Programmation Informatique Appliquée aux Sciences et Technologies

---

Chapitre II

Manipulation et Analyse de

Données Scientifiques

Par Mennour Hocine

Université de Mila

2025-2026

Table des Matières

Table des Matières 2
Introduction 3
1. Lecture de fichiers CSV avec read.csv2() 3
1.1. Qu'est-ce qu'un fichier CSV ? 3
1.2. Syntaxe de base de read.csv2() 3
1.3. Explication détaillée 4
1.4. Paramètres supplémentaires utiles 4
2. Statistiques Descriptives 4
2.1. La fonction summary() 4
2.2. La fonction str() 5
2.3. La fonction glimpse() 5
2.4. La fonction names() 6
2.5. Les fonctions head() et tail() 6
2.6. Autres fonctions statistiques avec l'opérateur $ 6
Exemples pratiques avec le jeu de données Mila.csv 7
3. Tests Préliminaires 7
3.1. Test de Shapiro-Wilk 8
3.2. Test de Levene 8
3.3. Types de distribution des données 9
Distribution normale (gaussienne) 9
Distribution non normale 9
3.4. Visualisation avec hist() 10
Comment interpréter un histogramme ? 10
Résumé et Conseils Pratiques 11

Introduction

L'analyse de données scientifiques constitue une compétence fondamentale pour tout chercheur ou étudiant en sciences et technologies. R, langage de programmation statistique, offre une multitude d'outils puissants pour importer, manipuler et analyser des jeux de données. Ce chapitre vous guidera à travers les étapes essentielles de l'analyse de données, depuis l'importation de fichiers jusqu'aux tests statistiques préliminaires, en utilisant des exemples concrets et accessibles aux débutants.

Tout au long de ce chapitre, nous travaillerons avec un jeu de données réel concernant l'étude des chênes-lièges dans la région de Mila. Ce fichier, nommé « Mila.csv », contient 60 observations d'arbres avec 13 variables différentes, incluant des mesures sur l'altitude, les précipitations, la température, le pH du sol, l'épaisseur du liège, le diamètre des arbres, leur hauteur, la production de glands, et les dégâts causés par le feu.

1. Lecture de fichiers CSV avec read.csv2()

1.1. Qu'est-ce qu'un fichier CSV ?

Un fichier CSV (Comma-Separated Values) est un format de fichier texte simple utilisé pour stocker des données tabulaires. Chaque ligne du fichier représente une observation (ou un enregistrement), et les valeurs de chaque colonne sont séparées par un délimiteur, généralement une virgule ou un point-virgule. Ce format est largement utilisé car il est lisible par de nombreux logiciels, incluant Microsoft Excel, LibreOffice Calc, et bien sûr R.

En France et dans de nombreux pays européens, les fichiers CSV utilisent souvent le point-virgule (;) comme séparateur car la virgule est utilisée comme séparateur décimal. C'est pourquoi R propose deux fonctions distinctes : **`read.csv()`** pour les fichiers avec virgule comme séparateur, et **`read.csv2()`** pour les fichiers avec point-virgule.

1.2. Syntaxe de base de read.csv2()

La fonction **`read.csv2()`** permet d'importer un fichier CSV et de le convertir automatiquement en data frame, la structure de données la plus courante dans R pour stocker des données tabulaires. Voici la syntaxe de base :

```
# Lecture d'un fichier CSV avec point-virgule comme séparateur
df <- read.csv2("chemin/vers/votre/fichier.csv")
# Exemple avec notre fichier Mila.csv
df <- read.csv2("C:/Users/acer/Desktop/Programmation/Mila.csv")
```

1.3. Explication détaillée

Analysons chaque élément de cette commande. Le symbole **`<-`** est l'opérateur d'assignation dans R. Il permet de stocker le résultat de la fonction **`read.csv2()`** dans une variable nommée **`df`** (abréviation de « data frame »). Le chemin du fichier doit être entouré de guillemets car il s'agit d'une chaîne de caractères.

Remarquez l'utilisation des barres obliques (/) dans le chemin du fichier, même sous Windows. R accepte également les barres obliques inverses doubles (\\) mais les barres obliques simples sont recommandées car elles sont plus portables entre les systèmes d'exploitation.

