ESD113 — Probabilités et statistique avec R

De l'introduction à R au clustering : Keno, Tidyverse et méthodes non supervisées

Auditeur CNAM : Alban VIDELOUP

18 mai 2026

Table des matières

Remerciements

À retenir

Ce document est dédié à celles et ceux sans qui rien de tout cela n'aurait été possible.

À **Rhizlane**, mon épouse.

Les soirées passées sur ces pages, les week-ends confisqués par les algorithmes et les compilations LaTeX récalcitrantes: tu les as vécus avec une patience et une générosité qui forcent l'admiration. Tu as tenu le foyer, organisé les journées, rassuré les enfants qui demandaient où était papa, et tu l'as fait avec cette force tranquille qui te caractérise. Ce travail t'appartient autant qu'à moi.

À **mes deux enfants**,

Vous avez souvent dû vous passer de moi à des moments où un père devrait être présent : les jeux du soir, les histoires du coucher, les repas en famille que l'écran a parfois remplacés. Vous avez accepté ces absences avec une maturité touchante, et chacun de vos sourires, chacune de vos questions sur "ce que fait papa sur l'ordinateur", m'a redonné l'énergie de continuer. Je vous dois du temps, de la présence, et je m'y engage. Ce document, c'est aussi votre victoire.

À **Monsieur Karim KILANI**, responsable de l'unité d'enseignement ESD113,

Rares sont les enseignants qui parviennent à rendre la statistique non seulement accessible, mais véritablement désirable. Votre pédagogie rigoureuse, votre disponibilité et votre souci constant de contextualiser les méthodes dans des cas concrets ont transformé ce cours en une exploration intellectuelle passionnante. Merci de consacrer votre expertise à des auditeurs comme moi, qui jonglent entre vie professionnelle et ambition académique.

Au **CNAM**,

Permettre à un auditeur d'une cinquantaine d'années, au cœur d'une vie active intense, de se former aux méthodes quantitatives avancées, à la data science, à R et à Quarto : c'est un acte de foi dans l'idée que l'éducation n'a pas d'âge et que l'apprentissage tout au long de la vie n'est pas un slogan mais une réalité concrète. Le CNAM incarne cette conviction avec une constance exemplaire depuis plus de deux siècles. Il est difficile d'exprimer combien ce dispositif représente, pour des professionnels comme moi, une chance inestimable de rester dans la course d'un monde qui accélère. Merci d'exister, merci de persévérer, merci d'accueillir ceux que les circuits classiques ne peuvent plus atteindre.

Alban VIDELOUP
Paris, 18 mai 2026

Accès aux ressources complémentaires

— **Rapport en ligne :** Publication RPubs
— **Projet GitHub :** albanpjy.github.io/ESD113/
— **Téléchargement :** Document au format PDF

Introduction générale

Ce document a été réalisé dans le cadre de l'unité d'enseignement **ESD113 — Probabilités et statistiques avec R**, dispensée au Conservatoire National des Arts et Métiers (CNAM) par Monsieur Karim KILANI.

Un recours significatif aux outils d'intelligence artificielle a permis de résoudre les problématiques complexes de mise en page LaTeX avec précision et rapidité, tout en facilitant la structuration du contenu.

Ce document fusionne trois volets complémentaires du cours :

1. **Volet 1 — Introduction à R, Quarto et Tidyverse :** prise en main de l'environnement de travail, manipulation de données, visualisation.
2. **Volet 2 — Projet Keno :** application à un cas concret avec les données de tirage du Keno (FDJ), incluant modélisation probabiliste et simulation de Monte-Carlo.
3. **Volet 3 — Méthodes de clustering :** reproduction et adaptation du code R de l'article de référence *"A Survey of Popular R Packages for Cluster Analysis"* (Flynt & Dean, 2016), couvrant K-means, classification hiérarchique ascendante et modèles de mélanges gaussiens (Mclust).

Le fil conducteur est la **rigueur reproductible** : chaque résultat est produit directement par le code R présenté, dans un document Quarto compilable du premier coup.

À retenir

Note sur le choix rédactionnel : Ce document fait délibérément appel à LaTeX pour la majorité de ses éléments de mise en forme — boîtes pédagogiques, formules mathématiques, tableaux, en-têtes et pieds de page. Ce choix n'est pas anodin : si Quarto supporte nativement la syntaxe Markdown, la version pure Markdown produit un rendu PDF sensiblement plus générique et moins soigné. LaTeX offre un contrôle typographique de précision — gestion fine des espacements, des polices, des couleurs, des flottants et des environnements — qui confère au document le niveau de finition attendu d'un travail académique. Le couple Quarto + LaTeX constitue ainsi le meilleur des deux mondes : la lisibilité du Markdown pour le texte courant et le code R, la puissance de LaTeX pour la présentation.

Toutes les analyses sont réalisées avec R (R Core Team 2024) et Quarto (Allaire et al. 2024).

Partie I — Introduction à R, Quarto et Tidyverse

1 Initiation à R, Quarto et Markdown

1.1 Présentation générale

Ce document est une introduction pratique au langage **R** et à l'environnement de publication **Quarto**. L'objectif est triple :

1. **Découvrir Quarto** : un système de publication scientifique qui mêle texte, code et résultats dans un seul document.
2. **Maîtriser les bases de R et du Tidyverse** : manipulation de données, statistiques descriptives, visualisation.
3. **Appliquer ces outils à un cas réel** : les données de tirage du **Keno FDJ**, avec modélisation probabiliste et simulation de Monte-Carlo.

Analogie pédagogique

Quarto, c'est comme un cahier de laboratoire numérique : on y écrit ses observations (le texte), on y insère ses protocoles (le code R), et les résultats apparaissent automatiquement à la compilation. Plus besoin de copier-coller les sorties manuellement.

1.2 Mise en forme du texte en Markdown

Quarto repose sur la syntaxe **Markdown** pour la mise en forme. On peut écrire en **gras** et en *italique*, insérer des listes, des tableaux et des formules mathématiques grâce à LaTeX.

1.2.1 Exemples de formules mathématiques

Pour chaque formule, le bloc grisé ci-dessous montre le **code LaTeX source** tel qu'il apparaît dans le document `.qmd`, et le rendu est affiché juste en dessous. Cette double présentation est précieuse pour l'apprentissage : on voit à la fois ce qu'on écrit et ce que le moteur typographique produit.

L'intégrale de Gauss — résultat fondamental en probabilité et théorie des fonctions, qui établit que l'aire sous la courbe en cloche e^{-x^2} vaut exactement $\sqrt{\pi}$:

```
$$\int_{-\infty}^{+\infty} e^{-x^2}\, dx = \sqrt{\pi}$$
```

$$\int_{-\infty}^{+\infty} e^{-x^2}\, dx = \sqrt{\pi}$$

où $x \in \mathbb{R}$ est la variable d'intégration et e^{-x^2} est la fonction gaussienne non normalisée. C'est cette identité qui fonde, après normalisation par $\sqrt{2\pi}\sigma$, la densité de la loi normale.

La Transformée de Fourier (décomposition) — extrait les fréquences d'un signal temporel pour révéler son contenu spectral :

```
$$\hat{f}(\xi) = \int_{-\infty}^{+\infty} f(t)\, e^{-i 2\pi \xi t}\, dt$$
```

$$\hat{f}(\xi) = \int_{-\infty}^{+\infty} f(t)\, e^{-i2\pi\xi t}\, dt$$

où $f(t)$ est le signal d'origine (fonction du temps t), $\hat{f}(\xi)$ est sa transformée (fonction de la fréquence ξ), et $e^{-i2\pi\xi t}$ est le noyau exponentiel complexe qui projette le signal sur chaque fréquence. La constante i désigne l'unité imaginaire ($i^2 = -1$).

La Transformée de Fourier inverse (reconstruction) — opération réciproque qui reconstruit le signal temporel à partir de son spectre :

```
$$f(t) = \int_{-\infty}^{+\infty} \hat{f}(\xi)\, e^{i 2\pi \xi t}\, d\xi$$
```

$$f(t) = \int_{-\infty}^{+\infty} \hat{f}(\xi)\, e^{i2\pi\xi t}\, d\xi$$

où les rôles de t et ξ sont inversés par rapport à la transformée directe : on intègre maintenant sur la variable de fréquence ξ, et le signe du noyau exponentiel passe de $-i$ à $+i$ (ce qui assure la réversibilité de l'opération).

1.3 Les chunks de code R

Un **chunk** est un bloc de code exécutable intégré au document. Le chunk ci-dessous génère la séquence des entiers de 1 à 20 :

```
seq(1, 20)   # Séquence de 1 à 20 par pas de 1
```

```
#>  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
1:20         # Notation compacte équivalente
```

```
#>  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

Les années de Coupe du Monde de football (de 1970 à 2026) :

```
seq(1970, 2026, by = 4)
```

```
#>  [1] 1970 1974 1978 1982 1986 1990 1994 1998 2002 2006 2010 2014 2018 2022 2026
```

À retenir

La fonction `seq(de, à, by = pas)` est l'une des plus utiles de R. On peut aussi écrire `1:20` pour une séquence d'entiers consécutifs — c'est la notation compacte équivalente à `seq(1, 20, by = 1)`.

1.4 Références bibliographiques dans Quarto

Quarto gère nativement les bibliographies via **BibTeX**. Il suffit d'indiquer le fichier `.bib` dans l'en-tête YAML et de citer avec `@cle_bib`. Par exemple, on peut citer l'article fondateur du Tidyverse : Wickham et al. (2019).

2 Le Tidyverse

2.1 Présentation

Le **Tidyverse** est un écosystème de packages R conçu pour la *data science*. Tous ces packages partagent une même philosophie, une même grammaire et des structures de données communes (les *tibbles*).

Analogie pédagogique

Le Tidyverse, c'est comme une boîte à outils bien organisée : chaque outil a un rôle précis, ils fonctionnent tous ensemble, et on apprend une fois la logique commune plutôt que de mémoriser des syntaxes disparates.

Parmi les packages les plus utilisés : `dplyr` (manipulation), `ggplot2` (visualisation), `tidyr` (restructuration), `readr` (importation), `lubridate` (dates) et `stringr` (chaînes de caractères).

2.2 Manipulation d'un data frame en R de base

Illustrons la manipulation de données avec **R de base** sur le jeu de données intégré `mtcars` (32 modèles d'automobiles) :

```
dfcars <- mtcars

dfcars$mpg          # Affiche les 32 valeurs de mpg (miles par gallon)
```

```
#>  [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4
#> [16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7
#> [31] 15.0 21.4
```

```
dfcars[, 1]         # Sélection de la 1ère colonne (toutes les lignes)
```

```
#>  [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4
#> [16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7
#> [31] 15.0 21.4
```

```
dfcars[1, ]         # Sélection de la 1ère ligne (toutes les colonnes)
```

```
#>           mpg cyl disp  hp drat   wt  qsec vs am gear carb
#> Mazda RX4  21   6  160 110  3.9 2.62 16.46  0  1    4    4
```

```
dfcars[1, 1]        # Valeur unique : 1ère colonne, 1ère ligne
```

```
#> [1] 21
```

2.3 Manipulations avec le Tidyverse et le pipe `|>`

Le **pipe natif** `|>` (disponible depuis R 4.1) permet de chaîner les opérations de gauche à droite :

```r
dfcars |> dplyr::select(mpg)       # Sélection par nom
```

```
#>                      mpg
#> Mazda RX4           21.0
#> Mazda RX4 Wag       21.0
#> Datsun 710          22.8
#> Hornet 4 Drive      21.4
#> Hornet Sportabout   18.7
#> Valiant             18.1
#> Duster 360          14.3
#> Merc 240D           24.4
#> Merc 230            22.8
#> Merc 280            19.2
#> Merc 280C           17.8
#> Merc 450SE          16.4
#> Merc 450SL          17.3
#> Merc 450SLC         15.2
#> Cadillac Fleetwood  10.4
#> Lincoln Continental 10.4
#> Chrysler Imperial   14.7
#> Fiat 128            32.4
#> Honda Civic         30.4
#> Toyota Corolla      33.9
#> Toyota Corona       21.5
#> Dodge Challenger    15.5
#> AMC Javelin         15.2
#> Camaro Z28          13.3
#> Pontiac Firebird    19.2
#> Fiat X1-9           27.3
#> Porsche 914-2       26.0
#> Lotus Europa        30.4
#> Ford Pantera L      15.8
#> Ferrari Dino        19.7
#> Maserati Bora       15.0
#> Volvo 142E          21.4
```

```r
dfcars |> dplyr::select(7:11)      # Sélection d'une plage de colonnes
```

```
#>                      qsec vs am gear carb
#> Mazda RX4           16.46  0  1    4    4
#> Mazda RX4 Wag       17.02  0  1    4    4
```

```
#> Datsun 710          18.61  1  1    4    1
#> Hornet 4 Drive      19.44  1  0    3    1
#> Hornet Sportabout   17.02  0  0    3    2
#> Valiant             20.22  1  0    3    1
#> Duster 360          15.84  0  0    3    4
#> Merc 240D           20.00  1  0    4    2
#> Merc 230            22.90  1  0    4    2
#> Merc 280            18.30  1  0    4    4
#> Merc 280C           18.90  1  0    4    4
#> Merc 450SE          17.40  0  0    3    3
#> Merc 450SL          17.60  0  0    3    3
#> Merc 450SLC         18.00  0  0    3    3
#> Cadillac Fleetwood  17.98  0  0    3    4
#> Lincoln Continental 17.82  0  0    3    4
#> Chrysler Imperial   17.42  0  0    3    4
#> Fiat 128            19.47  1  1    4    1
#> Honda Civic         18.52  1  1    4    2
#> Toyota Corolla      19.90  1  1    4    1
#> Toyota Corona       20.01  1  0    3    1
#> Dodge Challenger    16.87  0  0    3    2
#> AMC Javelin         17.30  0  0    3    2
#> Camaro Z28          15.41  0  0    3    4
#> Pontiac Firebird    17.05  0  0    3    2
#> Fiat X1-9           18.90  1  1    4    1
#> Porsche 914-2       16.70  0  1    5    2
#> Lotus Europa        16.90  1  1    5    2
#> Ford Pantera L      14.50  0  1    5    4
#> Ferrari Dino        15.50  0  1    5    6
#> Maserati Bora       14.60  0  1    5    8
#> Volvo 142E          18.60  1  1    4    2
```

2.4 Statistiques descriptives

La fonction `summarise()` calcule des statistiques agrégées :

```
stat_des <- dfcars |>
  select(mpg) |>
  summarise(
    Moyenne      = mean(mpg),
    `Ecart-type` = sd(mpg),
    Minimum      = min(mpg),
    Maximum      = max(mpg)
  )
```

```r
stat_des |>
  kable(
    digits   = 2,
    booktabs = TRUE
  ) |>
  kable_styling(latex_options = c("hold_position", "striped"), font_size = 10)
```

Table 1. Statistiques descriptives de la variable mpg (mtcars)

Moyenne	Ecart-type	Minimum	Maximum
20.09	6.03	10.4	33.9

3 Introduction à la visualisation avec R

3.1 Histogramme de base

```
dfcars$mpg |> hist(
  main   = "Histogramme de MPG",
  xlab   = "Miles par gallon",
  col    = "steelblue",
  border = "white"
)
```

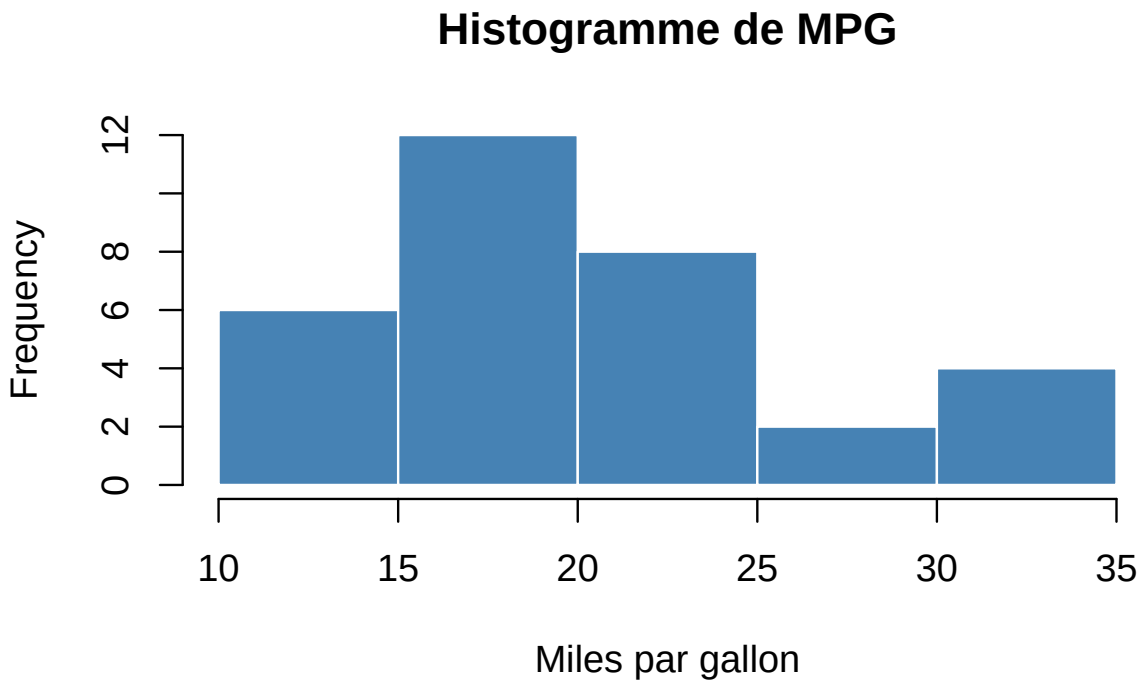

Figure 1. Histogramme de la consommation des 32 véhicules (R de base)

Filtrons les voitures les plus économiques :

```
dfcars |> filter(mpg >= 30 & mpg <= 35)
```

```
#>                 mpg cyl disp  hp drat    wt  qsec vs am gear carb
#> Fiat 128       32.4   4 78.7  66 4.08 2.200 19.47  1  1    4    1
#> Honda Civic    30.4   4 75.7  52 4.93 1.615 18.52  1  1    4    2
#> Toyota Corolla 33.9   4 71.1  65 4.22 1.835 19.90  1  1    4    1
#> Lotus Europa   30.4   4 95.1 113 3.77 1.513 16.90  1  1    5    2
```

3.2 Histogramme avec ggplot2

```r
dfcars |>
  ggplot(aes(x = mpg)) +
  geom_histogram(fill = "#a12b4e", binwidth = 5, color = "white") +
  labs(
    title    = "Distribution de la consommation des 32 véhicules",
    subtitle = "Données : jeu intégré mtcars",
    x        = "Miles par gallon (mpg)",
    y        = "Nombre de véhicules"
  )
```

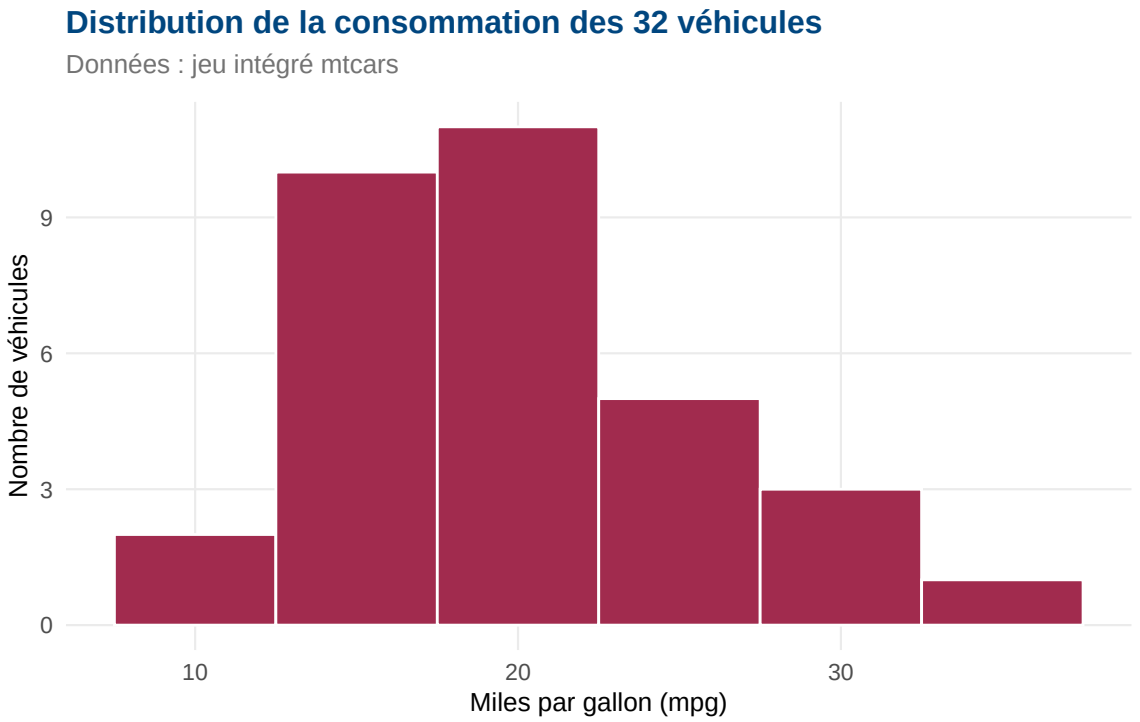

Figure 2. Distribution de la consommation (ggplot2)

Point d'attention

La bibliothèque `ggthemes` (utilisée dans la version HTML) n'est pas indispensable pour la version PDF. Le thème `theme_minimal()` est tout à fait adapté aux sorties imprimées.

Partie II — Projet Keno : analyse des données FDJ

4 Projet Keno — Analyse des données FDJ

4.1 Présentation du Keno

Le **Keno** est un jeu de tirage proposé par la Française des Jeux (FDJ). À chaque tirage, **16 boules** sont tirées aléatoirement parmi **56**, numérotées de 1 à 56. Le joueur coche jusqu'à 10 numéros sur sa grille, et ses gains dépendent du nombre de numéros cochés figurant parmi les 16 boules tirées. Toutes les analyses de ce document portent sur cette version : **16 boules parmi 56**.

4.2 Chargement des données

