

Pour tester l'égalité des variances, vous utiliserez le test de Bartlett, et avec le logiciel R la fonction **bartlett.test()**. L'hypothèse nulle de ce test est que les variances sont égales.

Pour réaliser ensuite le test statistique, si l'ensemble des conditions d'application sont respectées, vous utiliserez les fonctions **aov()** et **anova()**.

Exemple

Vous souhaitez comparer les moyennes de poids en fonction du sexe dans votre échantillon.

Les moyennes à comparer

>by(Matable$POIDS,Matable$SEXE,mean, na.rm=T)

```
Matable$SEXE: Femme
[1]     17.13333
--------------------
Matable$SEXE: Homme
[1]     17.84375
```

tester la normalité des distributions
Le nombre d'observations dans chaque groupe à comparer étant limité, la normalité sera testée sur l'ensemble des données transformées pour répondre à une distribution normale centrée. Il faut tester alors la normalité des résidus du modèle de régression linéaire (fonction **aov()**).

>Maregression<-aov(POIDS~SEXE,data=Matable)
>Residus<-residuals(Maregression)
>shapiro.test(`Residus`)

```
Shapiro-Wilk normality test

data:  Residus
W = 0.9527, p-value = 0.1847
```

NB: l'hypothèse nulle en relation avec le test de Shapiro-Wilk considère que la distribution de la variable suit une loi normale. La p-value étant strictement supérieure au risque de première espèce α fixé à 5%, le test n'est pas significatif. Vous ne rejettez pas l'hypothèse nulle.

tester l'égalité des variances de la variable POIDS entre chacun des groupes

>bartlett.test(Matable$POIDS~Matable$SEXE)

```
Bartlett test of homogeneity of variances
data:  Matable$POIDS by Matable$SEXE
Bartlett's K-squared = 0.005, df = 1, p-value = 0.9437
```

#Vous ne rejettez pas l'hypothèse nulle

réaliser pour finir le test statistique

>Maregression<- aov(POIDS~ SEXE,data=Matable)
>anova(Maregression)

```
Analysis of Variance Table
Response: Matable$POIDS

             Df Sum Sq Mean Sq F value Pr(>F)
Matable$SEXE  1   3.91  3.9073  0.1839 0.6712
Residuals    29 616.09 21.2446
```

Le calcul de la valeur du test F (F value=0.1839) est le rapport de la variance intergroupe (3.9073) sur la variance intragroupe (21.2446). La p-value est ici de 0.6712

Il n'a pas été mis en évidence de différence statistiquement significative des poids moyens en fonction du sexe.

NB: dans le cas de la comparaison de deux moyennes d'un grand échantillon (n>30), vous pouvez également utiliser le test de Student avec la fonction **t.test().**

>t.test(Matable$POIDS~Matable$SEXE,var.equal=T)

```
Welch Two Sample t-test
data:  Matable$POIDS by Matable$SEXE
t = -0.4286, df = 28.789, p-value = 0.6714
alternative hypothesis: true difference in means is not
equal to 0
95 percent confidence interval:
 -4.101615  2.680781
sample estimates:
mean in group Femme mean in group Homme
           17.13333            17.84375
```

NB: dans le cas de la comparaison de plus de deux moyennes avec une ANOVA, l'hypothèse alternative est qu'au moins une des moyennes est différente des autres. Si le test vous amène à retenir l'hypothèse alternative, vous aurez à identifier les différences entre les moyennes en les comparant alors deux à deux. Vous pouvez utiliser le test de Student avec la fonction **t.test()** mais cela peut-être fastidieux vous avez alors la possibilité d'utiliser la fonction **TukeyHSD().**

Exemple

#Création de la variable TAILLE

>Matable$TAILLE<- c(70.3, 66.1, 78.7, 78.3, 111.1, 71.3, 93.4, 114.1, 80.7, 144.5, 132.2, 72.2, 164.9, 99.3, 138.3, 166.6, 135.4, 111.9, 148.2, 144.5, 132.2, 129.7, 71.9, 99.3, 138.3, 71.2, 62.4, 67.2, 58.9, 59.2, 92.2)

#Calcul de l'indice de masse corporelle appelé IMC

>Matable$IMC<-Matable$POIDS/(Matable$TAILLE/100)^2

#Comparaison des IMC moyens en fonction du groupe d'âge

>Maregression<-aov(Matable$IMC~Matable$GROUPAGE)
vérification des conditions d'applications
>Residus<-residuals(Maregression)
>shapiro.test(Residus)

```
Shapiro-Wilk normality test W = 0.9456, p-value = 0.1181
```

