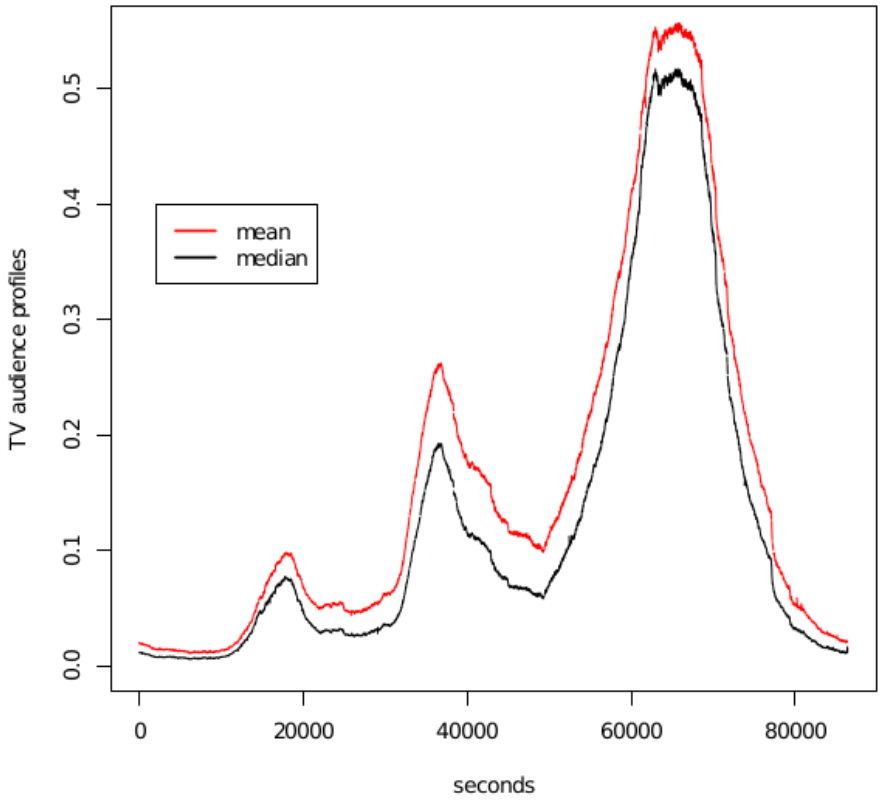
Université de Poitiers 2024-2025 MASTER 2 STDV

Projet de Modélisation et d'Algorithme Stochastique

Réalisé par :

Kodzo LIMA :
kodzo.lima@etu.univ-poitiers.fr

Rendu des TPs



Chargé de l'UE

MC HDR Yousri Slaoui

Table des matières

1 Introduction **3**

2 TP1 **3**

2.1 Exercice 1 . . . 3

2.1.1 Réalisation d'un script de R pour générer un mélange de trois lois normales : 3

2.1.2 Représentation graphique de ce mélange de lois normales . . . 6

2.1.3 Mélange de trois lois gaussiennes bidimensionnelles . . . 7

2.2 Exercice 2 . . . 9

2.2.1 Estimation théorique : Estimer les paramètres du modèle $\theta = (\tau_1, \tau_2, \mu_1, \mu_2, \Sigma_1, \Sigma_2)$ 9

2.2.2 Résultats de l'algorithme EM . . . 10

2.3 Exercice 3 . . . 13

2.3.1 Résultats de l'algorithme EM . . . 13

2.3.2 Résultats de l'algorithme EM . . . 14

3 TP2 **17**

3.1 Exercice 1 . . . 17

3.1.1 Écriture de X_n en fonction de Y_n . . . 17

3.1.2 Simulation des valeurs de X_n pour nallant de 1 à N . . . 17

3.1.3 Renvoie de la commande cumsum . . . 18

3.1.4 Déduction du vecteur $(S_1, ..., S_N)$ . . . 18

3.1.5 Que renvoie c(a,b)/c(c,d) . . . 20

3.1.6 Déduction de $(\overline{X}_1, ...\overline{X}_n)$ en fonction de $(S_1, ..., S_N)$ . . . 20

3.1.7 Montrer que $\overline{X}_n$ converge en probabilité vers 0. . . . 20

3.1.8 Simuler et afficher plusieurs trajectoires de $\overline{X}_n$ pour n allant de 1 à $N = 5000$ . . . 21

3.1.9 Simulation des variables . . . 22

3.1.10 Simuler et afficher plusieurs trajectoires de Sn pour n allant de 1 à $N = 5000$ 23

3.1.11 Calcul Approché de π . . . 27

3.1.12 Processus et Loi de S_n . . . 27

3.1.13 Détermination de y en fonction de x . . . 30

3.1.14 a) Calcul de $E[Y_n]$ . . . 30

4 Conclusion **32**

1 Introduction

Dans l'objectif nous apprendre les compétences requises pour une formation donnant la possibilité de continuer en thèse ou de rentrer dans le domaine professionnel, l'Ue Agorithme stochastique est un atout pour répondre au choix futur des étudiants. Le but de cette Ue est de nous aider à acquérir les compétences d'estimation des paramètres de lois ou de mélange de lois non seulement en dimension 1 mais en dimension d de façon théorique et appliquée en utilisant les méthodes de maximum de vraisemblance et l'algorithme EM. Dans la suite nous mettre en application sur R les théories apprises et si nécessaire faire voir d'abord par des démonstrations la théorie qui est appliquée.

2 TP1

2.1 Exercice 1

2.1.1 Réalisation d'un script de R pour générer un mélange de trois lois normales :

$\mathcal{N}(-2, 0.3^2)$, $\mathcal{N}(0, 0.2^2)$ et $\mathcal{N}(2, 0.2^2)$

```r
# Définir les paramètres
set.seed(123)
n <- 1000

# Générer les données pour chaque loi gaussienne
A <- rnorm(n, mean = -2, sd = 0.3)
B <- rnorm(n, mean = 0, sd = 0.2)
C <- rnorm(n, mean = 2, sd = 0.2)
# Combiner les données pour créer le mélange
data <- c(A, B, C)
```

* Nous avons ensuite représenter ces trois lois pour avoir une idée de ce qui pourrait être le graphique du mélange de lois

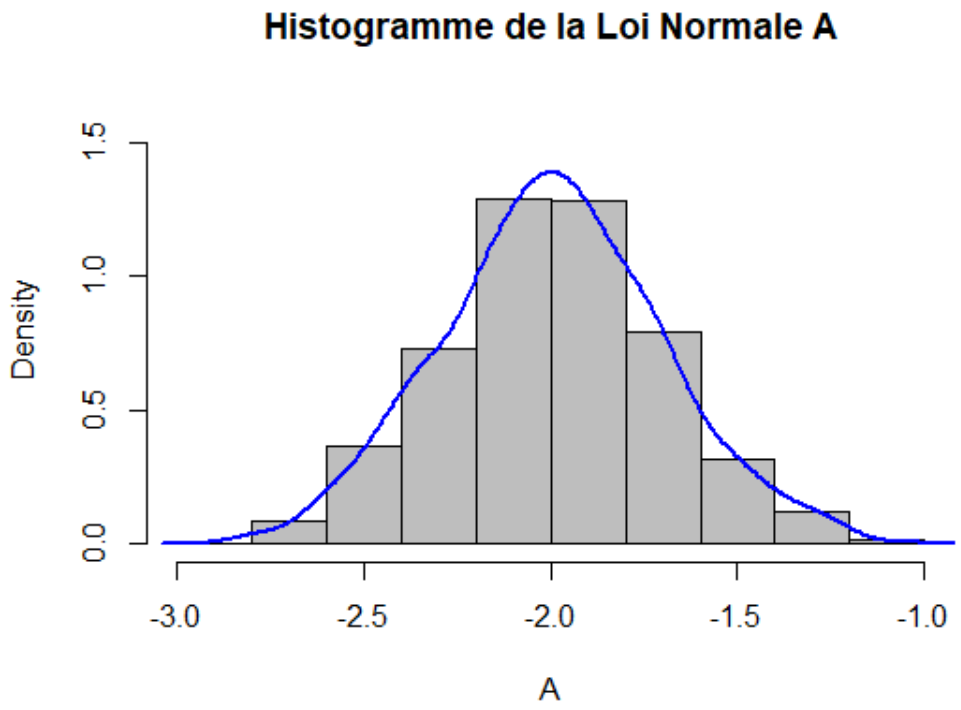


FIGURE 1 – Histogramme et courbe de la variable A

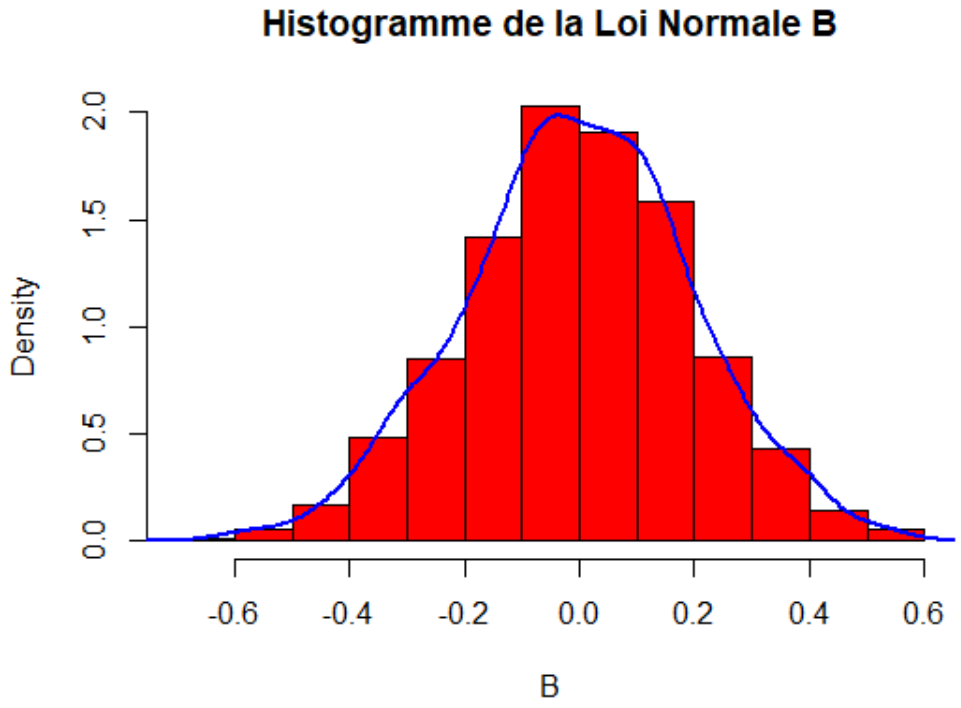


FIGURE 2 – Histogramme et courbe de la variable B

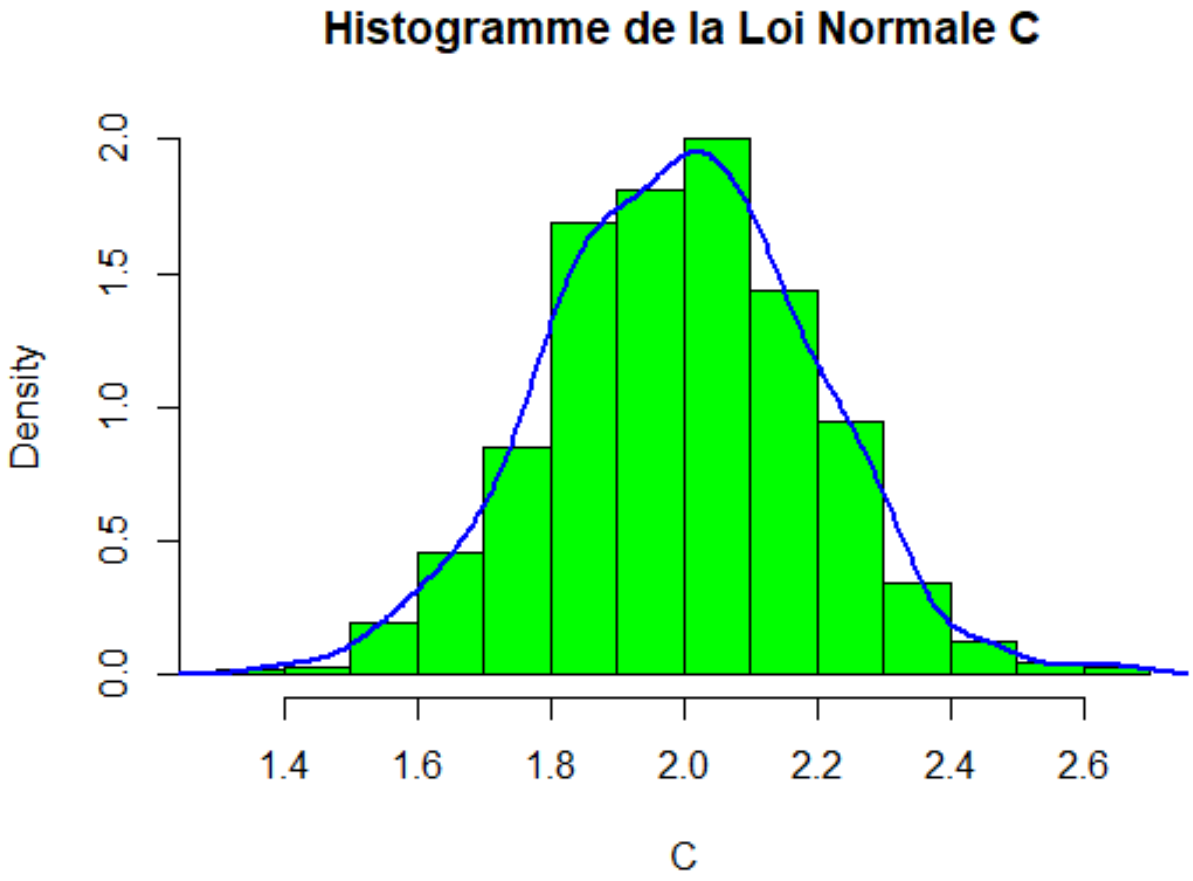


FIGURE 3 – Histogramme et courbe de la variable c

2.1.2 Représentation graphique de ce mélange de lois normales

```
# Représentation graphique de la densité du mélange
plot(density(data), col = "purple", lwd = 2,
     main = "Densité du mélange des trois lois gaussiennes",
     xlab = "Valeurs", ylab = "Densité")
```

On obtient donc le graphique suivant :

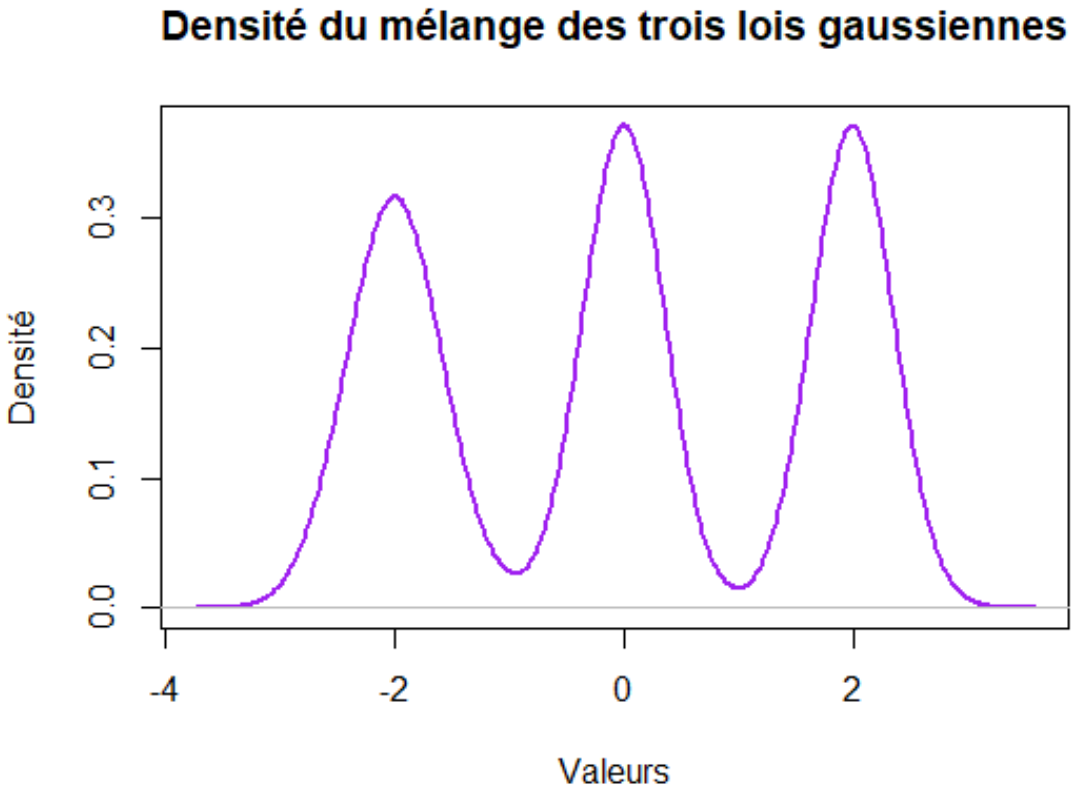


FIGURE 4 – Histogramme et courbe du mélange de lois

2.1.3 Mélange de trois lois gaussiennes bidimensionnelles

L'idée dans cette partie est de faire les mêmes questions pour un mélange de trois gaussiennes bidimensionnelles $\mathcal{N}(\mu_1, \Sigma_1)$,$\mathcal{N}(\mu_2, \Sigma_2)$ et $\mathcal{N}(\mu_3, \Sigma_3)$:

$$\mu_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \quad \mu_2 = \begin{pmatrix} 6 \\ 4 \end{pmatrix}, \quad \mu_3 = \begin{pmatrix} 12 \\ 9 \end{pmatrix},$$

$$\Sigma_1 = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}, \quad \Sigma_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \Sigma_3 = \begin{pmatrix} 1.5 & 0 \\ 0 & 1.5 \end{pmatrix}.$$

On met en application sur R ces informations de la façon suivante

```r
# Charger les bibliothèques nécessaires
library(MASS)
n <- 500

# Moyennes
mu1 <- c(2, 1)
mu2 <- c(6, 4)
mu3 <- c(12, 9)

# Matrices de covariance
Sigma1 <- matrix(c(0.5, 0, 0, 0.5), nrow = 2)
Sigma2 <- matrix(c(1, 0, 0, 1), nrow = 2)
Sigma3 <- matrix(c(1.5, 0, 0, 1.5), nrow = 2)

# Génération des échantillons
set.seed(123)  # Pour reproductibilité
data1 <- mvrnorm(n, mu = mu1, Sigma = Sigma1)
data2 <- mvrnorm(n, mu = mu2, Sigma = Sigma2)
data3 <- mvrnorm(n, mu = mu3, Sigma = Sigma3)

# Combiner les données pour former le mélange
data <- rbind(data1, data2, data3)

# Représenter graphiquement le nuage de points
plot(data, col = c(rep("red", n), rep("green", n), rep("blue", n)),
   pch = 16,
     main = "Nuage de points du mélange de trois lois normales",
     xlab = "X1", ylab = "X2")

# Ajouter une légende
legend("topleft", legend = c("Loi 1", "Loi 2", "Loi 3"),
       col = c("red", "green", "blue"), pch = 16)
```

*** Représenter graphiquement le nuage des points de l'échantillon mélange**

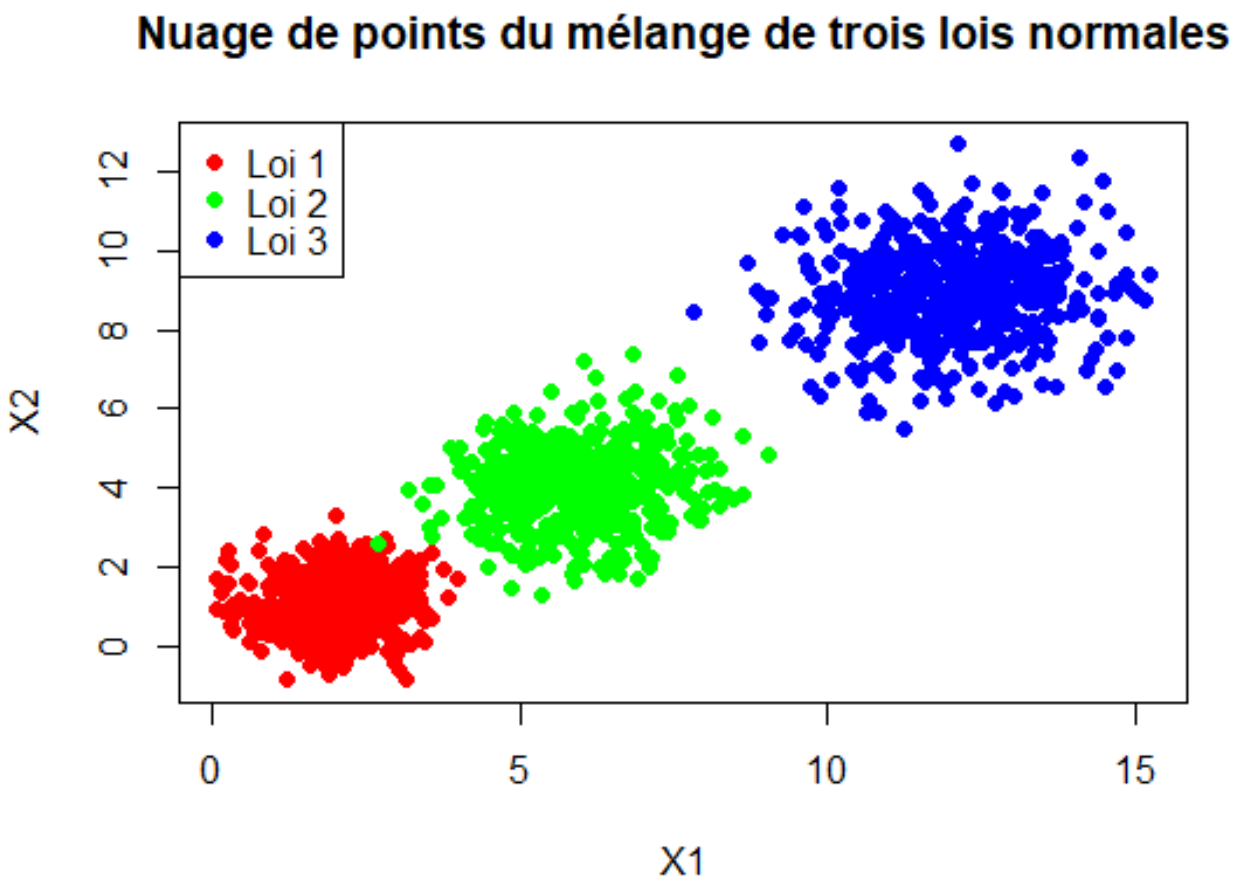


FIGURE 5 – Nuage de Points du mélange de lois gaussiennes bidimensionnelles

2.2 Exercice 2

2.2.1 Estimation théorique : Estimer les paramètres du modèle $\theta = (\tau_1, \tau_2, \mu_1, \mu_2, \Sigma_1, \Sigma_2)$

Soit $x = (x_1, x_2, \ldots, x_n)$ un échantillon de n observations indépendantes à partir de deux distributions normales multivariées de dimension d, et soit $z = (z_1, z_2, \ldots, z_n)$ les variables latentes qui déterminent la composante à partir de laquelle l'observation est originaire.

$$X_i|(Z_i = 1) \propto \mathcal{N}_d(\mu_1, \Sigma_1) \quad \text{et} \quad X_i|(Z_i = 2) \propto \mathcal{N}_d(\mu_2, \Sigma_2)$$

avec

$$\mathbb{P}(Z_i = 1) = \tau_1 \quad \text{et} \quad \mathbb{P}(Z_i = 2) = \tau_2 = 1 - \tau_1$$

D'après le cours nous pouvons écrire que :

$$L_n(X, Z|\theta) = \prod_{i=1}^{n} \left[\sum_{j=1}^{2} \mathbb{I}_{Z_i=j} \tau_j f_j(X_i) \right] \tag{3.1}$$

où $f_j : \mathbb{R}^N \to \mathbb{R}$ est la densité gaussienne de paramètre (μ_j, Σ_j) :

$$f_j(x) = \frac{1}{(2\pi)^{N/2} \det(\Sigma_j)^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_j)^T \Sigma_j^{-1} (x - \mu_j) \right).$$

Il vient donc, pour la log-vraisemblance des données complètes :

$$\log(L_n(X, Z|\theta)) = \sum_{i=1}^{n} \sum_{j=1}^{2} \mathbb{I}_{Z_i=j} \left(\log(\tau_j) - \frac{N}{2} \log(2\pi) - \frac{1}{2} \log(\det(\Sigma_j)) - \frac{1}{2}(x_i - \mu_j)^T \Sigma_j^{-1} (x_i - \mu_j) \right)$$

Nous pouvons écrire, d'après (3.1), que

$$\log(L_n(X, Z|\theta)) = \sum_{i=1}^{n} \sum_{j=1}^{2} \mathbb{I}_{Z_i=j} \left(\log(\tau_j) + \log(f_j(X_i)) \right) \tag{3.2}$$

par conséquent, on a

$$\mathbb{E}_{Z|X,\theta_m} \left[\log(L_n(X, Z|\theta)) \right] = \sum_{i=1}^{n} \sum_{j=1}^{2} \mathbb{P}(Z_i = j|X_i = x_i, \theta) \left(\log(\tau_j) + \log(f_j(X_i)) \right). \tag{3.3}$$

À l'itération, l'étape E nécessite de définir la distribution a posteriori de Z_j connaissant X_j et θ_m. On définit :

$$\hat{p}_{i,j}^{(m)} := \mathbb{P}\{Z_i = j|X_i = x_i, \theta_m\} = \frac{\tau_j f_j(x_i)}{\tau_1 f_1(x_i) + \tau_2 f_2(x_i)} \tag{3.4}$$

la probabilité a posteriori pour que le point X_i soit issu de la distribution $f_j \equiv \mathcal{N}(\mu_j, \Sigma_j)$, connaissant θ_m. Donc, d'après (eq : MLV :E), nous pouvons écrire que

$$\mathbb{E}_{Z|X,\theta_m}\left[\log(L_n(X,Z|\theta))\right] = \sum_{i=1}^{n}\sum_{j=1}^{2}\hat{p}_{i,j}^{(m)}\left(\log(\tau_j) + \log(f_j(X_i))\right). \tag{3.5}$$

La maximisation en θ de cette expression ne présente aucune difficulté :

$$\begin{aligned}\tau_1^{(m+1)} &= \arg\max_{\tau_1}\left\{\sum_{i=1}^{n}\hat{p}_{i,1}^{(m)}\log(\tau_1^{(m)}) + \sum_{i=1}^{n}\hat{p}_{i,2}^{(m)}\log(\tau_2^{(m)})\right\}\\ &= \arg\max_{\tau_1}\left\{\sum_{i=1}^{n}\hat{p}_{i,1}^{(m)}\log(\tau_1^{(m)}) + \sum_{i=1}^{n}(1-\hat{p}_{i,1}^{(m)})\log(1-\tau_1^{(m)})\right\}.\end{aligned}$$

Donc,

$$\frac{\partial\mathbb{E}_{Z|X,\theta_m}\left[\log(L_n(X,Z|\theta))\right]}{\partial\tau_1} = \frac{\sum_{i=1}^{n}\hat{p}_{i,1}^{(m)}}{\tau_1} - \frac{\sum_{i=1}^{n}(1-\hat{p}_{i,1}^{(m)})}{1-\tau_1},$$

ce qui implique que

$$\hat{\tau}_1^{(m)} = \frac{1}{n}\sum_{i=1}^{n}\hat{p}_{i,1}^{(m)}.$$

Il est facile aussi de montrer que

$$\begin{cases}\hat{\tau}_j^{(m)} = \frac{1}{n}\sum_{i=1}^{n}\hat{p}_{i,j}^{(m)}\\ \hat{\mu}_j^{(m)} = \frac{\sum_{i=1}^{n}\hat{p}_{i,j}^{(m)}x_i}{\sum_{i=1}^{n}\hat{p}_{i,j}^{(m)}}\\ \hat{\Sigma}_j^{(m)} = \frac{\sum_{i=1}^{n}\hat{p}_{i,j}^{(m)}(x_i-\hat{\mu}_j^{(m)})(x_i-\hat{\mu}_j^{(m)})^T}{\sum_{i=1}^{n}\hat{p}_{i,j}^{(m)}}.\end{cases}$$

2.2.2 Résultats de l'algorithme EM

On utilise le code suivant pour implémenter le code permettant d'obtenir les estimations des différents paramètres.

```
# Charger les bibliothèques nécessaires
library(mvtnorm)

# Générer des données simulées
set.seed(123)
n <- 500
tau1 <- 0.6
tau2 <- 1 - tau1
mu1 <- c(2, 3)
mu2 <- c(7, 8)
Sigma1 <- matrix(c(1, 0.5, 0.5, 2), nrow = 2)
Sigma2 <- matrix(c(1.5, 0.3, 0.3, 1), nrow = 2)

z <- rbinom(n, 1, tau1)
data <- matrix(0, nrow = n, ncol = 2)
data[z == 1, ] <- rmvnorm(sum(z == 1), mean = mu1, sigma = Sigma1)
```

```r
data[z == 0, ] <- rmvnorm(sum(z == 0), mean = mu2, sigma = Sigma2)

# Initialisation
tau1_hat <- 0.5
tau2_hat <- 1 - tau1_hat
mu1_hat <- colMeans(data) - 1
mu2_hat <- colMeans(data) + 1
Sigma1_hat <- diag(2)
Sigma2_hat <- diag(2)

# Algorithme EM
max_iter <- 100
tol <- 1e-6
log_likelihood_old <- -Inf

for (iter in 1:max_iter) {
  # E-step
  gamma1 <- tau1_hat * dmvnorm(data, mean = mu1_hat, sigma = Sigma1_
     hat)
  gamma2 <- tau2_hat * dmvnorm(data, mean = mu2_hat, sigma = Sigma2_
     hat)
  gamma_sum <- gamma1 + gamma2
  gamma1 <- gamma1 / gamma_sum
  gamma2 <- gamma2 / gamma_sum

  # M-step
  N1 <- sum(gamma1)
  N2 <- sum(gamma2)

  tau1_hat <- N1 / n
  tau2_hat <- N2 / n

  mu1_hat <- colSums(gamma1 * data) / N1
  mu2_hat <- colSums(gamma2 * data) / N2

  # Mise    jour de Sigma1
  Sigma1_hat <- matrix(0, nrow = 2, ncol = 2)
  for (i in 1:n) {
    diff <- matrix(data[i, ] - mu1_hat, ncol = 1)
    Sigma1_hat <- Sigma1_hat + gamma1[i] * (diff %*% t(diff))
  }
  Sigma1_hat <- Sigma1_hat / N1

  # Mise    jour de Sigma2
  Sigma2_hat <- matrix(0, nrow = 2, ncol = 2)
  for (i in 1:n) {
    diff <- matrix(data[i, ] - mu2_hat, ncol = 1)
    Sigma2_hat <- Sigma2_hat + gamma2[i] * (diff %*% t(diff))
  }
  Sigma2_hat <- Sigma2_hat / N2
```

```
  # Log-vraisemblance
  log_likelihood_new <- sum(log(gamma_sum))

  # Convergence
  if (abs(log_likelihood_new - log_likelihood_old) < tol) {
    cat("Convergence atteinte    l'itération :", iter, "\n")
    break
  }

  log_likelihood_old <- log_likelihood_new
}

# Résultats finaux
cat("Paramètres estimés :\n")
cat("tau1 =", tau1_hat, ", tau2 =", tau2_hat, "\n")
cat("mu1 =", mu1_hat, "\n")
cat("mu2 =", mu2_hat, "\n")
cat("Sigma1 =\n")
print(Sigma1_hat)
cat("Sigma2 =\n")
print(Sigma2_hat)
```

L'algorithme EM a convergé en **10 itérations**. Voici les paramètres estimés :

*** Proportions**

$$\tau_1 = 0.61185 \quad et \quad \tau_2 = 0.38815$$

Ces proportions indiquent que 61,185 % des données appartiennent à la première composante et 38,815 % à la deuxième composante.

*** Moyennes**

$$\mu_1 = \begin{pmatrix} 2.095059 \\ 2.945951 \end{pmatrix}, \quad \mu_2 = \begin{pmatrix} 7.051303 \\ 8.029634 \end{pmatrix}$$

Les moyennes estimées sont proches des moyennes réelles définies ($\mu_1 = (2, 3)$ et $\mu_2 = (7, 8)$).

*** Première composante :**

$$\Sigma_1 = \begin{pmatrix} 0.9968700 & 0.4655238 \\ 0.4655238 & 2.0046368 \end{pmatrix}$$

*** Deuxième composante :**

$$\Sigma_2 = \begin{pmatrix} 1.3596179 & 0.2240134 \\ 0.2240134 & 1.1269195 \end{pmatrix}$$

Ces matrices sont proches des matrices réelles :

$$\Sigma_1 = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 2 \end{pmatrix}, \quad \Sigma_2 = \begin{pmatrix} 1.5 & 0.3 \\ 0.3 & 1 \end{pmatrix}.$$

*** Analyse**

Les estimations des moyennes et des matrices de covariance sont très proches des valeurs réelles. L'algorithme a convergé rapidement (**10 itérations**), ce qui est attendu pour un mélange de deux lois normales avec des paramètres bien séparés. Les proportions (τ_1 et τ_2) reflètent les proportions réelles utilisées lors de la simulation ($\tau_1 = 0.6$, $\tau_2 = 0.4$).

2.3 Exercice 3

2.3.1 Résultats de l'algorithme EM

On utilise le code suivant pour estimer les paramètres de lois de l'exercice 1

On a le code suivant qui nous permet non suelement nous permet de faire les estimations des paramètres des lois normales du numéro 1 de l'exercice 1 mais aussi de faire un graphique de l'histogramme du mélange de ces lois.

```
    library(mixtools)
# Définir les paramètres réels
set.seed(123)
n <- 1000

# Moyennes et écart-types réels
mu_real <- c(-2, 0, 2)
sigma_real <- c(0.3, 0.2, 0.2)

# Proportions réelles
tau_real <- c(1/3, 1/3, 1/3)

# Génération des données simulées
z <- sample(1:3, n, replace = TRUE, prob = tau_real)
data <- numeric(n)
data[z == 1] <- rnorm(sum(z == 1), mean = mu_real[1], sd = sigma_real
   [1])
data[z == 2] <- rnorm(sum(z == 2), mean = mu_real[2], sd = sigma_real
   [2])
data[z == 3] <- rnorm(sum(z == 3), mean = mu_real[3], sd = sigma_real
   [3])

# Appliquer l'algorithme EM pour estimer les paramètres
em_result <- normalmixEM(data, k = 3)

# Afficher les paramètres estimés
cat("Paramètres estimés :\n")
for (k in 1:3) {
  cat(sprintf("tau_%d = %.5f\n", k, em_result$lambda[k]))
  cat(sprintf("mu_%d = %.5f\n", k, em_result$mu[k]))
```

```
  cat(sprintf("sigma_%d = %.5f\n", k, em_result$sigma[k]))
}

# Visualisation
hist(data, breaks = 50, probability = TRUE, col = "lightgray",
     main = "Histogramme du mélange de trois lois normales",
     xlab = "Valeurs", ylab = "Densité")

x_vals <- seq(min(data), max(data), length.out = 200)
lines(x_vals, em_result$lambda[1] * dnorm(x_vals, mean = em_result$mu
   [1], sd = em_result$sigma[1]), col = "red", lwd = 2)
lines(x_vals, em_result$lambda[2] * dnorm(x_vals, mean = em_result$mu
   [2], sd = em_result$sigma[2]), col = "blue", lwd = 2)
lines(x_vals, em_result$lambda[3] * dnorm(x_vals, mean = em_result$mu
   [3], sd = em_result$sigma[3]), col = "green", lwd = 2)
```

2.3.2 Résultats de l'algorithme EM

L'algorithme EM a estimé les paramètres suivants pour le mélange de trois lois normales univariées :

*** Proportions estimées :**

$$\tau_1 = 0.33200, \quad \tau_2 = 0.33400, \quad \tau_3 = 0.33400$$

Ces valeurs montrent que les proportions estimées sont proches de la valeur réelle attendue de $\frac{1}{3}$.

*** Moyennes estimées :**

$$\mu_1 = -2.01197, \quad \mu_2 = 0.01188, \quad \mu_3 = 2.00320$$

Les moyennes estimées sont très proches des valeurs réelles définies ($\mu_1 = -2$, $\mu_2 = 0$, $\mu_3 = 2$).

*** Écarts-types estimés :**

$$\sigma_1 = 0.29836, \quad \sigma_2 = 0.20531, \quad \sigma_3 = 0.19577$$

Les écarts-types estimés sont très proches des valeurs réelles attendues ($\sigma_1 = 0.3$, $\sigma_2 = 0.2$, $\sigma_3 = 0.2$).

*** Analyse :**

Les estimations des proportions, moyennes et écarts-types sont globalement très précises et confirment l'efficacité de l'algorithme EM pour identifier les paramètres du mélange de gaussiennes. Les petites variations par rapport aux valeurs réelles sont dues à la variabilité statistique et aux approximations de l'algorithme.

L'algorithme EM a estimé les paramètres suivants pour le mélange de trois lois normales bidimensionnelles :

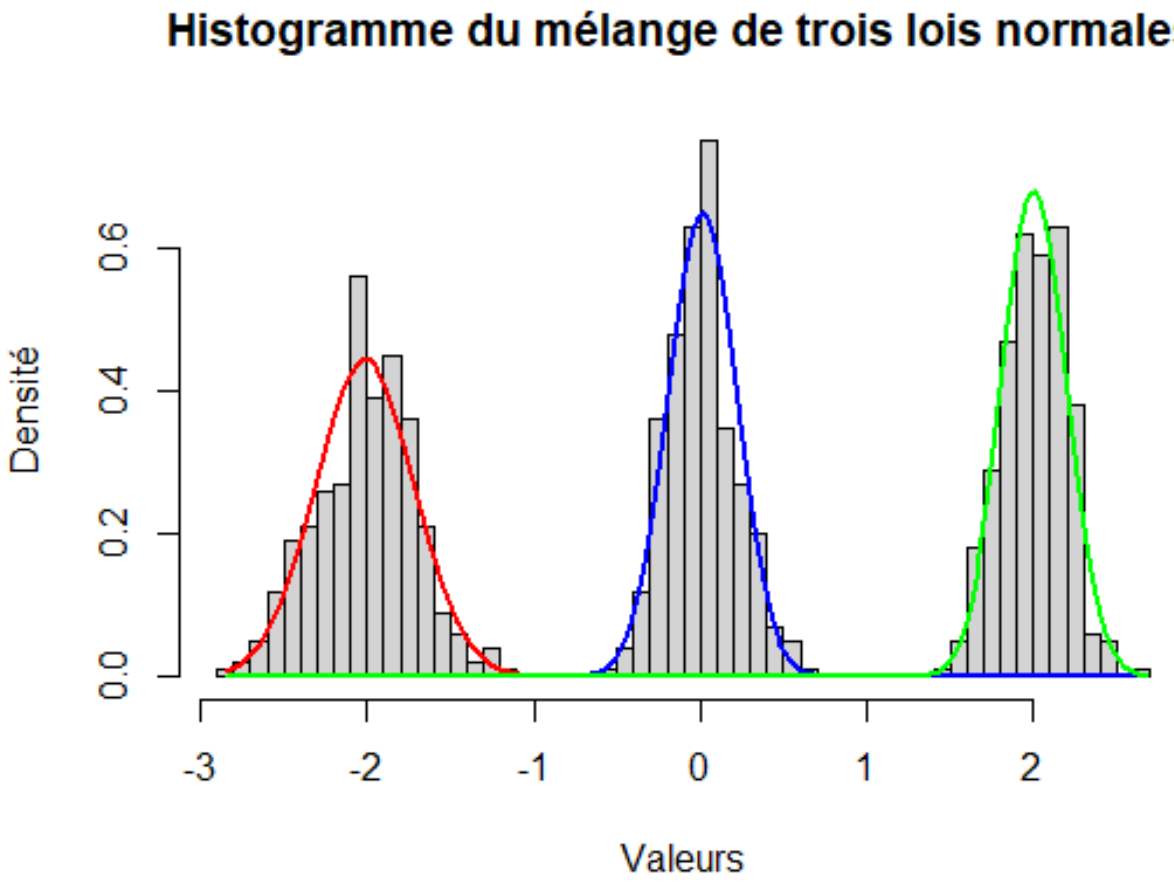


FIGURE 6 – Histogramme du mélange des lois gaussiennes

*** Proportions estimées**

$$\tau_1 = 0.32221, \quad \tau_2 = 0.17977, \quad \tau_3 = 0.49802$$

Ces valeurs indiquent que 32.221 % des données appartiennent à la première composante, 17.977 % à la deuxième et 49.802 % à la troisième.

*** Moyennes estimées**

$$\mu_1 = \begin{pmatrix} 2.12235 \\ 0.96453 \end{pmatrix}, \quad \mu_2 = \begin{pmatrix} 5.65465 \\ 3.62582 \end{pmatrix}, \quad \mu_3 = \begin{pmatrix} 10.08570 \\ 7.37152 \end{pmatrix}$$

Les moyennes estimées sont proches des moyennes réelles définies ($\mu_1 = (2, 1)$, $\mu_2 = (6, 4)$, $\mu_3 = (12, 9)$).

*** Matrices de covariance estimées**

*** Première composante :**

$$\Sigma_1 = \begin{pmatrix} 0.47752 & -0.04292 \\ -0.04292 & 0.45185 \end{pmatrix}$$

*** Deuxième composante :**

$$\Sigma_2 = \begin{pmatrix} 0.77047 & -0.22941 \\ -0.22941 & 0.95958 \end{pmatrix}$$

* Troisième composante :

$$\Sigma_3 = \begin{pmatrix} 8.61962 & 5.96750 \\ 5.96750 & 6.38867 \end{pmatrix}$$

*** Analyse :** Les estimations des moyennes et des matrices de covariance sont globalement proches des valeurs réelles. Cependant, on observe que la troisième composante (Σ_3) a une matrice de covariance plus grande, ce qui pourrait indiquer une variance plus élevée dans cette composante.

L'algorithme EM a donc permis de reconstituer de manière satisfaisante la structure sous-jacente du mélange de gaussiennes bidimensionnelles.

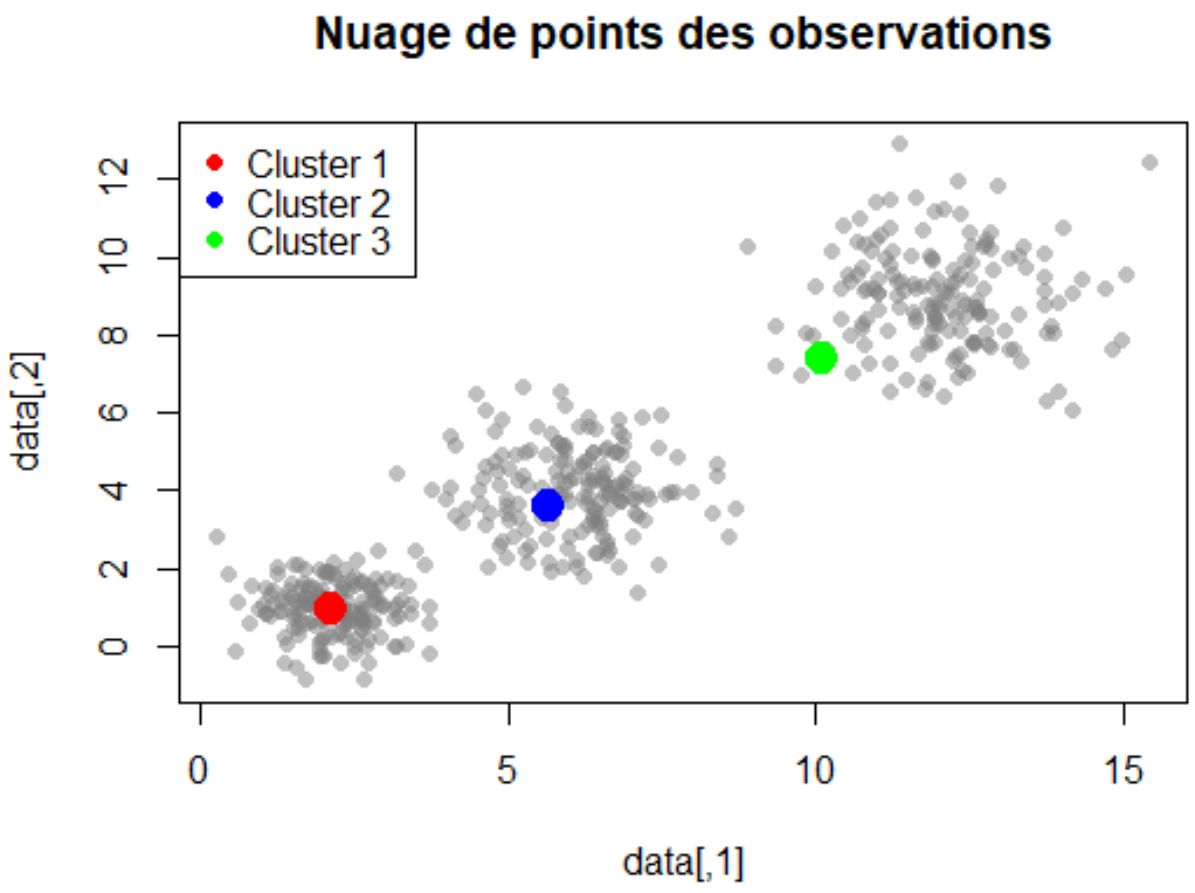


FIGURE 7 – Nuage des observations des lois gaussiennes bidimensionnelles

3 TP2

3.1 Exercice 1

Marche aléatoire simple symétrique On considère une suite $(X_n)_{n\geq 1}$ de variables aléatoires *indépendantes* telles que pour tout entier $n \geq 1$

$$\mathbb{P}(X_n = 1) = \frac{1}{2} \quad \text{et} \quad \mathbb{P}(X_n = -1) = \frac{1}{2}.$$

Pour tout entier $n \geq 1$, on définit les variables aléatoires S_n et $\overline{X}_n$ par

$$S_n = \sum_{k=1}^{n} X_k \quad \text{et} \quad \overline{X}_n = \frac{S_n}{n}.$$

3.1.1 Écriture de X_n en fonction de Y_n

On cherche à exprimer X_n en fonction d'une variable Y_n qui suit une **loi de Bernoulli de paramètre** $\frac{1}{2}$.

La loi de Bernoulli de paramètre $\frac{1}{2}$ est définie comme suit :

$$Y_n = \begin{cases} 1 & \text{avec probabilité } \frac{1}{2}, \\ 0 & \text{avec probabilité } \frac{1}{2}. \end{cases}$$

Ainsi, Y_n prend les valeurs 0 ou 1 avec une probabilité égale de $\frac{1}{2}$.

*** Relation entre X_n et Y_n**

Dans une **marche aléatoire symétrique**, on a :

$$X_n = \begin{cases} 1 & \text{avec probabilité } \frac{1}{2}, \\ -1 & \text{avec probabilité } \frac{1}{2}. \end{cases}$$

On remarque que X_n prend des valeurs dans $\{-1, 1\}$, tandis que Y_n les prend dans $\{0, 1\}$.

Une transformation simple qui convertit Y_n en X_n est :

$$X_n = 2Y_n - 1.$$

3.1.2 Simulation des valeurs de X_n pour nallant de 1 à N

```
N=100
Y<-rbinom(N,1,1/2)
X<-2*Y-1
```

*** Résultats des valeurs de X_n**

Les 100 valeurs obtenues sont représentées dans le tableau suivant :

Index	1	2	3	4	5	6	7	8	9	10
1	−1	1	−1	−1	1	1	1	1	−1	1
11	1	−1	−1	−1	1	−1	−1	−1	−1	1
21	1	−1	−1	−1	−1	1	−1	1	−1	1
31	1	−1	1	1	1	1	−1	1	−1	−1
41	1	−1	1	1	−1	1	1	−1	−1	1
51	1	1	−1	−1	1	1	1	1	1	1
61	1	−1	1	−1	1	−1	1	1	−1	−1
71	1	1	−1	−1	−1	−1	−1	1	−1	1
81	−1	1	−1	1	1	−1	1	−1	−1	−1
91	1	1	−1	1	−1	1	1	1	−1	1

3.1.3 Renvoie de la commande cumsum

```
cumsum(1:3)
cumsum(1:4)
```

La commande *cumsum(1 :3)* renvoie le vecteur $(1, 3, 6) = (1, 1+2, 1+2+3)$ et *cumsum(1 :4)* renvoie le vecteur $(1, 3, 6) = (1, 1+2, 1+2+3, 1+2+3+4)$.
En général la fonction *cumsum()* calcule la somme cumulative d'un vecteur numérique.

3.1.4 Déduction du vecteur $(S_1, ..., S_N)$

```
Sn<-cumsum(X)
print(Sn)
plot(Sn, type = "l", col = "blue", lwd = 2,
     xlab = "n", ylab = "S_n",
     main = "Marche aléatoire symétrique")
abline(h = 0, col = "red", lty = 2)
```

*** Valeurs cumulatives de S_n**

Les 100 valeurs de S_n sont représentées dans le tableau suivant :

Index	1	2	3	4	5	6	7	8	9	10
1	−1	0	−1	−2	−1	0	1	2	1	2
11	3	2	1	0	1	0	−1	−2	−3	−2
21	−1	−2	−3	−4	−5	−4	−5	−4	−5	−4
31	−3	−2	−1	−2	−1	−2	−3	−2	−3	−2
41	−1	−2	−1	0	−1	0	−1	−2	−1	0
51	−1	−2	−1	0	1	2	3	4	5	4
61	5	4	3	2	3	2	3	2	3	4
71	3	2	3	4	3	2	1	0	−1	0
81	−1	0	−1	0	−1	0	1	0	1	0
91	−1	−2	−1	0	−1	0	−1	0	1	2

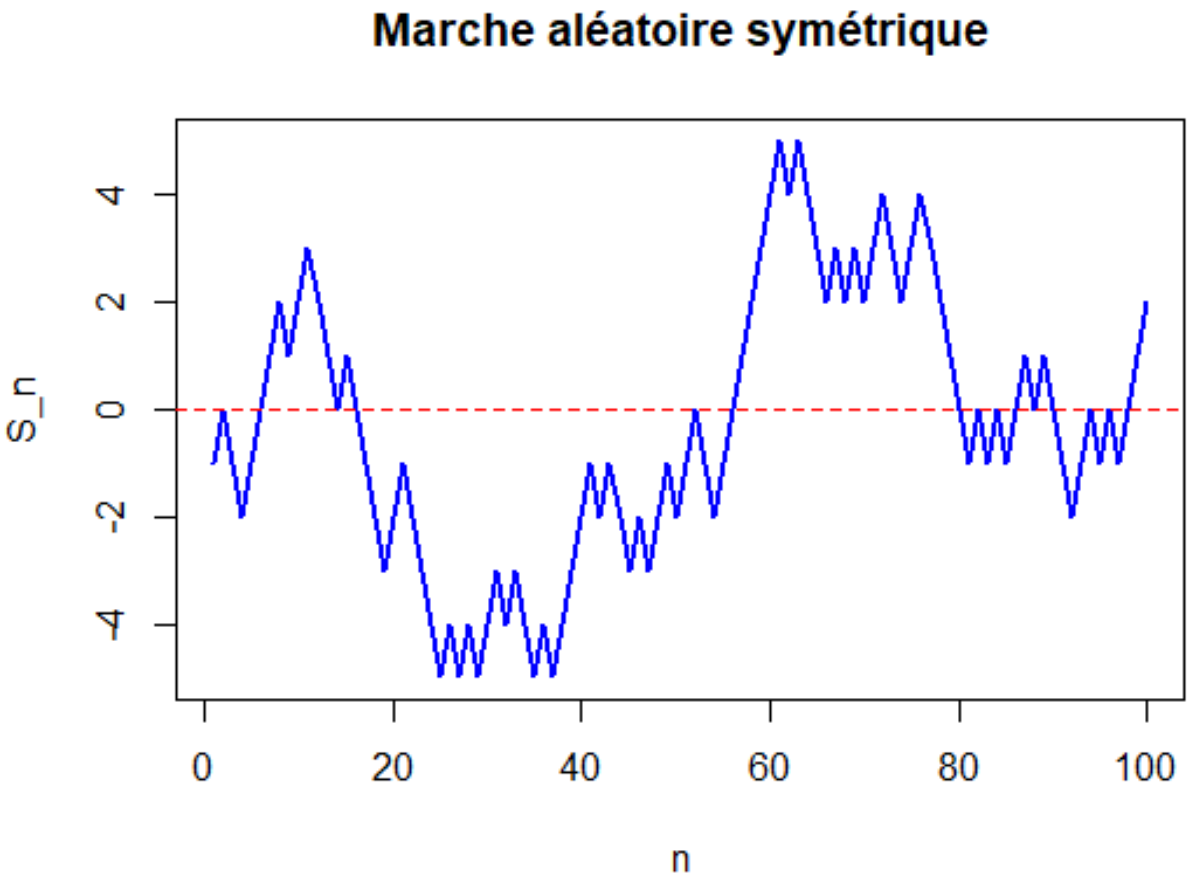


FIGURE 8 – Marche alèatoire symétrique

3.1.5 Que renvoie c(a,b)/c(c,d)

On remarque que c(1,2,4,8)/c(1,10,100,1000) renvoie la division élément par élément du premier vecteur sur le deuxième pour donner un autre vecteur autrement-dit (1/1,2/10,4/100,8/1000). En général $c(a,b)/c(c,d) = c(a/c, b/d)$.

3.1.6 Déduction de $(\overline{X}_1, ...\overline{X}_n)$ en fonction de $(S_1, ..., S_N)$

On a d'après ce qui précède :

```
Xbar<-Sn/(1:N)
print(Xbar)
```

On obtient donc le résultat suivant :

*** Valeurs de $\overline{X}_n$ (moyenne empirique)**

Les 100 valeurs de $\overline{X}_n$ sont représentées dans le tableau suivant :

Index	1	2	3	4	5	6	7	8	9	10
1	−1.000	0.000	−0.333	−0.500	−0.200	0.000	0.143	0.250	0.111	0.200
11	0.273	0.167	0.077	0.000	0.067	0.000	−0.059	−0.111	−0.158	−0.100
21	−0.048	−0.091	−0.130	−0.167	−0.200	−0.154	−0.185	−0.143	−0.172	−0.133
31	−0.097	−0.125	−0.091	−0.118	−0.143	−0.111	−0.135	−0.105	−0.077	−0.050
41	−0.024	−0.048	−0.023	−0.045	−0.067	−0.043	−0.064	−0.042	−0.020	−0.040
51	−0.020	0.000	−0.019	−0.037	−0.018	0.000	0.018	0.034	0.051	0.067
61	0.082	0.065	0.079	0.063	0.046	0.030	0.045	0.029	0.043	0.029
71	0.042	0.056	0.041	0.027	0.040	0.053	0.039	0.026	0.013	0.000
81	−0.012	0.000	−0.012	0.000	−0.012	0.000	0.011	0.000	0.011	0.000
91	−0.011	−0.022	−0.011	0.000	−0.011	0.000	−0.010	0.000	0.010	0.020

3.1.7 Montrer que $\overline{X}_n$ converge en probabilité vers 0.

La question est de savoir si Pour tout $\varepsilon > 0$

$$\overline{X}_n \xrightarrow{\mathbb{P}} 0 \quad ?$$

Ici il suffit d'utiliser Bienaymé-Tchebychev.
On sait que our tout nombre réel strictement positif α,

$$\mathbb{P}\left(|X - \mathbb{E}[X]| \geq \alpha\right) \leq \frac{\sigma^2}{\alpha^2}.$$

On peut donc écrire que

$$\mathbb{P}\left(|\overline{X}_n - \mu| \geq \varepsilon\right) \leq \frac{\mathrm{Var}(\overline{X}_n)}{\varepsilon^2}$$

On a $\mathbb{E}[\overline{X}_n] = 0$, et $\mathrm{Var}(\overline{X}_n) = \frac{1}{n}$.
Ainsi, pour tout $\varepsilon > 0$,

$$\mathbb{P}\left(|\overline{X}_n - 0| \geq \varepsilon\right) \leq \frac{1}{\varepsilon^2 n}.$$

Donc, lorsque $n \to \infty$,

$$\mathbb{P}\left(|\overline{X}_n - 0| \geq \varepsilon\right) \to 0.$$

3.1.8 Simuler et afficher plusieurs trajectoires de $\overline{X}_n$ pour n allant de 1 à $N = 5000$

```r
# Définition des paramètres
N <- 5000
nb_trajactoire <- 10

# Initialisation d'une matrice pour stocker les trajectoires
trajectoires <- matrix(NA, nrow = N, ncol = nb_trajactoire)

# Génération de plusieurs trajectoires
set.seed(123)
for (j in 1:nb_trajactoire) {
  Xn <- sample(c(-1, 1), size = N, replace = TRUE)
  Sn <- cumsum(Xn)
  Xbar <- Sn / (1:N)
  trajectoires[, j] <- Xbar
}

# Tracé des trajectoires
matplot(1:N, trajectoires, type = "l", lty = 1, col = rainbow(nb_
   trajactoire),
        xlab = "n", ylab = expression(bar(X)[n]),
        main = "Trajectoires de la moyenne empirique X n")
abline(h = 0, col = "black", lwd = 2, lty = 2)
```

On obtient donc le graphique suivant :

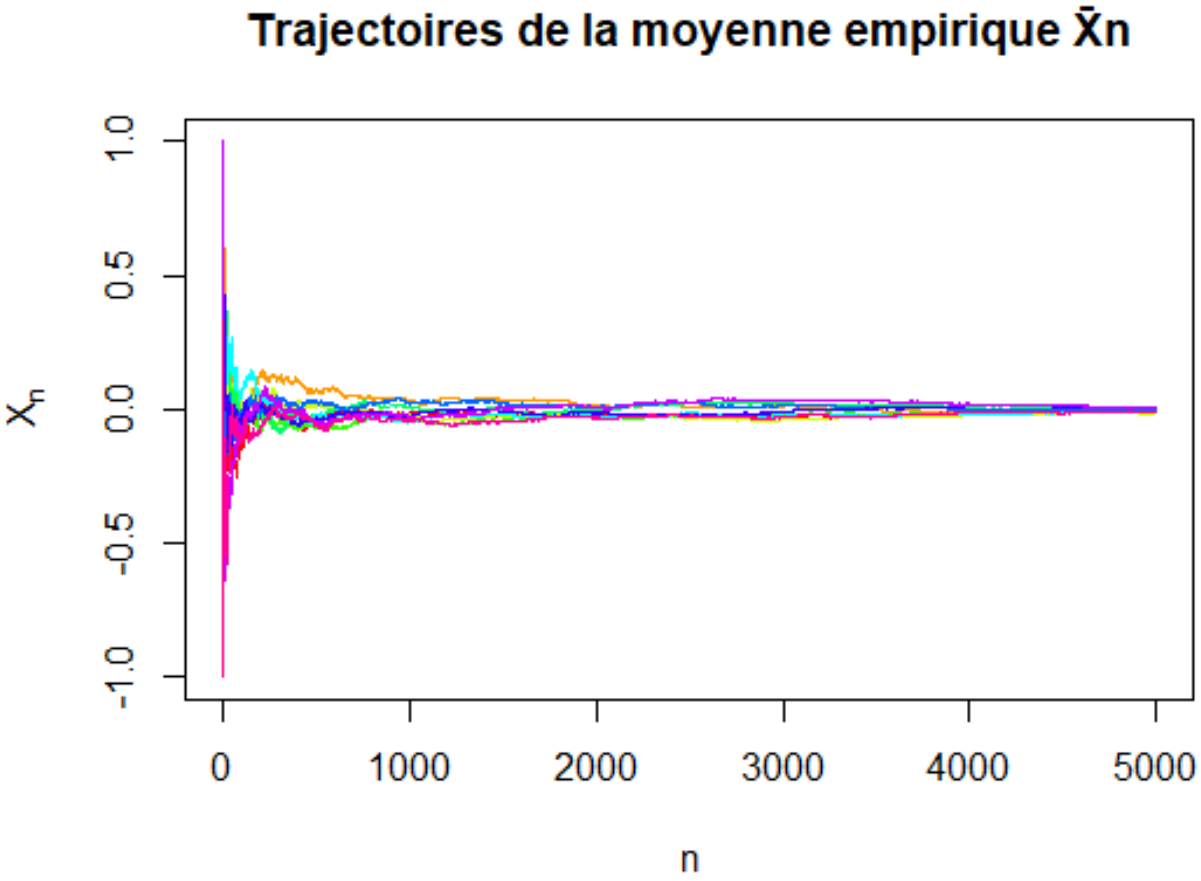


FIGURE 9 – Trajectoire de $\overline{X}_n$

3.1.9 Simulation des variables

Marche aléatoire simple biaisée À présent, on considère une suite $(X_n)_{n\geq 1}$ de variables aléatoires *indépendantes* telles que pour tout entier $n \geq 1$,

$$\mathbb{P}(X_n = 1) = p \quad \text{et} \quad \mathbb{P}(X_n = -1) = 1 - p,$$

avec p dans $]0, 1[$. Dans le cas où $p = \frac{1}{2}$, on retombe sur le cas précédent. On conserve les définitions des variables aléatoires S_n et $\overline{X}_n$.

On sait que :

$$Y_n \sim \mathcal{B}(p) \quad \text{(loi de Bernoulli de paramètre } p\text{)}$$

et la relation entre X_n et Y_n est donnée par :

$$X_n = 2Y_n - 1$$

Ici, on fait varier les valeurs de p dans $[0, 1]$. Donc pour $p = \frac{1}{3}$, on génère :

$$X_n = 2 \times \texttt{rbinom}(N, 1, P) - 1$$

3.1.10 Simuler et afficher plusieurs trajectoires de Sn pour n allant de 1 à $N = 5000$

```r
# Définition des paramètres
N <- 5000
num_traj <- 10
p_values <- c(1/2, 1/3, 2/3)

# Fixer une graine pour reproductibilité
set.seed(123)

# Création de la fenêtre graphique
par(mfrow=c(1,3))

# Génération et affichage des trajectoires pour chaque p
for (p in p_values) {
  trajectoires <- matrix(NA, nrow = N, ncol = num_traj)

  for (j in 1:num_traj) {
    Yn <- rbinom(N, 1, p)
    Xn <- 2 * Yn - 1
    Sn <- cumsum(Xn)
    trajectoires[, j] <- Sn
  }

  # Tracé des trajectoires
  matplot(1:N, trajectoires, type = "l", lty = 1, col = rainbow(num_
     traj),
          xlab = "n", ylab = expression(S[n]),
          main = paste("Trajectoires pour p =", p))
  abline(h = 0, col = "black", lwd = 2, lty = 2)
}


# Définition des paramètres
N <- 5000
num_traj <- 10
p_values <- c(1/2, 1/3, 2/3)

# Fixer une graine pour reproductibilité
set.seed(123)

# Génération et affichage des trajectoires séparément pour chaque p
for (p in p_values) {
  trajectoires <- matrix(NA, nrow = N, ncol = num_traj)
  for (j in 1:num_traj) {
    Yn <- rbinom(N, 1, p)
    Xn <- 2 * