Travaux pratiques de statistique pour L1 MASS sous  [1]

Thibault LAURENT[2]

23 décembre 2009

[1]Il s'agit des notes écrites au cours de l'année 2009

[2]Toulouse School of Economics, thibault.laurent@univ-tlse1.fr

Table des matières

1 Rudiments d'une enquête statistique **5**
- 1.1 Introduction . . . 5
- 1.2 La récolte de données . . . 5
  - 1.2.1 Rappel de vocabulaire . . . 5
  - 1.2.2 Les formes d'interview . . . 5
  - 1.2.3 Le choix des questions . . . 6
  - 1.2.4 Exemple d'enquête . . . 6
- 1.3 La gestion des données . . . 7
  - 1.3.1 Recueillir les données manuellement . . . 7
  - 1.3.2 Quelques conseils . . . 7
  - 1.3.3 Préparation d'un jeu de données . . . 7
  - 1.3.4 Sauvegarde . . . 8
- 1.4 Pour aller plus loin . . . 8

2 Introduction au logiciel R **9**
- 2.1 Introduction . . . 9
- 2.2 Présentation du logiciel . . . 9
- 2.3 Première prise en main . . . 10
  - 2.3.1 Opérations arithmétiques de base . . . 10
  - 2.3.2 Les vecteurs . . . 11
  - 2.3.3 Quelques opérateurs . . . 12
  - 2.3.4 Quelques opérateurs de comparaison . . . 13
  - 2.3.5 Les fonctions de base . . . 14
  - 2.3.6 Les classes simples d'objets . . . 16
  - 2.3.7 Les vecteurs de "chaîne de caractères" dits facteurs . . . 18
  - 2.3.8 Pour passer d'une classe d'objet à une autre . . . 19
  - 2.3.9 Le menu . . . 20
- 2.4 Exercices . . . 21
  - 2.4.1 Créer son environnement de travail . . . 21
  - 2.4.2 Créer différentes classes d'objet . . . 21

2.4.3 Manipulation d'objets 21

3 Manipulation d'une base de donnée 23
3.1 Introduction 23
3.2 Tableaux de données 23
3.3 Transformation du jeu de données 27
3.3.1 Le nom des variables 28
3.3.2 Le traitement des données manquantes 28
3.3.3 Le traitement des données aberrantes 28
3.3.4 Uniformiser les modalités 29
3.3.5 Agréger les tables 30
3.3.6 En résumé 31

4 Statistiques descriptives de base 32
4.1 Introduction 32
4.2 Les variables qualitatives : tableaux de fréquence et représentation graphique 32
4.2.1 Tableau de fréquence 32
4.2.2 Représentation graphique 34
4.2.3 Commentaires à faire sur les tableaux et graphiques 40
4.3 Les variables quantitatives discrètes 40
4.3.1 Diagramme en bâtons 40
4.3.2 Fonction de répartition empirique 42
4.4 Les variables quantitatives continues 43
4.4.1 L'histogramme 43
4.4.2 Résumé d'une variable quantitative 45
4.4.3 Boîte à moustache 47
4.4.4 Commentaires à faire sur les résumés et les graphiques 49

5 Etude multivariée 50
5.1 Introduction 50
5.2 Liaison entre deux variables quantitatives 50
5.2.1 Le nuage de points 50
5.2.2 Coefficients de corrélation linéaire 55
5.2.3 Régression linéaire 55
5.2.4 Commentaires sur l'étude de 2 variables quantitatives 56
5.3 Liaison entre deux variables qualitatives 56
5.3.1 Table de contingence 56
5.3.2 Distribution marginale 57
5.3.3 Distribution conditionnelle 57
5.3.4 Représentation graphique 59

5.3.5 Mesure de liaison 64
5.4 Laison entre une variable quantitative et une variable qualitative 65
5.4.1 Boîtes à moustache parallèle 65
5.4.2 Rapport de corrélation 65
5.4.3 Histogrammes parallèles 67
5.5 Analyse multivariée 67
5.5.1 Deux variables quantitatives et une variable qualitative 69
5.5.2 Plusieurs variables quantitatives 71

6 Annexe **73**
6.1 Enregistrer une fenêtre graphique 73
6.2 Les fonctions graphiques 74
6.3 Récapitulatif : commandes permettant d'ajouter des objets aux graphes existants (et quelques options ...) 75
6.4 Récapitulatifs : Options graphiques 75

Table des figures

3.1 Exemples de graphiques obtenus avec la commande 24

4.1 Exemple de diagramme en secteurs sans options 35
4.2 Exemple de diagramme en secteurs en utilisant des options 37
4.3 Exemple de diagramme en tuyaux d'orgue sans options 38
4.4 Exemple de diagramme en tuyaux d'orgue en utilisant des options 39
4.5 Exemple de diagramme en barres sans options 41
4.6 Fonction de répartition empirique 42
4.7 Exemple d'histogramme sans options 44
4.8 Exemple d'histogramme avec options 45
4.9 Exemple de boîte à moustaches avec options 48

5.1 Premier exemple de nuage de points 51
5.2 Second exemple de nuage de points 52
5.3 Troisième exemple de nuage de points 54
5.4 Distribution jointe 59
5.5 Exemple de diagramme en tuyaux d'orgue bidimensionnel 60
5.6 Exemple de diagramme en tuyaux d'orgue bidimensionnel 61
5.7 Représentation graphique des profils-lignes 62
5.8 Représentation graphique des profils-colonnes 63
5.9 Représentation graphique du lien entre variable quantitative et qualitative 66
5.10 Histogrammes parallèles 68
5.11 Nuage de points avec prise en compte d'une variable qualitative 69
5.12 Nuages de points où chaque quadrant représente les individus appartenant à une classe 70
5.13 Représentation matricielle de nuages de points 72

Chapitre 1

Rudiments d'une enquête statistique

1.1 Introduction

L'idée de ces séances de travaux pratiques est de réaliser une étude statistique à partir d'informations récoltées auprès de la population des L1 MASS. Dans ce premier TP, nous présentons les deux étapes préliminaires d'une étude statistique à savoir : la récolte des données auprès d'une population donnée et la saisie/stockage des données à l'aide d'outils informatiques. Les données qui seront construites seront ensuite traitées avec le logiciel R.

1.2 La récolte de données

1.2.1 Rappel de vocabulaire

On appelle individu l'objet étudié. Ici les individus seront les étudiants. On appelle population l'ensemble des individus qui participent à l'expérience, ici les étudiants de L1 MASS appartenant à un certain groupe de TP... On appelle variables ce qui est étudié chez les individus : ici, nous allons étudier l'âge, le sexe, le fait que vous soyez fumeur ou non, le type de transport que vous privilégiez et enfin le temps moyen que vous passez dans les transports par jour.

1.2.2 Les formes d'interview

Plusieurs types d'interview sont possibles. Les plus connus sont :
- en tête à tête
- par téléphone
- par mail

Par exemple, l'institut de sondages BVA réalise chaque semaine des enquêtes en utilisant ces trois moyens de récoltes de données. Pour chaque type d'enquête, les enquêteurs

doivent interroger une population de 1000 personnes dites représentative d'une partie de la population française (en général, les plus de 18 ans) sur des questions politiques, marketing, ...

Question : comment définiriez-vous un échantillon représentatif de la population française ?

1.2.3 Le choix des questions

En général, ce sont les sociétés (industries alimentaires,...), les partis politiques ou bien les particuliers (rare car un sondage coûte cher) qui commandent les questions aux instituts qui se chargent eux-mêmes de la récolte, de la gestion des données et du traitement... Une question doit être compris sans aucune ambiguïté par l'interviewé ainsi que le choix de la réponse qui est proposée. Par exemple : "pensez-vous que tel homme politique fasse du bon travail" ne donnera pas les mêmes résultats et n'a pas le même sens que "Voteriez-vous pour tel homme politique s'il se présentait aux prochaines élections présidentielles". En lisant certains journaux, les statisticiens peuvent être choqués de la façon dont leurs études ont été utilisées...

1.2.4 Exemple d'enquête

La population d'étude est les L1 MASS. L'échantillon est les étudiants appartenant à un groupe de TP. Chaque étudiant devra poser les questions suivantes à votre voisin et inversement, puis devra reporter les réponses au tableau. On ajoutera un numéro de ligne qui ira de 1 jusqu'au nombre d'tudiants présents dans la salle de TP.

Questionnaire

- Q1. Quel âge avez-vous ? (la réponse est un réel compris entre 15 et 100 :)
- Q2. Le sexe (réponse : M ☐ ou F ☐)
- Q3. Etes-vous fumeur ? (réponse : oui ☐ ou non ☐)
- Q4. Quel type de transport privilégiez-vous pour vous rendre à l'université : voiture/co-voiturage, transports publics (bus, métro, train, velo toulouse), marche/vélo, autres (1 seule réponse possible ; les réponses seront codées : VOIT ☐ - TPUB ☐ - NPOL ☐ - AUTRES ☐)
- Q5. Quel est votre degré de satisfaction des transports publics en terme de déservitude (réseau, fréquence, ...) dans l'agglomération toulousaine ? 1 (peu satisfait) ☐ , 2 (satisfait) ☐ , 3 (très satisfait) ☐
- Q6. Quel est le temps moyen (aller/retour) que vous mettez par jour pour vous rendre à l'université ? (la réponse est exprimée en minutes est un réel :)

- Q7. Parmi les questions 1 à 5, si je créé une variable par question, combien y-a-t-il de variables quantitatives ? (la réponse est un entier compris entre 0 et 5 :)
- Q8. Combien de variables sont qualitatives nominales ? (la réponse est un entier compris entre 0 et 5 :)

1.3 La gestion des données

Le développement de l'outil informatique permet de recueillir et stocker facilement les données. Par exemple, les sites de vente en ligne, les banques, les assurances,... possèdent des fichiers de données sur les clients qui sont gérées et mises à jour automatiquement dès qu'une transaction est effectuée par le client. Les géologues ou les botanistes peuvent utiliser des tablettes (ordinateurs portables) pour recueillir des informations sur des prélèvements de sol ou sur les plantes.

1.3.1 Recueillir les données manuellement

Moodle permet de faire des questionnaires, mais la récupération des données n'est pas encore facile. Pour stocker les données dans une "base de données", on peut utiliser des logiciels tels que **Access** ou **Excel**. C'est ce dernier que nous allons utiliser pour récolter les données sur les étudiants.

1.3.2 Quelques conseils

- Une disposition en lignes des individus et en colonnes des variables est usuelle
- La première colonne est en général l'identifiant de chaque individu
- La première ligne contient en général le nom des variables
- Il est important de coder simplement et judicieusement les modalités des variables qualitatives
- les valeurs manquantes correspondent à des blancs
- se méfier des types des variables (date, entier, réel,...)

1.3.3 Préparation d'un jeu de données

Il est primordial de maintenir et gérer une base de données efficacement. Les erreurs de saisie au moment du questionnaire, puis au moment de la saisie informatique auront des répercussions sur l'étude statistique. Par exemple, il serait étonnant de voir que la variable âge puisse prendre la valeur 120 dans une population d'étudiants ou que le temps moyen pour aller à l'université prenne une valeur négative... Dans ce cas là, on parle d'incohérences et le statisticien devra corriger ces problèmes en utilisant des méthodes spécifiques.

1.3.4 Sauvegarde

Le fichier pourra être enregistré dans un répertoire de travail TP1. Pour importer cette base de données sous R, il est préférable d'utiliser l'extension CSV (séparateur point virgule). On pourra appeller le fichier ressondages.csv

1.4 Pour aller plus loin

Les deux paragraphes que nous venons de voir sont généralisables à d'autres études. La récolte de données dépend bien évidemment de l'expérience menée. Certaines expérimentations scientifiques (dans le domaine de la biologie, physique, chimie, médecine) et/ou industrielles ne peuvent produire que peu de données car les coûts sont très élevés. Au contraire, les données dans les banques, les assurances, les ventes en ligne,... sont nombreuses car l'information est peu coûteuse.

La manière dont est menée l'expérience (ou le sondage) dépend bien évidemment de la problématique de l'étude.

Par ailleurs, le compte rendu d'une étude statistique devrait systématiquement faire figurer la manière dont les données ont été recueillies. Il s'agit au minimum de connaître la taille de l'échantillon et la manière dont il a été choisi. Par exemple, si je décide d'interroger des gens à une sortie de métro, il faut savoir que le profil de la population est déjà typé (des utilisateurs du métro) et dépend par exemple de l'heure à laquelle est fait le sondage.

La récolte et la préparation de ces données est la base de toute étude statistique et les logiciels statistiques ont été développés dans cette optique.

Les logiciels utilisés dans le monde des statistiques sont nombreux et on peut facilement les diviser en deux groupes :

- les logiciels "payants" comme par exemple **Stata**, **SPSS**, **SAS**, **S-plus** ... On peut également citer les logiciels **Matlab** et **Mathematica**, orientés davantage vers le calcul numérique ou **Eviews** et **Gauss** orientés vers l'économétrie...
- les logiciels "gratuits", la plupart étant inclus dans la famille des logiciels libres tels que **Scilab** (équivalent de **Matlab**), **Maxima** (équivalent de **Mathematica**) et enfin R(équivalent de **S-plus**)...

Chapitre 2

Introduction au logiciel R

2.1 Introduction

Nous découvrirons dans ce TP les origines et rudiments de base du logiciel R . L'objectif de ce TP est d'apprendre à construire et manipuler des objets de base comme les vecteurs (de valeurs numériques ou de caractères), se faliliariser avec des commandes de base et réussir à importer des jeux de données.

2.2 Présentation du logiciel

R est un système qui est communément appelé langage et logiciel. Il permet de réaliser des analyses statistiques. Plus particulièrement, il comporte des moyens qui rendent possibles la manipulation des données, le calcul de formules complexes et de belles représentations graphiques. R a aussi la possibilité d'exécuter des programmes et des fonctions qui ont été créés par ses utilisateurs. Le site `http://www.r-project.org` donne notamment des informations :

- la licence de type GNU : R est un logiciel libre, c'est-à-dire que les utilisateurs ont la possibilité d'exécuter, de copier, de distribuer, d'étudier, de modifier et d'améliorer le logiciel.
- les personnes qui ont contribué à ce projet depuis ses débuts dans les années 90.
- des exemples de graphiques qui ont été produits
- les dernières mises à jour, des forums où sont discutés des problèmes des utilisateurs
- de la documentation avec des manuels, une FAQ et un wiki.
- le journal qui publie des articles présentant le travail de chercheurs utilisant R

Le site du CRAN (`http://cran.cict.fr`) permet quant à lui de télécharger le logiciel et de télécharger également des "groupements" de fonctions ou packages ou librairies, qui ont été programmés et rendus disponibles par les utilisateurs. A noter que ce site contient

également d'excellentes documentations

2.3 Première prise en main

Lorsque la console est ouverte, la version du logiciel est indiquée et plusieurs informations sont données sur le logiciel. Les versions de R sont mises régulièrement à jour sur le serveur du CRAN, ce qui n'est pas forcément le cas dans les salles de TP informatique...

2.3.1 Opérations arithmétiques de base

Le texte d'accueil est suivi d'une ligne commençant par le caractère > et sur laquelle devrait se trouver votre curseur. Cette ligne est appelée l'invite de commande (ou prompt en anglais). Elle signifie que R est disponible et en attente de votre prochaine commande. Nous allons tout de suite lui fournir les commandes suivantes :

```
> 2 + 2

[1] 4

> 4/2

[1] 2

> 2.1 - 1 + 3.5 - 6 + 5/4

[1] -0.15

> (2.1 - 1 + 3.5 - 6 + 5)/4

[1] 0.9

> 7 * 1/3

[1] 2.333333

> 7/3

[1] 2.333333

> 2/5 - 3 * 4 + 2
```

```
[1] -9.6
```

```
> 2/(5 - 3 * (4 + 2))
```

```
[1] -0.1538462
```

Quelques remarques :

- On notera que la multiplication/division sont bien évidemment prioritaires sur l'addition/soustraction, d'où l'importance d'uilitiser des parenthèses.
- On utilise le point et non la virgule pour les chiffres décimaux.
- Lorsqu'on fournit une commande incomplète à R , celui-ci nous propose de la compléter en nous présentant une invite de commande spéciale utilisant le signe +.