>bartlett.test(Matable$IMC~Matable$GROUPAGE)

```
Bartlett's K-squared = 2.316, df = 2, p-value = 0.3141
```

#Réalisation de l'anova

>anova(Maregression)

```
Analysis of Variance Table
Response: Matable$IMC
                Df Sum Sq Mean Sq F value    Pr(>F)
Matable$GROUPAGE  2 1187.0  593.52  13.985 6.149e-05 ***
Residuals        28 1188.3   42.44
```

#Détermination des différences entre les groupes

>TukeyHSD(Maregression)

```
Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Matable$IMC ~ Matable$GROUPAGE)

$`Matable$GROUPAGE`
                diff        lwr       upr     p adj
 0-4 - >10  14.378488   7.234340 21.522637 0.0000847
 5-9 - >10   9.818267   2.931319 16.705215 0.0040622
 5-9 - 0-4  -4.560221 -12.392828  3.272386 0.3344227
```

NB : Vous retrouvez une différence entre le groupe de 10 ans et plus et les deux autres groupes.

7.3.2 - Tests non paramétriques

Dans les cas où la distribution n'est pas normale ou que l'égalité des variances n'est pas vérifiée alors il faudra utiliser des tests non paramétriques pour la comparaison d'une variable quantitative en relation avec une variable qualitative. Vous utiliserez un test de Wilcoxon s'il n'y a que deux classes et un test de Kruskall-Wallis s'il y a plus de deux classes dans la variable qualitative. Avec le logiciel R, les deux fonctions **wilcox.test()** dans le cas de comparaison de deux moyennes (test des rangs de Wilcoxon), ou **kruskal.test()** dans le cas de la comparaison de plus de deux moyennes (test de Kruskal-Wallis) sont disponibles.

Exemples

En supposant que les règles d'application des tests paramétriques ne sont pas respectées vous utiliseriez la fonction wilcox.test() de cette façon:

>wilcox.test(Matable$POIDS~Matable$SEXE)

```
Wilcoxon rank sum test

data:  Matable$POIDS by Matable$SEXE
W = 109, p-value = 0.6823
alternative hypothesis: true location shift is not equal
to 0
```

NB: Vous pouvez également utiliser le test de Kruskal-Wallis, pour la comparaison de deux moyennes. Ce test permet de comparer plus de deux moyennes.

>kruskal.test(Matable$POIDS~Matable$SEXE)

```
Kruskal-Wallis rank sum test
data:  Matable$POIDS by Matable$SEXE
```

```
Kruskal-Wallis chi-squared=0.1891, df=1, p-value=0.6637
```

NB: Si le test de Kruskal-Wallis peut être utilisé pour la comparaison de deux moyennes, le test de Wilcoxon ne permet pas la comparaison de plus de deux moyennes.

7.4 - Tests de liaison entre deux variables quantitatives

7.4.1 - Tests paramétriques

Pour étudier la liaison entre deux variables quantitatives, avec le logiciel R, vous utiliserez soit les fonctions **lm()** et **summary()** soit la fonction **cor.test()**.
Comme le paramètre étudié est ici le coefficient de corrélation, l'hypothèse nulle qui traduit l'absence de liaison entre les deux variables quantitatives, se définit par un coefficient de corrélation égal à zéro**.**
L'hypothèse alternative se définit par un coefficient de corrélation différent de 0 en situation bilatérale. En situation unilatérale, soit la liaison est positive et le coefficient de corrélation est testé strictement supérieur à zéro, soit la liaison est négative et le coefficient est testé strictement inférieur à zéro.

Conditions d'application :

- Les variables X et Y sont aléatoires
- L'association entre X et Y est linéaire
- Les observations de chaque variable sont indépendantes
- Les distributions de Y liées à chaque valeur de X sont normales et de variance constante et vice versa.

