Université de Montpellier
Département de Mathématiques Appliquées

Outstanding 4 : Gestion des références et imputations de valeurs manquantes

Atelier Projet — HAX916X

Réalisé par :
DIALLO Ousmane
ATTOUMANI Ibrahim

Année Universitaire 2025 – 2026

Table des matières

1. Introduction

Dans ce mini-cours, nous allons aborder trois sujets essentiels pour la manipulation et l'analyse de données dans R :

- Changer la modalité de référence dans un modèle ANOVA afin de mieux interpréter les coefficients.
- Remplacer les valeurs manquantes par la médiane, une méthode simple et efficace d'imputation.
- Représenter les contributions individuelles des observations dans un modèle linéaire (`lm`), l'équivalent des effets aléatoires (`ranef`) utilisés dans les modèles mixtes (`lmer`).

Nous finirons par vérifier la contrainte de normalité sur nos variables et tenterons de la corriger par des transformations si nécessaire.

2. Base de données

Ci-dessous, nous importons les données **Exp2_height.csv** puis réalisons plusieurs étapes de nettoyage indispensables.

Nous transformons d'abord les cellules vides en valeurs manquantes (**NA**), puis nous remplaçons les virgules par des points pour permettre la conversion correcte en format numérique

```r
# importation de données
data <- read.csv("data/Exp2_height.csv", sep = ";",stringsAsFactors = FALSE)
data$type_assoc <- ifelse(data$var_1 == data$var_2,
                      "intra", "inter")
# eviter A-B et B-A en double
data$comb <- as.factor(apply(data[,c("var_1", "var_2")],1,
                             function(x) paste(x, collapse = "")))

# Remplacer les cellules vides "" par NA
data[data == ""] <- NA

# changer les virg, en point.
data <- data.frame(lapply(data, function(col) {
  if (is.character(col)) {
    col <- gsub(",", ".", col)  # Remplace , par .
  }
  return(col)
}))
```

2.1. Imputation de données manquantes

Dans cette section, nous procédons à l'imputation des valeurs manquantes pour les variables de hauteur. Nous commençons par convertir les colonnes concernées au format numérique, puis nous remplaçons chaque valeur manquante par la moyenne correspondante de la colonne. Dans cette implémentation, les données sont **centrées autour de leur moyenne** avant l'imputation.

La fonction **imput** présentée en annexe permet de réaliser une **imputation itérative des valeurs manquantes** dans un jeu de données. Elle fonctionne en ajustant successivement un modèle linéaire pour chaque variable contenant des NA, en utilisant les autres variables comme prédicteurs.

Cette approche est particulièrement **robuste pour des données dont les variables présentent des relations linéaires**, et elle remplace les valeurs manquantes par les valeurs prédites de manière répétée sur plusieurs itérations afin d'améliorer la cohérence globale des données.

```r
# selectionner les variable
hauteur <- c("height_1_var_1", "height_2_var_2")

# séléctionner les variables sur la hauteur
data[,hauteur] <- lapply(data[,hauteur], as.numeric)

# Utilisation
data[,hauteur] <- imput(data[,hauteur])
```

```
##
## === Itération 1 ===
## variable 1 - indices NA: 16 20
## Valeurs prédites: 11.4555 11.1798
##
## variable 2 - indices NA: 6 15 51 145
## Valeurs prédites: 10.6892 10.735 10.6663 10.0709
##
##
## === Itération 2 ===
## variable 1 - indices NA: 16 20
## Valeurs prédites: 11.5732 11.3131
##
## variable 2 - indices NA: 6 15 51 145
## Valeurs prédites: 10.5491 10.6407 10.7552 10.6293
##
##
## === Itération 3 ===
## variable 1 - indices NA: 16 20
## Valeurs prédites: 11.5776 11.314
```

```
##
## variable 2 - indices NA: 6 15 51 145
## Valeurs prédites: 10.5513 10.6435 10.7588 10.632
##
##
## === Itération 4 ===
## variable 1 - indices NA: 16 20
## Valeurs prédites: 11.5777 11.314
##
## variable 2 - indices NA: 6 15 51 145
## Valeurs prédites: 10.5514 10.6436 10.7589 10.6321
```

2.2. Redéfinition de la référence

Dans cette section, nous montrons comment modifier la catégorie de référence d'une variable qualitative en la reparamétrant directement dans R. Ce changement de référence permet d'ajuster l'interprétation des coefficients et d'adapter le modèle à la comparaison souhaitée.