2.3.2 Les vecteurs

On peut créer un vecteur en utilisant la fonction `c`. Prenons un exemple en utilisant des valeurs numériques :

```
> c(1, 2, 3, 4, 5)
```

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

On peut également faire des opérations :

```
> c(1, 2, 3, 4, 5) + 1
```

```
[1] 2 3 4 5 6
```

est équivalent à :

```
> c(1, 2, 3, 4, 5) + c(1, 1, 1, 1, 1)
```

```
[1] 2 3 4 5 6
```

Et :

```
> c(1, 2, 3, 4, 5) + c(1, 5)
```

```
[1] 2 7 4 9 6
```

est équivalent à :

```
> c(1, 2, 3, 4, 5) + c(1, 5, 1, 5, 1)
```

```
[1] 2 7 4 9 6
```

Enfin :

```
> 5 * c(1, 2, 3, 4, 5)

[1]  5 10 15 20 25
```

est équivalent à

```
> c(5, 5, 5, 5, 5) * c(1, 2, 3, 4, 5)

[1]  5 10 15 20 25
```

2.3.3 Quelques opérateurs

On notera que l'opérateur : s'utilise de la façon suivante :

```
> 1:25

 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
[23] 23 24 25
```

Lorque R affiche un vecteur, le chiffre compris dans les crochets indiquent le numéro de la composante du premier chiffre de la ligne.

Pour chercher la composante d'un vecteur, il faut utiliser l'opérateur `[]`. Par exemple :

```
> c(1, 2, 3, 4, 5)[3]

[1] 3
```

affiche la troisième composante du vecteur alors que :

```
> c(1, 2, 3, 4, 5)[c(1, 2)]

[1] 1 2
```

affiche les composantes 1 et 2 du vecteur.

Pour utiliser la puissance, on utilise l'opérateur `^`.

```
> 5^5 == 25

[1] FALSE
```

2.3.4 Quelques opérateurs de comparaison

Pour vérifier une égalité ou une inégalité entre deux chiffres, on utilise les opérateurs ==, !=, <, >, >=, <=. Par exemple :

```
> 5 * 5 == 25

[1] TRUE

> 5 * 5 != 25

[1] FALSE

> (5 + 5) == 25

[1] FALSE

> (5 + 5) != 25

[1] TRUE

> 5 < 2

[1] FALSE

> 5 >= 5

[1] TRUE
```

Cela renvoie un "booléen", c'est-à-dire une des deux valeurs `TRUE` ou `FALSE`.

Pour vérifier simultanément deux conditions, on utilise l'opérateur `&` si les deux conditions doivent être vérifier en même temps ou l'opérateur `|` si seulement une des deux conditions est suffisante. Par exemple :

```
> (25 == 5 * 5) & (3 <= 2)

[1] FALSE

> (25 == 5 * 5) | (3 <= 2)

[1] TRUE
```

2.3.5 Les fonctions de base

R contient un nombre important de fonctions par défaut. Si vous tapez `help(cos)`[1], cela fait apparaître une fenêtre html qui vous donne des indications sur comment utiliser la fonction `cos`.

Dans la fenêtre de gauche, vous pouvez visualiser la liste des fonctions qui existent par défaut, soit un nombre considérable.

La partie `Description` décrit ce que fait la fonction, la partie `Usage` montre quels sont les paramètres d'entrée de cette fonction, la partie `Arguments` détaille la signification des paramètres d'entrée, la partie `Details` apporte des informations sur la façon dont a été programmé le code, etc. On trouve généralement une bibliographie ainsi que des liens vers d'autres fonctions qui ont un lien avec la fonction décrite. En fin de page, les exemples d'utilisation sont souvent très utiles pour avoir une démonstration de la fonction.

Une fonction s'utilise en appelant son nom, on ouvre une parenthèse et on remplit les paramètres[2]. Parmi ces paramètres, certains sont obligatoires. Il s'agit des paramètres qui n'ont pas de valeurs par défaut, ce qui peut être vu dans l'aide de la fonction dans la partie `Usage`.

Les fonctions avec 1 ou 2 paramètres

Les fonctions les plus faciles à utiliser sont celles qui n'utilisent qu'un seul argument d'entrée comme les fonctions trigonométriques (`cos`, `sin`, `tan`, `acos`, `asin`, `atan`) et les fonctions `abs` (renvoie la valeur absolue), `sqrt` (renvoie la racine carrée), `exp` (renvoie la valeur exponentielle).

La fonction `log` contient deux paramètres, le deuxième paramètre (optionnelle) indique quelle est la base du logarithme. Par défaut, le paramètre optionnel vaut `base=exp(1)` :

```
> log(100)

[1] 4.60517

> log(100, 5)

[1] 2.861353
```

[1] L'opérateur `?` est un raccourci de la fonction `help`. Ici, on aurait pu faire `?cos`

[2] Lorsqu'on tape seulement le nom de la fonction, R renvoie le code de la fonction. Essayer par exemple de taper simplement `help`

```
> log(100)/log(5)

[1] 2.861353
```

Une autre fonction de base pratique qui a deux paramètres est la fonction `which` qui renvoie les numéros des composantes d'un vecteur qui vérifie une certaine condition. Par exemple :

```
> which(1:10 >= 6)

[1]  6  7  8  9 10

> which(1:10 >= 6 & 1:10 < 7)

[1] 6

> which(1:10 >= 6 | 1:10 < 7)

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

Lorsqu'on travaille sur une matrice[3], l'option `arr.ind=TRUE` renvoie l'indice (numéro de ligne et numéro de colonne) des éléments de la matrice qui vérifient une certaine condition.

```
> which(matrix(1:10, 3, 3) >= 6, arr.ind = TRUE)

     row col
[1,]   3   2
[2,]   1   3
[3,]   2   3
[4,]   3   3
```

Les fonctions avec plus de 2 paramètres

Voyons un exemple d'utilisation de fonction avec la fonction `rep` qui permet de répéter une valeur ou un vecteur plusieurs fois :

```
> rep(1, times = 5)

[1] 1 1 1 1 1
```

est équivalent à :

[3] On reviendra sur la notion de matrices un peu plus tard

```
> rep(1, 5)

[1] 1 1 1 1 1
```

car le paramètre `times` est le premier paramètre optionnel et du coup, on n'est pas obligé de le renseigner même si cela est préférable pour éviter des mauvaises surprises.... Si on utilise la fonction `rep` à un vecteur, cela donne :

```
> rep(c(1, 2), times = 3)

[1] 1 2 1 2 1 2
```

Si au lieu du paramètre `times`, on utilise le paramètre `each`, on obtient :

```
> rep(c(1, 2), each = 3)

[1] 1 1 1 2 2 2
```

Une fonction de "base" pratique est la fonction `length` qui prend en entrée un vecteur et renvoie le nombre de composantes de ce vecteur :

```
> length(rep(c(1, 2), each = 3))

[1] 6
```

2.3.6 Les classes simples d'objets

L'idée serait par exemple de stoker un vecteur de valeurs numériques dans un objet et de pouvoir ensuite manipuler cet objet plutôt que d'avoir à répéter l'étape de création du vecteur. Prenons tout de suite un exemple :

```
> a <- c(2.2, 4, 5, 6.5)
```

Que signifie cette commande ? L'opérateur `<-` est appelé opérateur d'assignation. Il prend une valeur ou un vecteur quelconque à droite et la place dans l'objet indiqué à gauche. La commande pourrait donc se lire mettre le vecteur `c(2.2,4,5,6.5)` dans l'objet nommé `a`. On va ensuite pouvoir réutiliser cet objet dans d'autres calculs ou simplement afficher son contenu :

```
> a

[1] 2.2 4.0 5.0 6.5

> rep(a, times = 2)

[1] 2.2 4.0 5.0 6.5 2.2 4.0 5.0 6.5
```

Les réels

Il s'agit de la catégorie ou bien la classe d'objets que nous avons vus jusqu'à présent. On utilisera notamment cette classe d'objets pour définir des variables quantitatives. La fonction `class` permet de déterminer à quel type appartient un objet.

```
> class(a)

[1] "numeric"
```

donne les mêmes résultats que :

```
> class(5)

[1] "numeric"
```

car la fonction `class` s'intéresse au contenu du vecteur.

Les chaînes de caractères

Pour définir un objet de classe "chaîne de caractère", il faut utiliser les guillemets. Par exemple :

```
> b <- "voiture"
> class(b)

[1] "character"
```

On peut définir également un vecteur de chaînes de caractères. Par exemple :

```
> b.vecteur <- c("voiture", "pied", "transport", "transport")
```

ce qui pourra être très pratique pour coder des variables qualitatives. Lorsqu'on oublie les guillemets, R recherche dans ce cas là :

- un objet qui a été créé et qui porterait déjà ce nom,
- une fonction portant le même nom,
- du code propre au langage R comme `if`, `for`, ...

Par exemple, si on tape :

```
> b.function <- rep
```

cela a pour effet d'affecter à `b.function` le code de la fonction `rep` et on peut ensuite utilisé `b.function` exactement comme `rep`. Par exemple :

```
> b.function(b.vecteur, each = 2)

[1] "voiture"   "voiture"   "pied"      "pied"      "transport"
[6] "transport" "transport" "transport"
```

En revanche, il y aura un message d'erreur si vous faites `b<-voiture`.

Les booléens

Un objet appartient à la classe "booléen" s'il prend la valeur `TRUE` ou `FALSE`. Cette classe peut être intéressante pour créer des variables qualitatives dichotomiques comme "Etes-vous oui ou non fumeur ?". Par exemple :

```
> d <- c(TRUE, FALSE, TRUE)
> !d

[1] FALSE  TRUE FALSE
```

On notera que le point d'exclamation qui précède un booléen renvoie exactement les valeurs contraires.

2.3.7 Les vecteurs de "chaîne de caractères" dits facteurs

Utiliser un vecteur de type "chaîne de caractère" peut causer certains désagréments. Par exemple, si je définis :

```
> e.char <- c("vélo", "velo", "Velo")
```

pour R , le vecteur aura trois composantes différentes. Compte tenu qu'il peut y avoir des erreurs de saisie, la classe "facteur" définit dès le début les modalités existantes. On crée un objet de la classe facteur de la façon suivante :

```
> e.factor <- factor(c("vélo", "velo", "Velo", "transport",
+     "Transport"), levels = c("velo", "pied", "transport"))
> e.factor

[1] <NA>      velo      <NA>      transport <NA>
Levels: velo pied transport
```

Dans R , une valeur manquante est définie par le code `NA`. On remarque que les modalités qui n'ont pas été définies dans l'option `levels` sont déclarées manquantes. On notera que `levels` est également une fonction qui prend comme argument d'entrée un vecteur "facteur" et renvoie les différentes modalités.
Pour affecter de nouvelles valeurs à un vecteur "facteur", vous devrez utiliser l'opérateur `<-`. Par exemple :

```
> e.factor[6:8] <- c("transport", "transports", "pied")
> e.factor
```

```
[1] <NA>      velo      <NA>      transport <NA>      transport
[7] <NA>      pied
Levels: velo pied transport

> levels(e.factor)

[1] "velo"      "pied"      "transport"
```

On peut également créer des variables qualitatives ordinales en précisant l'option `ordered=TRUE`. Par exemple :

```
> e.factor2 <- factor(c("4", "3", "2", "3", "2", "4"),
+     levels = c("2", "3", "4"), ordered = TRUE)
> e.factor2

[1] 4 3 2 3 2 4
Levels: 2 < 3 < 4
```

2.3.8 Pour passer d'une classe d'objet à une autre

On utilise les fonctions `as.numeric`, `as.character`, `as.boolean`, `as.factor`. Par exemple :

```
> as.character(e.factor2)

[1] "4" "3" "2" "3" "2" "4"

> as.numeric(e.factor2)

[1] 3 2 1 2 1 3

> as.numeric(as.character(e.factor2))

[1] 4 3 2 3 2 4

> as.factor(e.char)

[1] vélo velo Velo
Levels: velo Velo vélo
```

Ces fonctions sont à utiliser avec précaution car elles utilisent des règles particulières !

2.3.9 Le menu

Les menus qui sont accessibles depuis la console sont en réalité des "raccourcis" de fonctions qui peuvent être tapées directement dans la console. Dans l'onglet "Fichier", les barres qui sont intéressantes sont :

- Sourcer du code R : pour donner le lien d'un fichier qui contient exclusivement du code R et qui sera compilé dans la console. La fonction qui permet cela en utilisant la console est la fonction `source`.
- Nouveau script : permet d'ouvrir une fenêtre dans laquelle vous pourrez écrire et sauvegarder votre code R [4]
- Sauver l'environnement de travail : permet de sauvegarder tout ce qui a été compilé dans la console. La fonction qui permet cela en utilisant la console est la fonction `save.image`. Pour récupérer le travail d'une précédente session, vous cliquez sur "Charger l'environnement de travail".
- Changer le répertoire courant : donne le répertoire où sera sauvegardé automatiquement le travail et où la console ira chercher des fichiers par défaut lorsque le chemin n'est pas précisé. La fonction qui permet cela en utilisant la console est la fonction `setwd`. La fonction `getwd` indique le répertoire courant de travail

Dans l'onglet "Edition"

- Editer les données : permet de modifier un jeu de données comme sur **Excel** en donnant juste le nom du jeu de données

Dans l'onglet "Misc" :

- Lister les objets : renvoie dans la console tous les objets qui ont été créés. La fonction qui permet cela en utilisant la console est la fonction `ls`.

Dans l'onglet "packages" :

- Charger le package : permet de rendre utilisable des objets et fonctions qui étaient contenus dans un package ou librairie qui a été préalablement téléchargé sur votre ordinateur (par défaut en installant le logiciel R ou par vous-même en téléchargeant le package sur le site du CRAN. La fonction qui permet cela en utilisant la console est la fonction `library` suivie du nom du package.

Dans l'onglet "Aides", il y a énormément de lien intéressants pour trouver l'aide sur une fonction, accéder à un manuel, au site du CRAN, ...

[4] Il est néanmoins préférable d'utiliser un éditeur de R comme Tinn-R (voir Vidéo dans Moodle) qui n'est cependant pas disponible dans les salles de TP...

2.4 Exercices

2.4.1 Créer son environnement de travail

A l'aide de la commande `setwd`, placer l'environnement de travail vers le répertoire où se trouve le fichier contenant les données créées dans le TP1. Attention, utiliser soit : `"C :/mes documents/TP1"` ou `"C :\\ mes documents \\TP1"` et tâcher d'être organisé pour arriver ensuite à récupérer facilement les informations désirées.

Ouvrir un nouveau script dans lequel vous écrirez votre code. Attention, il est primordial de structurer vos lignes de code en utilisant par exemple l'opérateur `#` qui permet de taper des lignes de commentaires.

2.4.2 Créer différentes classes d'objet

A partir des résultats du sondage effectué dans le TP1[5] :

- créer un vecteur de classe "numérique" appelé `age`, qui contient les valeurs obtenues dans la Q1 du questionnaire.
- créer un vecteur "facteur", appelé `sexe`, qui contient les valeurs obtenues dans la Q2 du questionnaire.
- créer un vecteur de classe "booléen", appelé `fumeur`, qui contient les valeurs obtenues dans la Q3 du questionnaire.
- créer un vecteur "facteur", appelé `transport`, qui contient les valeurs obtenues dans la Q4 du questionnaire.
- créer un vecteur "facteur" ordonné, appelé `degre.satisf`, qui contient les valeurs obtenues dans la Q5 du questionnaire.

2.4.3 Manipulation d'objets

En utilisant les fonctions `which` et `length` ainsi que les opérateurs de comparaison :

- Q1. affecter à l'objet `n` le nombre d'individus questionnés.
- Q2. donner le nombre d'individus, puis la proportion d'individus qui ont strictement moins de 18 ans.
- Q3. donner le nombre d'individus, puis la proportion d'individus qui ont 18 ou 19 ans
- Q4. donner le nombre d'individus, puis la proportion d'individus qui ont strictement plus de 19 ans
- Q5. affecter à l'objet `n1` le nombre d'individus de sexe masculin et `n2` le nombre d'individus de sexe féminin.

[5] Dans les questions qui suivent, on ne considèrera que les individus qui ont répondu au questionnaire...

- Q6. créer le vecteur `sexe.masc` de type booléen et de taille `n` qui prend la valeur `TRUE` si l'individu est de sexe masculin et `FALSE` sinon. Créer de la même façon le vecteur `sexe.femi`.
- Q7. parmi les individus de sexe masculin, donner le pourcentage d'individus qui sont fumeurs. On pourra s'aider du vecteur `sexe.masc`.
- Q8. parmi les individus de sexe féminin, quel est le pourcentage de non fumeurs. On pourra s'aider du vecteur `sexe.femi`.
- Q9. donner la proportion d'individus qui privilégie les transports en commun pour aller à l'université
- Q10. Construire un vecteur nommé `indice.transport.non.public` contenant les indices des individus qui ne privilégient pas les transports en commun et un scalaire `n3` donnant la taille de ce vecteur.
- Q11. Construire un vecteur nommé `indice.transport.public` contenant les indices des individus qui privilégient les transports en commun et un scalaire `n4` donnant la taille de ce vecteur.
- Q12. parmi les personnes qui ne privilégient pas les transports en commun, quel est le pourcentage d'individus à trouver la déservitude peu satisfaisantes.
- Q13. parmi les personnes qui privilégient les transports en commun, quel est le pourcentage d'individus à trouver la déservitude satisfaisantes ou très satisfaisantes.

Chapitre 3

Manipulation d'une base de donnée

3.1 Introduction

L'objectif de ce TP est d'arriver à importer sous R un jeu de données au format `.csv`. Ensuite, on verra comment on peut agréger plusieurs bases de données pour en créer une seule. Finalement, on construira et commentera les premiers tableaux et graphiques statistiques que vous avez étudiés en cours.

3.2 Tableaux de données

De la même façon qu'il existe de nombreuses fonctions prédéfinies sous R, il existe également de nombreux jeux de données disponibles qui permettent notamment d'illustrer des graphiques ou méthodes statistiques. En tapant `data()` dans votre console, la liste des jeux de données disponibles par défaut apparaît.
En tapant par exemple `edit(iris)`, vous affichez un tableur (comme excel) qui vous permet de consulter et/ou modifier votre jeu de données.
Si vous voulez obtenir de l'information sur le jeu de données, vous pouvez taper `help(iris)` et une fenêtre html vous donne des informations sur le jeu de données.
Plusieurs fonctions renvoient des informations sur le jeu données. Par exemple :