Exemple

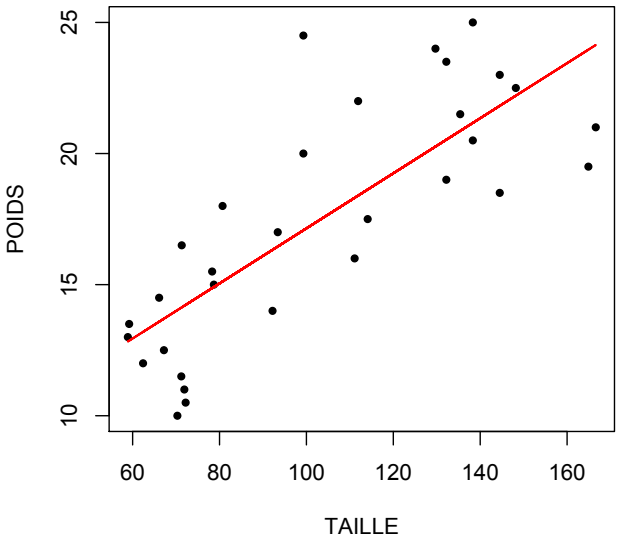
Vous souhaitez étudier la liaison qui existe entre le poids et la taille dans votre échantillon.

le graphique permet d'étudier la linéarité de l'association

>plot(Matable$TAILLE,Matable$POIDS, pch=20,xlab="TAILLE", ylab="POIDS")

ajouter la droite de régression (deux commandes possibles)

>lines(Matable$TAILLE,lm(Matable$POIDS~Matable$TAILLE) $fitted.values,type="l",col="red",lwd=1.5)
ou
>abline(lm(Matable$POIDS~Matable$TAILLE),type="l",col= "red",lwd=1.5)



puis réaliser le test du coefficient de corrélation.

>Maregression<- lm(Matable$POIDS~Matable$TAILLE)

>summary(Maregression)

```
Call:
lm(formula = Matable$POIDS ~ Matable$TAILLE)

Residuals:
    Min      1Q  Median      3Q     Max
-4.4610 -2.3193  0.1698  1.7744  7.4275

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    6.64532    1.67007   3.979 0.000423 ***
Matable$TAILLE  0.10501    0.01537   6.831 1.68e-07 ***
---
Signif.codes:0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05‘.’0.1‘ ’ 1

Residual standard error: 2.862 on 29 degrees of freedom
Multiple R-squared: 0.6167,   Adjusted R-squared: 0.6035
F-statistic: 46.67 on 1 and 29 DF,  p-value: 1.676e-07
```

Vous concluez alors à une association entre le poids et la taille.

Vous pouvez aussi utiliser la fonction cor.test() avec le paramètre "method" qui prendra la valeur "pearson".

Exemple

>cor.test(Matable$TAILLE,Matable$POIDS,alternative="two.sided",method="pearson",conf.level=0.95)

Vous obtiendrez alors ce résultat:

```
Pearson's product-moment correlation
data:  Matable$TAILLE and Matable$POIDS
t = 6.8313, df = 29, p-value = 1.676e-07
alternative hypothesis: true correlation is not equal to
0
95 percent confidence interval:
 0.5971602 0.8915679
sample estimates: cor 0.7853269
```

7.4.2 - Tests non paramétriques

Dans les cas où les conditions d'application ne sont pas vérifiées alors il vous faudra utiliser des tests non paramétriques. Vous utiliserez un test de corrélation des rangs de Spearman. Avec le logiciel R, vous utiliserez alors la fonction **cor.test()** avec le paramètre "method" qui prendra la valeur "spearman".

Exemple

>cor.test(Matable$TAILLE,Matable$POIDS,alternative="two.sided",method="spearman",conf.level=0.95)

Vous obtiendrez alors ce résultat:

```
Spearman's rank correlation rho

data:  Matable$TAILLE and Matable$POIDS
S = 1018.41, p-value = 9.379e-08
alternative hypothesis: true rho is not equal to 0
sample estimates:   rho  0.7946753
```

Vous rejettez donc l'hypothèse nulle et concluez à une relation entre le poids et la taille.

Exercice d'application

Ex 14

Créer un vecteur appelé Poids contenant 120 valeurs qui suit une loi normale de moyenne 2,7kg et d'écart type 0,07.
Créer un vecteur appelé Sexe qui suit une loi binomiale de taille 1 et de probabilité 0.45, comprenant 120 valeurs. Recodez la variable Sexe, qui prendra la valeur Masculin si la valeur est 0 et la valeur Féminin si la valeur est 1.
Donner la moyenne et la médiane du poids en fonction du sexe. Existe-til une différence significative des poids en fonction du sexe?