1.4. Paramètres supplémentaires utiles

La fonction **`read.csv2()`** accepte plusieurs paramètres optionnels qui permettent de personnaliser l'importation :

Paramètre	Description
`header`	TRUE si la première ligne contient les noms des colonnes (défaut: TRUE)
`sep`	Séparateur entre les valeurs (défaut: ";" pour read.csv2)
`dec`	Caractère utilisé pour les décimales (défaut: "," pour read.csv2)
`encoding`	Encodage du fichier (ex: "UTF-8", "latin1")

Tableau 1 : Principaux paramètres de read.csv2()

2. Statistiques Descriptives

Les statistiques descriptives permettent de résumer et de comprendre les caractéristiques principales d'un jeu de données. Elles incluent des mesures de tendance centrale (moyenne, médiane), de dispersion (écart-type, variance, étendue), et de position (quartiles, minimum, maximum). R offre plusieurs fonctions pour obtenir ces informations rapidement.

2.1. La fonction summary()

La fonction **`summary()`** est l'une des fonctions les plus utiles dans R pour obtenir un aperçu rapide de vos données. Elle fournit un résumé statistique pour chaque variable du data frame, adapté au type de données (numérique ou catégorielle).

```
# Afficher le résumé statistique du data frame
summary(df)
```

Résultat obtenu avec notre fichier Mila.csv :

```
   arbre_id            foret               altitude     pluie_annuelle
 Length:60          Length:60          Min.   :450.0   Min.   :550
 Class :character   Class :character   1st Qu.:450.0   1st Qu.:550
 Mode  :character   Mode  :character   Median :680.0   Median :720
                                       Mean   :683.3   Mean   :720
                                       3rd Qu.:920.0   3rd Qu.:890
                                       Max.   :920.0   Max.   :890
```

Interprétation : Pour les variables numériques comme `altitude`, la fonction affiche le minimum (450 m), le premier quartile (450 m), la médiane (680 m), la moyenne (683,3 m), le troisième quartile (920 m) et le maximum (920 m). Pour les variables caractères comme `arbre_id`, elle affiche la longueur (60 observations), la classe et le mode.

2.2. La fonction str()

La fonction **`str()`** (abréviation de « structure ») affiche la structure interne d'un objet R. Elle est particulièrement utile pour comprendre rapidement la nature de vos données, le type de chaque variable, et visualiser les premières observations. C'est souvent la première fonction à utiliser après avoir importé un nouveau jeu de données.

```
# Afficher la structure du data frame
str(df)
```

Résultat obtenu :

```
'data.frame':    60 obs. of  13 variables:
 $ arbre_id            : chr  "CL01" "CL02" "CL03" "CL04" ...
 $ foret               : chr  "Zouagha" "Zouagha" "Zouagha" ...
 $ altitude            : int  450 450 450 450 450 450 450 ...
 $ pluie_annuelle      : int  550 550 550 550 550 550 550 ...
 $ temperature_moy     : num  17.8 17.6 18 17.9 17.7 ...
```

Interprétation : La sortie de `str()` nous indique que notre data frame contient 60 observations et 13 variables. Le symbole **`$`** précède le nom de chaque variable. Les types de données sont abrégés : `chr` pour caractère, `int` pour entier, `num` pour numérique, et `logi` pour logique. Les premières valeurs de chaque variable sont également affichées.

2.3. La fonction glimpse()

La fonction **`glimpse()`** provient du package `dplyr` (inclus dans le tidyverse). Elle est similaire à `str()` mais présente les informations de manière plus compacte et transposée, ce qui la rend particulièrement utile pour les data frames avec de nombreuses colonnes.

```
# Charger le package dplyr (à faire une seule fois par session)
library(dplyr)

# Utiliser glimpse()
glimpse(df)
```

Remarque importante : Av d'utiliser `glimpse()`, vous devez installer le package dplyr si ce n'est pas déjà fait, avec la commande `install.packages("dplyr")`. Ensuite, chargez le package avec `library(dplyr)` au début de chaque session R.

2.4. La fonction names()

La fonction **`names()`** retourne les noms des colonnes (variables) d'un data frame. Cette fonction est très utile pour vérifier rapidement les variables disponibles ou pour accéder à une variable spécifique par son nom.

```
# Afficher les noms des colonnes
names(df)
```

Résultat : `[1] "arbre_id" "foret" "altitude" "pluie_annuelle" "temperature_moy" "ph_sol" "epaisseur_liege_2022" ...`

2.5. Les fonctions head() et tail()

Ces deux fonctions permettent de visualiser respectivement les premières et les dernières lignes d'un data frame. Par défaut, elles affichent 6 lignes, mais vous pouvez spécifier un nombre différent.

```
# Afficher les 6 premières lignes
head(df)

# Afficher les 10 premières lignes
head(df, n = 10)

# Afficher les 6 dernières lignes
tail(df)
```

2.6. Autres fonctions statistiques avec l'opérateur $

L'opérateur **`$`** permet d'accéder à une variable spécifique dans un data frame. La syntaxe est `nom_dataframe$nom_variable`. Une fois la variable isolée, vous pouvez appliquer diverses fonctions statistiques.

Fonction	Description	Exemple
`mean()`	Calcule la moyenne	`mean(df$altitude)`
`median()`	Calcule la médiane	`median(df$altitude)`
`min()`	Retourne la valeur minimale	`min(df$altitude)`
`max()`	Retourne la valeur maximale	`max(df$altitude)`
`sd()`	Calcule l'écart-type	`sd(df$altitude)`
`var()`	Calcule la variance	`var(df$altitude)`
`range()`	Retourne min et max	`range(df$altitude)`
`quantile()`	Calcule les quantiles	`quantile(df$altitude)`
`length()`	Retourne le nombre d'éléments	`length(df$altitude)`

Tableau 2 : Fonctions statistiques avec l'opérateur $

Exemples pratiques avec le jeu de données Mila.csv

Voici des exemples concrets d'utilisation de ces fonctions avec notre jeu de données sur les chênes-lièges :

```
# Calculer la moyenne de l'altitude
mean(df$altitude)  # Résultat : 683.3 m

# Calculer l'écart-type de la température moyenne
sd(df$temperature_moy)  # Résultat : environ 1.24 °C

# Obtenir l'étendue du pH du sol
range(df$ph_sol)  # Résultat : 5.4  8.2

# Calculer les quartiles de l'épaisseur du liège en 2024
quantile(df$epaisseur_liege_2024)
```

3. Tests Préliminaires

Avant d'appliquer des tests statistiques paramétriques (comme le test t de Student ou l'ANOVA), il est essentiel de vérifier certaines hypothèses sur vos données. Les deux conditions principales sont la normalité de la distribution (les données suivent-elles une loi normale ?) et l'homogénéité des variances (les groupes ont-ils des variances similaires ?). Ce chapitre vous présente les tests permettant de vérifier ces conditions.

3.1. Test de Shapiro-Wilk

Le test de Shapiro-Wilk est utilisé pour vérifier si un échantillon provient d'une population suivant une distribution normale (gaussienne). C'est l'un des tests les plus puissants pour détecter les écarts à la normalité, particulièrement efficace pour les petits échantillons ($n < 50$) mais également valide pour des échantillons plus grands.

Principe du test : L'hypothèse nulle (H_0) stipule que les données suivent une distribution normale. L'hypothèse alternative (H_1) stipule que les données ne suivent pas une distribution normale.

Hypothèse	Interprétation
H_0 : distribution normale	Les données suivent une loi normale
H_1 : distribution non normale	Les données ne suivent pas une loi normale

Tableau 3 : Hypothèses du test de Shapiro-Wilk

Règle de décision : Si la p-value est supérieure au seuil de significativité (généralement $\alpha = 0,05$), on ne rejette pas H_0 : les données peuvent être considérées comme suivant une distribution normale. Si la p-value est inférieure à 0,05, on rejette H_0 : les données ne suivent pas une distribution normale.

```
# Test de Shapiro-Wilk sur l'altitude
shapiro.test(df$altitude)

# Test de Shapiro-Wilk sur la température moyenne
shapiro.test(df$temperature_moy)
```

Interprétation des résultats : La fonction retourne deux valeurs : W (la statistique de test) et la p-value. Si p-value > 0,05, vous pouvez supposer que vos données suivent approximativement une distribution normale. Dans le cas contraire, vous devrez utiliser des tests non paramétriques.

3.2. Test de Levene

Le test de Levene permet de vérifier l'égalité des variances entre plusieurs groupes, une condition appelée homoscédasticité. Ce test est moins sensible aux écarts de normalité que le test de Bartlett et est donc généralement préféré.

Ce test est particulièrement utile lorsque vous souhaitez comparer plusieurs groupes (par exemple, comparer l'épaisseur du liège entre différentes forêts) et que vous prévoyez d'utiliser une ANOVA.

Principe du test : L'hypothèse nulle (H_0) stipule que toutes les variances des groupes sont égales. L'hypothèse alternative (H_1) stipule qu'au moins une variance diffère des autres.

```
# Charger le package car (nécessaire pour le test de Levene)
install.packages("car")  # À exécuter une seule fois
library(car)

# Test de Levene : comparer les variances d'épaisseur entre forêts
leveneTest(epaisseur_liege_2024 ~ foret, data = df)
```

Interprétation : Si la p-value est supérieure à 0,05, on ne rejette pas l'hypothèse d'égalité des variances. Les groupes peuvent être considérés comme ayant des variances similaires, ce qui justifie l'utilisation de tests paramétriques comme l'ANOVA.

3.3. Types de distribution des données

La distribution des données décrit la manière dont les valeurs sont réparties. Comprendre le type de distribution est crucial pour choisir les tests statistiques appropriés.

Distribution normale (gaussienne)

La distribution normale, également appelée distribution gaussienne ou « courbe en cloche », est caractérisée par sa symétrie autour de la moyenne. Elle est définie par deux paramètres : la moyenne (μ) qui détermine le centre de la distribution, et l'écart-type (σ) qui détermine sa dispersion. Environ 68% des observations se situent à moins d'un écart-type de la moyenne, 95% à moins de deux écarts-types, et 99,7% à moins de trois écarts-types.

Caractéristiques d'une distribution normale : la moyenne, la médiane et le mode sont égaux ; la courbe est symétrique ; les extrémités (queues) s'étendent à l'infini sans jamais toucher l'axe des abscisses. De nombreux phénomènes naturels suivent approximativement une distribution normale, ce qui explique son importance en statistiques.

Distribution non normale

Lorsque les données ne suivent pas une distribution normale, plusieurs situations peuvent se présenter : une asymétrie (skewness) où la distribution est étirée vers la gauche ou la droite ; une aplatissement (kurtosis) différent de celui d'une distribution normale ; ou une distribution multimodale avec plusieurs pics.

Dans ce cas, vous devez utiliser des tests non paramétriques qui ne font pas d'hypothèse sur la distribution des données. Ces tests incluent le test de Mann-Whitney (équivalent du test t), le test de Kruskal-Wallis (équivalent de l'ANOVA), ou le test de Wilcoxon pour échantillons appariés.

3.4. Visualisation avec hist()

L'histogramme est un outil graphique fondamental pour visualiser la distribution d'une variable numérique. Il permet de voir rapidement si les données sont symétriques, asymétriques, unimodales ou multimodales. La fonction **`hist()`** de R base crée cet histogramme de manière simple.

Principe de l'histogramme : L'axe horizontal représente les valeurs de la variable, divisées en intervalles (classes ou « bins »). L'axe vertical représente la fréquence (nombre d'observations) dans chaque intervalle. La forme de l'histogramme permet d'identifier le type de distribution.

```
# Histogramme simple de l'altitude
hist(df$altitude)

# Histogramme avec personnalisation
hist(df$temperature_moy,
     main = "Distribution de la température moyenne",
     xlab = "Température (°C)",
     ylab = "Fréquence",
     col = "lightblue",
     border = "darkblue")
```

Paramètres de la fonction hist() :

Paramètre	Description
`main`	Titre du graphique
`xlab`	Étiquette de l'axe des x (abscisses)
`ylab`	Étiquette de l'axe des y (ordonnées)
`col`	Couleur de remplissage des barres
`border`	Couleur des bordures des barres
`breaks`	Nombre de classes (intervalles)

Tableau 4 : Principaux paramètres de hist()

Comment interpréter un histogramme ?

Pour déterminer visuellement si vos données suivent une distribution normale, examinez la forme de l'histogramme. Une distribution normale présentera une forme symétrique en cloche, avec le pic au centre. Si l'histogramme montre une asymétrie marquée (une queue plus longue d'un côté), plusieurs pics distincts, ou une forme très aplatie ou très pointue, les données ne suivent probablement pas une distribution normale.

Important : L'histogramme donne une indication visuelle mais ne constitue pas un test formel. Il est recommandé de combiner l'observation de l'histogramme avec le test de Shapiro-Wilk pour prendre une décision éclairée sur la normalité des données.

Résumé et Conseils Pratiques

Ce chapitre vous a présenté les outils fondamentaux pour importer, explorer et préparer vos données pour l'analyse statistique. Voici un récapitulatif des bonnes pratiques à retenir pour vos travaux futurs en analyse de données avec R.

Workflow recommandé pour l'analyse de données :

1. Importez vos données avec `read.csv2()` pour les fichiers avec séparateur point-virgule.
2. Explorez la structure avec `str()` ou `glimpse()` pour comprendre les types de variables.
3. Obtenez un résumé statistique avec `summary()` pour identifier les valeurs extrêmes ou manquantes.
4. Visualisez la distribution avec `hist()` pour une première impression visuelle.
5. Testez la normalité avec `shapiro.test()` pour décider des tests statistiques appropriés.
6. Vérifiez l'homogénéité des variances avec `leveneTest()` si vous comparez plusieurs groupes.

En suivant cette démarche méthodique, vous serez en mesure de mener des analyses statistiques rigoureuses et d'interpréter correctement vos résultats. N'oubliez pas que la qualité de vos conclusions dépend directement de la qualité de votre préparation des données et de la vérification des hypothèses statistiques.