```
> nrow(iris)

[1] 150

> ncol(iris)

[1] 5

> names(iris)
```

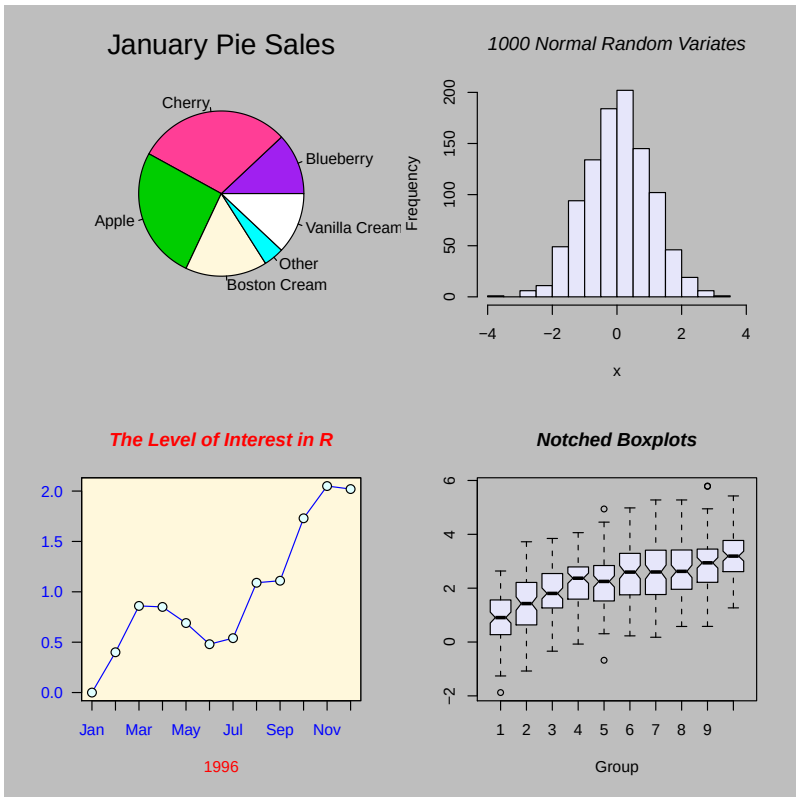

FIG. 3.1 – Exemples de graphiques obtenus avec la commande

```
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
[5] "Species"
```

On peut afficher une variable incluse dans un `data.frame` en utilisant le symbole `$`. Par exemple :

```
> iris$Species[c(49, 50, 51, 99, 100, 101)]

[1] setosa     setosa     versicolor versicolor versicolor
[6] virginica
Levels: setosa versicolor virginica
```

On peut également vouloir sélectionner seulement certaines variables chez certains individus. Pour ce faire, on utilise l'opérateur `[,]`. A gauche de la virgule, on indique quelles sont les lignes qu'on souhaite extraire et à droite de la virgule, on indique quelles sont les colonnes qu'on souhaite extraire. Par exemple, pour afficher les individus 2, 4 et 6 et seulement les variables 1 et 5, on utilise :

```
> iris[c(2, 4, 6), c(1, 5)]

  Sepal.Length Species
2          4.9  setosa
4          4.6  setosa
6          5.4  setosa
```

qui est équivalent à :

```
> iris[c(2, 4, 6), c("Sepal.Length", "Species")]

  Sepal.Length Species
2          4.9  setosa
4          4.6  setosa
6          5.4  setosa
```

Remarque importante pour la suite du TP : lorsqu'on veut sélectionner tous les individus sauf les individus 5, 10, 15 et 20, on utilise l'opérateur `-` devant la fonction `c`. Par exemple `iris[-c(5,10,15,20),]` renvoie le jeu de données `iris` "moins" les individus cités.

Un jeu de données est un objet dont la classe est `data.frame`. Il est très commode car il peut contenir toutes les variables étudiées, quelque soient leur nature[1] (quantitatives ou qualitatives). Il y a deux façons pour construire un `data.frame` :

[1] Ce qui ne sera pas le cas lorsque vous manipulerez des matrices qui ne pourront contenir que des variables quantitatives.

– On agrège des variables quantitatives ou qualitatives avec la fonction `data.frame`. Par exemple :

```
> v1 <- c(18, 19, 18, 17)
> v2 <- c(TRUE, FALSE, TRUE, FALSE)
> v3 <- factor(c("velo", "velo", "bus", "velo"), levels = c("velo",
+     "bus"))
> don <- data.frame(v1, v2, v3)
> names(don)
[1] "v1" "v2" "v3"
> names(don) <- c("age", "fumeur", "transport")
> dim(don)
[1] 4 3
```

– On importe directement un jeu de données qui est dans un format `.txt` à l'aide des fonctions `read.table` ou `read.delim`, un jeu de données qui est dans un format `.csv` à l'aide de la fonction `read.csv2`

Si on veut ajouter des individus au jeu de données `don`, on procède ainsi : d'abord on crée un nouveau `data.frame` `don2` avec le même nom et le même type de variables[2] :

```
> don2 <- data.frame(18, FALSE, "velo")
> names(don2) <- c("age", "fumeur", "transport")
> ncol(don2)

[1] 3
```

Ensuite, on utilise la fonction `rbind` qui permet de concaténer les 2 bases de données :

```
> don3 <- rbind(don, don2)
> don3

  age fumeur transport
1  18   TRUE      velo
2  19  FALSE      velo
3  18   TRUE       bus
4  17  FALSE      velo
5  18  FALSE      velo
```

[2]La première condition est obligatoire, la seconde est facultative et peut entraîner des changements de type l'objet créé...

La condition pour utiliser cette fonction est que les deux objets doivent avoir le même nombre de colonnes.

Si on veut ajouter une variable supplémentaire, on crée d'abord la variable sexe et ensuite on utilise la fonction `cbind` pour concaténer des colonnes entre elles :

```
> sexe <- factor(c("f", "f", "m", "m", "m"), levels = c("m",
+     "f"))
> length(sexe)

[1] 5

> don4 <- cbind(don3, sexe)
> don4

  age fumeur transport sexe
1  18   TRUE      velo    f
2  19  FALSE      velo    f
3  18   TRUE       bus    m
4  17  FALSE      velo    m
5  18  FALSE      velo    m
```

La condition est que les deux objets qu'on concatène aient le même nombre de lignes.

Q1 : Dans un premier temps, sauvegarder les trois fichiers "Bases de données G1", "Bases de données G2" et "Bases de données G3" dans un nouveau répertoire de travail TP3.
Ensuite, utiliser la fonction `setwd` pour placer votre environnement de travail R dans ce répertoire.
Importer les trois bases de données en utilisant la fonction `read.csv2`[3]. Vous les nommerez `groupe.1`, `groupe.2` et `groupe.3`.
Remarque : vérifier si la première ligne de votre base de données sous excel contient le nom des variables. Si oui, vous devrez utiliser l'option `header=TRUE` dans la fonction `read.csv2`.

3.3 Transformation du jeu de données

Le prochain objectif sera d'agréger les trois bases de données que nous venons d'importer. Le but est donc de ramener notre population d'étude de votre groupe de TP à l'ensemble de la promotion.

[3]Mettre le lien du fichier (avec l'extension) entre guillemets

3.3.1 Le nom des variables

Q2 : Vérifier si le nom des variables des objets `groupe.1`, `groupe.2` et `groupe.3` ont les même noms de variables. Si ce n'est pas le cas, donnez à chacun des objets les noms suivants[4] :

- `numero`
- `age`
- `sexe`
- `tabac`
- `transport`
- `gout.transport`
- `temps.transport`
- `nb.quanti`
- `nb.quali`

3.3.2 Le traitement des données manquantes

Certains étudiants du groupe 1 n'ont pas souhaité répondre au questionnaire... Lorsque vous affichez votre jeu de données (en faisant `edit(groupe.1)` ou simplement `groupe.1`), les quatre dernière lignes ne contiennent que des valeurs manquantes `NA`.
Lorsqu'une valeur est manquante de façon "accidentelle", on peut la remplacer par une valeur selon plusieurs méthodes. Les plus intuitives sont de remplacer une valeur manquante par la moyenne observée sur l'échantillon entier dans le cas d'une variable quantitative ou par la modalité la plus représentée s'il s'agit d'une variable qualitative.
Dans notre cas, ces valeurs manquantes n'apportent aucune information... C'est pourquoi nous n'allons pas conserver ces lignes.

Q3 : re-définir l'objet `groupe.1` en enlevant les lignes 24 à 27. On pourra utiliser l'opérateur `-` vu ci-dessus.

3.3.3 Le traitement des données aberrantes

Il arrive qu'un jeu de données contienne des incohérences. Dans la première base de données, un enquêteur a malencontreusement exprimé la variable `temps.transport` en heure au lieu de minutes et un étudiant affirme mettre 0 minutes pour se rendre à l'université.

Q4 : remplacer dans l'objet `groupe.1` ces valeurs incohérentes. On remplacera la valeur 1.3 par 90 et la valeur 0 par la valeur 1. On pourra utiliser la fonction `which` pour trouver les indices de ces individus.

[4] On pourra s'inspirer de la fonction `names` utilisée ci-dessus

3.3.4 Uniformiser les modalités

La dernière chose à vérifier est que les modalités des variables qualitatives aient été codées de la même façon dans toutes les bases de données. On peut utiliser la fonction `str` pour connaître la structure "globale" d'un `data.frame`. Par exemple :

```
> str(groupe.1)
```

```
'data.frame':        23 obs. of  9 variables:
 $ numero         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ age            : int  18 19 18 22 19 20 20 18 18 18 ...
 $ sexe           : Factor w/ 3 levels "","f","m": 2 2 3 3 2 3 3 2 2 3 ...
 $ tabac          : Factor w/ 3 levels "","non","oui": 2 2 3 2 3 2 3 2 2 3 ...
 $ transport      : Factor w/ 5 levels "","AUTRES","NPOL",..: 3 3 3 3 3 3 3 2 4 2 .
 $ gout.transport : int  1 1 2 3 3 3 3 2 2 2 ...
 $ temps.transport: num  30 90 1 30 20 5 20 70 80 40 ...
 $ nb.quanti      : int  2 2 2 2 2 2 2 2 2 2 ...
 $ nb.quali       : int  3 3 3 3 3 3 3 3 3 4 ...
```

On notera que les modalités ou `levels` des variables `sexe`, `tabac` et `transport` incluent la valeur `""`. En effet, même après avoir supprimé les lignes contenant les valeurs manquantes, cela n'a pas supprimé cette valeur parmi les modalités. Voici une manière de recoder les modalités de variables qualitatives ou `factor`. Prenons par exemple :

```
> a <- factor(c("m", "f", "m", ""))
> a <- a[1:3]
> levels(a)
```

```
[1] ""  "f" "m"
```

```
> a <- factor(as.character(a))
> levels(a)
```

```
[1] "f" "m"
```

Q5 : effectuer la transformation ci-dessus aux variables `sexe`, `tabac` et `transport` de l'objet `groupe.1`. Pour faire cela, on utilisera le début de code suivant : `groupe.1$sexe <- ...` en remplaçant les ... par du code basé sur l'exemple ci-dessus.

Pour changer les modalités d'une variable `factor`, on procède de la façon suivante :

```
> levels(a) <- c("masc", "fem")
> a
```

```
[1] fem  masc fem
Levels: masc fem
```

Q6 : uniformiser les modalités des variables `sexe`, `tabac` et `transport` sur les trois bases de données. On prendra comme modalités ou `levels` de référence, celles de la première base :

```
> levels(groupe.1$sexe)
[1] "f" "m"
> levels(groupe.1$tabac)
[1] "non" "oui"
> levels(groupe.1$transport)
[1] "AUTRES" "NPOL"   "TPUB"   "VOIT"
```

et on pourra utiliser le début de code suivant : `levels(groupe.2$sexe)<-` suivi du code approprié.

3.3.5 Agréger les tables

Q7 : à l'aide la fonction `rbind` présentée ci-dessus, concaténer les trois bases de données en une seule qu'on nommera `promo.09`

Q8 : changer les valeurs de la variable `numero`. On ira de la valeur 1 jusqu'à `nrow(promo.09)`.

Enfin, nous allons transformer la variable `tabac` en variable booléenne. Pour cela on va procéder de la manière suivante : on va dans un premier temps transformer la variable `tabac` en valeur numérique en utilisant la fonction `as.numeric`. Ensuite, on va modifier cette variable numérique de sorte qu'il n'y ait plus que deux valeurs possibles 0 et 1. Enfin, on va transformer cette variable numérique en variable booléenne ou `logical` en utilisant la fonction `as.logical`. Par exemple :

```
> a <- factor(c("oui", "non", "non"))
> a
[1] oui non non
Levels: non oui
> a <- as.numeric(a)
> a
[1] 2 1 1
> a <- a - 1
> a <- as.logical(a)
> a
[1]  TRUE FALSE FALSE
```

Q9 : transformer la variable `tabac` en variable booléenne en utilisant l'exemple ci-dessus.

3.3.6 En résumé

Si vous avez bien respecté les consignes, en appelant la fonction `str` à l'objet `nrow(promo.09)`, vous devriez obtenir les résultats suivants :

```
> str(promo.09)

'data.frame':        78 obs. of  9 variables:
 $ numero         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ age            : int  18 19 18 22 19 20 20 18 18 18 ...
 $ sexe           : Factor w/ 2 levels "f","m": 1 1 2 2 1 2 2 1 1 2 ...
 $ tabac          : logi  FALSE FALSE TRUE FALSE TRUE FALSE ...
 $ transport      : Factor w/ 4 levels "AUTRES","NPOL",..: 2 2 2 2 2 2 2 1 3 1 ...
 $ gout.transport : int  1 1 2 3 3 3 3 2 2 2 ...
 $ temps.transport: num  30 90 1 30 20 5 20 70 80 40 ...
 $ nb.quanti      : int  2 2 2 2 2 2 2 2 2 2 ...
 $ nb.quali       : int  3 3 3 3 3 3 3 3 3 4 ...
```

Chapitre 4

Statistiques descriptives de base

4.1 Introduction

L'objectif de ce TP est de construire, puis commenter les premiers tableaux et graphiques statistiques que vous avez étudiés en cours.

4.2 Les variables qualitatives : tableaux de fréquence et représentation graphique

4.2.1 Tableau de fréquence

On obtient un tableau d'effectifs d'une variable qualitative en utilisant la fonction `table`, suivi du vecteur contenant les valeurs observées. Par exemple :

```
> res.tab <- table(iris$Species)
> res.tab

    setosa versicolor  virginica
        50         50         50
```

Si j'applique la fonction `sum` à l'objet créé, j'obtiens la somme des effectifs des modalités. Si j'utilise la fonction `cumsum`, cela crée un vecteur avec les effectifs cummulés.