Annexes

Utilisation des tests

			Variable à expliquer			
			Qualitative nominale (2 classes)	**Qualitative nominale (plus de 2 classes)**	**Qualitative ordinale**	**Quantitative**
Facteur étuidié (variable explicative)	**Qualitatifs (deux groupes)**	**Indépendants**	chisq.test() fisher.test()	chisq.test() fisher.tes()	Test de Mann-whitney**	t.test() wilcox.test()*
		Appariés	mcnemar.test()	Test Q de Cochran**	wilcox.test()*	t.test() wilcox.test()*
	Qualitatifs (plus de deux groupes)	**Indépendants**	chisq.test() fisher.test()	chisq.test() fisher.test()	kruskal.test()*	aov() & anova() aov() & summary() kruskal.test()*
		Appariés	Test Q de Cochran**	Test Q de Cochran**	friedman.test()**	
	Quanti-tatifs		Régression logistique**	Régression logistique multinomiale**	Régression logistique ordinale**	lm() corr(,pearson) corr(,spearman)*

** Tests non paramétriques*
***Non abordé dans ce livre*

Corrigés des exercices

Ex1

```
rm(list=ls())
ls()
```

Ex2

```
#Création de l'objet Prenom
        Prenom<-"Vincent"
#Création de l'objet Nom
        Nom<-"Richard"
#Création de l'objet Identite
        Identite<-paste(Nom,Prenom,sep=',')
#Affichage du contenu de l'objet Identite
        Identite
```

Ex3

```
#Création de l'objet Femme
        Femme<-257836
#Création de l'objet Homme
        Homme<-247643
#Création de l'objet Population
        Population<-sum(Femme,Homme)
        #ou Population<-Femme+Homme
#Afficher le contenu de l'objet Population
        Population
#Création de l'objet Sexratio
        Sexratio<-Homme/Femme
```

```
#Affichage de l'objet Sexratio
        Sexratio
#Création de l'objet Dcfemme
        Dcfemme<-236
#Création de l'objet Dchomme
        Dchomme<-328
#Création de l'objet Dctotal pour la somme des décès
        Dctotal<-Dcfemme+Dchomme
        # ou DcTotal<-sum(Dcfemme,Dchomme)
#Création de l'objet Mortfemme pour mortalité des femmes
        Mortfemme<-(Dcfemme/Femme)*1000
#Création de l'objet Morthomme pour mortalité des hommes
        Morthomme<-(Dchomme/Homme)*1000
#Création de l'objet Mortpop pour mortalité de la population
        Mortpop<-(Dctotal/Population)*1000
#Affichage du contenu des objets sur la mortalité
Mortfemme;Morthomme;Mortpop
```

Ex4

```
ls()
```

Ex5 - Manipulation de champs "dates"

```
#création de l'objet Datdeb
Datdeb<-c("12/01/90","02/01/90","22/04/95","05/06/91")
#Création de l'objet Datfin
Datfin<-
c("25/02/1990","15/06/1991","20/05/1996","14/04/1992")
#Afficher le contenu des objets Datdeb,Datfin
Datdeb;Datfin
#Conversion au format date
Datdeb<-as.Date(Datdeb,"%d/%m/%y")
Datfin<-as.Date(Datfin, ,"%d/%m/%Y")
```

```
#Afficher le contenu des objets Datdeb,Datfin
Datdeb;Datfin
#Création d'un objet Duree
Duree<-difftime(Datfin,Datdeb,units=days)
# ou Duree<-Datfin-Datdeb
#Afficher le contenu de l'objet Duree
Duree
#Créer un objet An et afficher son contenu
An<-as.character(Datdeb,"%Y")
#Création d'un objet Mois et affichage du contenu
Mois<-as.character(Datdeb,"%m")
#Création d'un objet Jour et affichage du contenu
Jour<-as.character(Datdeb,"%d")
#Afficher An, Mois, Jour
An; Mois; Jour
```