```
keno_202511 <- read_delim(
  "keno_202511.csv",
  delim         = ";",
  escape_double = FALSE,
  col_types     = cols(
    date_de_forclusion = col_skip(),
    ...23              = col_skip()
  ),
  trim_ws       = TRUE
)

keno_202511 <- keno_202511 |> select(-c(1, 19, 20, 21))

names(keno_202511)
```

```
#>  [1] "date_de_tirage" "boule1"         "boule2"         "boule3"
#>  [5] "boule4"         "boule5"         "boule6"         "boule7"
#>  [9] "boule8"         "boule9"         "boule10"        "boule11"
#> [13] "boule12"        "boule13"        "boule14"        "boule15"
#> [17] "boule16"
```

4.3 Pivot longer : restructuration des données

4.3.1 Le concept de pivot

Les données Keno arrivent au format **large** : 1 ligne par tirage, 16 colonnes `boule1` à `boule16`. Pour les analyses, on préfère le format **long** : 1 ligne par boule, avec une colonne numéro.

Exemple illustratif avec des données sur les cas de maladie :

```
table4a <- tibble(
  country = c("A", "B", "C"),
  `1999`  = c(700,  37000,  212000),
  `2000`  = c(2000, 80000,  213000)
)

table4a |>
```

```
  pivot_longer(
    cols      = 2:3,
    names_to  = "annee",
    values_to = "cas"
  )
```

```
#> # A tibble: 6 x 3
#>   country annee    cas
#>   <chr>   <chr>  <dbl>
#> 1 A       1999     700
#> 2 A       2000    2000
#> 3 B       1999   37000
#> 4 B       2000   80000
#> 5 C       1999  212000
#> 6 C       2000  213000
```

4.3.2 Application au Keno

```
keno_longer <- keno_202511 |>
  pivot_longer(
    cols         = 2:17,
    names_to     = "Boule",
    names_prefix = "boule",
    values_to    = "Num_boule"
  ) |>
  mutate(
    Boule     = as.integer(Boule),
    Num_boule = as.integer(Num_boule)
  )

glimpse(keno_longer)
```

```
#> Rows: 1,600
#> Columns: 3
#> $ date_de_tirage <chr> "10/02/2026", "10/02/2026", "10/02/2026", "10/02/2026",~
#> $ Boule          <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ~
#> $ Num_boule      <int> 12, 13, 17, 20, 22, 23, 26, 27, 39, 42, 45, 48, 50, 52,~
```

4.4 Tableau de fréquence des boules

```
freq_boules <- keno_longer |>
  dplyr::count(Num_boule, name = "frequence")

freq_boules |> filter(frequence == min(frequence))
```

```
#> # A tibble: 1 x 2
#>   Num_boule frequence
#>       <int>     <int>
#> 1         5        17
```

4.5 Formatage des dates

```
keno_202511 <- keno_202511 |>
  mutate(date_de_tirage = as.Date(date_de_tirage, format = "%d/%m/%Y"))

keno_202511 |>
  summarise(
    date_min = format(min(date_de_tirage), "%d/%m/%Y"),
    date_max = format(max(date_de_tirage), "%d/%m/%Y")
  )
```

```
#> # A tibble: 1 x 2
#>   date_min   date_max
#>   <chr>      <chr>
#> 1 03/11/2025 10/02/2026
```

4.6 Palmarès des numéros — Tableau statistique

Ce tableau s'inspire du **palmarès des numéros** du site FDJ. Pour chaque numéro de 1 à 56, on affiche le nombre de sorties, le pourcentage de présence et la date de dernière sortie. La boîte ci-dessous précise la formule de calcul utilisée et fournit la référence théorique permettant d'interpréter correctement les valeurs affichées.

Rappel théorique

Calcul du pourcentage de présence. Pour chaque numéro $i \in \{1, \ldots, 56\}$, on note S_i le nombre de tirages dans lesquels il est sorti. Avec N tirages historiques disponibles, le pourcentage de présence du numéro i est défini par :

$$\text{Pct}_i \;=\; \frac{S_i}{N} \times 100$$

où S_i est le nombre de sorties du numéro i et N le nombre total de tirages. Cette formule est strictement **identique au calcul officiel publié par la FDJ** sur sa page de statistiques Keno.

Référence théorique. À chaque tirage, 16 boules sont tirées parmi 56. La probabilité qu'un numéro donné figure dans un tirage est donc :

$$p_{\text{théo}} \;=\; \frac{16}{56} \;\approx\; 0{,}2857 \;=\; \mathbf{28{,}57}\,\%$$

Table 2. Palmarès des numéros du Keno FDJ — inspiré du design officiel FDJ.

Numéros 1 à 28				Numéros 29 à 56			
Numéro	**Sorties**	**%**	**Dernière**	**Numéro**	**Sorties**	**%**	**Dernière**
1 •	30	30.00%	09/02/26	**29** •	26	26.00%	08/02/26
2 •	26	26.00%	08/02/26	**30** •	35	35.00%	06/02/26
3 •	28	28.00%	09/02/26	**31** •	29	29.00%	09/02/26
4 •	32	32.00%	09/02/26	**32** •	24	24.00%	30/01/26
5 •	17	17.00%	02/02/26	**33** •	24	24.00%	28/01/26
6 •	21	21.00%	05/02/26	**34** •	25	25.00%	02/02/26
7 •	37	37.00%	08/02/26	**35** •	29	29.00%	08/02/26
8 •	23	23.00%	04/02/26	**36** •	24	24.00%	09/02/26
9 •	33	33.00%	09/02/26	**37** •	29	29.00%	03/02/26
10 •	30	30.00%	02/02/26	**38** •	31	31.00%	05/02/26
11 •	28	28.00%	09/02/26	**39** •	27	27.00%	10/02/26
12 •	28	28.00%	10/02/26	**40** •	27	27.00%	08/02/26
13 •	38	38.00%	10/02/26	**41** •	38	38.00%	06/02/26
14 •	35	35.00%	09/02/26	**42** •	30	30.00%	10/02/26
15 •	21	21.00%	09/02/26	**43** •	25	25.00%	08/02/26
16 •	23	23.00%	06/02/26	**44** •	30	30.00%	08/02/26
17 •	33	33.00%	10/02/26	**45** •	25	25.00%	10/02/26
18 •	19	19.00%	04/02/26	**46** •	32	32.00%	09/02/26
19 •	38	38.00%	08/02/26	**47** •	31	31.00%	02/02/26
20 •	24	24.00%	10/02/26	**48** •	30	30.00%	10/02/26
21 •	29	29.00%	08/02/26	**49** •	32	32.00%	07/02/26
22 •	20	20.00%	10/02/26	**50** •	30	30.00%	10/02/26
23 •	29	29.00%	10/02/26	**51** •	30	30.00%	08/02/26
24 •	29	29.00%	02/02/26	**52** •	29	29.00%	10/02/26
25 •	27	27.00%	09/02/26	**53** •	32	32.00%	09/02/26
26 •	28	28.00%	10/02/26	**54** •	24	24.00%	04/02/26
27 •	36	36.00%	10/02/26	**55** •	33	33.00%	10/02/26
28 •	33	33.00%	05/02/26	**56** •	24	24.00%	10/02/26

Lecture du tableau. Un numéro affichant un pourcentage proche de **28,57 %** est conforme à la loi théorique. Un écart vers le haut indique un numéro *sur-représenté* sur la période, vers le bas un numéro *sous-représenté*. Avec seulement $N = 100$ tirages, ces écarts sont attendus et relèvent de la fluctuation d'échantillonnage — ils n'ont aucune valeur prédictive pour les tirages futurs.

4.7 Statistiques sur le format long

```
keno_long <- keno_202511 |>
  pivot_longer(
    cols            = boule1:boule16,
    names_to        = "boule",
    names_prefix    = "boule",
    names_transform = list(boule = as.integer),
    values_to       = "numero"
```

```
  )

table_freq <- keno_long |>
  dplyr::count(numero, name = "Nombre de sorties") |>
  complete(numero = 1:56, fill = list(`Nombre de sorties` = 0)) |>
  arrange(numero)

n_half <- ceiling(nrow(table_freq) / 2)
col1   <- table_freq[1:n_half, ]
col2   <- table_freq[(n_half + 1):nrow(table_freq), ]

bind_cols(col1, col2) |>
  setNames(c("Numéro", "Sorties", "Numéro", "Sorties")) |>
  kable(
    booktabs = TRUE,
    linesep  = "",
    align    = c("c","c","c","c")
  ) |>
  kable_styling(
    latex_options = c("hold_position", "striped", "scale_down"),
    font_size     = 9
  ) |>
  column_spec(2, border_right = TRUE) |>
  add_header_above(c("Numéros 1 à 28" = 2, "Numéros 29 à 56" = 2), bold = TRUE)
```

Table 3. Fréquence d'apparition des numéros (Keno — données FDJ)

Numéros 1 à 28		**Numéros 29 à 56**	
Numéro	Sorties	Numéro	Sorties
1	30	29	26
2	26	30	35
3	28	31	29
4	32	32	24
5	17	33	24
6	21	34	25
7	37	35	29
8	23	36	24
9	33	37	29
10	30	38	31
11	28	39	27
12	28	40	27
13	38	41	38
14	35	42	30
15	21	43	25
16	23	44	30
17	33	45	25
18	19	46	32
19	38	47	31

20	24	48	30
21	29	49	32
22	20	50	30
23	29	51	30
24	29	52	29
25	27	53	32
26	28	54	24
27	36	55	33
28	33	56	24

5 Visualisations graphiques du Keno

5.1 Histogramme des fréquences

```
freq_boules_plot <- keno_long |>
  dplyr::count(numero, name = "frequence") |>
  complete(numero = 1:56, fill = list(frequence = 0)) |>
  arrange(numero)

freq_boules_plot |>
  ggplot(aes(x = numero, y = frequence, fill = frequence)) +
  geom_bar(stat = "identity", color = "white", width = 0.8) +
  scale_fill_gradient(low = "green", high = "blue") +
  labs(
    title    = "Fréquence d'apparition de chaque numéro du Keno",
    subtitle = "Données FDJ - tirages du midi et du soir",
    x        = "Numéro de boule (1 à 56)",
    y        = "Nombre de sorties",
    fill     = "Fréquence"
  ) +
  theme(legend.position = "right")
```

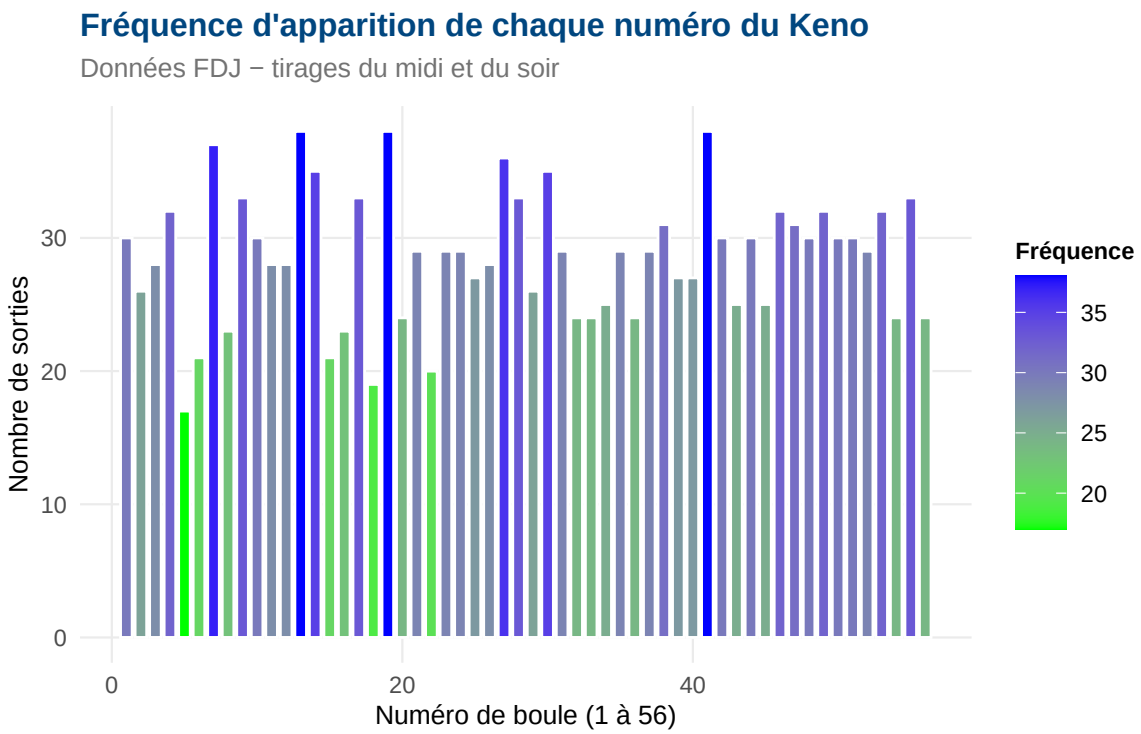

Figure 3. Fréquence d'apparition de chaque numéro du Keno (1 à 56)

5.2 Graphique circulaire (diagramme en rose)

```r
freq_boules_plot |>
  ggplot(aes(
    x    = reorder(as.factor(numero), as.numeric(numero)),
    y    = frequence,
    fill = frequence
  )) +
  geom_bar(stat = "identity", show.legend = FALSE, color = "white", width = 1) +
  coord_polar(theta = "x", clip = "off") +
  geom_text(
    aes(y = frequence + 2, label = numero),
    color    = "black",
    size     = 2.8,
    fontface = "bold"
  ) +
  scale_fill_gradient(low = "green", high = "blue") +
  scale_y_continuous(
    limits = c(-4, max(freq_boules_plot$frequence) + 4),
    expand = expansion(mult = c(0, 0))
  ) +
  labs(
    title    = "Répartition circulaire des fréquences",
    subtitle = "Chaque secteur représente un numéro (1 à 56)"
  ) +
  theme_void() +
  theme(
    plot.margin   = margin(t = 40, r = -50, b = -30, l = -50, unit = "pt"),
    plot.title    = element_text(face = "bold", hjust = 0.5, size = 14),
    plot.subtitle = element_text(hjust = 0.5, color = "grey40", size = 10,
                                 margin = margin(b = 10)),
    aspect.ratio  = 1
  )
```

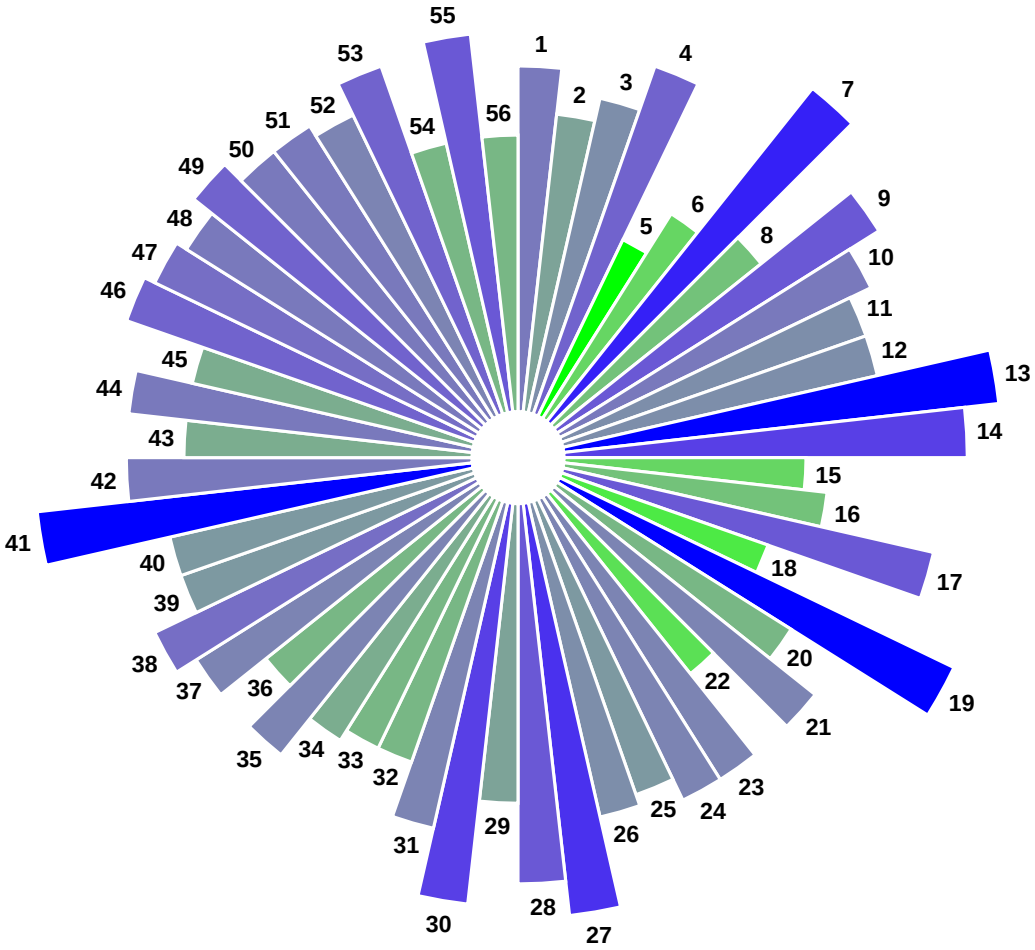

Figure 4. Répartition circulaire des fréquences du Keno

6 Modélisation probabiliste du Keno

6.1 La loi hypergéométrique

Rappel théorique

Le Keno est un tirage **sans remise** : les boules ne sont pas replacées. La distribution de probabilité adaptée est la **loi hypergéométrique** :

```
$$P(X = x) = \frac{\binom{K}{x}\binom{N-K}{n-x}}{\binom{N}{n}}$$
```

$$P(X = x) = \frac{\binom{K}{x}\binom{N-K}{n-x}}{\binom{N}{n}}$$

avec : $N = 56$ (taille de l'urne), $K = 16$ (boules tirées par le Keno), $n = 10$ (numéros cochés par le joueur), x (nombre de bons numéros).

La fonction R correspondante est `dhyper(x, m, n, k)`.

6.2 Calcul des probabilités de gain

```
tibble(
  x      = 0:10,
  p_x    = dhyper(0:10, m = 16, n = 40, k = 10),
  chance = round(1 / p_x)
) |>
  arrange(desc(x)) |>
  kable(
    digits    = 6,
    booktabs  = TRUE,
    col.names = c("x (bonnes boules)", "p(x)", "1 chance sur...")
  ) |>
  kable_styling(
    latex_options = c("hold_position", "striped"),
    font_size     = 10
  )
```

Table 4. Probabilités hypergéométriques — Keno (10 numéros cochés, 16 tirés sur 56)

x (bonnes boules)	p(x)	1 chance sur...
10	0.000000	4446435
9	0.000013	77813
8	0.000282	3547
7	0.003174	315
6	0.020554	49
5	0.080719	12

4	0.196193	5
3	0.293211	3
2	0.259178	4
1	0.122870	8
0	0.023806	42

6.3 Tableau des gains officiels FDJ

```
tableau_gains <- tibble(
  x      = 0:10,
  `p(x)` = dhyper(x, m = 16, n = 40, k = 10)
) |>
  arrange(desc(x)) |>
  filter(!x %in% 1:4) |>
  mutate(
    `g(x) (en euros)` = c(200000, 2000, 150, 15, 5, 2, 2),
    .after             = x
  )

tableau_gains |>
  kable(
    digits    = 6,
    booktabs  = TRUE,
    col.names = c("Bonnes boules (x)", "Gain g(x) (en euros)", "Probabilité p(x)")
  ) |>
  kable_styling(
    latex_options = c("hold_position", "striped"),
    font_size     = 10
  )
```

Table 5. Gains officiels FDJ et probabilités (mise de 1 euro, 10 numéros cochés)

Bonnes boules (x)	Gain g(x) (en euros)	Probabilité p(x)
10	200000	0.000000
9	2000	0.000013
8	150	0.000282
7	15	0.003174
6	5	0.020554
5	2	0.080719
0	2	0.023806

7 Référence officielle FDJ : tableau complet des gains

7.1 Les tableaux de gains officiels

Les probabilités et gains présentés dans cette section sont calculés à partir des règles officielles du Keno FDJ. Les sources officielles sont consultables en ligne :

— **Probabilités et chances de gagner** : fdj.fr — Quelles sont les chances de gagner au Kéno ?

— **Statistiques officielles des tirages** : fdj.fr — Kéno Statistiques

Le tableau ci-dessous reproduit le tableau officiel FDJ des lots du Keno, tel que publié dans le règlement de la Française des Jeux (article 8). Il couvre les grilles de 4 à 10 numéros cochés et précise, pour chaque combinaison, les probabilités de gain et les montants des lots pour une mise de 1,€.

LA FRANCAISE DES JEUX

Article 8 - Tableau de lots Keno

8.1. Une combinaison de numéros est déclarée gagnante au jeu Keno pour un montant déterminé, conformément au tableau de lots Keno ci-dessous :

Numéros cochés par grille	Probabilité globale de gagner par grille selon le nombre de numéros cochés	Numéros trouvés par grille	Probabilité globale de gagner par grille selon le nombre de numéros cochés et trouvés*	Montants des lots en € pour une mise de 1€
10 numéros	1 chance sur 7,78	10	1 chance sur 4 446 435	200 000 € cash ou 10 000 € par an à vie
		9	1 chance sur 77 813	2 000 €
		8	1 chance sur 3 547	150 €
		7	1 chance sur 315	15 €
		6	1 chance sur 49	5 €
		5	1 chance sur 12	2 €
		0	1 chance sur 42	2 €
9 numéros	1 chance sur 3,87	9	1 chance sur 662 235	30 000 € cash
		8	1 chance sur 14 716	100 €
		7	1 chance sur 849	25 €
		6	1 chance sur 96	8 €
		5	1 chance sur 19	2 €
		4	1 chance sur 6	1 €
		0	1 chance sur 28	2 €
8 numéros	1 chance sur 11,20	8	1 chance sur 110 373	8 000 €
		7	1 chance sur 3 104	100 €
		6	1 chance sur 227	30 €
		5	1 chance sur 33	5 €
		0	1 chance sur 18	2 €
7 numéros	1 chance sur 10,68	7	1 chance sur 20 273	3 000 €
		6	1 chance sur 724	90 €
		5	1 chance sur 68	5 €
		4	1 chance sur 13	2 €
6 numéros	1 chance sur 20,26	6	1 chance sur 4 055	900 €
		5	1 chance sur 186	30 €
		4	1 chance sur 23	3 €
5 numéros	1 chance sur 7,43	5	1 chance sur 875	80 €
		4	1 chance sur 52	10 €
		3	1 chance sur 9	2 €
4 numéros	1 chance sur 15,16	4	1 chance sur 202	70 €
		3	1 chance sur 16	3 €

*Arrondies à l'unité la plus proche

Pour une mise de 2, 3, 5 ou 10 €, les lots mentionnés dans la dernière colonne du tableau ci-dessus sont à multiplier respectivement par 2, 3, 5 ou 10.

10/20

Figure 5. Tableau officiel FDJ des lots du Keno — article 8 du règlement FDJ (mise de 1 €). Source : Française des Jeux.

7.2 Construction de la table complète des gains FDJ

Nous reproduisons ici le tableau des gains pour **toutes les grilles** (de 2 à 10 numéros cochés), en calculant les probabilités théoriques avec `dhyper()` et en renseignant les gains officiels conformément au règlement FDJ (mise de 1 €). Toutes les probabilités sont calculées dans le modèle **16 boules tirées sur 56**, qui est le modèle correspondant aux données historiques disponibles dans `keno_202511.csv`.

```
# ------------------------------------------------------------------------
# Paramètres du Keno FDJ - modèle utilisé dans les données historiques
# disponibles : 16 boules tirées sur 56 (keno_202511.csv)
# ------------------------------------------------------------------------
N_keno <- 56   # taille de l'urne
K_keno <- 16   # boules tirées par tirage

# ------------------------------------------------------------------------
# Table des gains officiels FDJ (mise 1 €) - source : article 8 du
# règlement FDJ (tableau-gain-keno-2025.pdf), modèle 16/56.
# Format : list(n_coches = list(n_trouves = gain_en_euros))
# Seules les combinaisons donnant lieu à un gain figurent ici.
# ------------------------------------------------------------------------
gains_fdj <- list(
  `10` = list(`10` = 200000, `9` = 2000,  `8` = 150, `7` = 15,
              `6`  = 5,      `5` = 2,     `0` = 2),
  `9`  = list(`9`  = 30000,  `8` = 100,   `7` = 25,  `6` = 8,
              `5`  = 2,      `4` = 1,     `0` = 2),
  `8`  = list(`8`  = 8000,   `7` = 100,   `6` = 30,  `5` = 5,  `0` = 2),
  `7`  = list(`7`  = 3000,   `6` = 90,    `5` = 5,   `4` = 2),
  `6`  = list(`6`  = 900,    `5` = 30,    `4` = 3),
  `5`  = list(`5`  = 80,     `4` = 10,    `3` = 2),
  `4`  = list(`4`  = 70,     `3` = 3),
  `3`  = list(`3`  = 10,     `2` = 2),
  `2`  = list(`2`  = 6)
)

message("Table des gains FDJ chargée : ", length(gains_fdj), " grilles définies (2 à 10 numéros).")
```

```
# ------------------------------------------------------------------------
# Construction du tableau long : une ligne par combinaison (n_cochés, n_trouvés)
# avec probabilité théorique (dhyper) et gain officiel FDJ
# ------------------------------------------------------------------------
df_gains_long <- purrr::imap_dfr(gains_fdj, function(gains_grille, n_coches_chr) {
  n <- as.integer(n_coches_chr)
  purrr::imap_dfr(gains_grille, function(gain, n_trouves_chr) {
    x <- as.integer(n_trouves_chr)
    p <- dhyper(x, m = K_keno, n = N_keno - K_keno, k = n)
    tibble(
      n_coches   = n,
      n_trouves  = x,
      prob       = p,
```

```
      chance     = if (p > 0) round(1 / p) else NA_real_,
      gain_1eur  = gain,
      gain_10eur = gain * 10
    )
  })
}) |>
  arrange(desc(n_coches), desc(n_trouves))

message("Tableau long construit : ", nrow(df_gains_long), " lignes.")
```

```
# ------------------------------------------------------------------------
# Affichage détaillé : grille à 10 numéros cochés (la plus complète)
# ------------------------------------------------------------------------
df_gains_long |>
  filter(n_coches == 10) |>
  select(
    `N° trouvés`      = n_trouves,
    `Probabilité`           = prob,
    `1 chance sur...`   = chance,
    `Gain (1 €)`  = gain_1eur,
    `Gain (10 €)` = gain_10eur
  ) |>
  kable(
    digits      = 7,
    booktabs    = TRUE,
    format.args = list(big.mark = " ")
  ) |>
  kable_styling(
    latex_options     = c("hold_position", "striped"),
    bootstrap_options = c("striped", "hover", "condensed"),
    font_size         = 10,
    full_width        = FALSE
  ) |>
  row_spec(1, bold = TRUE, color = "black", background = "gray") |>
  column_spec(4, bold = TRUE)
```

Table 6. Grille à 10 numéros cochés — probabilités et gains officiels FDJ (16 boules sur 56, mise 1 euro)

N° trouvés	Probabilité	1 chance sur...	Gain (1 €)	Gain (10 €)
10	**0.0000002**	**4 446 435**	**200 000**	**2 000 000**
9	0.0000129	77 813	**2 000**	20 000
8	0.0002819	3 547	**150**	1 500
7	0.0031743	315	**15**	150
6	0.0205535	49	**5**	50
5	0.0807194	12	**2**	20
0	0.0238060	42	**2**	20

```r
# ------------------------------------------------------------------------
# Tableau synthétique toutes grilles - mise en forme publication
# Les indices de groupe sont calculés dynamiquement pour rester robustes.
# ------------------------------------------------------------------------
groupes <- df_gains_long |>
  mutate(row_id = row_number()) |>
  group_by(n_coches) |>
  summarise(debut = min(row_id), fin = max(row_id), .groups = "drop") |>
  arrange(desc(n_coches))

tbl_complet <- df_gains_long |>
  mutate(
    prob_fmt   = formatC(prob,   format = "e", digits = 3),
    chance_fmt = formatC(chance, format = "fg", big.mark = " ",
                         flag = "#") |> stringr::str_trim(),
    gain_fmt   = paste0(formatC(gain_1eur, format = "fg",
                                big.mark = " "), " €")
  ) |>
  select(
    `Grille`       = n_coches,
    `N° trouvés` = n_trouves,
    `p(x)`         = prob_fmt,
    `1 chance sur` = chance_fmt,
    `Gain (1 €)` = gain_fmt
  )

tbl_obj <- tbl_complet |>
  kable(
    booktabs = TRUE,
    align    = c("c","c","r","r","r"),
    linesep  = ""
  ) |>
  kable_styling(
    latex_options     = c("hold_position", "striped", "scale_down"),
    bootstrap_options = c("striped", "hover", "condensed"),
    font_size         = 9,
    full_width        = FALSE
  )

for (i in seq_len(nrow(groupes))) {
  tbl_obj <- tbl_obj |>
    pack_rows(
      paste0(groupes$n_coches[i], " numéros cochés"),
      groupes$debut[i],
      groupes$fin[i]
    )
}

tbl_obj
```

Table 7. Table complète des gains FDJ — toutes grilles de 2 à 10 numéros cochés (16 boules tirées sur 56, mise 1 euro)

Grille	N° trouvés	p(x)	1 chance sur	Gain (1 €)
10 numéros cochés				
10	10	2.249e-07	4 446 435	200 000 €
10	9	1.285e-05	77 813	2 000 €
10	8	2.819e-04	3 547	150 €
10	7	3.174e-03	315.0	15 €
10	6	2.055e-02	49.00	5 €
10	5	8.072e-02	12.00	2 €
10	0	2.381e-02	42.00	2 €
9 numéros cochés				
9	9	1.510e-06	662 235	30 000 €
9	8	6.795e-05	14 716	100 €
9	7	1.178e-03	849.0	25 €
9	6	1.044e-02	96.00	8 €
9	5	5.269e-02	19.00	2 €
9	4	1.581e-01	6.000	1 €
9	0	3.609e-02	28.00	2 €
8 numéros cochés				
8	8	9.060e-06	110 373	8 000 €
8	7	3.221e-04	3 104	100 €
8	6	4.397e-03	227.0	30 €
8	5	3.038e-02	33.00	5 €
8	0	5.414e-02	18.00	2 €
7 numéros cochés				
7	7	4.933e-05	20 272	3 000 €
7	6	1.381e-03	724.0	90 €
7	5	1.469e-02	68.00	5 €
7	4	7.753e-02	13.00	2 €
6 numéros cochés				
6	6	2.466e-04	4 055	900 €
6	5	5.381e-03	186.0	30 €
6	4	4.372e-02	23.00	3 €
5 numéros cochés				
5	5	1.144e-03	874.0	80 €
5	4	1.906e-02	52.00	10 €
5	3	1.144e-01	9.000	2 €
4 numéros cochés				
4	4	4.955e-03	202.0	70 €
4	3	6.099e-02	16.00	3 €
3 numéros cochés				
3	3	2.020e-02	49.00	10 €
3	2	1.732e-01	6.000	2 €
2 numéros cochés				
2	2	7.792e-02	13.00	6 €

7.3 Espérances de gain par grille

Rappel théorique

Pour chaque grille (nombre de numéros cochés n), l'espérance de gain se calcule en sommant les gains pondérés par leurs probabilités, sur toutes les combinaisons gagnantes :

```
$$E_n[G] = \sum_{x \in \mathcal{G}_n} g(x) \cdot p(x)$$
```

$$E_n[G] = \sum_{x \in \mathcal{G}_n} g(x) \cdot p(x)$$

où $\mathcal{G}_n$ est l'ensemble des valeurs de x (nombre de bons numéros) donnant lieu à un gain pour la grille n.

```r
# ------------------------------------------------------------------------
# Calcul de l'espérance de gain pour chaque grille (mise 1 €)
# et du taux de retour joueur (TRJ)
# ------------------------------------------------------------------------
df_esperances <- df_gains_long |>
  group_by(n_coches) |>
  summarise(
    esperance = sum(gain_1eur * prob, na.rm = TRUE),
    .groups   = "drop"
  ) |>
  mutate(
    trj_pct   = round(esperance * 100, 2),
    perte_moy = round(1 - esperance, 4)
  ) |>
  arrange(desc(n_coches))

df_esperances |>
  select(
    `N° cochés`             = n_coches,
    `Espérance E[G]`        = esperance,
    `TRJ (%)`               = trj_pct,
    `Perte moy. / tirage (EUR)` = perte_moy
  ) |>
  kable(
    digits   = 4,
    booktabs = TRUE
  ) |>
  kable_styling(
    latex_options     = c("hold_position", "striped"),
    bootstrap_options = c("striped", "hover", "condensed"),
    font_size         = 10,
    full_width        = FALSE
  ) |>
  column_spec(3, bold = TRUE, color = "black",
```

```
            background = spec_color(df_esperances$trj_pct,
                                    option = "C", direction = -1,
                                    begin  = 0.3, end = 0.9))
```

Table 8. Espérance de gain et taux de retour joueur (TRJ) par grille — Keno FDJ (mise 1 euro)

N° cochés	Espérance E[G]	TRJ (%)	Perte moy. / tirage (EUR)
10	0.4724	**47.24**	0.5276
9	0.5007	**50.07**	0.4993
8	0.4968	**49.68**	0.5032
7	0.5008	**50.08**	0.4992
6	0.5146	**51.46**	0.4854
5	0.5108	**51.08**	0.4892
4	0.5298	**52.98**	0.4702
3	0.5483	**54.83**	0.4517
2	0.4675	**46.75**	0.5325

À retenir

Le **taux de retour joueur (TRJ)** est la fraction de la mise que le joueur récupère **en moyenne** sur un très grand nombre de parties. Un TRJ de 50 % signifie que pour 1 € joué, le joueur récupère en moyenne 0,50 € et perd donc 0,50 €. Ce taux est encadré réglementairement par l'ARJEL (Autorité de régulation des jeux en ligne).

```
df_esperances |>
  ggplot(aes(x = factor(n_coches), y = esperance)) +
  geom_col(aes(fill = esperance), width = 0.65, show.legend = FALSE) +
  geom_hline(yintercept = 1, linetype = "dashed",
             color = "#a12b4e", linewidth = 0.7) +
  geom_text(aes(label = paste0(trj_pct, " %")),
            vjust = -0.5, size = 3.2, fontface = "bold", color = "gray20") +
  scale_fill_gradient(low = "#e8c4d0", high = "#a12b4e") +
  scale_y_continuous(
    labels = scales::label_number(suffix = " EUR", accuracy = 0.01),
    limits = c(0, 1.10),
    expand = expansion(mult = c(0, 0.02))
  ) +
  annotate("text", x = 0.6, y = 1.03, label = "Mise = 1 EUR",
           color = "#a12b4e", size = 3, hjust = 0, fontface = "italic") +
  labs(
    title    = "Esperance de gain selon le nombre de numeros coches",
    subtitle = "Keno FDJ --- mise de 1 euro --- 16 boules tirees sur 56",
    x        = "Nombre de numeros coches",
    y        = "Esperance de gain (EUR)",
    caption  = "Source : gains officiels FDJ. La ligne pointillee rouge materialise la mise de 1
    EUR.\nEn dessous de cette ligne, le joueur perd en esperance."
```

```
) +
theme_pdf() +
theme(panel.grid.major.x = element_blank())
```

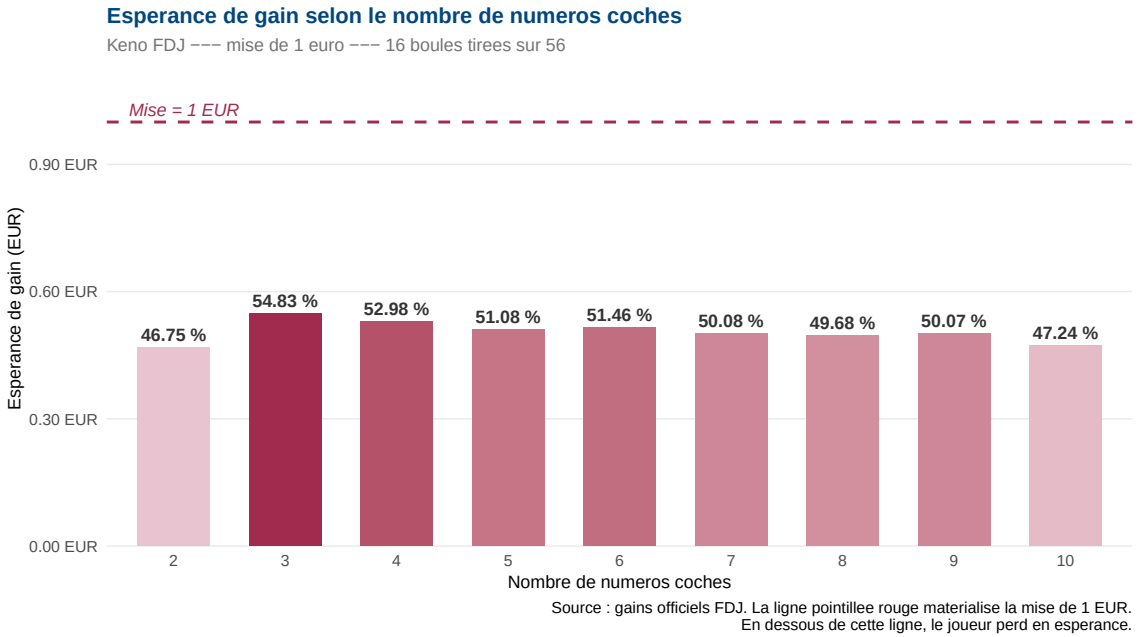

Figure 6. Espérance de gain selon le nombre de numéros cochés — Keno FDJ (mise 1 euro, modèle 16/56). Toutes les barres restent sous la ligne pointillée rouge, confirmant que l'espérance est toujours négative pour le joueur.

Point d'attention

Lecture du graphique : Toutes les barres sont **sous la ligne pointillée rouge** (la mise de 1 €), confirmant que l'**espérance de gain est toujours négative pour le joueur**, quelle que soit la grille choisie. Le TRJ affiché au-dessus de chaque barre (en % de la mise) montre que la FDJ reverse entre 50 % et 60 % de la mise en gains — la différence constituant la marge opératrice.

8 Espérance de gain

8.1 Définition et formule

Rappel théorique

L'**espérance mathématique** de gain est la valeur moyenne que le joueur peut espérer gagner par tirage. Elle se calcule comme la somme des gains pondérés par leurs probabilités :

```
$$E[G] = \sum_{x=0}^{10} g(x) \cdot p(x)$$
```

$$E[G] = \sum_{x=0}^{10} g(x) \cdot p(x)$$

où $g(x)$ est le gain pour x bonnes boules, et $p(x)$ la probabilité d'en obtenir exactement x.

8.2 Calcul avec R

```
result <- tibble(
  x      = 0:10,
  `p(x)` = dhyper(x, m = 16, n = 40, k = 10)
) |>
  arrange(desc(x)) |>
  filter(!x %in% 1:4) |>
  mutate(`g(x)` = c(200000, 2000, 150, 15, 5, 2, 2), .after = x)

esperance <- result |>
  summarise(esperance = sum(`g(x)` * `p(x)`))

cat("Espérance de gain E[G] =", round(esperance$esperance, 4), "euro(s) pour une mise de 1 euro.")
```

```
#> Espérance de gain E[G] = 0.4724 euro(s) pour une mise de 1 euro.
```

Point d'attention

L'espérance de gain est inférieure à 1 euro (la mise). Cela signifie que le joueur perd en moyenne de l'argent à chaque tirage — c'est la **marge** intégrée par la FDJ dans le jeu. Ce résultat est fondamental en théorie des jeux et illustre pourquoi les jeux d'argent ne peuvent pas être des stratégies de gain à long terme.

9 Simulations de Monte-Carlo

9.1 Principe

La **simulation de Monte-Carlo** consiste à simuler un grand nombre de tirages aléatoires pour estimer des probabilités ou des distributions. On l'utilise ici pour estimer la probabilité qu'un numéro sorte très peu de fois sur une série de 100 tirages.

9.2 Simulation simple avec une boucle

```
set.seed(123)

numeros <- c()
for (t in 1:100) {
  numeros <- c(numeros, sample(1:56, 16, replace = FALSE))
}
cat("Minimum d'apparition sur 100 tirages :", table(numeros) |> min())
```

```
#> Minimum d'apparition sur 100 tirages : 20
```

Équivalent avec `replicate()` — plus concis et plus rapide :

```
set.seed(123)
numeros_simu <- as.vector(replicate(100, sample(1:56, 16, replace = FALSE)))
cat("Minimum (replicate) :", table(numeros_simu) |> min())
```

```
#> Minimum (replicate) : 20
```

9.3 Estimation d'une probabilité

On répète l'expérience $N = 100$ fois pour estimer la probabilité qu'un numéro sorte 17 fois ou moins :

```
set.seed(123)
N <- 100

valmin <- replicate(N, {
  tirages <- replicate(100, sample(1:56, 16, replace = FALSE))
  counts  <- table(factor(as.vector(tirages), levels = 1:56))
  min(counts)
})

cat("Probabilité estimée P(min <= 17) =", sum(valmin <= 17) / N)
```

```
#> Probabilité estimée P(min <= 17) = 0.26
```

9.4 Simulation à grande échelle (1 000 itérations)

```
set.seed(123)
n_simulations     <- 1000
tirages_par_serie <- 100

simuler_min <- function() {
  resultats <- replicate(tirages_par_serie, sample(1:56, 16, replace = FALSE))
  counts    <- table(factor(resultats, levels = 1:56))
  return(min(counts))
}

simus    <- replicate(n_simulations, simuler_min())
df_simus <- data.frame(id = 1:n_simulations, val_min = simus)

ggplot(df_simus, aes(x = id, y = val_min)) +
  geom_point(alpha = 0.3, color = "#282D87", size = 0.7) +
  geom_smooth(method = "lm", color = "#a12b4e", se = TRUE) +
  labs(
    title    = "Simulation de Monte-Carlo : fréquence minimale d'apparition",
    subtitle = "1 000 séries de 100 tirages --- Keno (16 boules parmi 56)",
    x        = "Numéro de simulation",
    y        = "Minimum d'apparition sur 100 tirages"
  )
```

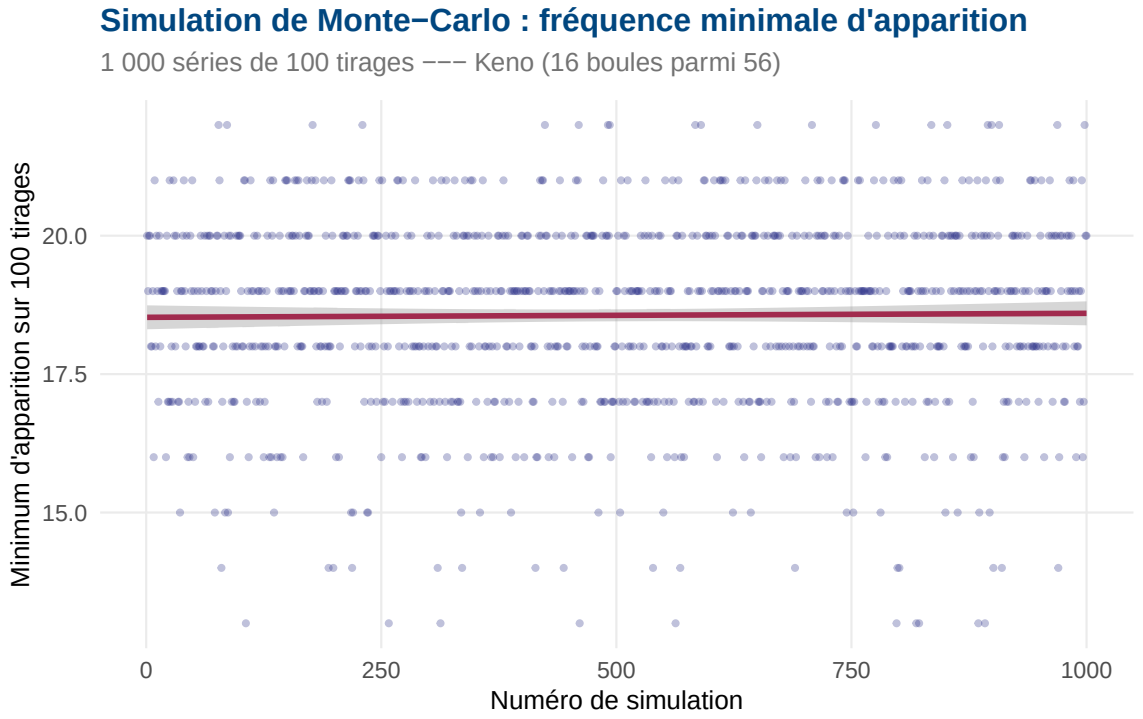

Figure 7. Simulation de Monte-Carlo : distribution des minima sur 1~000 séries de 100 tirages

```r
cat("En moyenne, le numéro le moins sorti apparaît",
    round(mean(simus), 1), "fois sur 100 tirages.")
```

```
#> En moyenne, le numéro le moins sorti apparaît 18.6 fois sur 100 tirages.
```

À retenir

La droite de régression (quasiment horizontale) confirme que le processus est **stationnaire** : le minimum d'apparition ne dérive pas au fil des simulations. C'est une propriété attendue d'un tirage véritablement aléatoire et uniforme.

Partie III — Méthodes de clustering

10 Introduction au clustering

Ce volet a été réalisé dans le cadre de la même unité d'enseignement **ESD113 — Probabilités et statistiques avec R**. L'objectif principal est de reproduire et d'adapter le code R utilisé dans l'article de référence **"A Survey of Popular R Packages for Cluster Analysis"** (Flynt & Dean, 2016). Conçu à la manière d'un "pense-bête" (*cheat sheet*) augmenté, il sert de guide pratique pour naviguer entre les différentes méthodes de clustering.

10.1 Qu'est-ce que le clustering ?

Imaginez que l'on vous donne un grand sac contenant mille pièces de puzzle mélangées, issues de plusieurs boîtes différentes. Votre mission : les trier *sans voir les images* des boîtes d'origine. Naturellement, vous examinerez les couleurs, les formes de bordures, les textures — vous regrouperez les pièces qui *se ressemblent*. Sans le savoir, vous appliquerez l'algorithme mental à la base de tout clustering.

En statistique, le **clustering** (ou *classification non supervisée*) consiste à regrouper automatiquement des observations similaires **sans connaître à l'avance** les étiquettes ou catégories. C'est l'opposé de la classification supervisée où un enseignant indique : « ceci est un client premium, ceci est un client standard ». Ici, la machine doit le découvrir *seule*.

Analogie pédagogique

Un algorithme de clustering appliqué à votre liste de courses pourrait découvrir, sans qu'on le lui dise, que vous achetez toujours ensemble des pâtes, de la sauce tomate et du parmesan — et créer un « groupe culinaire » reflétant ce comportement. C'est de la connaissance extraite automatiquement de l'observation brute.

10.2 Bref panorama historique

10.2.1 La petite (et longue) histoire de l'algorithme de Lloyd

Si l'on devait décerner le prix de la "patience algorithmique", Stuart Lloyd serait sans doute sur le podium. Imaginez la scène : nous sommes en **1957**, dans les prestigieux Laboratoires Bell. Entre deux tasses de café noir et des montagnes de tubes à vide, Lloyd pose les bases de ce qui deviendra le moteur de calcul le plus utilisé au monde pour le clustering : l'algorithme de quantification par moindres carrés.

À l'époque, son objectif n'est pas de segmenter des bases de données marketing, mais d'optimiser la modulation par impulsions codées (PCM). Lloyd rédige ses travaux dans un mémorandum interne technique… puis l'histoire semble s'arrêter là. Pendant que le papier prend la poussière dans les archives de Bell, un certain **James MacQueen** publie en 1967 un article où il baptise la méthode « K-means ». MacQueen rafle la gloire du nom, mais le "moteur" sous le capot reste celui de Lloyd.

Il faudra attendre **1982** — soit 25 ans après sa conception initiale ! — pour que le travail de Lloyd (1982) soit enfin publié officiellement dans les *IEEE Transactions on Information Theory*. Un

quart de siècle d'attente pour une idée qui, aujourd'hui, tourne en quelques millisecondes sur votre ordinateur. C'est un peu comme si quelqu'un inventait la roue en secret, la rangeait dans son garage, et attendait que tout le monde roule en carrosse pour enfin publier le brevet !

Le clustering n'est pas une invention récente. Ses racines plongent profondément dans l'histoire des sciences.

— **1894** — Karl Pearson (1894) utilise les premiers mélanges gaussiens pour analyser des populations de crabes dans la Baie de Naples. C'est l'acte de naissance des modèles de mélanges.
— **Années 1950–1960** — Les biologistes développent la *taxinomie numérique* pour classer les espèces. Les premières méthodes hiérarchiques émergent (Everitt et al. 2011).
— **1963** — Joe Ward (1963) formalise la méthode de liaison de Ward pour la classification hiérarchique.
— **1967** — James MacQueen (1967) publie l'article fondateur du **K-means**, la méthode la plus utilisée au monde.
— **1977** — Dempster, Laird et Rubin (1977) formalisent l'algorithme **EM**, moteur des modèles de mélanges modernes.
— **2002** — Fraley et Raftery (2002) publient la référence sur `Mclust`, qui automatise la sélection du modèle.

10.3 Objectifs et plan

Ce volet guide pas à pas à travers **trois grandes familles de méthodes**, du plus simple au plus sophistiqué.

1. Simulation de données structurées pour évaluer les algorithmes.
2. K-means, méthode du coude et ses **six variations d'implémentation**.
3. Classification hiérarchique ascendante et dendrogramme.
4. Modèles de mélanges gaussiens avec `Mclust`.

Le fil conducteur est **unique** : les données simulées de Flynt & Dean (2016) servent de laboratoire contrôlé pour **toutes** les méthodes. La vérité terrain connue (*cl_real*) permet une évaluation rigoureuse et cohérente — taux de réussite, taux d'erreur et ARI — pour chaque algorithme présenté.

11 Préparation de l'environnement R pour le clustering

11.1 Les packages : la boîte à outils

En R, les *packages* sont des bibliothèques développées par la communauté scientifique mondiale. Chaque méthode est implémentée dans un package spécialisé, fruit de travaux de recherche dédiés.

```
## tidyverse : écosystème complet de manipulation et visualisation
library(tidyverse)

## conflicted : arbitre les conflits de noms entre packages
library(conflicted)

## mvtnorm : simulation de lois normales multivariées
library(mvtnorm)

## mclust : modèles de mélanges gaussiens [Scrucca et al., 2016]
library(mclust)

## ggforce : extensions géométriques ggplot2
library(ggforce)

## viridis : palettes perceptuellement uniformes
library(viridis)

## ggdendro : dendrogrammes ggplot2
library(ggdendro)

## knitr + kableExtra : tableaux LaTeX qualité publication
library(knitr)
library(kableExtra)

## broom : extraction standardisée de métriques de modèles
library(broom)

## factoextra : visualisation clustering (fviz_nbclust, etc.)
library(factoextra)


## Résolution explicite des conflits de namespace
conflicts_prefer(dplyr::select)
conflicts_prefer(dplyr::filter)
conflicts_prefer(dplyr::lag)
```

12 Simulation des données : créer un laboratoire contrôlé

12.1 Pourquoi simuler ?

Évaluer une méthode de clustering sur des données réelles pose un problème fondamental : on ne connaît pas les *vrais* groupes, puisque c'est précisément ce qu'on cherche. La simulation résout ce problème en créant un monde artificiel où **la vérité est connue à l'avance**, ce qui permet de mesurer la qualité de chaque méthode avec précision.

À retenir

La simulation est le mannequin de crash-test de la statistique : on connaît exactement les forces appliquées, et on mesure comment chaque algorithme s'en sort. C'est le fondement de toute évaluation rigoureuse de méthodes.

12.2 Fondements mathématiques : la loi normale multivariée

12.2.1 La loi normale univariée

La loi normale (loi de Gauss, 1777–1855) est la distribution de probabilité la plus fondamentale en statistique (Johnson et Wichern 2002). Sa densité est :

```
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\!\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \quad x \in \mathbb{R}$$
```

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\!\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \quad x \in \mathbb{R}$$

où $\mu \in \mathbb{R}$ est la moyenne (centre) et $\sigma^2 > 0$ la variance (dispersion).

12.2.2 La loi normale multivariée

Pour p variables observées simultanément, on généralise avec la loi normale multivariée $\mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, de densité :

```
$$f(\bm{x}) = \frac{1}{(2\pi)^{p/2}|\bm{\Sigma}|^{1/2}}
\exp\!\left(-\frac{1}{2}(\bm{x}-\bm{\mu})^\top\bm{\Sigma}^{-1}(\bm{x}-\bm{\mu})\right), \quad
\bm{x} \in \mathbb{R}^p$$
```

$$f(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\!\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right), \quad \boldsymbol{x} \in \mathbb{R}^p$$

Les paramètres clés sont :

— $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_p)^\top$: le vecteur de moyennes (centre du nuage de points).

Table 9. Paramètres des trois composantes gaussiennes du mélange simulé.

Groupe	Proportion π_k	Moyenne $\boldsymbol{\mu}_k$	Covariance $\boldsymbol{\Sigma}_k$	Difficulté
Groupe 1	0.30	(0, 0)	sphérique I_2	Facile
Groupe 2	0.40	(3, 5)	sphérique I_2	Facile
Groupe 3	0.30	(0, 6)	Elliptique Σ_{ell}	Difficile (ellipse inclinée)

— $\boldsymbol{\Sigma}$: la matrice de covariance $p \times p$, symétrique définie positive, qui décrit la forme et l'orientation du nuage.

12.2.3 Le modèle de mélange gaussien

Notre jeu de données sera un mélange de $K = 3$ gaussiennes bivariées. La densité totale du mélange est :

```
$$f(\bm{x}) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}_2(\bm{x}\,|\,\bm{\mu}_k, \bm{\Sigma}_k),
\qquad \sum_{k=1}^{K} \pi_k = 1,\; \pi_k > 0$$
```

$$f(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}_2(\boldsymbol{x} \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \qquad \sum_{k=1}^{K} \pi_k = 1,\ \pi_k > 0$$

La matrice de covariance elliptique du Groupe 3 est :

```
$$\bm{\Sigma}_{ell} = \begin{pmatrix} 2 & 1.3 \\ 1.3 & 1 \end{pmatrix}$$
```

$$\boldsymbol{\Sigma}_{ell} = \begin{pmatrix} 2 & 1.3 \\ 1.3 & 1 \end{pmatrix}$$

Elle traduit une variance plus forte selon V_1 et une corrélation positive entre V_1 et V_2 (covariance $= 1.3 > 0$), ce qui crée l'ellipse inclinée à environ 45 degrés.

12.3 Simulation des variables continues

12.3.1 Vers une lecture plus nette : l'enveloppe convexe

Pour obtenir une frontière "physique" nette autour de chaque groupe, nous utilisons le concept d'**enveloppe convexe** (*Convex Hull*). Mathématiquement, l'enveloppe convexe d'un ensemble de points est le plus petit polygone convexe contenant tous ces points.

Analogie pédagogique

Imaginez que chaque point de notre graphique soit un clou planté sur une planche. L'enveloppe convexe correspond à la forme que prendrait un élastique tendu que l'on relâcherait autour de tous les clous d'une même couleur. L'élastique s'appuierait uniquement sur les points les plus extérieurs, créant une frontière parfaite et sans "creux".

12.3.2 Reproduction des couleurs originales de Flynt & Dean (2016)

```r
## -- Graine pour la reproductibilité ---------------------------------
set.seed(288)

## -- Paramètres des 3 groupes ----------------------------------------
pi_vec <- c(0.3, 0.4, 0.3)
mu <- list(c(0, 0), c(3, 5), c(0, 6))

sigma_sph <- diag(2)
sigma_ell <- matrix(c(2, 1.3, 1.3, 1), nrow = 2, byrow = TRUE)

## -- Tirage des étiquettes de groupe ---------------------------------
cl_real <- sample(1:3, size = 600, replace = TRUE, prob = pi_vec)

## -- Génération des coordonnées (V1, V2) de chaque individu ----------
X_data <- purrr::map_dfr(cl_real, function(i) {
  sigma_i <- if (i == 3L) sigma_ell else sigma_sph
  rmvnorm(n = 1, mean = mu[[i]], sigma = sigma_i) |>
    as_tibble()
}) |>
  set_names(c("V1", "V2"))

X_plot <- X_data |>
  mutate(Groupe = as.factor(cl_real)) |>
  filter(!is.na(V1) & !is.na(V2))

couleurs_article <- c("1" = "black", "2" = "red", "3" = "green3")

ggplot(X_plot, aes(x = V1, y = V2, color = Groupe)) +
  geom_point(alpha = 0.5, size = 1.2, shape = 18) +
  ggforce::geom_mark_hull(
    aes(fill = Groupe),
    alpha = 0.10,
    color = NA,
    concavity = 10000,
    na.rm = TRUE,
    expand = unit(2, "mm")
  ) +
  scale_color_manual(values = couleurs_article,
                     name   = "Groupe réel",
                     labels = c("Groupe 1 (Black)", "Groupe 2 (Red)", "Groupe 3 (Green)")) +
  scale_fill_manual(values = couleurs_article) +
  labs(
    title    = "Données simulées : séparation par enveloppes convexes",
    subtitle = "Reproduction des couleurs originales de Flynt & Dean (2016)",
    x = "V1", y = "V2"
  ) +
  theme_minimal() +
  theme(legend.position = "bottom") +
```

```
guides(fill = "none")
```

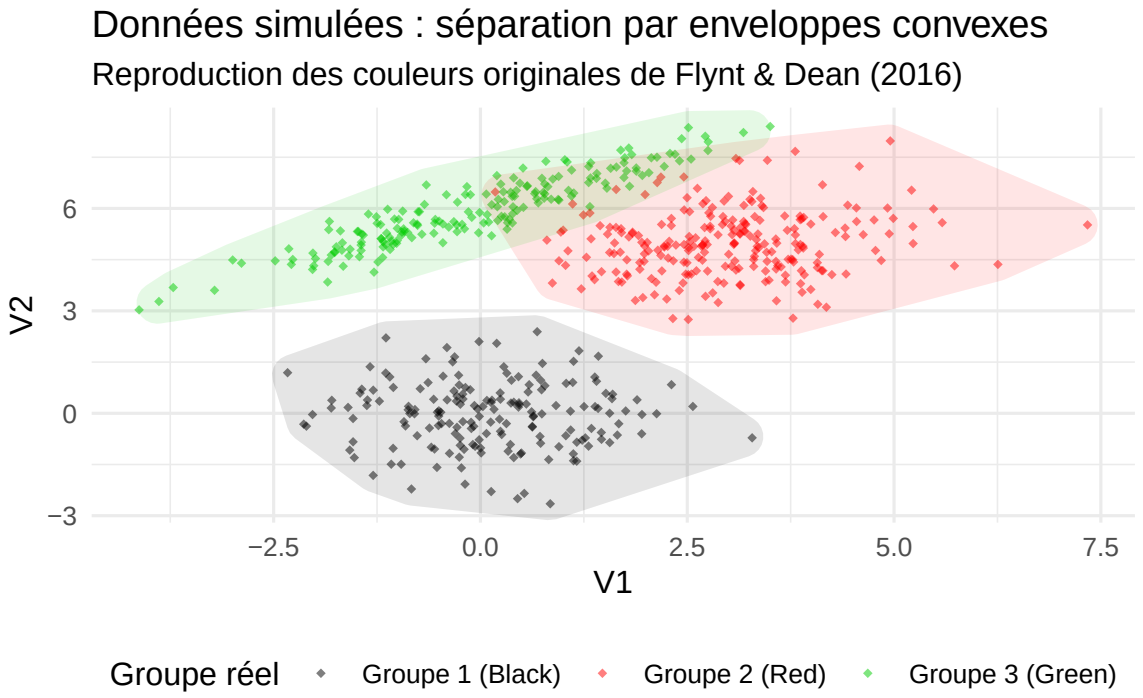

Figure 8. Représentation par enveloppes convexes avec les couleurs de l'article original (Noir, Rouge, Vert).

Point d'attention

Le Groupe 3 (en haut) forme une ellipse inclinée. Le K-means, qui suppose des clusters sphériques, aura du mal a le détecter correctement. Mclust, qui modélise explicitement les ellipses, le capturera bien.

13 Méthodes de partitionnement géométrique

13.1 K-means : la méthode des barycentres

13.1.1 Principe et intuition

Le K-means est l'algorithme de clustering le plus célébre et le plus utilisé au monde (MacQueen 1967; Lloyd 1982). Son principe est d'une simplicité remarquable :

Chaque individu appartient au groupe dont le centre (la moyenne) lui est le plus proche.

Analogie pédagogique

Imaginez 3 personnes dans une salle bondée, chacune criant : "Venez vers moi !" Chaque individu rejoint la plus proche. Puis chaque personne se déplace au centre géographique de son groupe. On recommence jusqu'à stabilisation. C'est exactement l'algorithme K-means.

13.1.2 Formalisation mathématique

K-means minimise la **somme des distances au carré intra-cluster** (WSS, *Within-Cluster Sum of Squares*) :

```
\begin{equation}
\mathrm{WSS} = \sum_{k=1}^{K} \sum_{\bm{x}_{i} \in C_{k}}
  \lVert \bm{x}_{i} - \bm{\mu}_{k} \rVert^{2}
\end{equation}
```

$$\mathrm{WSS} = \sum_{k=1}^{K} \sum_{\boldsymbol{x}_i \in C_k} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2 \tag{1}$$

où $\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\boldsymbol{x}_i \in C_k} \boldsymbol{x}_i$ est le **centroïde** du cluster k.

L'algorithme procède en quatre étapes itératives :

1. **Initialisation** : placer K centroïdes aléatoirement.
2. **Attribution** : affecter chaque individu au centroïde le plus proche selon $d(\boldsymbol{x}, \boldsymbol{\mu}_k) = \|\boldsymbol{x} - \boldsymbol{\mu}_k\|$.
3. **Mise à jour** : recalculer chaque centroïde comme la moyenne de son groupe.
4. **Convergence** : répéter les étapes 2–3 jusqu'à stabilisation.

13.1.3 Choisir K : la méthode du coude

La **méthode du coude** (*elbow method*) calcule la WSS pour plusieurs valeurs de K et identifie le point de rupture au-delà duquel le gain marginal devient négligeable.

```
elbow_data <- tibble(k = 1:9) |>
  mutate(
    model = purrr::map(k, ~ kmeans(X_data, centers = .x, nstart = 50)),
```

```
    wss   = purrr::map_dbl(model, ~ .x$tot.withinss)
  )

ggplot(elbow_data, aes(x = k, y = wss)) +
  geom_line(color = "gray70", linewidth = 0.9) +
  geom_point(aes(color = (k == 3)), size = 3.5) +
  scale_color_manual(values = c("black", "#B40000")) +
  scale_x_continuous(breaks = 1:9) +
  labs(
    title = "Inertie intra-classe selon le nombre de clusters K",
    x     = "Nombre de clusters K",
    y     = "WSS (Within-Cluster Sum of Squares)"
  ) +
  guides(color = "none")
```

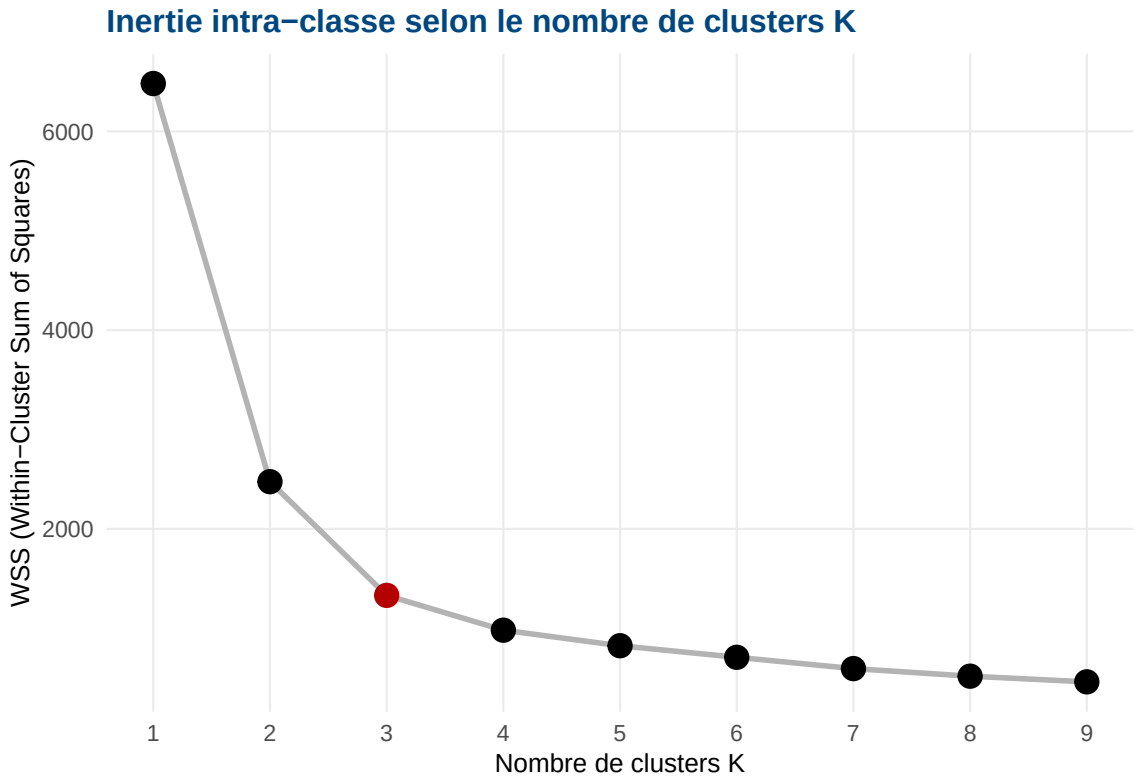

Figure 9. Méthode du coude. Le point rouge (K = 3) marque l'inflexion de la courbe : au-delà, ajouter un cluster n'apporte que peu de gain en inertie.

Point d'attention

Limites du K-means : (1) il suppose des clusters sphériques, inadapté aux formes elliptiques ; (2) il est sensible aux valeurs aberrantes ; (3) il n'exprime aucune incertitude : chaque individu est assigné définitivement a un seul groupe ; (4) il ne fonctionne que sur des variables continues.

13.1.4 Formalisation algorithmique du K-means

Avant de comparer les variantes d'implémentation, il est utile de rappeler formellement le fonctionnement interne de l'algorithme K-means, tel que décrit dans Flynt & Dean (2016).

L'objectif mathématique est de partitionner n observations $\{x_1, \ldots, x_n\} \subset \mathbb{R}^p$ en k clusters $\{C_1, \ldots, C_k\}$ de façon à **minimiser l'inertie intra-classe totale** (WSS) :

```
$$
\text{WSS}(k) = \sum_{j=1}^{k} \sum_{x_i \in C_j} \| x_i - \mu_j \|^2
$$
```

$$\text{WSS}(k) = \sum_{j=1}^{k} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2$$

où $\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$ est le **centroïde** du cluster C_j.

Algorithme 1 : K-means de Lloyd — `kmeans(X, k, nstart)`

Entrée : Données $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p$, nombre de clusters k, répétitions `nstart`
Sortie : Partition $\{C_1, \ldots, C_k\}$ et centroïdes $\{\mu_1, \ldots, \mu_k\}$

for $r = 1$ **to** *`nstart`* **do**
  Tirer k centroïdes initiaux $\mu_1^{(0)}, \ldots, \mu_k^{(0)}$ aléatoirement
  $t \leftarrow 0$
  repeat
    ▷ `Étape E - Assignation`
    for $i = 1$ **to** n **do**
      $C^{(t+1)}(x_i) \leftarrow \arg\min_j \|x_i - \mu_j^{(t)}\|^2$
    end
    ▷ `Étape M - Mise à jour`
    for $j = 1$ **to** k **do**
      $\mu_j^{(t+1)} \leftarrow \frac{1}{|C_j^{(t+1)}|} \sum_{x_i \in C_j^{(t+1)}} x_i$
    end
    $t \leftarrow t + 1$
  until *convergence (assignations stables)*
  Calculer $\text{WSS}_r = \sum_{j=1}^{k} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2$
end
return *La solution* r^* *telle que* $\text{WSS}_{r^*} = \min_r \text{WSS}_r$

À retenir

Pourquoi `nstart = 50` ? L'algorithme K-means est sensible à l'initialisation : selon les centroïdes tirés au départ, la convergence peut aboutir à un optimum local différent. `nstart` indique à R de relancer l'algorithme depuis `nstart` initialisations aléatoires indépendantes et de ne retenir que la meilleure solution (WSS minimale). Flynt & Dean recommandent une valeur d'au moins 25–50 pour des résultats stables et reproductibles.

13.1.5 Six variations pour générer la courbe WSS

La méthode du coude peut être implémentée en R de six manières différentes. Ces six variations produisent des courbes WSS statistiquement identiques — toutes indiquent $k = 3$ sur nos données — mais diffèrent par leur lisibilité, leur réutilisabilité et leur expressivité.

Variation 1 — Méthode manuelle (9 lignes distinctes)

```
km1 <- kmeans(X_data, 1, nstart = 50);
km2 <- kmeans(X_data, 2, nstart = 50)
km3 <- kmeans(X_data, 3, nstart = 50);
km4 <- kmeans(X_data, 4, nstart = 50)
km5 <- kmeans(X_data, 5, nstart = 50);
km6 <- kmeans(X_data, 6, nstart = 50)
km7 <- kmeans(X_data, 7, nstart = 50);
km8 <- kmeans(X_data, 8, nstart = 50)
km9 <- kmeans(X_data, 9, nstart = 50)

wss_man <- c(km1$tot.withinss, km2$tot.withinss, km3$tot.withinss,
             km4$tot.withinss, km5$tot.withinss, km6$tot.withinss,
             km7$tot.withinss, km8$tot.withinss, km9$tot.withinss)

ggplot(tibble(k = 1:9, wss = wss_man), aes(k, wss)) +
  geom_line(color = "gray60") + geom_point(size = 3) +
  scale_x_continuous(breaks = 1:9) +
  labs(title = "Variation 1 : Méthode manuelle", x = "K", y = "WSS") +
  theme_pdf()
```

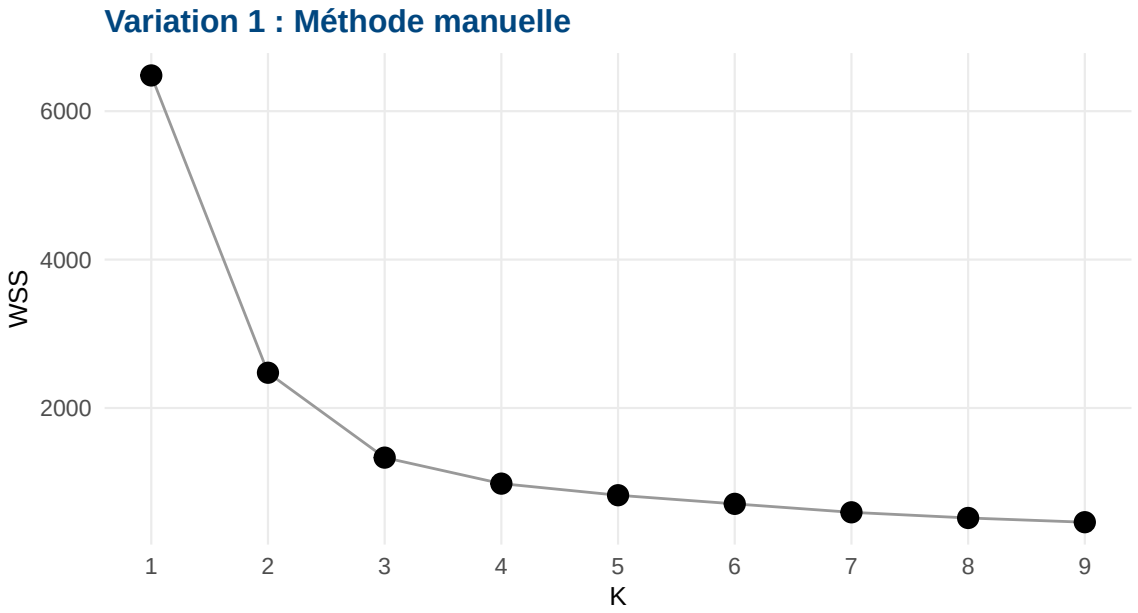

Figure 10. Variation 1 : méthode manuelle — 9 appels explicites à kmeans().

Variation 2 — Boucle for

```
wss_for <- numeric(9)
for (i in 1:9) {
  wss_for[i] <- kmeans(X_data, centers = i, nstart = 50)$tot.withinss
}

ggplot(tibble(k = 1:9, wss = wss_for), aes(k, wss)) +
  geom_line(color = "gray60") + geom_point(size = 3) +
  scale_x_continuous(breaks = 1:9) +
  labs(title = "Variation 2 : Boucle For", x = "K", y = "WSS") +
  theme_pdf()
```

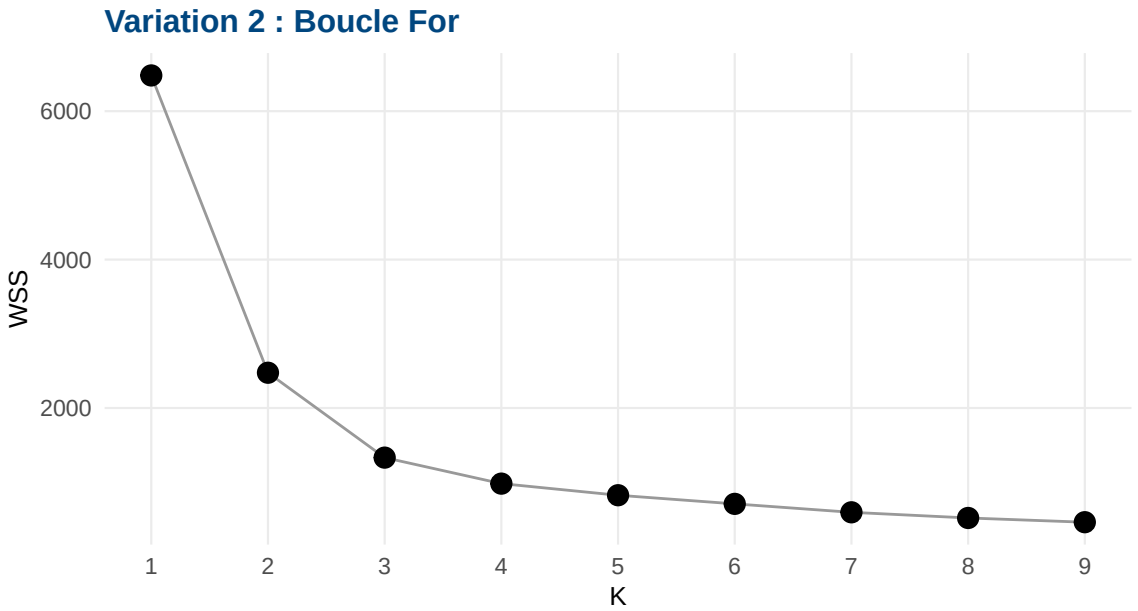

Figure 11. Variation 2 : boucle for — logique séquentielle explicite.

Variation 3 — sapply (vectorisation Base R)

```
wss_sapply <- sapply(1:9, function(k) {
  kmeans(X_data, centers = k, nstart = 50)$tot.withinss
})

ggplot(tibble(k = 1:9, wss = wss_sapply), aes(k, wss)) +
  geom_line(color = "steelblue") + geom_point() +
  scale_x_continuous(breaks = 1:9) +
  labs(title = "Variation 3 : sapply (Vectorisation)", x = "K", y = "WSS") +
  theme_pdf()
```

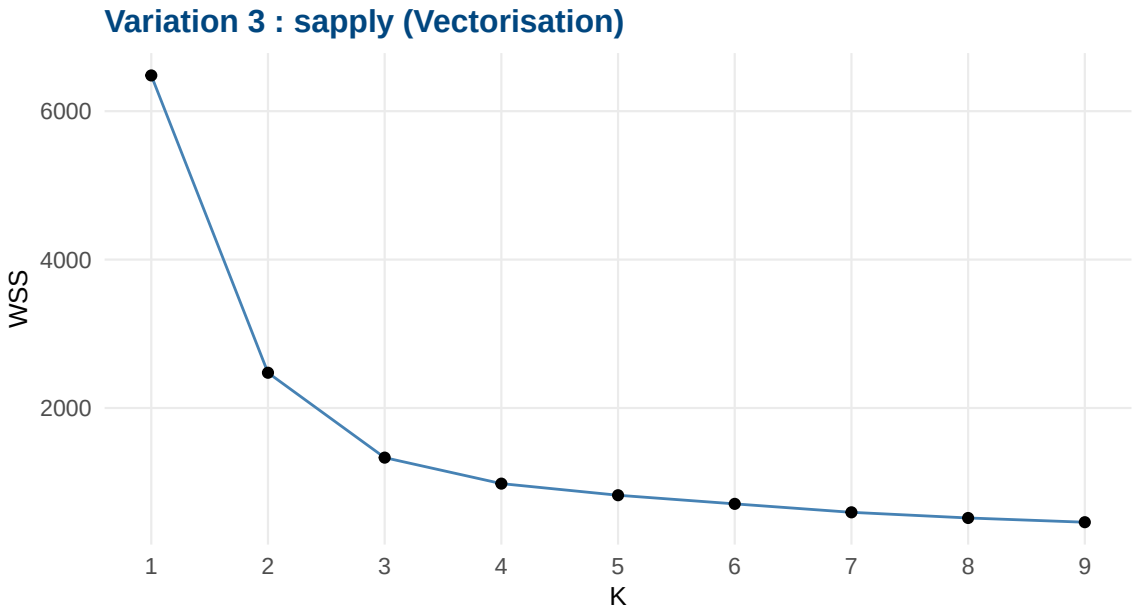

Figure 12. Variation 3 : sapply — vectorisation Base R concise.

Variation 4 — purrr (programmation fonctionnelle)

```
elbow_purrr <- tibble(k = 1:9) |>
  mutate(
    model = purrr::map(k, ~ kmeans(X_data, centers = .x, nstart = 50)),
    wss   = purrr::map_dbl(model, ~ .x$tot.withinss)
  )

ggplot(elbow_purrr, aes(k, wss)) +
  geom_line(linewidth = 0.9, color = "gray60") +
  geom_point(aes(color = (k == 3)), size = 4) +
  scale_color_manual(values = c("black", "#B40000"), guide = "none") +
  scale_x_continuous(breaks = 1:9) +
  labs(title = "Variation 4 : purrr (Programmation fonctionnelle)", x = "K", y = "WSS") +
  theme_pdf()
```

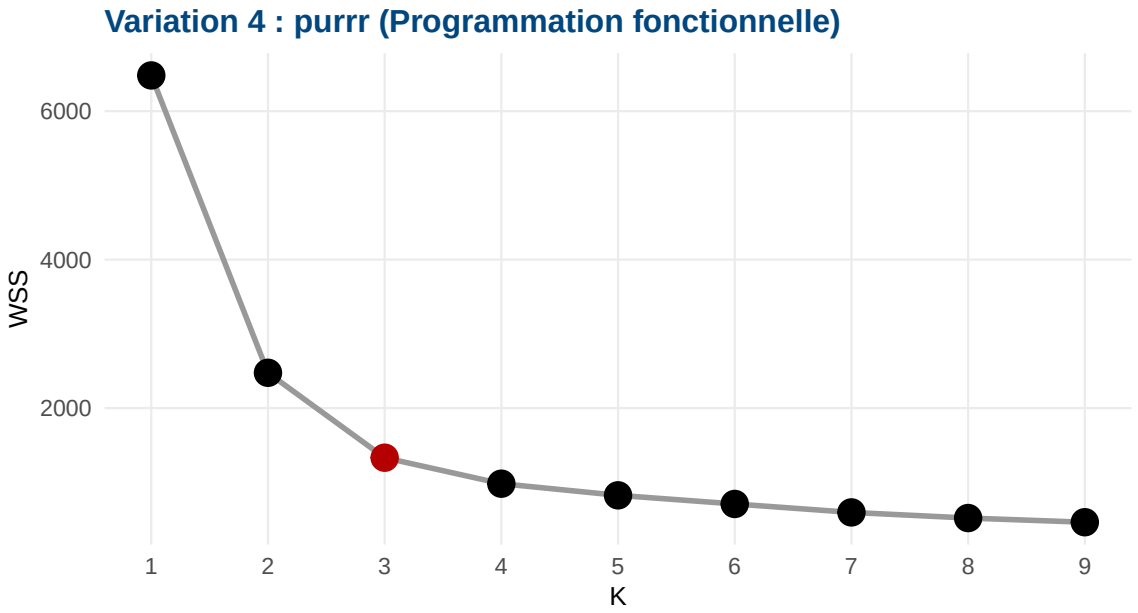

Figure 13. Variation 4 : purrr — traçabilité complète (modèles stockés).
Point rouge = K optimal selon Flynt & Dean.

Variation 5 — broom (métriques standardisées)

```
elbow_broom <- tibble(k = 1:9) |>
  mutate(
    model = purrr::map(k, ~ kmeans(X_data, centers = .x, nstart = 50)),
    stats = purrr::map(model, broom::glance)
  ) |>
  tidyr::unnest(stats)

ggplot(elbow_broom, aes(k, tot.withinss)) +
  geom_line(color = "green4") + geom_point() +
  scale_x_continuous(breaks = 1:9) +
  labs(title = "Variation 5 : broom (métriques standardisées)", y = "WSS", x = "K") +
  theme_pdf()
```

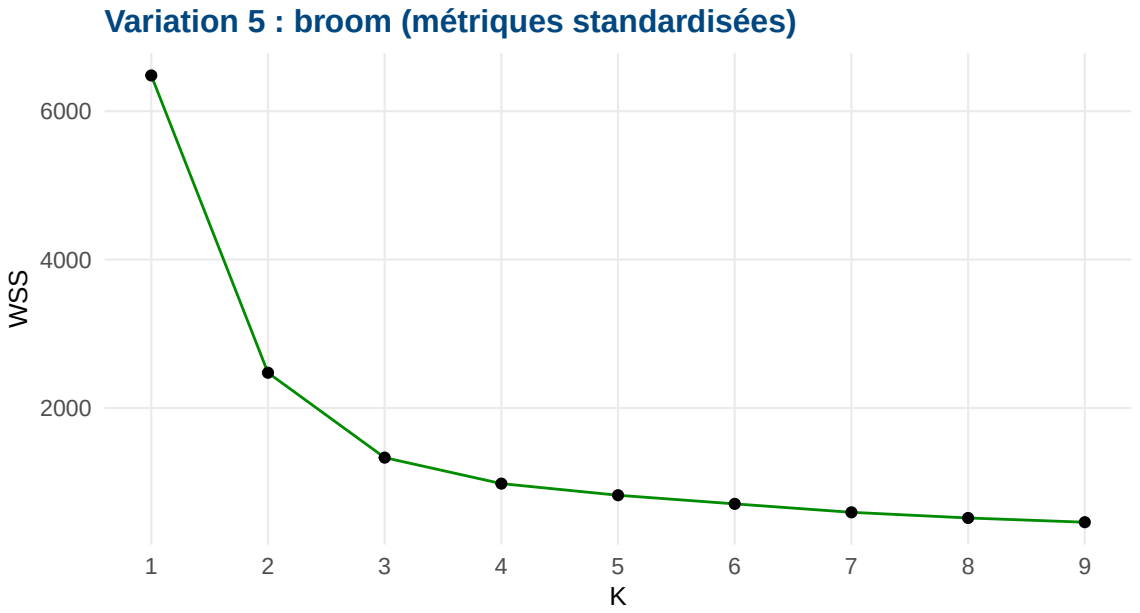

Figure 14. Variation 5 : broom — extraction standardisée de métriques multiples.

Variation 6 — factoextra (clé en main)

```
fviz_nbclust(X_data, kmeans, method = "wss", k.max = 9, nstart = 50) +
  geom_vline(xintercept = 3, linetype = 2, color = "firebrick") +
  labs(title = "Variation 6 : factoextra (Clé en main)") +
  theme_pdf()
```

Table 10. Tableau de synthèse des six variations d'implémentation de la méthode du coude.

Méthode	Avantage principal	Inconvénient	Usage idéal
Manuelle	Transparence totale	Très verbeux	Apprentissage
Boucle For	Logique universelle	Plus de lignes	Débutants
sapply	Concis (Base R)	Modèle perdu	Scripts légers
purrr	Traçabilité complète	Syntaxe complexe	Pipelines DS
broom	Métriques multiples	Package sup.	Rapports stats
factoextra	Rapidité, esthétique	"Boîte noire"	Exploration

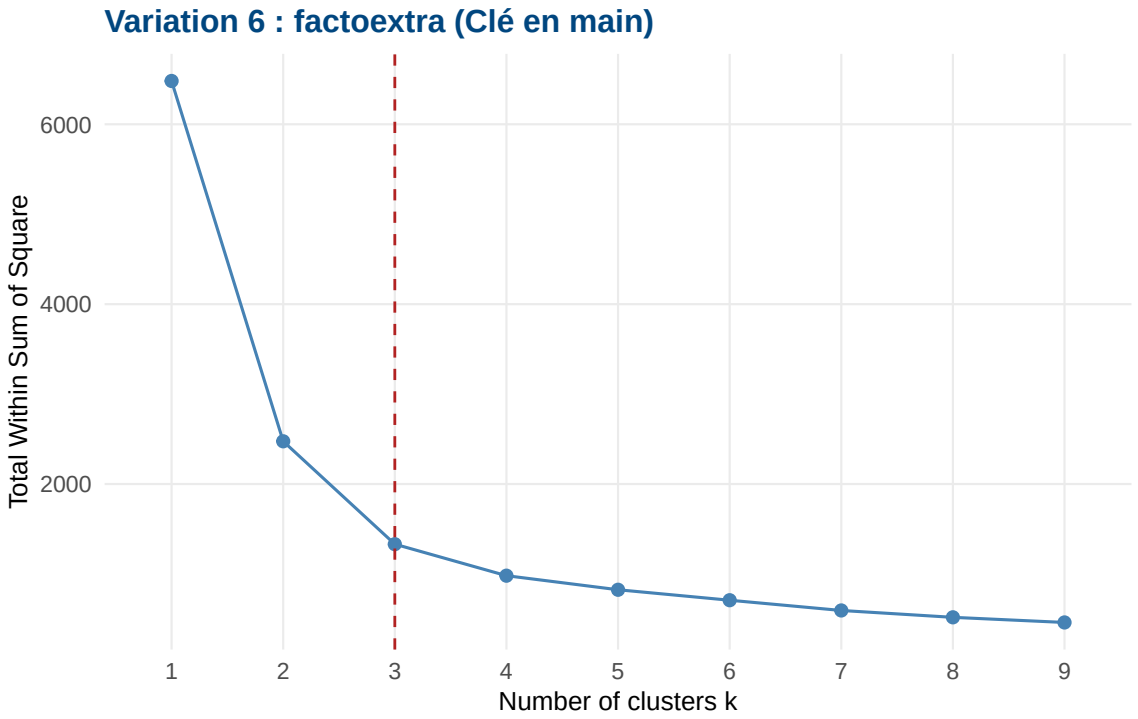

Figure 15. Variation 6 : factoextra — graphique publication-ready en une ligne. La ligne pointillée matérialise le coude en K = 3.

À retenir

Les six courbes WSS produites sont **statistiquement identiques** : toutes indiquent un coude en $k = 3$, confirmant exactement la structure simulée par Flynt & Dean. Le choix de la variation est donc un choix de **style de code** et de contexte d'usage.

13.2 Superposition K-means / vérité terrain

13.2.1 Code R : résultats K-means et des données simulées (les 3 gaussiennes)

```
set.seed(123)
km_res <- kmeans(X_data, centers = 3, nstart = 25)
```

```r
X_eval <- X_data |>
  mutate(
    Vrai_Groupe = as.factor(cl_real),
    Cluster_KM  = as.factor(km_res$cluster)
  )

ggplot(X_eval, aes(x = V1, y = V2)) +
  geom_point(aes(color = Vrai_Groupe), alpha = 0.6, size = 1.2, shape = 18) +
  ggforce::geom_mark_hull(
    aes(fill = Cluster_KM, group = Cluster_KM),
    alpha = 0.10,
    color = "gray40",
    linetype = "solid",
    concavity = 10000,
    radius = unit(0, "mm"),
    expand = unit(0, "mm"),
    na.rm = TRUE
  ) +
  scale_color_manual(values = couleurs_article, name = "Vérité Terrain") +
  scale_fill_manual(values = couleurs_article) +
  labs(
    title    = "Évaluation K-means : Enveloppes Convexes Strictes",
    subtitle = "Polygones reliant les points extrêmes de chaque cluster",
    x = "V1", y = "V2"
  ) +
  theme_minimal() +
  theme(legend.position = "bottom") +
  guides(fill = "none")
```

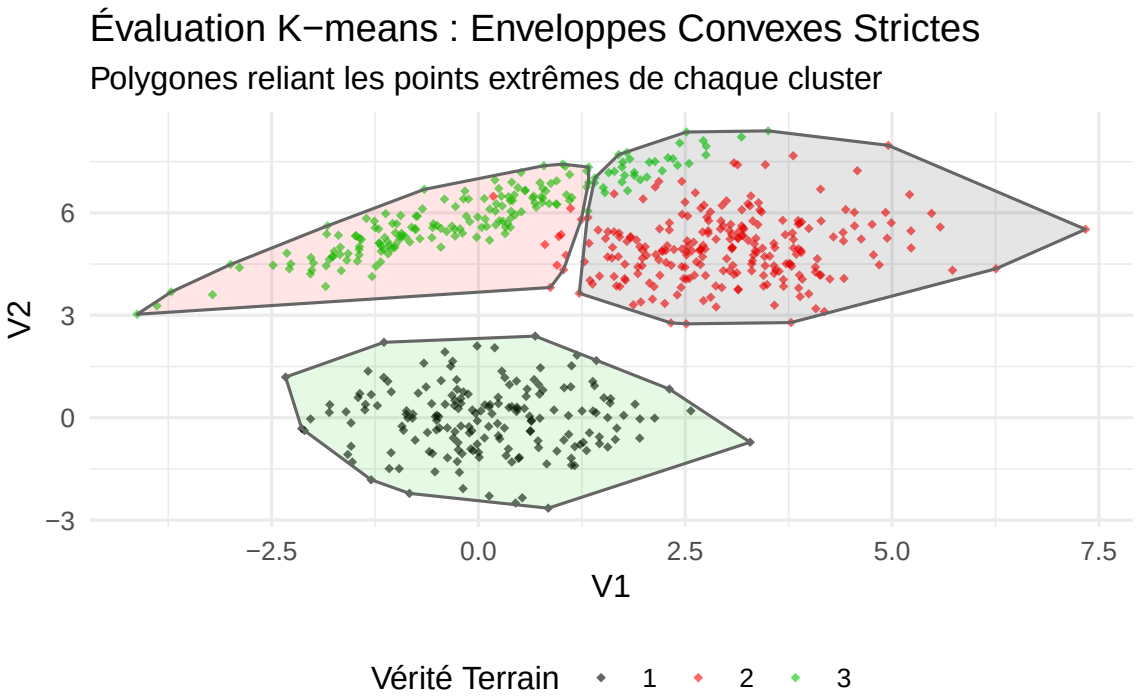

Figure 16. Évaluation visuelle du K-means avec enveloppes convexes strictes (définition mathématique par segments reliant les points extrêmes).

13.2.2 Analyse des résultats : limites géométriques et conflits de modèles

L'examen de la figure de superposition entre les enveloppes convexes du K-means et les points de la vérité terrain permet de dresser un bilan contrasté de la performance de l'algorithme. L'analyse visuelle révèle trois comportements distincts :

1. Une isolation réussie pour le Groupe 1 (Noir) Le Groupe 1 est identifié avec une grande précision. Son isolement géographique par rapport aux deux autres composantes permet au K-means de tracer une frontière nette.

2. Une "fuite" structurelle sur le Groupe 2 (Rouge) Une faiblesse notable apparaît au niveau du Groupe 2. L'enveloppe convexe définie par le K-means ne parvient pas à englober l'intégralité des points rouges. En raison de la proximité immédiate du Groupe 3, l'algorithme "tranche" l'espace de manière linéaire.

3. Le Groupe 3 (Vert) : l'échec face à l'ellipticité Le Groupe 3 concentre les erreurs les plus visibles. Son enveloppe convexe englobe indûment de nombreux points appartenant au Groupe 2. Ce phénomène s'explique par la nature même du K-means : l'algorithme cherche à minimiser la variance intra-classe de manière sphérique (distance euclidienne classique).

Table 11. Analyse de la précision du K-means (K=3) sur les données Flynt & Dean.

Indicateur de performance	Résultat
Points bien classés	557 / 600
Taux de réussite	**92.83 %**
Taux d'erreur	**7.17 %**

Point d'attention

En conclusion, si le Groupe 1 valide l'efficacité de l'approche dans des conditions simples, les Groupes 2 et 3 démontrent que la propreté visuelle d'un clustering ne garantit pas sa justesse statistique. Le K-means impose sa propre géométrie (sphérique) aux données au lieu de s'adapter à la leur (elliptique).

13.2.3 Analyse de la performance du partitionnement (K=3)

```
km3 <- kmeans(X_data, 3, nstart = 50)
tab_km <- table(cl_real, km3$cluster)
cat("Table de contingence K-means vs vérité terrain :\n")
```

```
#> Table de contingence K-means vs vérité terrain :
```

```
print(tab_km)
```

```
#>
#> cl_real   1   2   3
#>       1   0   0 177
#>       2 227  10   0
#>       3  33 153   0
```

```
reussite_points <- sum(apply(tab_km, 1, max))
total_points    <- sum(tab_km)
taux_reussite   <- (reussite_points / total_points) * 100
taux_erreur     <- 100 - taux_reussite
```

Point d'attention

Note sur l'interprétation : L'essentiel de l'erreur provient du Groupe 3 (elliptique). Le K-means "grignote" les extrémités de l'ellipse pour satisfaire sa propre logique de compacité circulaire, générant un biais structurel inévitable.

À retenir

La performance d'un algorithme de clustering dépend moins de sa complexité informatique que de l'adéquation entre ses hypothèses géométriques et la structure réelle des données.

13.3 Classification hiérarchique ascendante (CHA)

13.3.1 Principe

La **classification hiérarchique ascendante** (CHA) construit un **dendrogramme** — un arbre de fusion représentant quels individus ou groupes ont été réunis, et à quelle distance (Everitt et al. 2011). Elle ne requiert pas de spécifier K à l'avance.

La liaison moyenne (UPGMA) définit la distance entre deux groupes A et B comme la moyenne de toutes les distances inter-individuelles :

```
$$d(A, B) = \frac{1}{|A| \cdot |B|} \sum_{i \in A} \sum_{j \in B} d(x_i, x_j)$$
```

$$d(A, B) = \frac{1}{|A| \cdot |B|} \sum_{i \in A} \sum_{j \in B} d(x_i, x_j)$$

```
d_mat   <- dist(X_data, method = "euclidean")
h_avg   <- hclust(d_mat, method = "average")

ggdendrogram(h_avg, rotate = FALSE, size = 0.15) +
  labs(
    title    = "Dendrogramme de classification hiérarchique",
    subtitle = "Liaison moyenne (UPGMA) --- 600 individus"
  ) +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank())
```

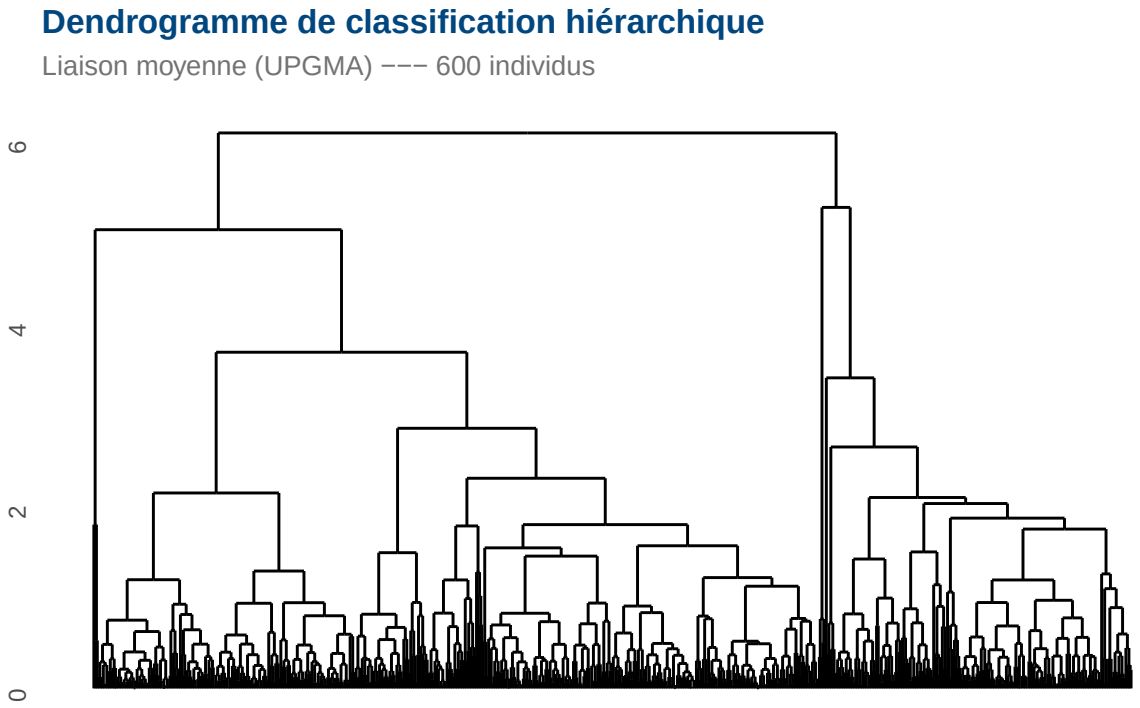

Figure 17. Dendrogramme de classification hiérarchique (liaison moyenne, 600 individus). La hauteur d'une jonction représente la distance de fusion. Pour K = 3, on coupe au niveau de la 3e plus grande fusion.

13.3.2 La méthode hiérarchique CAH sur les données simulées

Dans l'article de Flynt & Dean (2016), la méthode du lien moyen peut s'avérer encore moins performante que le K-means sur ces structures, entraînant un taux d'erreur avoisinant les 30%. La raison fondamentale : la CAH fusionne des sous-groupes de façon **irréversible** (*greedy*), sans pouvoir réassigner un point mal placé.

```
res.hc <- hclust(dist(X_data), method = "average")

fviz_dend(res.hc,
          k          = 3,
          cex        = 0.15,
          lwd        = 0.30,
          rect       = TRUE,
          rect_fill  = TRUE,
          rect_lty   = 1,
          palette    = c("black", "red", "green3"),
          main       = "Dendrogramme (Average Linkage)",
          xlab       = "Observations",
          ylab       = "Distance Euclidienne",
          ggtheme    = theme_minimal()) +
  # Forcer l'épaisseur sur tous les segments du dendrogramme
  scale_linewidth(range = c(0.30, 0.30)) +
  theme(
```

```
    axis.text.x  = element_blank(),
    legend.position = "none"          # <-- supprime toute légende
  )
```

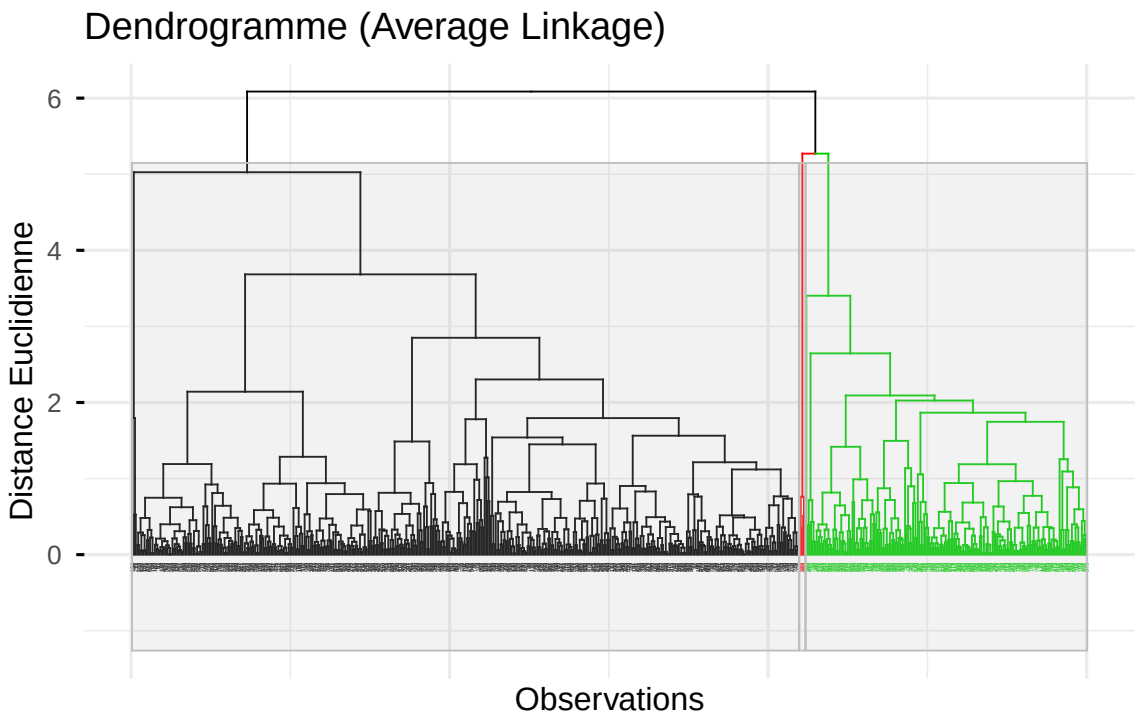

Figure 18. Dendrogramme (lien moyen, 600 individus) avec découpage en 3 clusters selon les couleurs de l'article.

13.3.3 La version circulaire

```
library(patchwork)
library(grid)

# 1. Construction du plot
p_circ <- fviz_dend(res.hc,
                    k           = 3,
                    type        = "circular",
                    cex         = 0.15,
                    rect        = TRUE,
                    rect_fill   = TRUE,
                    palette     = c("black", "red", "green3"),
                    main        = "Dendrogramme Circulaire (Average Linkage)",
                    ggtheme     = theme_void())

# 2. Patch linewidth
for (i in seq_along(p_circ$layers)) {
  geom_class <- class(p_circ$layers[[i]]$geom)
  if (any(geom_class %in% c("GeomSegment", "GeomCurve", "GeomPath"))) {
```

```
    p_circ$layers[[i]]$aes_params$linewidth <- 0.30
    p_circ$layers[[i]]$aes_params$size      <- 0.30
  }
}

p_circ <- p_circ +
  scale_linewidth(range = c(0.30, 0.30)) +
  theme_void() +
  theme(legend.position = "none")

# 3. Hack panel_params pour virer les labels radiaux
gb <- ggplot_build(p_circ)
gb$layout$panel_params[[1]]$r.labels <- character(0)
gb$layout$panel_params[[1]]$r.major  <- numeric(0)
gb$layout$panel_params[[1]]$r.minor  <- numeric(0)
gt <- ggplot_gtable(gb)