```
> n <- sum(res.tab)
> n

[1] 150
```

```
> eff.cum <- cumsum(res.tab)
> eff.cum

    setosa versicolor  virginica
        50        100        150
```

Pour obtenir un tableau des fréquences, il suffit de diviser l'objet `res.tab` par la taille `n` de la population. On fait la même chose pour avoir les fréquences cummulées.

```
> res.tab/n

    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333

> eff.cum/n

    setosa versicolor  virginica
 0.3333333  0.6666667  1.0000000
```

Pour agréger les effectifs, les fréquences et les fréquences cummulées dans un même tableau, on utilise la fonction `rbind`. On peut ensuite donner des noms de ligne à l'objet créé. Par exemple :

```
> res.tab2 <- rbind(res.tab, res.tab/n, eff.cum/n)
> row.names(res.tab2) <- c("effectif", "fréquence",
+     "fréquence cummulée")
> res.tab2

                      setosa versicolor  virginica
effectif           50.0000000 50.0000000 50.0000000
fréquence           0.3333333  0.3333333  0.3333333
fréquence cummulée  0.3333333  0.6666667  1.0000000
```

Enfin, on peut créer un vecteur qui donnera la somme par ligne des effectifs et fréquences en utilisant la fonction `apply` de la façon suivante :

```
> total <- apply(res.tab2[c(1, 2), ], 1, sum)
> total

 effectif fréquence
      150         1
```

Cette ligne de commande veut dire la chose suivante : on applique à la première, puis à la seconde ligne de l'objet `res.tab2` la fonction `sum`. Si on avait remplacé le chiffre 1 par 2, on aurait appliquer la fonction `sum` sur les colonnes.

Pour finir, on agrége la colonne des totaux au tableau en utilisant la fonction `cbind`. Avant cela, on ajoute la valeur `NA` au vecteur `total` pour signaler que le total des fréquences cummulées n'a aucun sens...

```
> total <- c(total, NA)
> res.tab3 <- cbind(res.tab2, total)
> res.tab3
```

```
                      setosa versicolor  virginica total
effectif           50.0000000 50.0000000 50.0000000   150
fréquence           0.3333333  0.3333333  0.3333333     1
fréquence cummulée  0.3333333  0.6666667  1.0000000    NA
```

Présentation des résultats dans un rapport

Dans un rapport, vous ne pouvez pas copier/coller les sorties fournies par R dans l'état. Vous devrez recopier vos résultats dans un tableau de la façon suivante.

	setosa	versicolor	virginica	total
effectif	50.00	50.00	50.00	150.00
fréquence	0.33	0.33	0.33	1.00
fréquence cummulée	0.33	0.67	1.00	

TAB. 4.1 – Variable Espèce du jeu de données Iris

Q1 : on peut également représenter ce type de tableau sur les variables quantitatives discrètes. En vous aidant des lignes de code ci-dessus, donner le tableau de fréquence de la variable `age` du `data.frame promo.09`. Quel est le mode de cette variable ?

4.2.2 Représentation graphique

Diagramme en secteur

On utilise la fonction `pie` suivie du résultat de la fonction `table`. Par exemple :

```
> pie(res.tab)
```

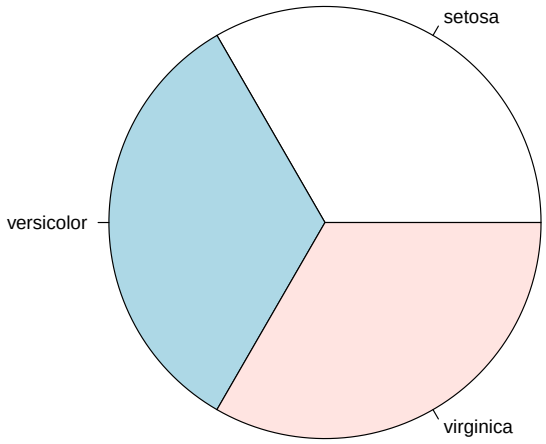

FIG. 4.1 – Exemple de diagramme en secteurs sans options

Il existe de nombreuses options (voir `help(pie)`) pour rendre le graphique plus parlant. Le choix des couleurs par exemple est très important. En effet, si on souhaite porter l'attention sur une modalité en particulier, on pourra choisir une couleur vive comme le rouge.

Pour changer de couleurs, on utilise l'option `col=` , suivi d'un vecteur avec des numéros (1 : noir, 2 : rouge, 3 : vert, 4 : bleu foncé, 5 : bleu ciel, 6 : rose, 7 : jaune, 8 : gris). Attention, si vous redéfinissez les couleurs, elles seront attribués en respectant l'ordre des modalités affichées quand vous utilisez la fonction `table`. Ici, la première couleur sera attribuée à la modalité `setosa`, la seconde à `versicolor` et la dernière à `virginica`.
On peut également changer le nom des modalités sur le graphique en utilisant l'option `labels`. Ceci peut être très utile car pour que le diagramme en secteurs soit "parlant", il faut que les modalités soient explicites. Par exemple, afficher "TPUB" n'est pas très parlant alors que "transport public" l'est...

On peut également utiliser les options suivantes qui sont par ailleurs communes (comme l'option `col`) à l'ensemble des fonctions qui réalisent des graphiques :

- `main` permet d'ajouter un titre au graphique
- `xlab` et `ylab` permettent d'ajouter un titre aux axes des abscisses et des ordonnées

Par exemple :

```
> pie(res.tab, col = c(2, 5, 7), labels = paste(c("setosa :",
+     "versicolor :", "virginica :"), as.character(round(res.tab/n,
+     2)), "%"), main = "Répartition des espèces d'iris",
+     xlab = "Source : Fisher, R. A. (1936)")
```

Vous aurez noté que l'option `labels` vaut :

```
> paste(c("setosa :", "versicolor :", "virginica :"),
+     as.character(round(res.tab/n, 2)), "%")

[1] "setosa : 0.33 %"     "versicolor : 0.33 %"
[3] "virginica : 0.33 %"
```

où la fonction `paste` permet de concaténer des chaînes de caractère et la fonction `round` permet de faire des arrondis en précisant comme première argument le scalaire ou le vecteur à arrondir et comme deuxième argument le nombre de décimales à garder.

Q2 : représenter le diagramme circulaire de la variable `transport` en utilisant les options `col` et `labels`.

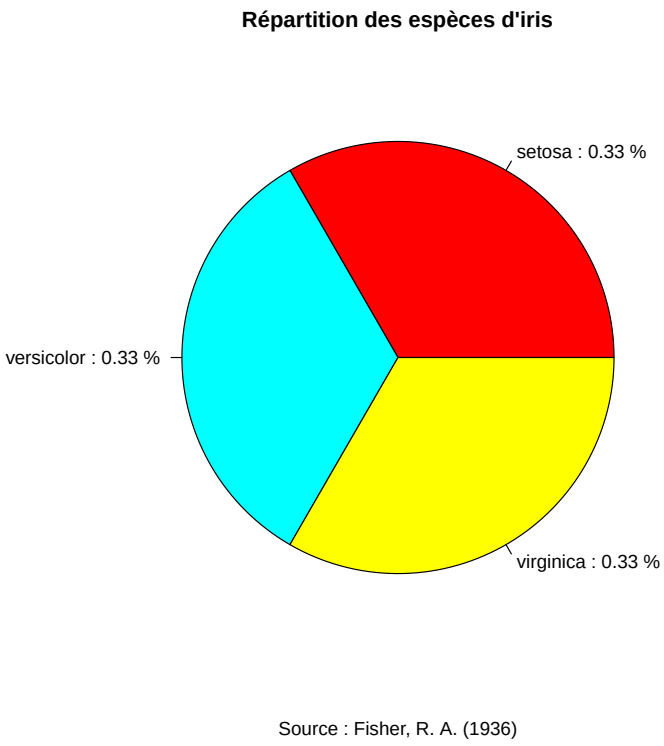

FIG. 4.2 – Exemple de diagramme en secteurs en utilisant des options

Diagramme en tuyaux d'orgue

On utilisera la fonction `barplot` avec le résultat de la fonction `table` exprimé en effectif ou bien en fréquence. Par exemple :

```
> barplot(res.tab)
```

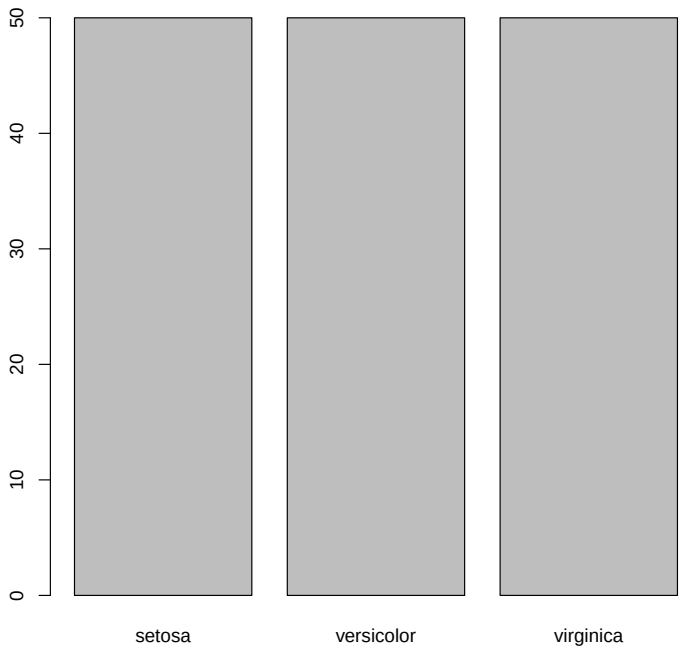

FIG. 4.3 – Exemple de diagramme en tuyaux d'orgue sans options

Il existe de nombreuses options (voir `help(barplot)`). Les plus utiles ont déjà été vues comme `col`, `main`, `xlab`, `ylab`, mais on pourra également utilisé :

- `names.arg`, pour changer le nom des modalités
- `horiz`, `FALSE` par défaut et `TRUE` pour changer l'orientation des barres.

Par exemple :

```
> barplot(res.tab/n, horiz = TRUE, xlab = "Fréquence",
+     ylab = "modalités observées", col = colors()[100:102],
+     names.arg = c("espèce 1", "espèce 2", "espèce 3"),
+     main = "Répartition des espèces d'iris")
```

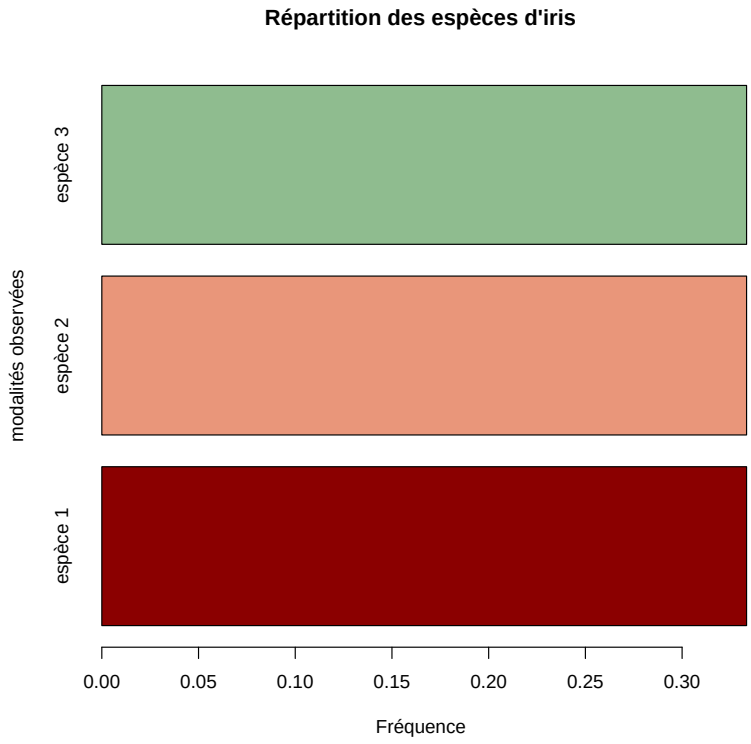

FIG. 4.4 – Exemple de diagramme en tuyaux d'orgue en utilisant des options

Compléments sur le choix des couleurs : on a utilisé ici la fonction `colors()` qui renvoie une liste de 657 couleurs possibles parmi lesquelles : 'green', 'purple', 'pink', ... Au lieu de donner un vecteur de numéro dans l'option `col`, on va donner un vecteur de `character` avec les couleurs choisies.

Compléments sur les options : essayer les différents exemples proposés dans l'aide de la fonction `pie` et `barplot`. Certaines options permettent d'utiliser des dégradés de couleur grisâtre ou d'utiliser des hachures avec différentes inclinaisons. Cela peut être utile pour différencier les classes lorsqu'on imprime en noir et blanc...

Q3 : représenter le diagramme en tuyaux d'orgue de la variable `gout.transport` en utilisant des options vues ci-dessus.

4.2.3 Commentaires à faire sur les tableaux et graphiques

Dans un rapport, il est d'usage de commenter une variable qualitative en exposant les chiffres et pourcentages "bruts" et de comparer les modalités une à une. Par exemple, on dira que telle modalité est deux fois plus présente qu'une autre. Par ailleurs, on s'interesse aux évenements qui sont rares ou abondants : telle modalité est sur-représentée alors qu'une autre est sous-représentée. Attention à ce que l'analyse de graphiques n'aboutissent pas à des commentaires tels que : "on constate ...., c'est parce que ...". Il faut tempérer ce genre de commentaires par : "on constate que ... ; il se pourrait que cela s'explique par..."

Q4 : Commenter le graphique que vous avez réalisé dans la Q12.

4.3 Les variables quantitatives discrètes

4.3.1 Diagramme en bâtons

Pour repésenter un diagramme en bâtons, il suffit d'appliquer la fonction `plot` au résultat de la fonction `table` exprimé en effectif ou bien en fréquence relative. On verra que la fonction `plot` est utilisée pour de nombreux graphiques en utilisant des options spécifiques à chacun de ces graphiques.