Ex6 - Extraction de données

```
#Création d'un objet appelé Poids  et affichage du contenu
Poids<-seq(20,45,0.5)
#Extraire la 16° valeur de l'objet Poids
Poids[16]
#Extraire les 1°, 16° et 31° valeurs de l'objet Poids
Poids[c(1,16,31)]
#Extraire  les poids dont la valeur est supérieure à 38
Poids[Poids>38]
#Extraire les poids dont les valeurs sont à la fois supérieures à
25 et inférieures à 37.
Poids[Poids>25 & Poids<37]
#Créer un objet Classpoids
Classpoids<-ifelse(Poids<25,1,ifelse(Poids>30,3,2))
# afficher l'effectif de chacun des groupes
table(Classpoids) #ou summary(as.factor(Classpoids))
```

```
#Création d'une matrice
Mat.1<-
matrix(c(1,2,0,2,1,2,0,3,2,0,4,2,0,0,1,0,1,0,0,2,2,3,1,2)
,ncol=4)
#Afficher la matrice
Mat.1
#Créer l'objet Matable de type table de données à partir de
l'objet Mat.1.
Matable<-data.frame(Mat.1)
#Renommer les colonnes avec les valeurs suivantes : SEXE,
AGEJOUR, INFECTION, FRATRIE.
names(Matable)<-
c("SEXE","AGEJOUR","INFECTION","FRATRIE")
#Renommer les lignes avec des valeurs allant de EPI1 à EPI6.
row.names(Matable)<-paste(rep("EPI",6),c(1:6),sep="")
#Extraire la ligne correspondant à l'individu EPI4
Matable[rownames(Matable)=="EPI4",]
#Afficher le contenu de la colonne nommée FRATRIE
Matable$FRATRIE
#Ajouter une colonne DECES ayant les valeurs: 0,0,1,1,0,0
Matable$DECES<-c(0,0,1,1,0,0)
#Ajouter une colonne SEXE2
Matable$SEXE2<-ifelse(Matable$SEXE==1,"M",ifelse(
Matable$SEXE==2,"F","NP") )
#Ajouter une colonne POIDS
Matable$POIDS<-2950+(Matable$AGE*126)
#Ajouter une colonne POIDSKG
Matable <-transform(Matable,POIDSKG=POIDS/1000).
#Afficher le contenu de Matable dans une fenêtre différente
de la console.
View(Matable)
```

Ex8 – Variables quantitatives

```
#Création de l'objet Mesure
Mesure<-runif(30,20,35)
#Calculer la médiane et la moyenne
summary(Mesure)
#Calculer l'écart-type
sd(Mesure)
#Calculer l'intervalle de confiance de la moyenne
t.test(Mesure)$conf.int

#Création de l'objet Mesure1
Mesure1<-runif(100,20,35)
#Calculer la médiane et la moyenne
summary(Mesure1)
#Calculer l'écart-type
sd(Mesure1)
#Calculer l'intervalle de confiance de la moyenne
t.test(Mesure1)$conf.int
```

Ex9 – Variables qualitatives

```
#Création de l'objet Caracter1
Caracter1<-rbinom(110,1,0.15)
#Création de l'objet Caracter2
Caracter2<-rbinom(110,2,0.40)

#Calcul des effectifs de Caracter1
A1<-table(Caracter1)
#Calcul des pourcentages de Caracter1
P1<-round(prop.table(A1)*100,0)
#Table pour Caracter1
Matable1<-
cbind(A1[1],P1[1],A1[2],P1[2],margin.table(A1),100)
```

```
colnames(Matable1)<-c('Eff1',"%","Eff2","%","Total","%")
rownames(Matable1)<-""
Matable1

#Calcul des effectifs de Caracter2
A2<-table(Caracter2)
#Calcul des pourcentages de Caracter2
P2<-round(prop.table(A2)*100,0)
#Table pour Caracter2
Matable2<-
cbind(A2[1],P2[1],A2[2],P2[2],A2[3],P2[3],margin.table(A2),
100)
colnames(Matable2)<-c('Eff1',"%","Eff2","%","Eff3","%",
"Total","%")
rownames(Matable2)<-""
Matable2
```

Ex10

```
# utiliser la fonction layout()
Zone<-matrix(c(1,1,2,3),ncol=2)
layout(Zone)

#afficher la fenêtre graphique
layout.show(max(Zone))

#réaliser chacun des graphiques à insérer
boxplot(Mesure,main="Boite à moustache \nde la variable
Mesure")
 hist(Mesure, breaks=seq(20,35,by=2.5),main="Histogram me
\n de la variable Mesure", xlab="Groupes",ylab=
"Effectifs")
A<-cut(Mesure,breaks=seq(20,35,by=2.5))
plot(table(A),type="o",main="Polygone de fréquence \n de la
variable Mesure", cex.axis=0.5, ylab="Effectifs")
```

Ex11

```
#Création de l'objet Caracter1
Caracter1<-rbinom(100,1,0.15)
#Création d'un objet Sexe en fonction des valeurs de Caracter1
Sexe<- ifelse (Caracter1==0,"Masculin", "Féminin")
#Création d'un graphique en barre
barplot(table(Sexe),main='Sexe', col=c("pink","blue"))

#Création de l'objet Caracter2
Caracter2<-rbinom(100,2,0.40)

#Création d'un objet contenant les valeurs des  pourcentages pour chaque classe
Percent<-round(prop.table(table(Caracter2))*100,0)

#ouvrir une nouvelle fenêtre graphique
quartz() # avec R pour Mac
#windows()  # avec R pour windows

#Faire un graphique de type camembert
pie(Percent, main ='Niveaux scolaires',
labels=c(paste("Primaire","\n",Percent[1],"%"),paste("Secondaire","\n
",Percent[2],"%"),paste("Universitaire","\n",Percent[3],"%")))
```