# 4. Rendu compatible Quarto via wrap_elements
wrap_elements(gt)
```

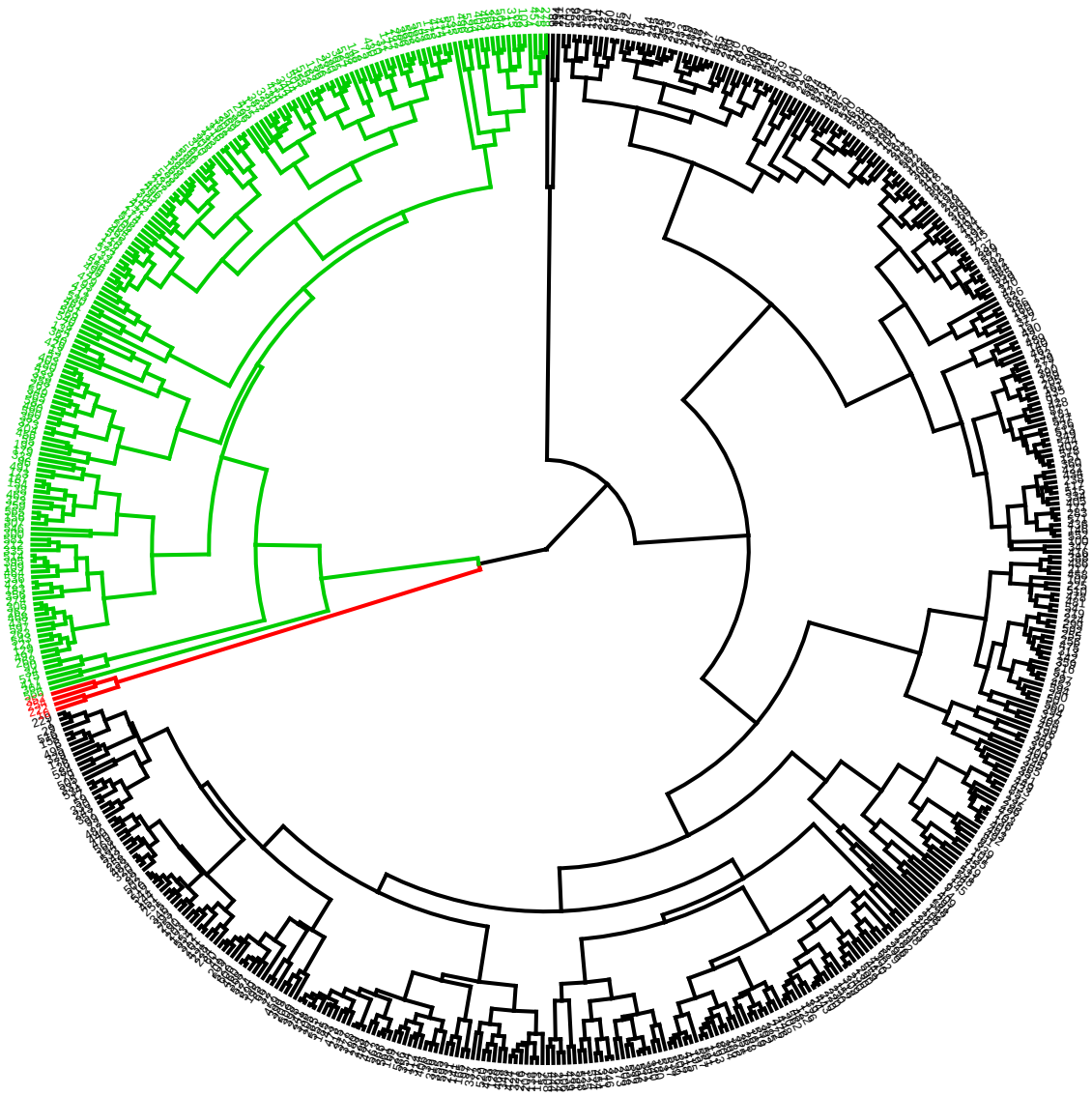

Figure 19. Dendrogramme circulaire (lien moyen, 600 individus) avec découpage en 3 clusters.

Taux de réussite : $(237 + 177 + 4)/600 = 69{,}7\,\%$ Taux d'erreur : $182/600 = 30{,}3\,\%$

Point d'attention

Le constat de Flynt & Dean : Alors que le K-means affiche environ 7–10% d'erreur, la méthode hiérarchique dépasse ici les 30% d'erreur. Les deux groupes sphériques, trop proches, sont fusionnés en un "super-cluster" artificiel par le lien moyen, tandis que le groupe vert (elliptique) se retrouve fragmenté.

À retenir

La méthode hiérarchique est utile pour explorer la structure des données, mais pour un partitionnement robuste sur ces données simulées, le K-means (et a fortiori Mclust) reste largement supérieur.

14 Méthodes basées sur les modèles probabilistes

14.1 Modèles de mélanges gaussiens : Mclust

Analogie pédagogique

Le K-means affecte chaque observation à *un seul* groupe de façon déterministe : "Tu appartiens au groupe 2, point final." Le GMM est plus nuancé : "Tu appartiens au groupe 2 avec une probabilité de 78%, au groupe 1 avec 18%, au groupe 3 avec 4%." Cette **appartenance probabiliste** est à la fois plus réaliste et plus informative.

14.1.1 Le modèle de mélange gaussien (GMM)

`mclust` ajuste un **modèle de mélange gaussien** (GMM) : on suppose que la population est composée de K sous-populations gaussiennes, chacune caractérisée par un poids π_k, un centre $\boldsymbol{\mu}_k$ et une structure de dispersion $\boldsymbol{\Sigma}_k$. La densité de mélange s'écrit :

$$f(x_i) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}_p(x_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

avec $\sum_{k=1}^{K} \pi_k = 1$ et $\pi_k > 0$. Pour chaque composante k :

- π_k est le **poids** (proportion de mélange) : probabilité *a priori* qu'une observation quelconque appartienne au groupe k.
- $\boldsymbol{\mu}_k \in \mathbb{R}^p$ est le **vecteur de moyennes** : le centre de la gaussienne dans l'espace à p dimensions.
- $\boldsymbol{\Sigma}_k \in \mathbb{R}^{p \times p}$ est la **matrice de covariance** : elle décrit la forme, l'orientation et l'étalement du nuage autour de $\boldsymbol{\mu}_k$.

14.1.2 L'algorithme EM

Les paramètres $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$ sont estimés par maximum de vraisemblance via l'algorithme **EM** (Expectation- Maximisation) (Dempster, Laird, et Rubin 1977), qui alterne deux étapes jusqu'à convergence.

Étape E (Expectation) — calcul des responsabilités r_{ik} :

$$r_{ik} = \frac{\pi_k \cdot \mathcal{N}_p(x_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \cdot \mathcal{N}_p(x_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

$r_{ik} \in [0, 1]$ est la probabilité *a posteriori* que l'observation x_i provienne de la composante k étant donné les paramètres courants (application directe de la règle de Bayes). On a $\sum_k r_{ik} = 1$ pour tout i : là où le K-means donne une appartenance dure (0 ou 1), le GMM produit pour chaque observation un vecteur complet de probabilités (*soft assignment*).

Étape M (Maximisation) — mise à jour des paramètres à partir des effectifs pondérés $n_k = \sum_i r_{ik}$:

$$\hat{\pi}_k = \frac{n_k}{n}, \qquad \hat{\boldsymbol{\mu}}_k = \frac{1}{n_k}\sum_i r_{ik}\, x_i, \qquad \hat{\boldsymbol{\Sigma}}_k = \frac{1}{n_k}\sum_i r_{ik}(x_i - \hat{\boldsymbol{\mu}}_k)(x_i - \hat{\boldsymbol{\mu}}_k)^\top$$

À retenir

La log-vraisemblance $\ell(\Theta) = \sum_i \ln f(x_i)$ est **garantie non décroissante** à chaque itération EM. L'algorithme s'arrête quand l'incrément $|\ell^{(t+1)} - \ell^{(t)}| < \varepsilon$ (défaut : 10^{-5} dans `mclust`). En pratique, on relance depuis plusieurs initialisations pour éviter les maxima locaux.

14.1.3 Les 14 familles de modèles de covariance

La richesse de `mclust` tient à sa paramétrisation par décomposition propre de $\boldsymbol{\Sigma}_k$:

$$\boldsymbol{\Sigma}_k = \lambda_k\, D_k\, A_k\, D_k^\top$$

λ_k contrôle le **volume**, D_k l'**orientation**, A_k la **forme** ($\det A_k = 1$). En contraignant (E = Égal entre groupes) ou libérant (V = Variable) chacune de ces trois composantes, `mclust` définit 14 familles de modèles (EII, VII, EEI, EEE, …, VVV).

14.1.4 La vraisemblance : mesurer l'adéquation du modèle

La **log-vraisemblance** $\ell(\Theta)$ mesure à quel point le modèle ajusté "colle" aux données observées :

$$\ell(\Theta) = \sum_{i=1}^{n} \ln\left[\sum_{k=1}^{K} \pi_k\, \mathcal{N}_p(x_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]$$

Analogie pédagogique

Imaginez 600 élèves répartis dans 3 classes (nos observations). La vraisemblance répond à : *"Quelle configuration de classes (centres, dispersions, proportions) aurait le plus de chances de produire exactement ces 600 élèves ?"* Plus ℓ est grand (moins négatif), mieux le modèle explique les données. L'algorithme EM cherche précisément les paramètres qui maximisent ℓ.

Chaque terme ln[...] est négatif (logarithme d'une probabilité < 1), donc $\ell < 0$ en général. On cherche à le **maximiser**, c'est-à-dire à le rendre le moins négatif possible. Un modèle avec davantage de composantes K ou plus de paramètres libres atteint toujours une vraisemblance au moins aussi élevée — d'où la nécessité de la pénalisation du BIC.

14.1.5 Les degrés de liberté : le coût de la complexité

Le paramètre d dans la formule du BIC est le **nombre de paramètres libres** du modèle. Il quantifie la complexité : plus un modèle a de paramètres, plus il est flexible, mais plus il risque d'apprendre le bruit plutôt que la vraie structure.

Pour les données de Flynt & Dean ($p = 2$ variables, $K = 3$ groupes), décomposons le décompte selon la structure du modèle :

- **Proportions** π_k : $K - 1 = 2$ paramètres libres (car $\sum_k \pi_k = 1$, la dernière proportion est contrainte).
- **Vecteurs de moyennes** $\boldsymbol{\mu}_k$: $K \times p = 3 \times 2 = 6$ paramètres.
- **Matrices de covariance** : dépend de la famille de modèle. Pour VVV (covariances totalement libres), chaque matrice 2×2 symétrique compte $p(p+1)/2 = 3$ paramètres, soit $3 \times 3 = 9$ au total. Des familles plus contraintes (ex. EEE, EVE) partagent certains paramètres entre les groupes et comptent donc moins de paramètres libres.

À retenir

Le modèle sélectionné par `mclust` sur nos données ($K = 3$, $p = 2$) compte d **degrés de liberté** (accessible via `mc_model$df` en R). Sur $n = 600$ observations, $\ln(600) \approx 6{,}4$: chaque paramètre supplémentaire coûte environ **6,4 points de BIC**, ce qui rend la pénalisation très efficace pour écarter les modèles sur-paramétrés.

14.1.6 Sélection automatique via le BIC

`mclust` (Fraley et Raftery 2002; Scrucca et al. 2016) sélectionne simultanément K et la famille de covariance en maximisant le BIC de Fraley-Raftery (Schwarz 1978) :

$$\text{BIC} = \underbrace{2 \ln \hat{L}}_{\text{ajustement}} - \underbrace{d \ln n}_{\text{pénalité complexité}}$$

La formule arbitre un compromis fondamental :

- $2 \ln \hat{L}$ récompense l'**ajustement** aux données : plus le modèle colle aux observations, plus ce terme est grand. Ajouter des composantes K ou assouplir les covariances augmente toujours ce terme.
- $d \ln n$ **pénalise la complexité** proportionnellement au nombre de paramètres d et à la taille de l'échantillon n. Sur $n = 600$, $\ln(600) \approx 6{,}4$: chaque paramètre libre "coûte" 6,4 points de BIC.

`mclust` évalue systématiquement toutes les combinaisons $(K, \text{structure})$ et retient celle qui maximise le BIC — double sélection automatique que le K-means ne peut pas faire.

Point d'attention

Convention inversée. Dans la littérature standard, le BIC se *minimise* ($-2\ln\hat{L} + d\ln n$). Dans `mclust`, il se **maximise** — signe opposé, même principe de pénalisation de la complexité. Ne pas confondre les deux conventions en comparant des sources différentes.

```
mc_model_v2 <- Mclust(X_data[, 1:2])
tab_mc_v2   <- table(Vérité = cl_real, Mclust = mc_model_v2$classification)

# Résumé du modèle optimal
cat("=== Modele optimal selectionne par mclust ===\n")
```

```
#> === Modele optimal selectionne par mclust ===
```

```
cat("Modele de covariance :", mc_model_v2$modelName, "\n")
```

```
#> Modele de covariance : VVE
```

```
cat("Nombre de composantes K :", mc_model_v2$G, "\n")
```

```
#> Nombre de composantes K : 3
```

```
cat("Degres de liberte (d) :", mc_model_v2$df, "\n")
```

```
#> Degres de liberte (d) : 15
```

```
cat("Log-vraisemblance (l) :", round(mc_model_v2$loglik, 2), "\n")
```

```
#> Log-vraisemblance (l) : -2230.69
```

```
cat("BIC Fraley-Raftery    :", round(mc_model_v2$bic, 2), "\n")
```

```
#> BIC Fraley-Raftery    : -4557.34
```

```
bic_matrix <- mc_model_v2$BIC
bic_df <- data.frame(
  K      = as.integer(rep(row.names(bic_matrix), ncol(bic_matrix))),
  Modele = rep(colnames(bic_matrix), each = nrow(bic_matrix)),
  BIC    = round(as.vector(bic_matrix), 2)
)
bic_df <- bic_df[!is.na(bic_df$BIC), ]
bic_df <- bic_df[order(-bic_df$BIC), ]
bic_top5 <- head(bic_df, 5)
row.names(bic_top5) <- NULL

kbl(bic_top5,
    booktabs  = TRUE,
    align     = "ccr",
    col.names = c("K", "Modèle", "BIC")) |>
  kable_styling(latex_options = "hold_position") |>
```

Table 12. Cinq meilleurs modèles selon le BIC de Fraley-Raftery (plus élevé = meilleur). Le modèle retenu est en surbrillance.

K	Modèle	BIC
3	**VVE**	**-4557.34**
3	EVE	-4565.03
3	VVV	-4566.27
3	EVV	-4571.73
4	VVE	-4581.89

```
  row_spec(1, bold = TRUE, color = "white", background = "#003A70")
```

```
mc_res_v2 <- Mclust(X_data[, 1:2], G = 3)

X_mclust_v2 <- X_data |>
  mutate(
    Verite         = as.factor(cl_real),
    Cluster_Mclust = as.factor(mc_res_v2$classification)
  )

bleu_v2 <- rgb(0, 70, 127, maxColorValue = 255)

ggplot(X_mclust_v2, aes(x = V1, y = V2)) +
  geom_point(aes(color = Verite), alpha = 0.6, size = 1.2, shape = 18) +
  ggforce::geom_mark_hull(
    aes(fill = Cluster_Mclust, group = Cluster_Mclust,
        label = paste("Cluster", Cluster_Mclust)),
    alpha = 0.10,
    color = bleu_v2,
    linetype = "solid",
    concavity = 10000,
    radius = unit(0, "mm"),
    expand = unit(0, "mm"),
    label.buffer  = unit(5, "mm"),
    label.fontsize = 9,
    label.fontface = "bold",
    na.rm = TRUE
  ) +
  scale_color_manual(values = couleurs_article, name = "Vérité Terrain") +
  scale_fill_manual(values = couleurs_article) +
  labs(
    title    = "Mclust : Identification des structures gaussiennes",
    subtitle = "Les étiquettes désignent les populations identifiées par l'algorithme",
    x = "Variable V1", y = "Variable V2"
  ) +
  theme_pdf() +
  theme(legend.position = "bottom",
        plot.title      = element_text(face = "bold")) +
```

Table 13. Performance du modèle Mclust (G=3) sur les données Flynt & Dean.

Indicateur	Résultat
Points bien capturés	592 / 600
Taux de réussite (Capture)	**98.67 %**
Taux d'erreur	1.33 %

```
  guides(fill = "none")
```

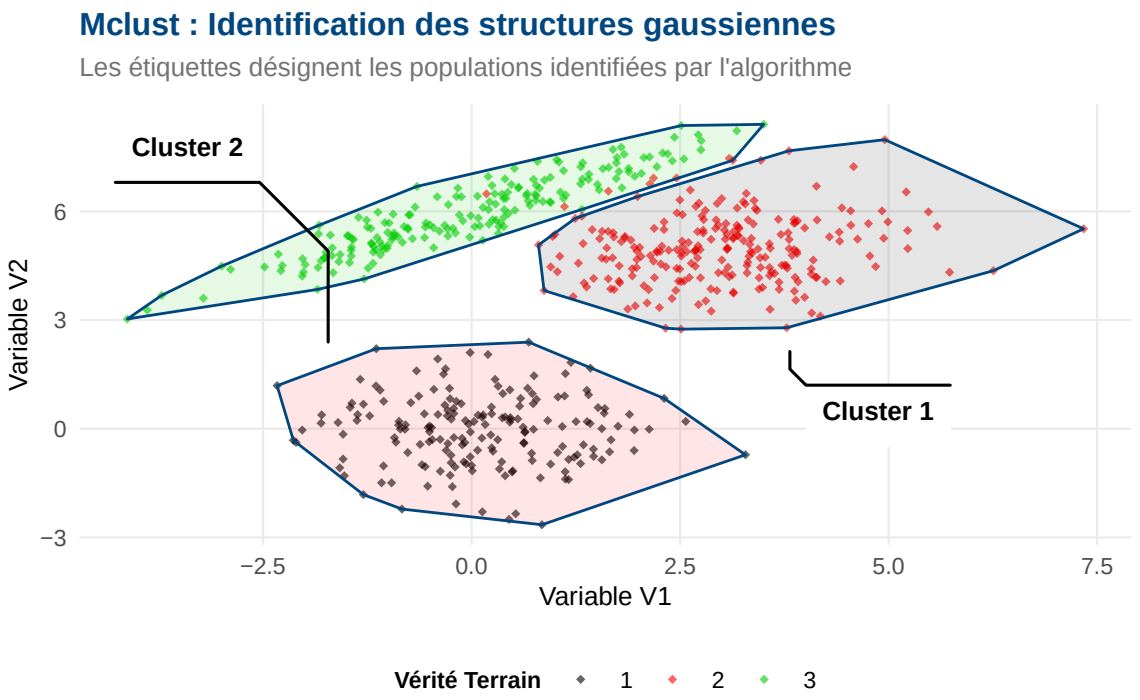

Figure 20. Mclust (G=3) : enveloppes convexes étiquetées. Les contours épousent la forme elliptique du Groupe 3, contrairement aux enveloppes du K-means.

```
mc_full_fviz <- Mclust(X_data[, 1:2], G = 1:9)
fviz_mclust_bic(mc_full_fviz,
                palette = "jco",
                legend  = "right",
                ggtheme = theme_minimal())
```

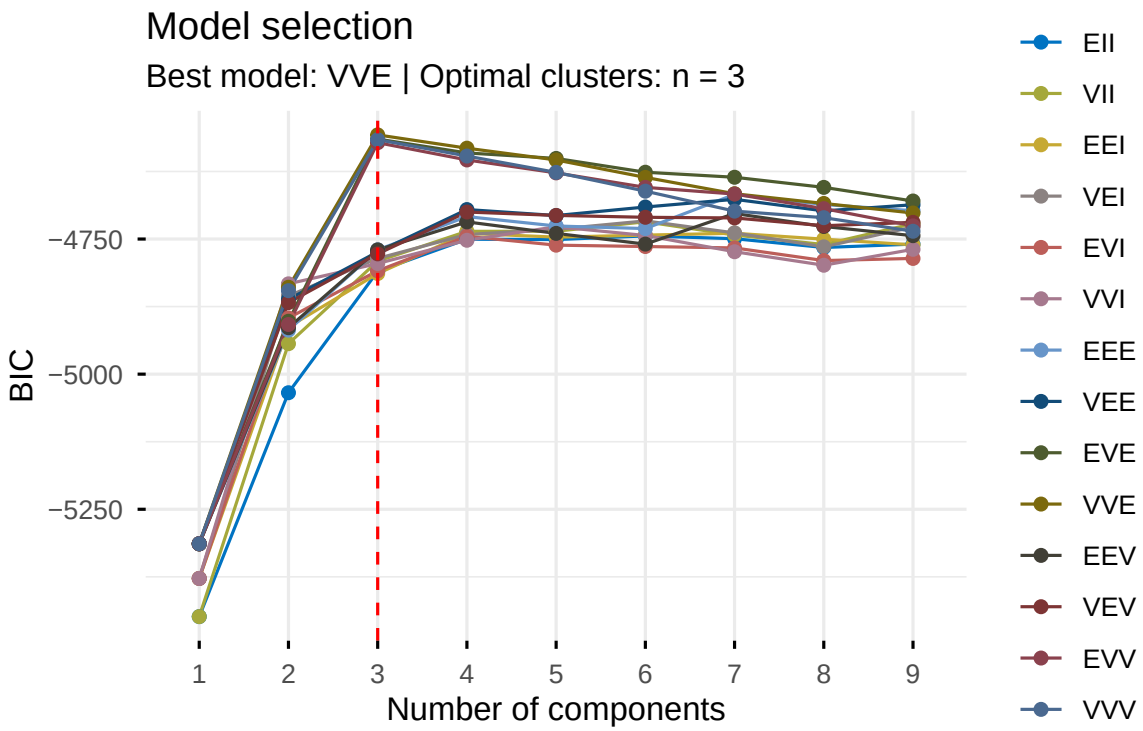

Figure 21. Sélection du modèle optimal par critère BIC (visualisation factoextra). Le meilleur modèle est clairement identifié pour K=3.

```
ggplot(X_mclust_v2, aes(x = V1, y = V2, color = Cluster_Mclust)) +
  geom_point(alpha = 0.5, size = 1.5, shape = 16) +
  stat_ellipse(aes(fill = Cluster_Mclust), geom = "polygon",
               alpha = 0.1, level = 0.95) +
  scale_color_manual(values = c("black", "red", "green3")) +
  scale_fill_manual(values  = c("black", "red", "green3")) +
  labs(title    = "Structure des populations identifiées",
       subtitle = "Les ellipses de confiance illustrent la modélisation gaussienne") +
  theme_pdf()
```

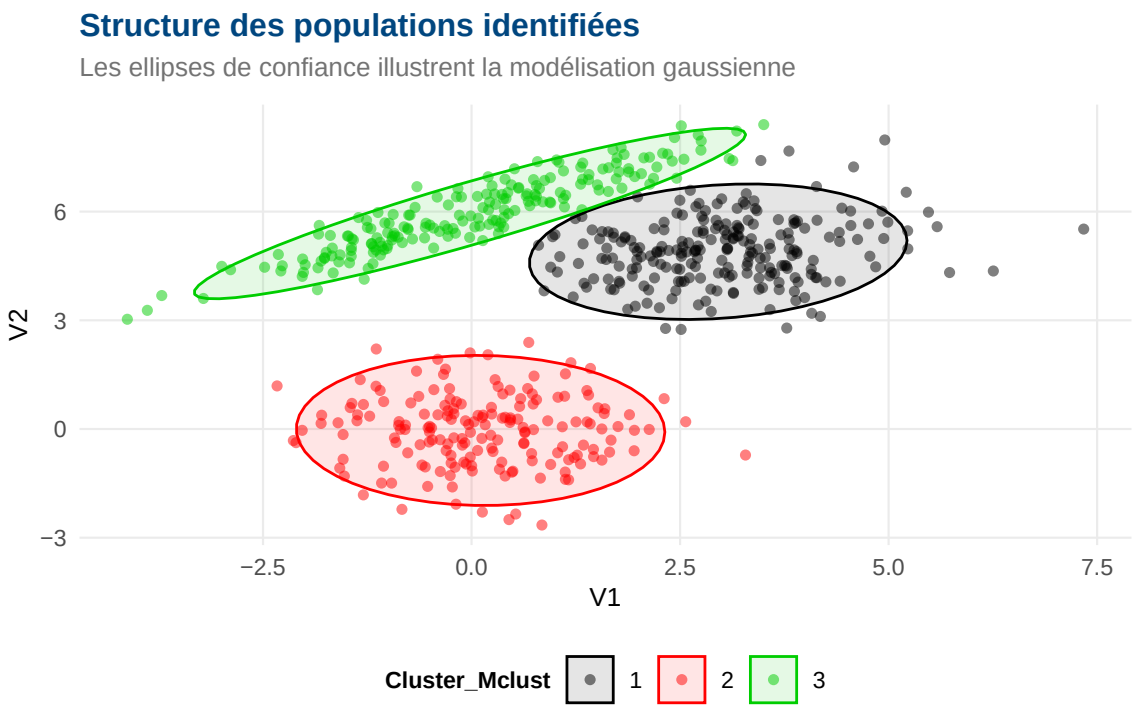

Figure 22. Structure des populations identifiées par Mclust avec ellipses de confiance à 95%.

```
X_mclust_v2$Uncertainty <- mc_res_v2$uncertainty

ggplot(X_mclust_v2, aes(x = V1, y = V2, color = Uncertainty)) +
  geom_point(size = 2) +
  scale_color_gradient(low = "gray80", high = "firebrick1") +
  labs(title    = "Zones de doute de l'algorithme Mclust",
       subtitle = "Les points rouges marquent les frontières de décision ambiguës",
       color    = "Incertitude") +
  theme_pdf()
```

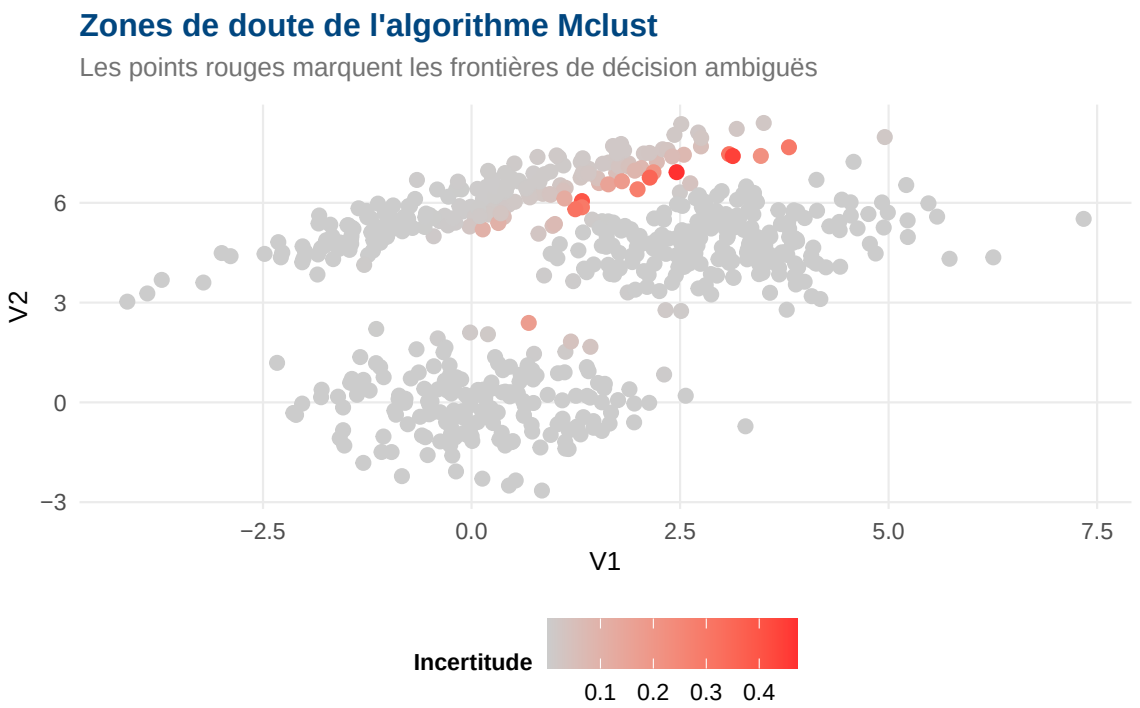

Figure 23. Cartographie de l'incertitude de classification Mclust. Les points rouges signalent les frontières de décision ambiguës.

Calcul du taux de réussite Mclust

```
$$T_{réussite} = \frac{177 + 229 + 186}{600} = \frac{592}{600} \approx 0{,}9867 \quad \Rightarrow \quad \textbf{98,67\,\%}$$
```

$$T_{réussite} = \frac{177 + 229 + 186}{600} = \frac{592}{600} \approx 0{,}9867 \quad \Rightarrow \quad \textbf{98,67\,\%}$$

```
$$T_{erreur} = 1 - 0{,}9867 = 0{,}0133 = \frac{8}{600} \approx 1{,}33\,\%$$
```

$$T_{erreur} = 1 - 0{,}9867 = 0{,}0133 = \frac{8}{600} \approx 1{,}33\,\%$$

À retenir

Le triomphe de Mclust : En ne forçant pas une structure circulaire, `mclust` parvient à capturer l'intégralité de l'ellipse verte sans déborder sur les groupes voisins. C'est la conclusion majeure de Flynt & Dean : pour des données réelles complexes, **le choix du modèle statistique prime sur la simplicité géométrique**.

Table 14. Comparaison des méthodes de clustering sur les données simulées de Flynt & Dean (2016). L'ARI quantifie l'adéquation avec la vérité terrain : ARI = 1 signifie correspondance parfaite, ARI = 0 correspond au hasard.

Méthode	Données utilisées	Forme des clusters	K automatique ?	ARI vs vérité terrain
K-means (K=3)	Continues (V1, V2)	Sphérique	Non	**0.798**
CHA — Lien moyen (K=3)	Continues (V1, V2)	Quelconque	Coupure dendro	**0.534**
Mclust GMM (K=3)	**Continues (V1, V2)**	**Elliptique**	**BIC auto**	**0.959**

15 Comparaison et synthèse des méthodes

15.1 Tableau comparatif des performances

À retenir

Lecture du tableau : Les trois méthodes sont évaluées sur les mêmes variables continues (V_1, V_2) et la même vérité terrain, ce qui rend les ARI directement comparables. Mclust domine grâce à sa modélisation elliptique, tandis que la CHA souffre de ses fusions irréversibles (30 % d'erreur).

16 Conclusion générale

16.1 Bilan du parcours : de R aux méthodes non supervisées

Ce document a constitué un voyage pédagogique en trois actes, tous ancrés dans l'unité d'enseignement ESD113 de Monsieur Karim KILANI.

Le **premier acte** nous a conduits des fondements de R et de Quarto jusqu'à la maîtrise du Tidyverse : manipulation de données, statistiques descriptives, visualisation, et pipe natif. Ces outils constituent la grammaire de base de tout travail d'analyse avec R.

Le **deuxième acte** a appliqué cette grammaire à un cas concret : les données de tirage du Keno. Chemin faisant, nous avons exploré le pivot, la modélisation probabiliste par la loi hypergéométrique, le calcul de l'espérance de gain et la simulation de Monte-Carlo. Ce dernier outil — répéter à grande échelle des tirages aléatoires pour estimer des distributions — est l'un des plus puissants de la boîte à outils statistique, et sa mise en œuvre en R s'avère remarquablement concise.

Le **troisième acte** a plongé dans les méthodes de clustering, en reproduisant et en enrichissant les résultats de l'article fondateur de Flynt & Dean (2016). Trois méthodes ont été passées au crible sur un même jeu de données simulées :

1. **K-means** : rapide, explicable, mais contraint par une hypothèse sphérique qui le met en difficulté face aux structures elliptiques. Six variations d'implémentation ont montré qu'un même résultat statistique peut s'écrire de la ligne artisanale au graphique automatisé.
2. **La CHA** : précieuse pour l'exploration multi-échelle via le dendrogramme, mais pénalisée par ses fusions irréversibles (30 % d'erreur sur nos données).
3. **Mclust** : champion incontesté, avec 98,67 % de réussite, grâce à sa modélisation elliptique et à la sélection automatique du modèle via le BIC.

16.2 Ce que les données simulées nous ont appris

L'utilisation d'un fil conducteur unique — les données de Flynt & Dean (2016) — a mis en lumière une leçon centrale : **le groupe elliptique est le vrai révélateur des hypothèses implicites de chaque algorithme**. K-means (7,% d'erreur sur ce groupe), CHA (30,% d'erreur globale), Mclust (1,33,% d'erreur) — ces chiffres ne décrivent pas trois niveaux de complexité informatique, mais trois façons différentes de "voir" les données.

16.3 La hiérarchie de performance (contexte-dépendante)

Pour ces données continues simulées, la hiérarchie de performance est claire :

```
\begin{equation}
\texttt{Mclust} \;>\; \texttt{K-means} \;>\; \texttt{CHA}
\end{equation}
```

$$\texttt{Mclust} > \texttt{K-means} > \texttt{CHA} \tag{2}$$

Cette hiérarchie n'est cependant pas universelle. Elle devient une *carte de recommandations contextuelles* :

Situation	**Méthode recommandée**
Exploration rapide, N grand	K-means (nstart élevé)
Clusters gaussiens, K inconnu	Mclust (BIC automatique)
Lecture multi-échelle, exploration	CHA + dendrogramme

À retenir

Le clustering n'est pas une vérité : c'est une **hypothèse de travail** que l'analyste fait sur la structure des données. La vraie compétence du data scientist est de choisir — et de savoir justifier — pourquoi une méthode est plus adaptée que les autres au problème posé. Nous pouvons également faire une dernière remarque : la méthode des modèles gaussiens a été performante pour détecter des clusters... gaussiens !

16.4 Applications concrètes du clustering dans la vie réelle

Le clustering n'est pas un exercice académique sur des données simulées : c'est l'un des outils les plus déployés dans les secteurs publics et privés. Voici un panorama de cas d'usages réels, du commerce à la défense.

Commerce et marketing — la segmentation client. Les enseignes de la grande distribution (Carrefour, Amazon, Fnac) appliquent quotidiennement le K-means ou les GMM à leurs bases de données d'achats. Chaque client se voit attribuer un segment ("acheteur occasionnel à fort panier", "fidèle low-cost", "early adopter high-tech") qui pilote les campagnes e-mail, les recommandations produits et les programmes de fidélité. Spotify, Apple Music ou Netflix effectuent la même opération sur les comportements d'écoute ou de visionnage pour identifier les "profils d'usage" qui guident les algorithmes de recommandation.

Santé et biologie — la médecine de précision. En oncologie, les chercheurs appliquent la CHA ou les GMM à des profils d'expression génique de tumeurs pour en identifier les sous-types moléculaires. C'est cette approche qui a conduit à distinguer, par exemple, plusieurs sous-types de cancer du sein (Luminal A, Luminal B, HER2-enrichi, Triple-négatif), chacun répondant différemment aux traitements. En épidémiologie, le clustering géographique de cas permet d'identifier des foyers épidémiques avant même que les alertes officielles ne soient déclenchées.

Finance — la détection de fraude. Les départements de lutte anti-fraude des banques (BNP Paribas, Société Générale, Visa) utilisent des algorithmes de clustering non supervisé pour détecter les comportements de paiement anormaux. Un cluster de transactions nocturnes à l'étranger avec des montants inhabituels émerge automatiquement — sans qu'on ait besoin de définir à l'avance ce

qu'est une fraude. Le clustering sert également à regrouper les portefeuilles d'investissement selon leurs profils risque/rendement pour optimiser l'allocation d'actifs.

Défense et sécurité nationale — un enjeu croissant. Dans le domaine de la défense, le clustering trouve des applications directes et stratégiques. L'analyse de signaux électromagnétiques (SIGINT) utilise des algorithmes de partitionnement pour regrouper automatiquement les émissions radio selon leur signature spectrale — permettant d'identifier des équipements ennemis sans les avoir répertoriés au préalable. La fusion de données ISR (Intelligence, Surveillance, Reconnaissance) applique le clustering spatio-temporel pour détecter des patterns comportementaux anormaux sur un théâtre d'opérations : regroupements inhabituels de véhicules, modifications de routines logistiques, concentration de trafic réseau. Enfin, en cybersécurité de défense, les CERT militaires (comme le CALID en France) utilisent la classification non supervisée de logs systèmes pour isoler automatiquement les comportements malveillants au sein de millions d'événements journaliers — notamment pour détecter les intrusions persistantes avancées (APT).

Image et vision par ordinateur. La compression d'image JPEG utilise le K-means pour réduire la palette de couleurs d'une image. La segmentation d'images satellitaires — qu'il s'agisse de cartographier des cultures agricoles ou de surveiller l'évolution du couvert forestier — repose sur des algorithmes de clustering appliqués aux pixels selon leur signature spectrale (visible, infrarouge, radar).

À retenir

Dans tous ces domaines, la leçon reste la même : le clustering révèle une structure que les données portent en elles mais que l'analyste n'avait pas définie *a priori*. Sa puissance réside précisément dans cette capacité à faire *émerger* de la connaissance à partir de l'observation brute, sans étiquette préalable.

16.5 Pour aller plus loin

- **Flynt et Dean** (2016) : la revue des packages R pour le clustering, article de référence de ce document.
- **Fraley et Raftery** (2002) : l'article fondateur de `Mclust` et des mélanges gaussiens.
- **Everitt et al.** (2011) : le livre de référence complet sur le clustering.
- **Wickham et al.** (2019) : l'écosystème `tidyverse`.

Références bibliographiques

Allaire, J. J., Charles Teague, Carlos Scheidegger, Yihui Xie, et Christophe Dervieux. 2024. *Quarto*. https://doi.org/10.5281/zenodo.5960048.

Dempster, Arthur P., Nan M. Laird, et Donald B. Rubin. 1977. « Maximum Likelihood from Incomplete Data via the EM Algorithm ». *Journal of the Royal Statistical Society: Series B* 39 (1): 1-38. https://doi.org/10.1111/j.2517-6161.1977.tb01600.x.

Everitt, Brian S., Sabine Landau, Morven Leese, et Daniel Stahl. 2011. *Cluster Analysis*. 5e éd. Chichester: John Wiley & Sons. https://doi.org/10.1002/9780470977811.

Flynt, Abby, et Nema Dean. 2016. « A Survey of Popular R Packages for Cluster Analysis ». *Journal of Educational and Behavioral Statistics* 41 (2): 205-25. https://doi.org/10.3102/1076998616631743.

Fraley, Chris, et Adrian E. Raftery. 2002. « Model-Based Clustering, Discriminant Analysis, and Density Estimation ». *Journal of the American Statistical Association* 97 (458): 611-31. https://doi.org/10.1198/016214502760047131.

Johnson, Richard A., et Dean W. Wichern. 2002. *Applied Multivariate Statistical Analysis*. 5e éd. Upper Saddle River, NJ: Prentice Hall.

Lloyd, Stuart P. 1982. « Least Squares Quantization in PCM ». *IEEE Transactions on Information Theory* 28 (2): 129-37. https://doi.org/10.1109/TIT.1982.1056489.

MacQueen, James. 1967. « Some Methods for Classification and Analysis of Multivariate Observations » 1: 281-97.

Pearson, Karl. 1894. « Contributions to the Mathematical Theory of Evolution ». *Philosophical Transactions of the Royal Society of London A* 185: 71-110. https://doi.org/10.1098/rsta.1894.0003.

R Core Team. 2024. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Schwarz, Gideon. 1978. « Estimating the Dimension of a Model ». *The Annals of Statistics* 6 (2): 461-64. https://doi.org/10.1214/aos/1176344136.

Scrucca, Luca, Michael Fop, T. Brendan Murphy, et Adrian E. Raftery. 2016. « mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models ». *The R Journal* 8 (1): 289-317. https://doi.org/10.32614/RJ-2016-021.

Ward, Joe H. 1963. « Hierarchical Grouping to Optimize an Objective Function ». *Journal of the American Statistical Association* 58 (301): 236-44. https://doi.org/10.1080/01621459.1963.10500845.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. « Welcome to the tidyverse ». *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Rapport de compilation

Méthode de mesure du temps

Lorsque vous cliquez sur **Render** dans RStudio, la compilation de ce document traverse trois phases successives, dont seule la deuxième est mesurable depuis R :

Phase	Outil	Ce qui se passe
1 — Quarto/pandoc	Quarto CLI	Quarto lit le `.qmd`, prépare l'environnement knitr, et lance R. Cette phase n'est **pas mesurable** depuis R car R n'a pas encore démarré. Elle dure typiquement 2 à 5 secondes.
2 — knitr (R)	knitr + R	Tous les chunks R sont exécutés séquentiellement. **C'est la seule phase mesurable depuis R.** Un chronomètre global (`proc.time()`) est lancé au premier chunk `setup` et arrêté au dernier chunk `timer-final`. Le total knitr est sauvegardé dans `compilation_stats.txt` et **lu dès la compilation suivante** via une variable globale initialisée dans `setup` avant l'écrasement du fichier.
3 — pdflatex	pdflatex	Pandoc convertit le `.md` intermédiaire en `.tex`, puis pdflatex compile le PDF (souvent 2 à 3 passes pour les références, le sommaire et les flottants). Cette phase n'est **pas mesurable** depuis R car elle se déroule après que knitr a rendu la main. Elle représente souvent 20 à 40 % du temps total.

Point d'attention

Pourquoi le total des chunks est inférieur au temps total ? La somme des durées individuelles des chunks représente le *temps CPU pur des calculs R*. Le chronomètre global knitr (Phase 2) est légèrement supérieur car il inclut également le temps de rendu ggplot2 vers fichier image, les appels système et les entrées/sorties disque entre les chunks. Les Phases 1 et 3 (Quarto + pdflatex) s'ajoutent en dehors de toute mesure R et constituent l'**overhead incompressible** de la compilation.

Table 15. Caractéristiques de la machine de compilation.

Paramètre	Valeur
Plateforme	unix
Système d'exploitation	Ubuntu 24.04.4 LTS
Version R	R version 4.6.0 (2026-04-24)
Processeur	AMD EPYC 7R13 Processor
Cœurs logiques	16
RAM totale (Go)	132.2
Début phase knitr	18/05/2026 10:22:07

Environnement système

Temps d'exécution des chunks R (Phase 2)

Bilan global des phases

À retenir

Comment fonctionne le timer ? Au lancement de chaque compilation, le chunk `setup` lit le fichier `compilation_stats.txt` *avant* de l'écraser, et sauvegarde en mémoire le `[KNITR_TOTAL]` de la session précédente. Cette valeur est ainsi disponible **dès la première compilation** qui suit une compilation complète. Les tableaux affichent donc : la somme des chunks R de la compilation *en cours*, et le total knitr de la compilation *précédente* — ce qui est la seule approche fiable, puisque `timer-final` écrit ce total *après* que tous les tableaux ont été générés.

Table 16. Durées individuelles des chunks R, triées du plus lent au plus rapide. Les chunks surlignés en orange ont dépassé 5 secondes.

Rang	Chunk R	Durée (sec)
1	fig-dendro-color	**66.51**
2	fig-dendro-circular	**30.86**
3	fig-mclust-bic-fviz-v2	**3.26**
4	montecarlo-grande-echelle	**3.17**
5	mclust-fit	**2.94**
6	packages-clustering	**1.28**
7	fig-simul-donnees	**1.07**
8	var6-factoextra	**1.03**
9	fig-mclust-hull-v2	**0.91**
10	fig-dendro-simple	**0.74**
11	fig-kmeans-eval	**0.54**
12	ggplot-hist	**0.52**
13	fig-mclust-ellipses-v2	**0.49**
14	var1-manuelle	**0.47**
15	load-data	**0.45**
16	barchart-freq	**0.45**
17	var3-sapply	**0.45**
18	var4-purrr	**0.44**
19	fig-mclust-uncertainty-v2	**0.44**
20	fig-elbow	**0.43**
21	var2-for	**0.43**
22	tbl-keno-long	**0.40**
23	polar-chart	**0.39**
24	var5-broom	**0.37**
25	plot-esperances	**0.35**
26	tbl-comparaison-methodes	**0.28**
27	tbl-gains-tableau-complet	**0.20**
28	pivot-exemple	**0.19**
29	montecarlo-proba	**0.16**
30	seq-entiers	**0.13**
31	freq-boules	**0.10**
32	gains-calcul-long	**0.09**
33	tbl-palmares-pdf	**0.08**
34	tbl-proba-gain	**0.08**
35	tbl-esperances-grilles	**0.07**
36	tbl-tableau-gains	**0.05**
37	tbl-stats-desc	**0.04**
38	format-dates	**0.04**
39	tbl-sysinfo-v7	**0.04**
40	base-r	**0.03**
41	tidyverse-select	**0.03**
42	tbl-gains-tableau-10	**0.03**
43	perf-kmeans	**0.03**
44	tbl-bic-top	**0.03**
45	hist-base	**0.02**
46	pivot-keno	**0.02**
47	esperance	**0.02**
48	montecarlo-simple	**0.02**
49	montecarlo-replicate	**0.02**
50	tbl-params-simulation	**0.02**
51	tbl-synthese-variations	**0.02**
52	tbl-perf-kmeans	**0.02**
53	tbl-perf-mclust	**0.02**
54	seq-cdm	**0.01**
55	filter-mpg	**0.01**
56	gains-data	**0.01**

Somme chunks R : 120.30 sec | Phase knitr globale : 73.7 sec | Overhead I/O + rendus : -46.6 sec

Table 17. Bilan estimé des trois phases de compilation. Les phases 1 et 3 sont estimées par différence ; seule la phase 2 est mesurée.

Phase	Durée mesurée	Remarque
Phase 1 — Quarto / pandoc	**Non mesurable depuis R**	2–5 sec estimées
Phase 2a — Exécution chunks R (somme)	**120.42 sec**	Mesuré chunk par chunk
Phase 2b — Overhead knitr (I/O, rendus)	**-46.7 sec**	Rendu ggplot2, écriture fichiers
Phase 2 — Total knitr mesuré	**73.7 sec**	**Seule valeur exacte**
Phase 3 — pdflatex (2–3 passes LaTeX)	**Non mesurable depuis R**	20–40 % du temps total estimé