```
> plot(table(groupe.1$age), main = "")
```

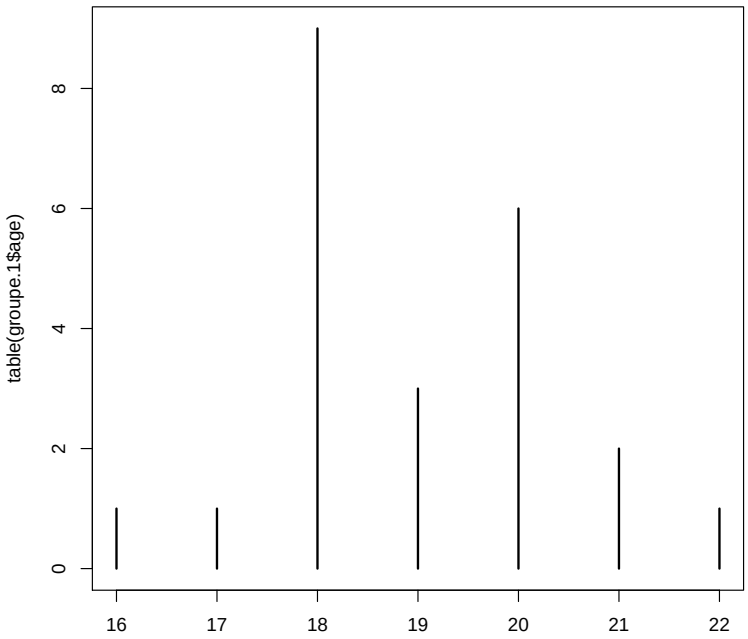

FIG. 4.5 – Exemple de diagramme en barres sans options

Q5 : Dessiner le diagramme en bâtons de la variable `age` inclus dans le `data.frame` `promo.09`. Vous pourrez utiliser les options `col`, `main`, `xlab` et `ylab` vues précedemment.

4.3.2 Fonction de répartition empirique

Sur R, on trace une fonction de répartition empirique en appliquant à la fonction `plot` le résultat de la fonction `ecdf` (empirical cummulative distribution function), qui prend comme argument d'entrée un vecteur de type `numeric`. Par exemple :

```
> res.ecdf <- ecdf(promo.09$age)
> plot(res.ecdf, main = "Fonction de répartition empirique de la variable age",
+     col = "royalblue")
> abline(h = 0.5)
```

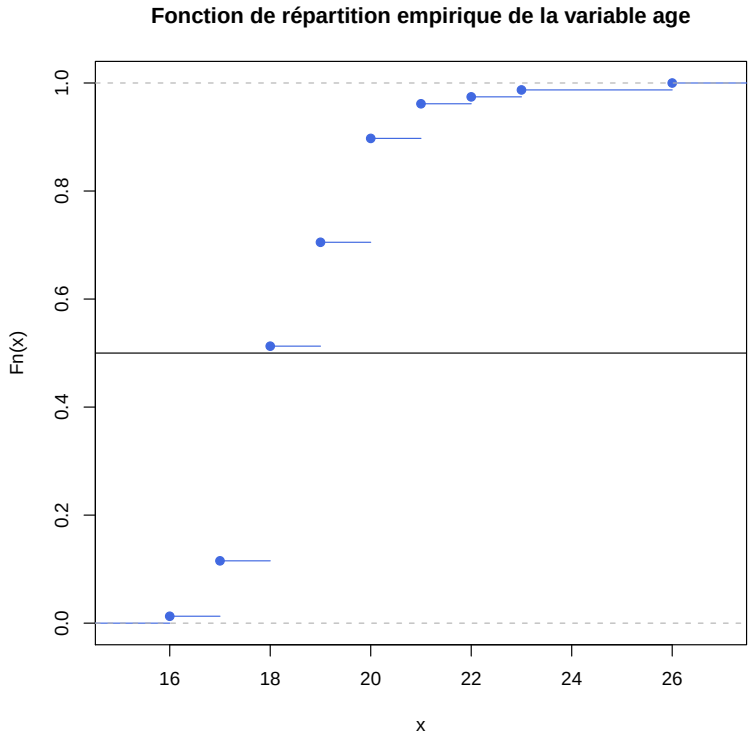

FIG. 4.6 – Fonction de répartition empirique

Q6 : à l'aide de la fonction de répartition empirique ci-dessus, donner la valeur de la médiane de la variable `age` du `data.frame promo.09`.

Q7 : écrire un vecteur `age2` qui contient les valeurs (16, 17, 18, 19, 20, 21, 22, 23, 24, 25). Représenter la fonction de répartition empirique de cette variable. Quelle est la valeur médiane de cette variable ? Quelle différence constatez-vous entre l'allure de la fonction de répartition empirique de `age2` et celle de `age` du `data.frame promo.09`.

4.4 Les variables quantitatives continues

4.4.1 L'histogramme

On peut représenter simplement un histogramme en utilisant la fonction `hist`. A l'aide d'un algorithme, R choisit le nombre de barres qu'il va représenter ainsi que l'amplitude des barres qui seront par défaut toutes égales. De plus, il donne par défaut en ordonnées les effectifs ou fréquences absolues. Pour représenter un histogramme et avoir le tableau-type tel qu'il a été présenté en cours, il suffit de faire :

```
> hist.sepal.length <- hist(iris$Sepal.Length)

> hist.sepal.length

$breaks
[1] 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0

$counts
[1]  5 27 27 30 31 18  6  6

$intensities
[1] 0.06666665 0.36000000 0.36000000 0.40000000 0.41333333
[6] 0.24000000 0.08000000 0.08000000

$density
[1] 0.06666665 0.36000000 0.36000000 0.40000000 0.41333333
[6] 0.24000000 0.08000000 0.08000000

$mids
[1] 4.25 4.75 5.25 5.75 6.25 6.75 7.25 7.75

$xname
```

```
[1] "iris$Sepal.Length"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
```

L'objet retourné donne les informations suivantes : `breaks` donne les coordonnées des barres, `counts` (qui contient un élément de moins que `breaks`) contient les effectifs observés à l'intérieur des classes d'intervalles. `intensities` et `density` contiennent la même information et donnent la densité de proportion (cf. cours). `mids` donne le barycentre de chaque barre créée.

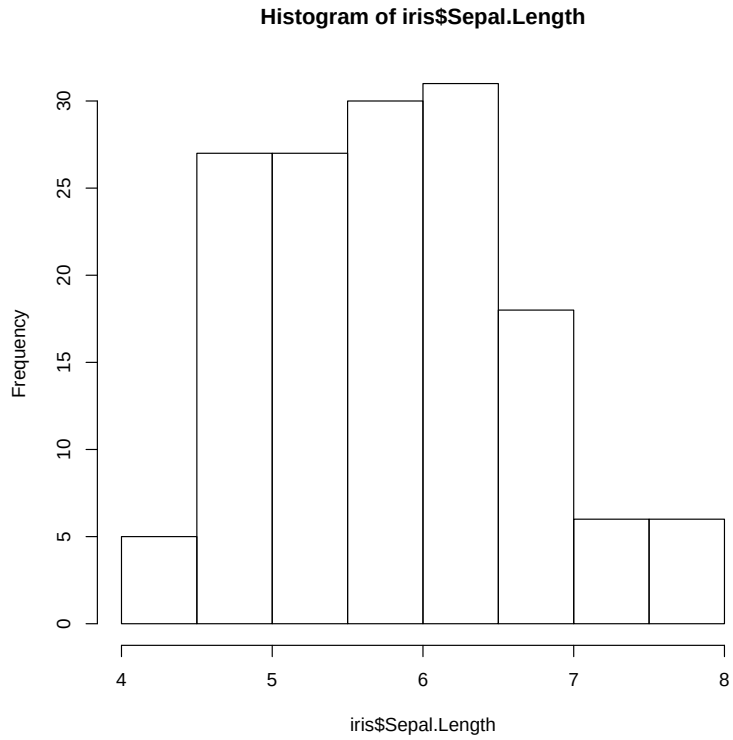

FIG. 4.7 – Exemple d'histogramme sans options

Pour la fonction `hist`, une option utile est l'option `freq=FALSE` pour représenter en ordonnée la densité de proportion. On utilise les mêmes options vues précemment pour rendre plus agréable la lecture du graphique. Par exemple :

```
> hist(iris$Sepal.Length, freq = FALSE, col = "royalblue",
+     xlab = "Taille du sépale", ylab = "densité de proportion",
+     main = "")
```

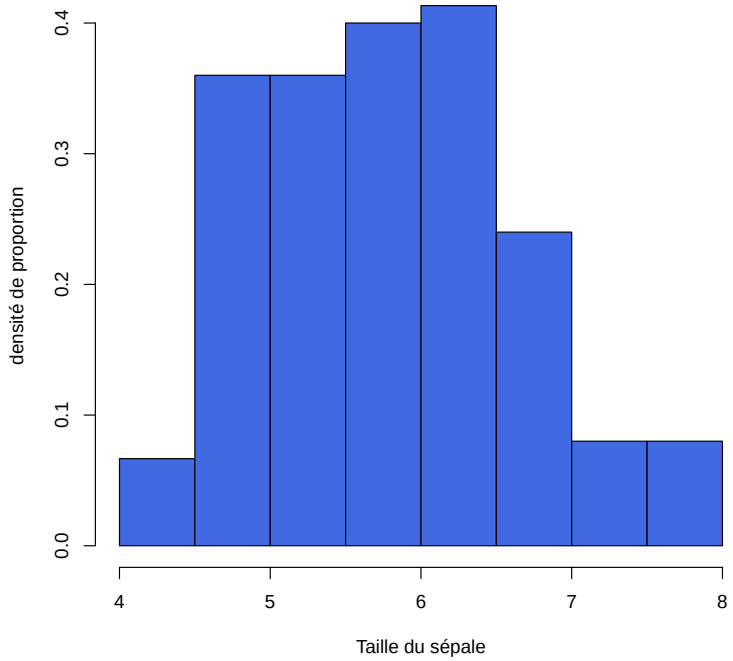

FIG. 4.8 – Exemple d'histogramme avec options

Q8 : dessiner l'histogramme de la variable `temps.transport` de l'objet `promo.09` en représentant la densité de proportion en ordonnée. On utilisera également les options de base `col`,... En considérant les classes proposées par R, quel est le mode de cette variable ?

4.4.2 Résumé d'une variable quantitative

Les fonctions suivantes donnent des statistiques de base sur une variable :
- `min` : renvoie le minimum d'un vecteur
- `median` : renvoie la valeur médiane d'un vecteur. Elle correspond à la $\frac{n+1}{2}$-ième observation triée si n est impair et égal au barycentre de la $\frac{n}{2}$-ième et $\frac{n}{2}+1$-ième observation si n est pair.

- `quantile` : renvoie la valeur des quantiles d'ordre $0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1$ d'un vecteur. Il existe plusieurs façons de calculer des valeurs quantiles (voir `help(quantile)`). Pour obtenir celle proposée en cours dans le cas discret, il suffit d'utiliser l'option `type=1` alors que pour celle utilisée dans le cas continu, il faut utiliser l'option `type=4`.
- `mean` : renvoie la valeur moyenne d'un vecteur de valeurs `numeric`
- `max` : renvoie la valeur maximum d'un vecteur de valeurs `numeric`

```
> a <- c(10, 11, 11.5, 12, 13, 15, 15.5, 16, 16, 17)
> min(a)

[1] 10

> median(a)

[1] 14

> mean(a)

[1] 13.7

> quantile(a, type = 1)

  0%  25%  50%  75% 100%
10.0 11.5 13.0 16.0 17.0

> quantile(a, type = 4)

   0%   25%   50%   75%  100%
10.00 11.25 13.00 15.75 17.00

> max(a)

[1] 17

> range(a)

[1] 10 17
```

Q9 : en utilisant les fonctions `sum` et `mean`, calculer le terme $var1 = \frac{1}{n}\sum_i^n (a_i - \bar{a})^2$. Calculer ensuite $var2 = \frac{n}{n-1} var1$. Enfin, calculer la variance de `a` en utilisant la fonction `var`. Que constatez-vous ?

Les fonctions suivantes calculent les paramètres de dispersion :

- `range` : renvoie l'étendue d'un vecteur (max(x)-min(x))
- `var` : renvoie la variance corrigée
- `sd` (pour "standard deviation") : renvoie l'écart-type corrigée

Q10 : en utilisant les fonctions `mean` et `sd`, calculer le vecteur `age.cr` correspondant à la variable `age` du `data.frame promo.09` centrée et réduite ?

4.4.3 Boîte à moustache

Pour représenter la boîte à moustache d'une variable quantitative, on utilise la fonction `boxplot`. Cette fonction possède les mêmes options que les fonctions précédentes (`main`, `col`, `xlab`, ...). A noter les options :

- `xlim` : par défaut, `c(min(x),max(x))` (bornes de l'axe des abscisses)
- `ylim` : par défaut, `c(min(y),max(y))` (bornes de l'axe des ordonnées)

```
> boxplot(iris$Sepal.Length, ylab = "Taille du sépale",
+     col = "purple", ylim = c(3, 9))
> res.quantile <- quantile(iris$Sepal.Length)
> min.x <- res.quantile[1]
> Q1 <- res.quantile[2]
> Q2 <- res.quantile[3]
> Q3 <- res.quantile[4]
> max.x <- res.quantile[5]
> text(0.67, c(Q1, Q2, Q3), c("Quartile Q1", "Médiane",
+     "Quartile Q3"), cex = 0.9, col = "royalblue")
> text(1.33, c(min.x, max.x), c("Max(Q1-1.5(Q3-Q1) , min(x))",
+     "Min(Q3+1.5(Q3-Q1), max(x))"), cex = 0.9, col = "royalblue")
> points(c(1, 1), c(Q1 - 1.5 * (Q3 - Q1), Q3 + 1.5 *
+     (Q3 - Q1)), pch = c(2, 6), col = "red")
> text(1, c(Q1 - 1.5 * (Q3 - Q1), Q3 + 1.5 * (Q3 -
+     Q1)) + 0.2, c("Q1-1.5(Q3-Q1)", "Q3+1.5(Q3-Q1)"),
+     col = "red", cex = 0.8)
```

Q11 : représenter la boîte à moustache de la variable `temps.transport` du `data.frame promo.09` et en vous inspirant du graphique ci-dessus, préciser quelles sont les valeurs correspondant aux moustaches.

Présentation des résultats dans un rapport

Vous pourrez recopier vos résultats dans un tableau de la façon suivante et mettre vos graphiques à côté du tableau.

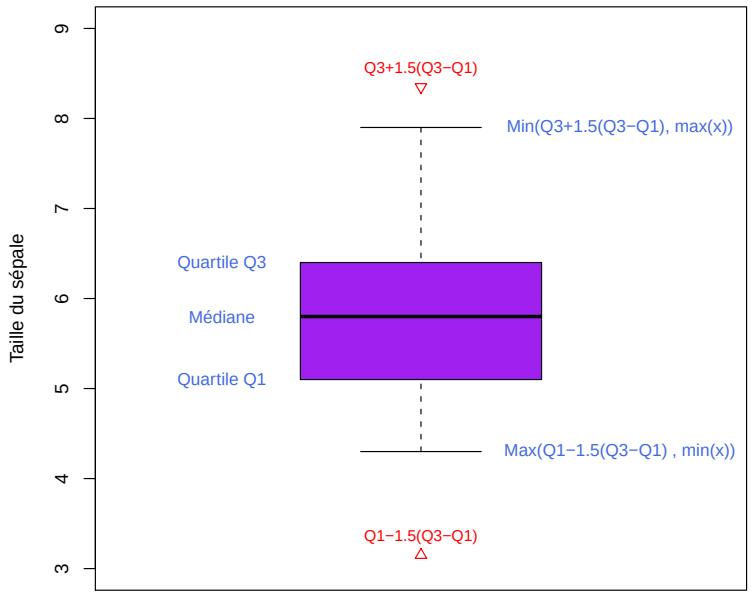

FIG. 4.9 – Exemple de boîte à moustaches avec options

Description de la variable `a`

Statistiques	Valeurs
Minimum	10
1er quartile	11.5
Moyenne	13.7
Médiane	13
3ème quartile	15.75
Maximum	17
Etendue	7
Variance	6.18
Ecart-type	2.49
Effectifs	10

4.4.4 Commentaires à faire sur les résumés et les graphiques

Dans un rapport, il est d'usage de commenter une variable quantitative de la façon suivante : "Sur l'échantillon observée de taille n, le temps moyen mis pour se rendre à l'université est ... La moyenne quadratique des écarts à la moyenne (écart-type) vaut ... 75% des étudiants mettent moins de ... pour venir, 50 % moins de ..., et 25% moins de ... On observe certaines valeurs atypiques, c'est-à-dire des gens qui mettent plus de Q3 + 1.5 (Q3-Q1) (respectivement moins de Q1-1.5(Q3-Q1)) pour venir. Il est intéressant de constater que ces gens ... (on regarde alors plus précisément qui sont ces individus et ce qu'on observe éventuellement sur les autres variables : par exemple, viennent-ils à voiture, à vélo, ...). La différence entre celui qui met le plus de temps et celui qui en met le moins vaut ... "

Q12 : à partir du résumé et des graphiques réalisés sur la variable `temps.transport` faites vos commentaires sur cette variable.

Chapitre 5

Etude multivariée

5.1 Introduction

Dans ce TP, on s'intéresse à l'analyse descriptive de données bivariées. On verra quels tableaux et quels graphiques utilisés lorsqu'on étudie simultanément :

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

Prérequis : avant de commencer ce TP, vous compilerez le code correspondant à la feuille de TP 3, téléchargeable sur Moodle voir `http://cours.univ-tlse1.fr`

5.2 Liaison entre deux variables quantitatives

5.2.1 Le nuage de points

Le graphique utilisé pour représenter simultanément deux variables quantitatives est le nuage de points. On le crée avec la fonction `plot` en indiquant comme premier argument la variable en abscisse et comme second argument, la variable en ordonnée. On peut utiliser les options qui ont été vues dans le précédent TP et qui sont résumées dans ce TP en annexe. Par exemple, on représente ci-dessous le nuage de points de la variable `Petal.Width` en fonction de la variable `Sepal.Length`, où ces deux variables appartiennent au `data.frame` (jeu de données) `iris` :

```
> plot(iris$Sepal.Length, iris$Petal.Width, xlab = "taille du sépale",
+     ylab = "taille du pétale")
> title(main = "Nuage de points")
```

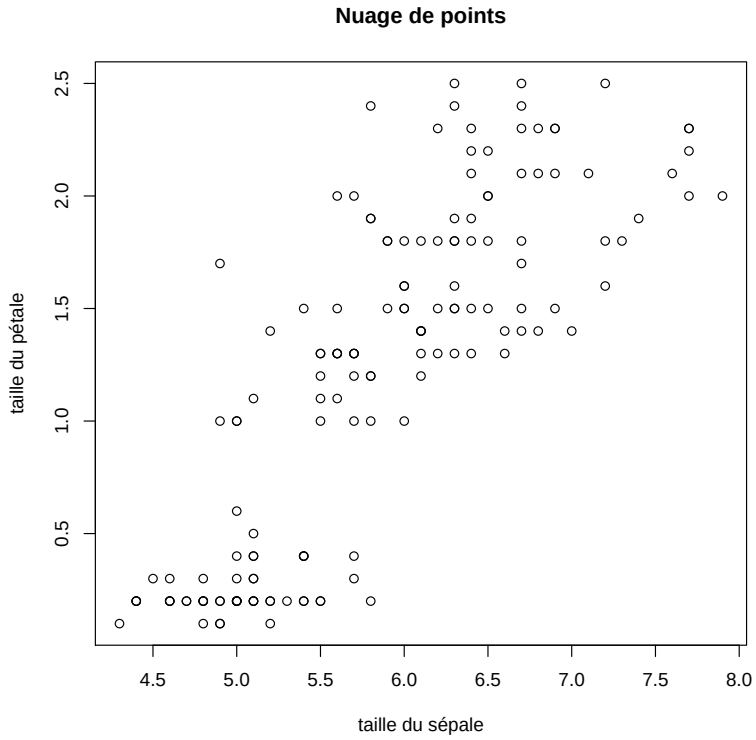

FIG. 5.1 – Premier exemple de nuage de points

On peut ajouter un diagramme de dispersion unidimensionnel dans les marges avec la fonction `rug`. On notera que cette fonction peut s'utiliser après avoir représenté un histogramme ou une boîte à moustache. Dans l'exemple suivant, on a ajouté une "perturbation" (avec la fonction `jitter`name=jitter, description=perturbe les données (utiles pour des données répétées)), i.e. qu'on a ajouté aux valeurs observées de faibles valeurs (positives ou négatives), afin que les valeurs répétées puissent être repérées dans les marges.

```
> plot(iris$Sepal.Length, iris$Petal.Width, xlab = "taille du sépale",
+     ylab = "taille du pétale", col = "darkblue")
> title(main = "Nuage de points")
> rug(jitter(iris$Sepal.Length, 0.75), side = 1, col = "royalblue")
> rug(jitter(iris$Petal.Width, 0.8), side = 2, col = "royalblue")
```

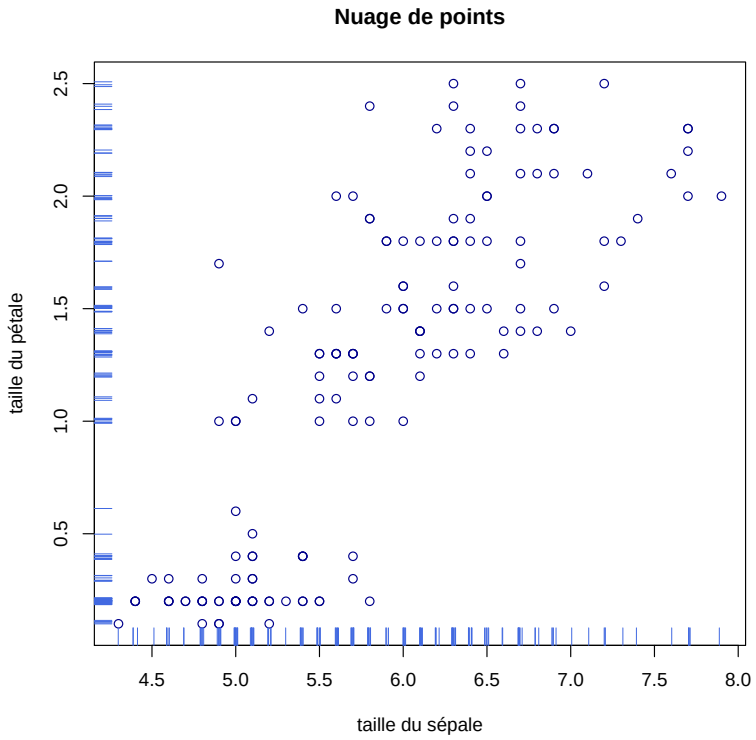

FIG. 5.2 – Second exemple de nuage de points

Enfin, en utilisant le code suivant, on peut représenter sur le nuage de points la droite d'équation $y = a + bx$ en utilisant la fonction `abline(a = ..., b = ...)`[1]. De plus, nous

[1]On notera que la fonction `abline` ne marche que si la fonction `plot` a été appelée au préalable.

avons ajouté en-dessous de l'axe des abscisses, la boîte à moustache de la variable X et à gauche de l'axe des ordonnées la boîte à moustache de la variable Y.

```
> op <- par()
> layout(matrix(c(2, 1, 0, 3), 2, 2, byrow = T), c(1,
+     6), c(4, 1))
> par(mar = c(1, 1, 5, 2))
> plot(iris$Sepal.Length, iris$Petal.Width, xlab = "taille du sépale",
+     ylab = "taille du pétale", col = "darkblue")
> abline(a = -3.2, b = 0.75, col = "red", lty = 2)
> title(main = "Nuage de points")
> legend("topleft", legend = "droite de régression linéaire",
+     col = "red", lty = 2)
> rug(jitter(iris$Sepal.Length, 0.75), side = 1, col = "royalblue")
> rug(jitter(iris$Petal.Width, 0.8), side = 2, col = "royalblue")
> par(mar = c(1, 2, 5, 1))
> boxplot(iris$Sepal.Length, axes = F, col = "lightblue")
> title(ylab = "Taille des sépales", line = 0)
> par(mar = c(5, 1, 1, 2))
> boxplot(iris$Petal.Width, horizontal = T, axes = F,
+     col = "lightblue")
> title(xlab = "Taille des pétales", line = 1)
> par(op)
```

Q1 : dans le jeu de données `promo.09`, on ne dispose que d'une seule variable quantitative (`temps.transport`). Nous allons construire la variable `promo.09$cout.transport` par simulation, c'est-à-dire que nous allons construire nous-mêmes cette variable, en supposant que celle-ci est proportionnelle à 1.5 fois la variable `temps.transport`, plus un terme d'erreur. Autrement dit :

$$cout.transport = 1.5 \times temps.transport + \epsilon$$

Vous construirez d'abord la variable `erreur` (correspondant à ϵ), en lui affectant le code suivant : `rnorm(n,0,5)`. On remplacera n par la taille de l'échantillon. Vérifier ensuite et en utilisant les fonctions `var` et `cov` que :

$$var(aX + Y) = a^2 \times var(X) + var(Y) + 2 \times a \times cov(X, Y)$$

On remplacera bien évidemment dans la formule ci-dessus $aX+Y$ par `promo.09$cout.transport`, X par `promo.09$temps.transport` et Y par `erreur`.

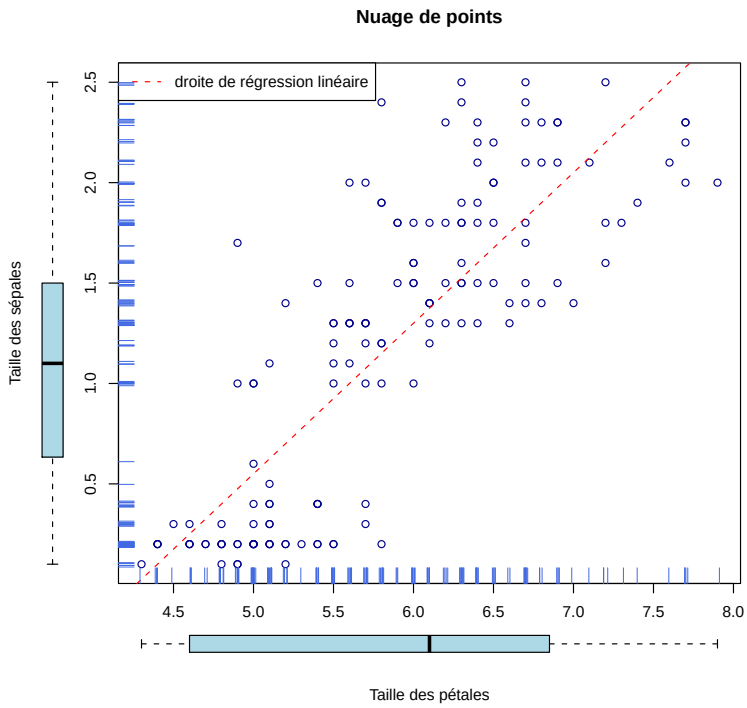

FIG. 5.3 – Troisième exemple de nuage de points

Q2 : en utilisant le code du troisième exemple, tracer le nuage de points de la variable `cout.transport` en fonction de `temps.transport`. Pour la fonction `abline`, on prendra les paramètres exacts de la droite d'équation linéaire, à savoir $a = 0$ et $b = 1.5$.

5.2.2 Coefficients de corrélation linéaire

On rappelle que :

$$r(x, y) = \frac{cov(x, y)}{\sqrt{var(x)}\sqrt{var(y)}}$$

Ce coefficient s'obtient avec la fonction `cor` et est compris entre -1 et 1. On a les trois possibilités suivantes :

- quand r est proche de 1, cela implique que plus x a des valeurs fortes, plus y a des valeurs fortes et que ce lien est linéaire.
- quand r est proche de -1, cela implique que plus x a des valeurs fortes, plus y a des valeurs faibles et ce lien est linéaire.
- quand r est proche de 0, cela implique qu'il n'y a aucun lien linéaire entre x et y

Q3 : Construire la variable[2] $x = (0.05,\ 0.1,\ 0.15,\ ...,\ 1)$ de taille 20 , puis la variable $y = 1/4 \times x^4 - x^3/6 - 5 \times x^2/36 + x/9$. Représenter le nuage de points de y en fonction de x en utilisant la fonction `plot`. Quelle type de relation y-a-t-il entre x et y ? Calculer ensuite le coefficient de corrélation linéaire entre les variables x et y en utilisant la fonction `cor`. Que constatez-vous ?

5.2.3 Régression linéaire

Soit la droite d'équation :

$$y = a + bx$$

On estime la pente de régression b et la constante a par :

$$\hat{b} = \frac{cov(x, y)}{var(x)}$$
$$\hat{a} = \bar{y} - \hat{b}\bar{x}$$

Q4 : en utilisant les fonctions `cov` et `var`, affecter à l'objet `hat.b` la valeur de la pente de régression de la variable `temps.transport` sur `cout.transport`. Affecter ensuite à l'objet `a.hat` l'estimation de la constante a. Comparer les valeurs $\hat{a}$ et $\hat{b}$ aux valeurs a et b exacts.

[2] On pourra utiliser la fonction `seq(from = ..., to = ..., by = ...)`, `from` indique la première valeur du vecteur, `to` indique la dernière valeur du vecteur et `by` indique la longueur de l'intervalle de discrétisation.

Remarque : dans la réalité, `a` et `b` sont inconnus. C'est pourquoi on représente dans les nuages de points vus ci-dessus la droite d'équation linéaire $y = a + bx$ en utilisant $a = \hat{a}$ et $b = \hat{b}$.

5.2.4 Commentaires sur l'étude de 2 variables quantitatives

Voici un exemple de commentaires à faire : "Le nuage de points semble indiquer que plus la variable x augmente, plus la variable y augmente (ou diminue). Par ailleurs, ce lien semble linéaire (si ce n'est pas le cas, préciser s'il existe un lien quadratique, polynomiale, exponentielle, ...). Ce constat se vérifie (ou non) par la valeur du coefficient de corrélation linéaire qui est proche de ... Si on ajuste les données observées par un modèle de régression linéaire, alors, une augmentation de 1 unité de x entraînerait une augmentation (ou diminution) de y d'une valeur de b."

5.3 Liaison entre deux variables qualitatives

5.3.1 Table de contingence

Soient X et Y deux variables qualitatives. Pour créer la table de contingence de X et Y, on utilise la fonction `table(X,Y)`. Pour obtenir les effectifs totaux, on utilise la fonction `addmargins` appliquée à un objet `table`. Prenons pour exemple les variables `sexe` et `tabac` du `data.frame promo.09` :

```
> tab.bi <- table(promo.09$sexe, promo.09$tabac, dnn = c("sexe",
+     "tabac"))
> dimnames(tab.bi)[[1]] <- c("feminin", "masculin")
> dimnames(tab.bi)[[2]] <- c("non fumeur", "fumeur")
> addmargins(tab.bi)
```

```
          tabac
sexe       non fumeur fumeur Sum
  feminin          22      8  30
  masculin         29     19  48
  Sum              51     27  78
```

Présentation des résultats dans un rapport
Vous devrez recopier vos résultats dans un tableau de la façon suivante.

Q5 : représenter la table de contingence des variables `gout.transport` et `transport`.

	Tabac		
Sexe	non fumeur	fumeur	Total
feminin	22	8	30
masculin	29	19	48
Total	51	27	78

TAB. 5.1 – Table de contingence

5.3.2 Distribution marginale

La distribution marginale revient à appliquer la fonction `table` sur chaque variable prise séparément, comme cela a été fait dans le TP précédent. Ces distributions marginales pourront être comparées aux profils-lignes et profils-colonnes pour vérifier si les variables X et Y sont indépendantes ou non.

5.3.3 Distribution conditionnelle

Profils-lignes

On obtient les profils-lignes en divisant les cellules de la table de contingence par les totaux obtenus dans la colonne "Total". Ainsi la somme de chque ligne est égale à 1. Voici le code à utiliser :

```
> profil.ligne = round(sweep(addmargins(tab.bi, 2,
+     list(All = sum)), 1, apply(tab.bi, 1, sum)/100,
+     "/"), 1)
> profil.ligne

          tabac
sexe       non fumeur fumeur   All
  feminin        73.3   26.7 100.0
  masculin       60.4   39.6 100.0
```

On représente bien entendu les résultats dans un tableau :

Commentaires : on dira qu'à telle modalité de X fixée, on observe telle répartition des modalités de Y. En comparant les profils lignes avec la distribution marginale de Y, cela donne une idée sur l'indépendance ou non des variables X et Y.

	tabac		
sexe	non fumeur	fumeur	Total
feminin	73.30	26.70	100
masculin	60.40	39.60	100

TAB. 5.2 – Profils-lignes en pourcentage

Profils-colonnes

On obtient les profils-colonnes en divisant les cellules de la table de contingence par les totaux obtenus dans la ligne "Total". Ainsi, la somme de chaque colonne est égale à 1. Voici le code à utiliser pour calculer les profils-colonnes :

```
> profil.colonne <- round(sweep(addmargins(tab.bi,
+     1, list(All = sum)), 2, apply(tab.bi, 2, sum)/100,
+     "/"), 1)
> profil.colonne

          tabac
sexe       non fumeur fumeur
  feminin        43.1   29.6
  masculin       56.9   70.4
  All           100.0  100.0
```

On présentera les profils-colonnes dans un tableau de la façon suivante :

	tabac	
Sexe	non fumeur	fumeur
feminin	43.10	29.60
masculin	56.90	70.40
Total	100	100

TAB. 5.3 – Profils-colonnes en pourcentages

Commentaires : on dira qu'à telle modalité de Y fixée, on observe telle répartition des modalités de X. En comparant les profils-colonnes avec la distribution marginale de X, cela donne une idée sur l'indépendance ou non des variables X et Y.

Q6 : calculer les profils-lignes et les profils-colonnes des variables `gout.transport` et `transport`.

5.3.4 Représentation graphique

Distribution jointe : premier type de graphique

Voici un premier type de représentation pour deux variables qualitatives en utilisant la fonction `plot` appliquée à un objet de type `table` bidimensionnel. On constate que la largeur des colonnes est proportionnelle à la répartition des modalités de la variable X. On observe dans chaque colonne la répartition des modalités de la variable Y. A noter qu'on peut inverser lignes et colonnes en remplaçant `tab.bi` par `t(tab.bi)` dans le code suivant :

```
> plot(tab.bi, main = "Distibution jointe", color = TRUE)
```

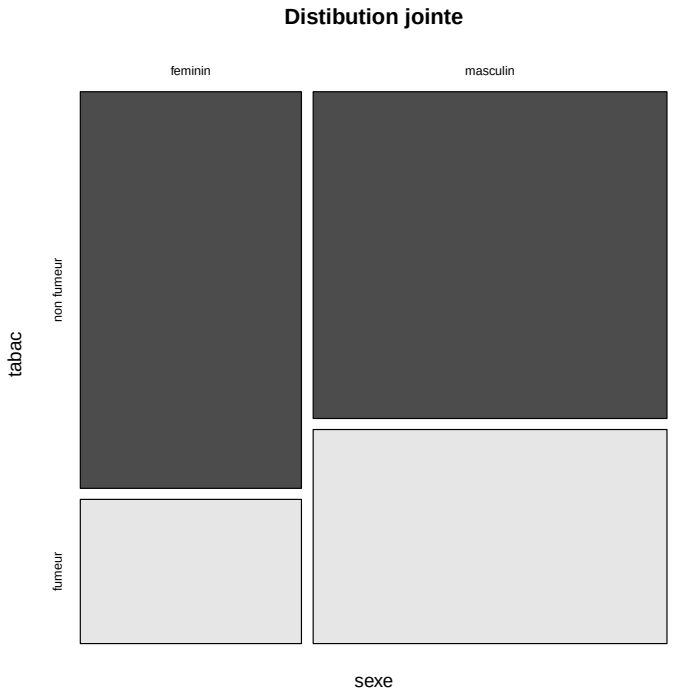

FIG. 5.4 – Distribution jointe

Commentaires : dans ce graphique, on s'interesse d'une part à l'"aire" des rectangles pour vérifier s'il n'y a pas des cases sur-représentées (resp. sous-représentées). D'autre part, on peut vérifier si les modalités de Y sont réparties de la même façon, selon les modalités de X. Pour cela, on regarde si les barres horizontales sont toutes à la même hauteur.

Distribution jointe : second type de graphique

Dans le graphique suivant, on étudie la répartition de la variable `sexe` par modalité de la variable `tabac`. La fonction utilisée est la fonction `barplot` appliquée à un objet de type `table`.

```
> barplot(tab.bi, legend.text = c("feminin", "masculin"),
+     col = c("pink", "lightblue"))
> title(main = "Répartition du sexe en fonction des fumeurs")
```

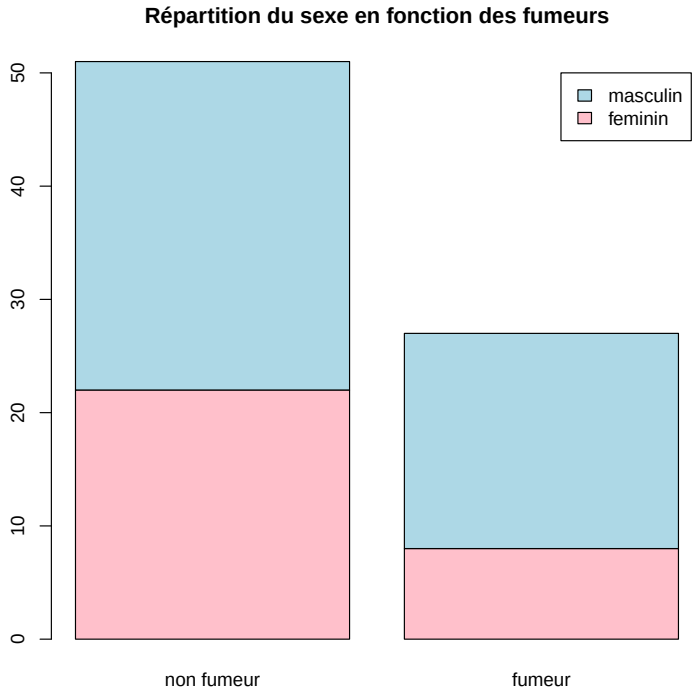

FIG. 5.5 – Exemple de diagramme en tuyaux d'orgue bidimensionnel

On peut présenter le graphique précédent d'une autre manière en utilisant l'option `beside=TRUE`. Les modalités de la variable Y sont représentées les unes à côté des autres. De plus, dans l'exemple suivant, nous avons inversé l'ordre des variables (i.e. répartition de la variable `tabac` par modalité de la variable `sexe`), en utilisant la fonction `t()` appliquée à `tab.bi`.

```
> barplot(t(tab.bi), legend.text = c("non fumeur",
+     "fumeur"), col = c("white", "grey"), beside = TRUE)
> title(main = "Répartition des fumeurs en fonction du sexe")
```

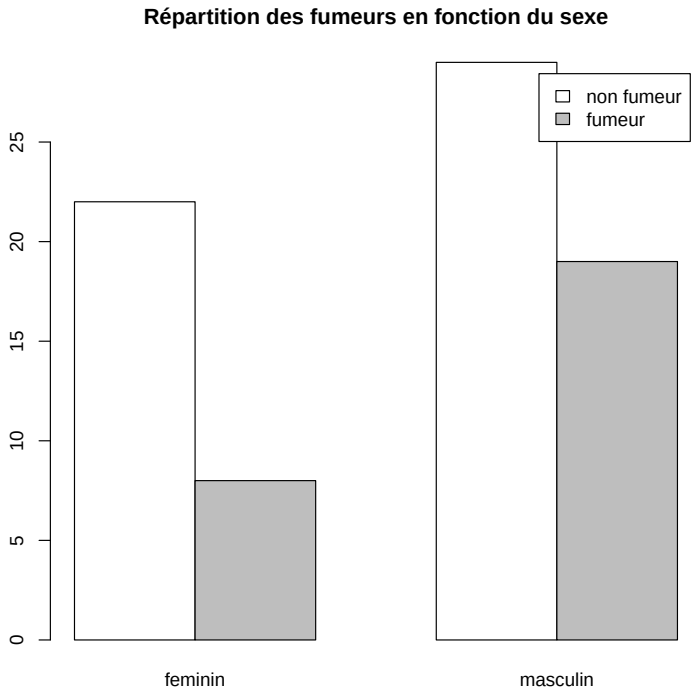

FIG. 5.6 – Exemple de diagramme en tuyaux d'orgue bidimensionnel

Q7 : représenter les deux types de graphiques vus ci-dessus pour visualiser la distribution jointe des variables `gout.transport` en fonction de `transport`.

Distibution conditionnelle

Pour représenter les distributions conditionnelles, on représente les mêmes graphiques que précédemment à partir des profils-lignes (en enlevant la colonne des totaux) ou profils-colonnes (en enlevant la ligne des totaux). Par exemple, on représente graphiquement les profils-lignes avec le code suivant :

```
> barplot(t(profil.ligne[, c(1, 2)]), legend.text = c("non fumeur",
+     "fumeur"), col = c("royalblue", "purple"))
> title(main = "Distribution conditionnelle du tabac en fonction du sexe")
```

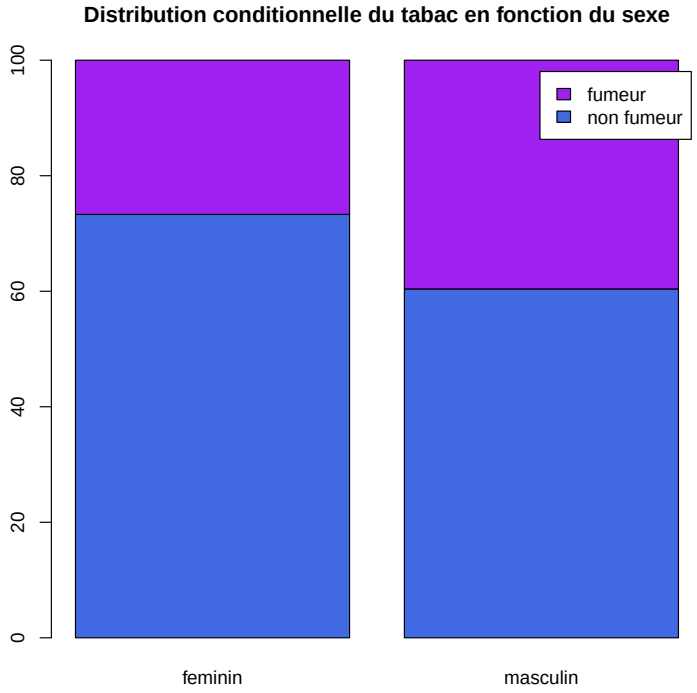

FIG. 5.7 – Représentation graphique des profils-lignes

Pour représenter graphiquement les profils-colonnes, on procède ainsi :

```
> barplot(profil.colonne[c(1, 2), ], legend.text = c("feminin",
+     "masculin"), col = c("pink", "lightblue"))
> title(main = "Répartition du sexe en fonction des fumeurs")
```

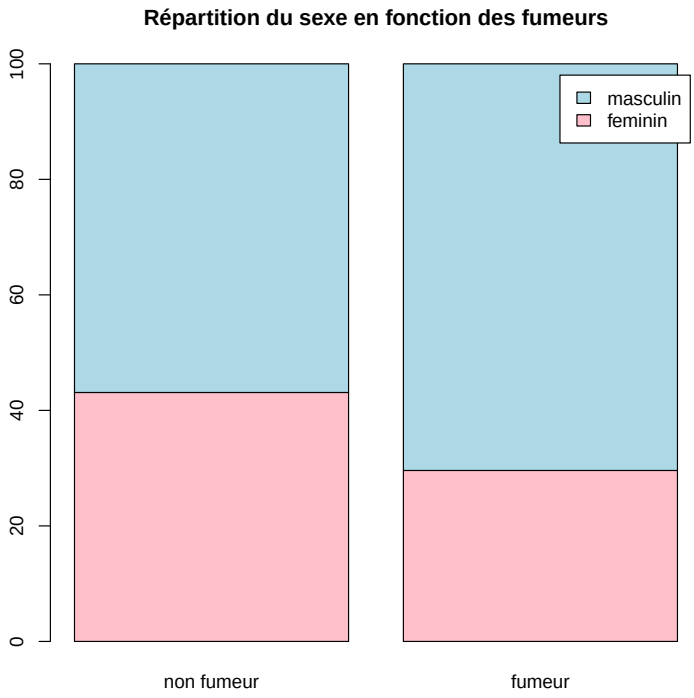

FIG. 5.8 – Représentation graphique des profils-colonnes

Q8 : à partir des profils-lignes et profils-colonnes des variables `gout.transport` et `transport`, représenter graphiquement les distributions conditionnelles.

5.3.5 Mesure de liaison

La distance du χ^2

On obtient la distance du χ^2 en utilisant la fonction `chisq.test` appliquée à un objet de type `table`. Par exemple :

```
> b <- chisq.test(tab.bi, correct = FALSE)
> b

        Pearson's Chi-squared test

data:  tab.bi
X-squared = 1.3609, df = 1, p-value = 0.2434
```

Commentaires : la distance du χ^2 est égale ici à 1.3609. On dira que les variables sont indépendantes si la p-value est supérieure à 0.05. On peut vérifier la valeur de la distance du χ^2 à partir de la table de contingence. On obtient les valeurs exactes avec `b$observed` et les valeurs attendues avec `b$expected`. Du coup, on obtient les contributions au χ^2 de la façon suivante :

```
> b$observed

          tabac
sexe       non fumeur fumeur
  feminin          22      8
  masculin         29     19

> b$expected

          tabac
sexe       non fumeur   fumeur
  feminin    19.61538 10.38462
  masculin   31.38462 16.61538

> (b$observed - b$expected)^2/b$expected

          tabac
sexe       non fumeur    fumeur
  feminin   0.2898944 0.5475783
  masculin  0.1811840 0.3422365
```

Q9 : calculer la distance du χ^2 entre les variables `gout.transport` et `transport`. Est-ce que ces variables sont indépendantes ?

5.4 Laison entre une variable quantitative et une variable qualitative

5.4.1 Boîtes à moustache parallèle

Pour visualiser le lien entre une variable quantitative et une variable qualitative, on utilise les boîtes à moustaches parallèles qui s'obtiennent avec la fonction `boxplot`. L'option `varwidth=TRUE` permet de représenter des épaisseurs de boîtes différentes selon la taille de chaque groupe. Dans l'exemple des données `iris`, comme les classes ont les mêmes effectifs, l'épaisseur des boîtes est la même pour chaque classe.

```
> boxplot(Petal.Length ~ Species, data = iris, varwidth = TRUE,
+     names = c("espèce setosa", "espèce versicolor",
+         "espèce virginica"), col = c("purple", "pink",
+         "lightblue"))
> title(main = "Boîte à moustache parallèle")
```

Commentaires : l'analyse des boîtes à moustaches parallèles permet de comparer la distribution de la variable Y entre les classes. Si une boîte n'est pas au même "niveau" que les autres, cela implique que la variable Y se comporte différemment (valeurs plus fortes ou valeurs plus faibles selon la position de la boîte) lorsque les individus appartiennent à cette classe. Par exemple, sur les données `iris`, on constate que tous les individus appartenant à l'espèce 1 ont des valeurs plus faibles que celles des autres classes. La majorité des individus appartenant à l'espèce 2 ont également des valeurs plus faibles que celles de l'espèce 3

5.4.2 Rapport de corrélation

On appelle rapport de corrélation le rapport de "variance inter/variance totale". Sa formule est :

$$e = \frac{\frac{1}{n}\sum_{i=1}^{I} n_{i.}(\bar{y}_i - \bar{y})^2}{\sigma_Y^2}$$

On calcule donc la taille de l'échantillon totale, la moyenne de Y, puis la variance de Y, puis la taille de chaque classe y et la moyenne de Y à l'intérieur de chaque classe i :

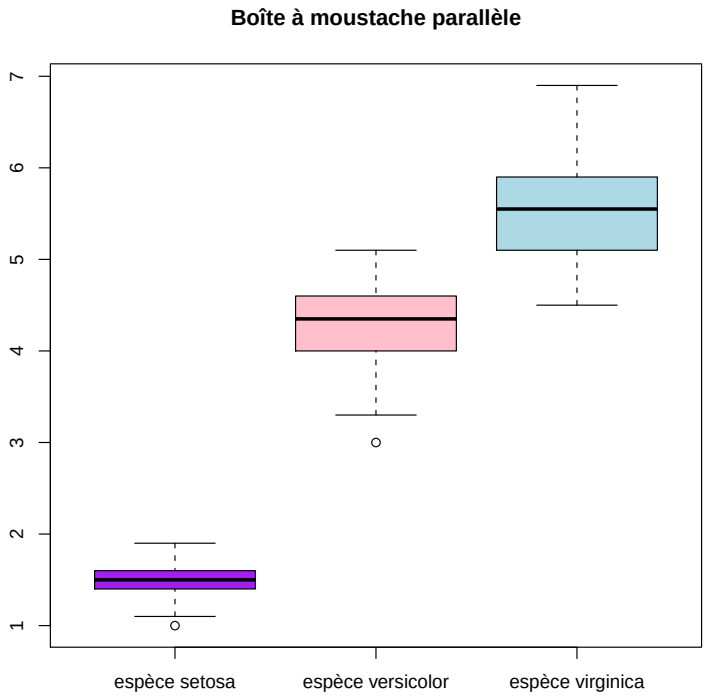

FIG. 5.9 – Représentation graphique du lien entre variable quantitative et qualitative

```
> n <- length(iris$Petal.Length)
> ybar <- mean(iris$Petal.Length)
> yvar <- var(iris$Petal.Length)
> ybar.i <- by(iris$Petal.Length, iris$Species, mean)
> y.ni <- by(iris$Petal.Length, iris$Species, length)
```

La valeur de e veut :

```
> e <- sum(y.ni * (ybar.i - ybar)^2)/n/yvar
> e
[1] 0.935096
```

Commentaires : si e est proche de 0, les moyennes intra sont presque équivalentes, alors il y a absence de dépendance en moyenne entre X et Y. Si e est proche de 1, la variance intra est proche de 0, alors il y a une forte liaison entre X et Y

Q10 : représenter les boîtes à moustaches parallèles de la variable `temps.transport` en fonction de la variable `tabac` du jeu de données `promo.09`. Que constatez-vous

Q11 : calculer le rapport de corrélation entre la variable `temps.transport` en fonction de la variable `tabac` du jeu de données `promo.09`. Que constatez-vous ?

5.4.3 Histogrammes parallèles

On peut représenter plusieurs histogrammes les uns au-dessous des autres lorsqu'on étudie une variable quantitative et une variable qualitative simultanément. La fonction à utiliser n'est pas accessible de suite ; en effet, elle est incluse dans un "package" ou librairie, qui n'est pas chargée par défaut sous R, mais qui se charge en utilisant la fonction `library`. La fonction à utiliser est `ldahist` avec comme premier argument, la variable quantitative et comme second argument la variable qualitative. Le package à charger s'appelle **MASS** et cette opération se fait avec la fonction `library`.

```
> library(MASS)
> ldahist(iris$Petal.Length, iris$Species)
```

5.5 Analyse multivariée

Au-delà de deux variables à étudier simultanément, il est possible d'effectuer d'autres représentations inspirées de celles vues jusqu'à présent. Ces représentations seront appréciées dans vos rapports à condition qu'elles soient accompagnées de commentaires pertinents.

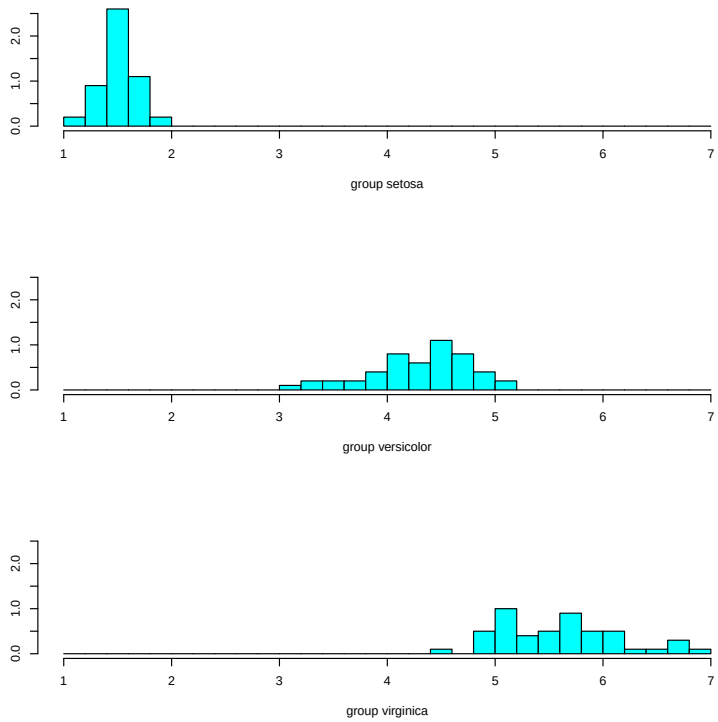

FIG. 5.10 – Histogrammes parallèles

5.5.1 Deux variables quantitatives et une variable qualitative

On peut représenter sur un nuage de points (une variable quantitative Y en fonction d'une variable quantitative X), les points avec des couleurs et/ou symboles différents selon leurs appartenances aux classes d'une variable qualitative. Par exemple :

```
> plot(iris$Petal.Length, iris$Sepal.Length, col = as.integer(iris$Species),
+     pch = as.integer(iris$Species), xlab = "Taille du pétale",
+     ylab = "Taille du sépale")
> legend("topleft", legend = c("Espèce 1", "Espèce 2",
+     "Espèce 3"), col = c(1, 2, 3), pch = 1:3)
```

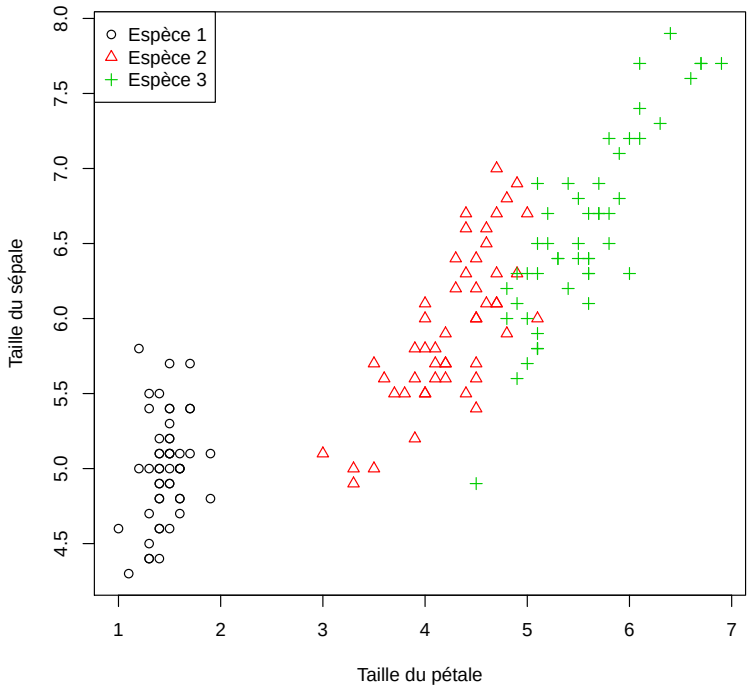

FIG. 5.11 – Nuage de points avec prise en compte d'une variable qualitative

Une autre manière de représenter la même chose est d'utiliser la fonction `xyplot` incluse dans le package **lattice**. Par exemple :

```
> library(lattice)
> xyplot(Sepal.Length ~ Petal.Length | Species, data = iris)
```

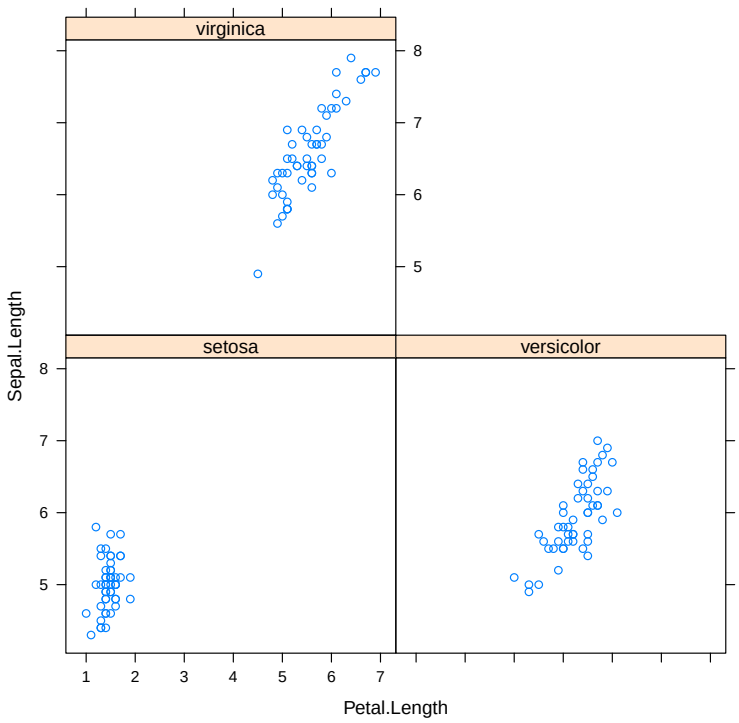

FIG. 5.12 – Nuages de points où chaque quadrant représente les individus appartenant à une classe

Commentaires : on voit sur ces deux types de graphique que X et Y semblent corrélées positivement et linéairement pour les espèces 2 et 3 alors que cela ne semble pas être le cas pour l'espèce 1 (on pourrait vérifier ce constat en calculant le coefficient de corrélation linéaire entre X et Y pour chaque espèce...). De plus, la variable X semble avoir un pouvoir discriminant très fort sur la classe 1. En effet, si on observe un individu dont $X < 2.5$, alors il est très probable que cet individu appartienne à la classe 1. Les espèces 2 et 3 semblent se ressembler davantage même si la variable X semble discriminer légèrement les deux classes autour de la valeur 4.8.

5.5.2 Plusieurs variables quantitatives

Soient 4 variables quantitatives. On peut représenter dans les cellules (hormis la diagonale) d'une matrice de taille 4×4, les 9 nuages de points avec en ordonnée la variable nommée à droite ou à gauche sur la diagonale, et en abscisse la variable nommée au-dessus ou en dessous sur la diagonale. On peut également ajouter comme information supplémentaire, les valeurs prises par une variable qualitative en utilisant une couleur (et/ou symbole) différente selon la modalité. Pour cela, on utilise la fonction `pairs` de la façon suivante :

```
> pairs(iris[1:4], col = as.integer(iris$Species))
```

Commentaires : les variables X_3 et X_4 semblent être fortement corrélées positivement et linéairement entre elles, quelque soit la modalité prise par la variable qualitative. La variable X_2 semble être légèrement liée à X_3 et X_4 en ce qui concerne les espèces 2 et 3. Les lien entre X_1 et X_2 semblent différent selon que l'individu appartient à l'espèce 2 ou 3 et selon que l'individu appartient à l'espèce 3, etc...

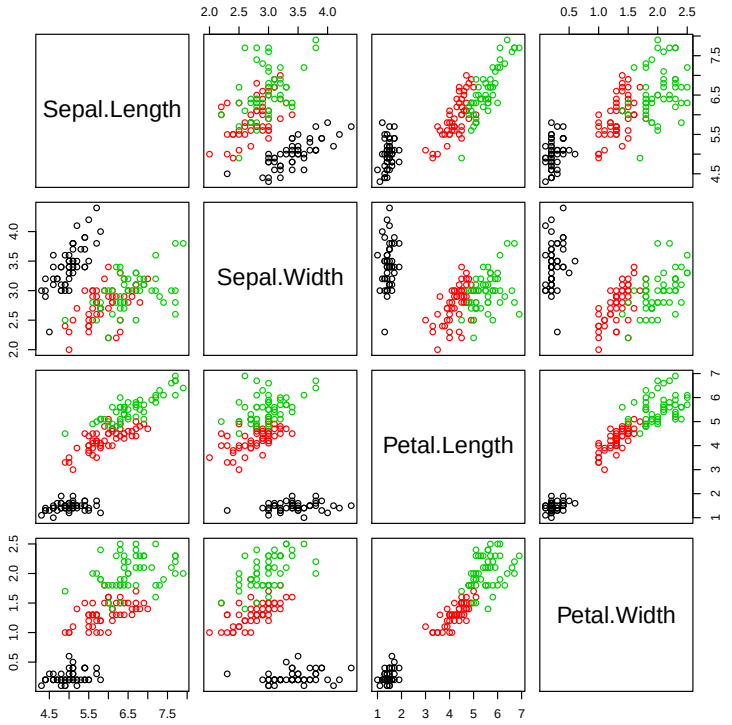

FIG. 5.13 – Représentation matricielle de nuages de points

Chapitre 6

Annexe

6.1 Enregistrer une fenêtre graphique

Pour sauver une fenêtre graphique, il suffit de cliquer sur l'onglet `Fichier`, puis `Sauver sous` et choisir le format approprié (le plus usuel est le format `jpeg`, qui peut être inséré facilement dans les rapports.

6.2 Les fonctions graphiques

plot(x)	graphe des valeurs de x (sur l'axe des y) ordonnées sur l'axe des x
plot(x, y)	graphe bivarié de x (sur l'axe des x) et y (sur l'axe des y)
sunflowerplot(x,y)	idem que plot() mais les points superposés sont dessinés en forme de feurs dont le nombre de pétales représente le nombre de points
pie(x)	graphe en camembert
boxplot(x)	graphe boites et moustaches
stripchart(x)	graphe des valeurs de x sur une ligne (une alternative à boxplot() pour des petits échantillons)
coplot(x y | z)	graphe bivarié de x et y pour chaque valeur (ou intervalle de valeurs) de z
interaction.plot (f1, f2, y)	si f1 et f2 sont des facteurs, graphe des moyennes de y (sur l'axe des y) en fonction des valeurs de f1 (sur l'axe des x) et de f2 (diffééerentes courbes) ; l'option fun permet de choisir la statistique résumée de y (par défaut fun=mean)
matplot(x,y)	graphe bivarié de la 1ère colonne de x contre la 1ère de y, la 2ème de x contre la 2ème de y, etc.
dotchart(x)	si x est un tableau de données, dessine un graphe de Cleveland (graphes superposés ligne par ligne et colonne par colonne)
fourfoldplot(x)	visualise, avec des quarts de cercles, l'association entre deux variables dichotomiques pour différentes populations (x doit être un tableau avec dim=c(2, 2, k) ou une matrice avec dim=c(2, 2) si k = 1)
pairs(x)	si x est une matrice ou un tableau de données, dessine tous les graphes bivariés entre les colonnes de x
plot.ts(x)	si x est un objet de classe "ts", graphe de x en fonction du temps, x peut être multivarié mais les séries doivent avoir les mêmes fréquence et dates
hist(x)	histogramme des fréquences de x
barplot(x)	histogramme des valeurs de x
qqplot(x, y)	quantiles de y en fonction des quantiles de x
stars(x)	si x est une matrice ou un tableau de données, dessine un graphe en segments ou en étoile où chaque ligne de x est représentée par une étoile et les colonnes par les longueurs des branches
symbols(x, y, ...)	dessine aux coordonnées données par x et y des symboles (cercles, carrés, rectangles, étoiles, ...) dont les tailles, couleurs, etc, sont spécifiés par des arguments supplémentaires

6.3 Récapitulatif : commandes permettant d'ajouter des objets aux graphes existants (et quelques options ...)

Commande	Description
`abline`	`aline(h=.)` permet de tracer une ligne horizontale `abline(v=.)` une ligne verticale $x = v$ et `abline(a,b)` une droite d'équation $y = bx + a$.
`arrows(x,y,z,t)`	Ajoute une flèche du point (x ; y) au point (z ; t).
`axis(n=.,at=.,labels=.,pos=.,las=.)`	Ajoute un axe (n = 1, abscisses et n = 2, ordonnée place les valeurs labels aux positions `at` ; `las=0` ajoute ces valeurs parallèlement aux axes, `las=1` les ajoute horizontalement et `las=2` les ajoute en en effectuant une rotation de 4 `pos` oblige l'axe à être tracé à partir de cette coord
`box()`	Ajoute une boîte autour de la figure.
`lines(x,y,...)`	Ajoute des courbes à la figure.
`points(x,y)`	Ajoute des points de coordonnées $(x; y)$ à la figure.
`segments(x,y,t,z)`	Ajoute les segments qui partent de $(x; y)$ et arriven
`text(x,y,text)`	Ajoute `text` à partir du point $(x; y)$.
`title("title","subtitle")`	Ajoute un titre et un sous-titre (en bas).
`legend(x,y,legend=.,col=.,lty=.)`	Ajoute `legend` au point $(x; y)$ pour les paramètres

6.4 Récapitulatifs : Options graphiques

Paramètres	Option par défaut	Description
`type`	"p"	"p", "l", "b", "h", "o" ou "n" : type (voir le 1er graphique)
`axes`	T	T / F : avec ou sans axe
`main`	""	Titre
`sub`	""	Sous-titre
`xlab`	Nom de la variable	Enoncé axe des abscisses
`ylab`	Nom de la variable	Enoncé axe des ordonnées
`xlim`	`c(min(x),max(x))`	Bornes de l'axe des abscisses
`ylim`	`c(min(y),max(y))`	Bornes de l'axe des ordonnées
`pch`	"o"	Caractère marquant le point
`lwd`	1	Epaisseur de la ligne : 1 est l'épaisseur courante, 2 est le dou
`lty`	1	Type de la ligne : 1 est une ligne solide, 2 une ligne pointillé
`col`	1	Couleur : 1 est noir, 2 est rouge, ...
`box`	T	T / F : avec ou sans boite autour de la figure