Ex12

```
#Création de l'objet Sexe
Sexe<-rbinom(120,1,0.55)

#Création de l'objet Nivscol
```

```
Nivscol<-rbinom(120,2,0.40)

# Les étapes de construction du tableau de contingence du
niveau scolaire en fonction des sexes
Table2<-table(Sexe,Nivscol)# table des valeurs des niveaux
scolaires en fonction de sexe
Total<-margin.table(Table2) #Effectif total de la table
Efflign<-margin.table(Table2, 1) #total des effectifs par ligne
Effcol<-margin.table(Table2, 2) #total des effectifs par
colonne
Plign<-prop.table(Table2, margin=1) #Pourcentage par ligne
prop.table(Efflign) #Pourcentage du total des effectifs par
ligne
Peffcol<-prop.table(Effcol) #Pourcentage du total des effectifs
par colonne

Plign2<-round(rbind(Plign,Peffcol)*100,0)#Ajouter une ligne
contenant le pourcentage des effectifs globaux des niveaux
scolaires aux lignes des pourcentages obtenus en fonction du
sexe et faire un arrondi.
Tablecont<-cbind(Table2,Efflign) #ajouter une colonne
contenant le total des effectifs en lignes à la table des valeurs
Totalcol<-c(Effcol,Total) #créer un vecteur contenant les
effectifs des colonnes et le total global
Tablecont<-rbind(Tablecont,Totalcol) #l'ajouter sur une ligne
au tableau de contingence
Tablecont<-
cbind(Tablecont[,1],Plign2[,1],Tablecont[,2],Plign2[,2],Tableco
nt[,3],Plign2[,3],Tablecont[,4])#insérer entre chacune des
colonnes des valeurs la colonne des pourcentages
correspondant.
Dcol<-c(round(prop.table(Efflign)*100,0),100)#créer un
vecteur contenant les pourcentages de l'objet Sexe et le
nombre 100 correspondant à 100%
```

```
Tablecont<-cbind(Tablecont,Dcol)#ajouter une colonne à la
table de contingence  contenant les valeurs du vecteur Dcol
#Renommer le nom des colonnes
colnames(Tablecont)<-
c("Prim.","%","Sec.","%","Uni.","%","Total","%")
#renommer le nom des lignes
rownames(Tablecont)<-c("Masculin","Féminin","Total")
#Visualiser la table de contingence
Tablecont
```

Ex 13

```
#Création du vecteur Poids
Poids<-rnorm(120,mean=2.8, sd=0.08)
#Création du vecteur Sexe
Sexe<-rbinom(120,1,0.55)
#Recodifier la variable Sexe
Sexe<-ifelse(Sexe==0,'Masculin','Féminin')
#Afficher les mesures résumant la distribution du poids en
fonction du sexe.
aggregate(Poids,list(Sexe),summary)
#Réaliser un graphique de type "boite à moustache"
boxplot(Poids~Sexe, main="Poids en fonction du
sexe",ylab='Poids en kg',xlab='Sexe')
points(1,mean(Poids[Sexe=='Féminin']),pch=4, col='red')
points(2,mean(Poids[Sexe=='Masculin']),pch=4,col='red')
```

Ex 14

```
#Création du vecteur Poids
Poids<-rnorm(120,mean=2.8, sd=0.07)
#Création du vecteur Sexe
Sexe<-rbinom(120,1,0.45)
```

```
#Modification de la variable Sexe
Sexe<-ifelse(Sexe==0,'Masculin','Féminin')
#Afficher les mesures résumant la distribution du poids en
fonction du sexe
aggregate(Poids,list(Sexe),summary)
#Réaliser une comparaison des poids en fonction du sexe.
Maregression<-aov(Poids~Sexe)
#Vérifier la normalité des données
Residus<-residuals(Maregression)
shapiro.test(Residus)
#Vérifier l'homogénéité des variances entre les groupes
bartlett.test(Poids~Sexe)
#Réaliser le test Anova
anova(Maregression)
#Il aurait été possible de réaliser un test T de Student
t.test(Poids~Sexe)
```

Index des fonctions utilisées