Considérons le modèle suivant:

$$Y = Z\beta + \epsilon \quad (1)$$

où:

- le vecteur $Y \in \mathbb{R}^n$ contient les hauteurs des modalités de la variété,
- le vecteur $\beta \in \mathbb{R}^J$ contient les effets des modalités de la variété,
- La matrice $Z = [Z_1, \ldots, Z_J] \in \mathbb{R}^{n \times J}$ regroupe les variables indicatrices associées à la variété `var1`, qui comporte 10 modalités notées `A` à `J`. Pour chaque individu $i = 1, \ldots, n$, si $\texttt{var1}_i = D$, alors la ligne Z_i contient des zéros partout, sauf dans la colonne correspondant à la modalité D, où elle vaut 1. Ainsi, chaque ligne vérifie la relation $\sum_{j=1}^{J} Z_j = \mathbf{1}_n$, c'est-à-dire qu'une seule modalité est active pour chaque individu.,
- le vecteur $\epsilon \in \mathbb{R}^{\kappa}$ est le bruit,
- la contrainte $\sum_{j=1}^{J} \beta_j = 0$ est vérifié,
- $n = 165$ est le nombre d'observations.

le modèle de peut être réécrit en prenant une modalité de variété comme référence :

$$
\begin{aligned}
Y &= \sum_{j=1}^{J} \beta_j Z_j + \epsilon \\
&= \sum_{j=1}^{J-1} \beta_j Z_j + \beta_J Z_J + \epsilon \\
&= \sum_{j=1}^{J-1} \beta_j Z_j + \beta_J Z_J - Z_J \sum_{j=1}^{J} \beta_j + \epsilon \quad \text{(par le contrainte} \quad \sum_{j=1}^{J} \beta_j = 0) \qquad (*) \\
&= \sum_{j=1}^{J-1} \beta_j Z_j + \beta_J Z_J - \sum_{j=1}^{J-1} \beta_j Z_J - \beta_J Z_J + \epsilon \\
&= \sum_{j=1}^{J-1} \beta_j (Z_j - Z_J) + \epsilon
\end{aligned}
$$

On retrouve le modèle (1) à partir du modèle (2) si $\mu = \frac{1}{J}\sum_{j=1}^{J} \beta_j$ et si cette quantité vaut 0 ci-dessous :

$$
\begin{aligned}
y_{i,j} = \mu + (\beta_j - \mu) + \epsilon_{i,j} \iff Y &= \mu 1_n + \sum_{j=1}^{J} (\beta_j - \mu) Z_j + \epsilon \\
\iff Y &= \mu 1_n + \sum_{j=1}^{J} \alpha_j Z_j + \epsilon \quad (\text{avec} \quad \alpha_j = \beta_j - \mu) \\
\iff Y &= \mu 1_n + \sum_{j=A}^{J-1} \alpha_j (Z_j - Z_J) + \epsilon \quad (\text{ d'après } (*))
\end{aligned}
$$

2.2.1. Application sous R: Redéfinition de la référence

Nous allons dans la suite voir comment effectuer à un changement de référence sous R.

```r
# s'assurer que var_1 est bien un facteur
data$var_1 <- factor(data$var_1)

# On choisit la nouvelle référence pour var_1
data$var_1 <- relevel(data$var_1, ref = "B")

# On relance le modèle avec la nouvelle référence
mod1 <- lmer(height_1_var_1 ~ var_1 + (1 | var_1:var_2),
             data = data)

# Résumé et ANOVA
summary(mod1)
anova(mod1)

# Effets aléatoires
```

```
rr1 <- ranef(mod1)
dotplot(rr1)
```

2.2.2. Equivalent de ranef et doplot pour lm()

Dans cette section, nous allons créer un équivalent **dotplot_lm** des effets des interactions pour un modèle linéaire classique (lm). Nous visualiserons les coefficients par combinaison de variables à l'aide d'un dotplot avec intervalles de confiance. Pour plus de détaille sur la fonction **dotplot_lm** voir en annexe.

```
mod2 <- lm(data$height_1_var_1 ~ var_1 + comb,
           data = data)
summary(mod2)

dotplot_lm(mod2, "comb")
```

Annexe

```r
# equivalent de ranef
dotplot_lm <- function(mod, nom_var_inter, col="blue"){
  # Extraire les coefficients des combinaisons
rrr <- sort(coef(mod)[grep(nom_var_inter, names(coef(mod)))])

# Calculer les intervalles de confiance pour le modèle
ci <- confint(mod)
ci_comb <- ci[grep(nom_var_inter, rownames(ci)), ]

# Trier les coefficients
rrr_ord <- sort(rrr)
# aligne les IC sur les coefficients triés
ci_comb_ord <- ci_comb[names(rrr_ord), ]

# Définir les limites de l'axe x
xlim_range <- range(c(ci_comb_ord[,1]-0.85, ci_comb_ord[,2], rrr_ord)+0.85)

# Dotplot
dotplot(rrr_ord,
        xlab = "Coefficient",
        ylab = "Combinaison",
        main = "Effets par combinaison avec IC",
        xlim = xlim_range,  # forcer l'axe x
        panel = function(x, y) {
          panel.dotplot(x, y)
          for(i in seq_along(x)) {
            panel.segments(ci_comb_ord[i, 1], i, ci_comb_ord[i, 2], i, col = col, lwd =
          }
        })
}
```

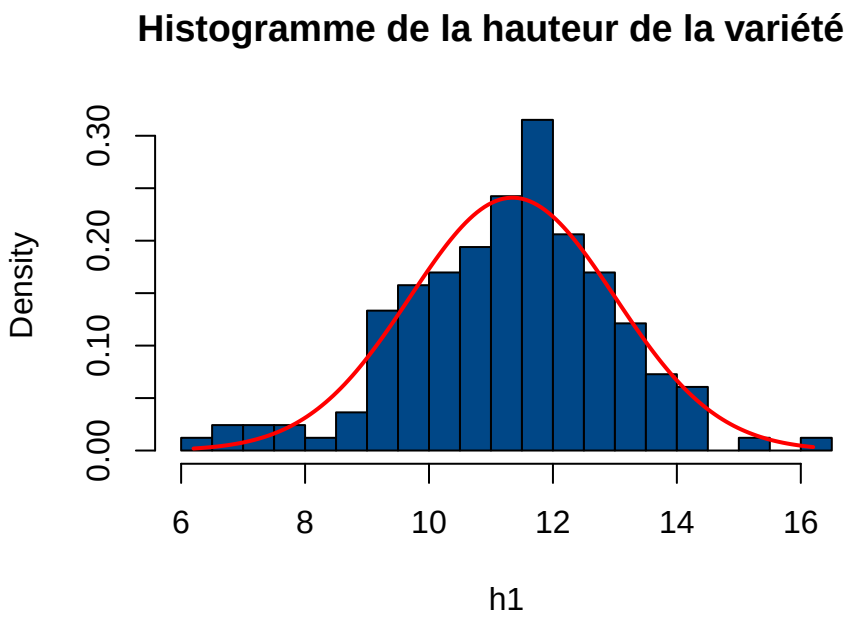
```r
imput <- function(X, iter = 4){
  p <- ncol(X)
  X_inter <- X

  for (k in 1:iter) {
    cat("\n=== Itération", k, "===\n")

    for (j in 1:p) {
      Y <- X_inter[, j]
      na_rows <- which(is.na(Y))

      if (length(na_rows) > 0) {
```

```
        Y <- X[, j]
        # Construire le data.frame des prédicteurs
        exp <- X[, -j, drop = FALSE]

        # Si j = 1, remplacer les NA des prédicteurs par leur moyenne
        if (j == 1) {
          for (col in 1:ncol(exp)) {
            na_pred <- is.na(exp[, col])
            if (any(na_pred)) {
              exp[na_pred, col] <- mean(exp[, col], na.rm = TRUE)
            }
          }
        }

        df <- data.frame(Y = Y, exp)

        # Modèle linéaire
        mod <- lm(Y ~ ., data = df)

        # Valeurs prédites pour les NA
        pred_na <- mod$fitted.values[na_rows]

        # Affichage des valeurs imputées
        cat("variable", j, "- indices NA:", na_rows, "\n")
        cat("Valeurs prédites:", round(pred_na, 4), "\n\n")

        # Remplacer les NA par les valeurs prédites
        Y[na_rows] <- pred_na
        X[, j] <- Y
      }
    }
  }

  return(X)
}
```

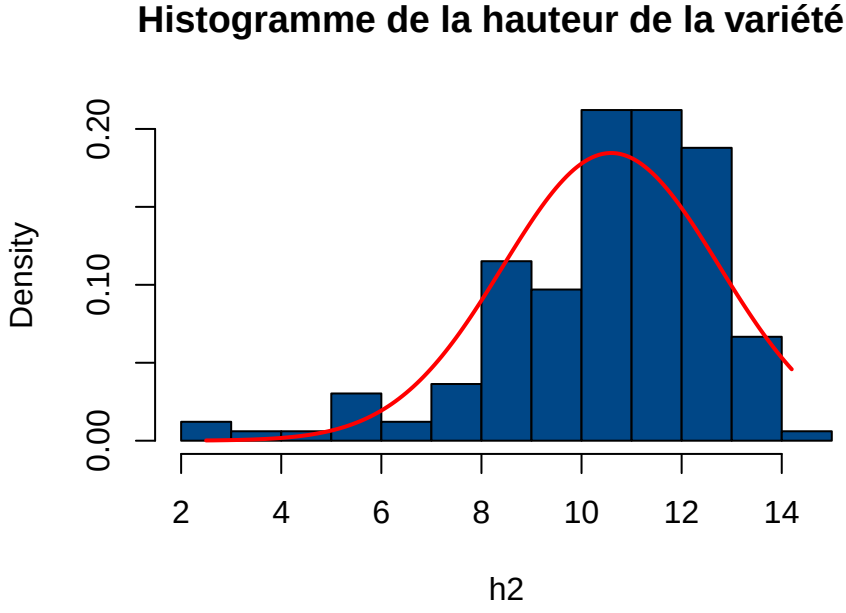
Pour la variable **h2** du code ci-dessous, une transformation a été appliquée pour corriger son asymétrie : inversion autour de sa valeur maximale, suivie d'une racine carrée.
Cette approche permet de rendre la distribution plus symétrique et plus proche d'une distribution normale.
Les histogrammes avant et après transformation montrent clairement l'amélioration de la forme de **h2**.

```
h1 <- data$height_1_var_1
h2 <- data$height_2_var_2
```

```r
# correction de l'asymétrie
# skewness(h2) # quantifier l'asymétrie

# Inversion autour de la valeur maximale
h2_t <- max(h2) - h2

# Puis racine carrée pour adoucir la distribution
h2_t <- sqrt(h2_t)

# grilles pour les densités
x1   <- seq(from = 6.2, to = 16.2, length = 165)
x2   <- seq(from = 2.5, to = 14.2, length = 165)
x2_t <- seq(from = min(h2_t), to = max(h2_t), length = 165)


# les densités théorique sur la grille
d1 <- dnorm(x1, mean = mean(h1), sd= sd(h1))
d2 <- dnorm(x2, mean = mean(h2), sd= sd(h2))
d2_t <- dnorm(x2_t, mean = mean(h2_t), sd= sd(h2_t))


# histogrammes + densités
hist(h1, breaks = 15, freq = FALSE,
     main="Histogramme de la hauteur de la variété 1", col = "#004787")
lines(x1, d1, col="red", lwd = 2)
```




```r
hist(h2, breaks = 15, freq = FALSE,
     main="Histogramme de la hauteur de la variété 2", col = "#004787")
lines(x2, d2, col="red", lwd = 2)
```




```r
hist(h2_t, breaks = 15, freq = FALSE,
     main="Histogramme de la hauteur de la variété 2 transformée",
     cex.main = 0.9, col = "#004787")
```

```r
lines(x2_t, d2_t, col="red", lwd = 2)
```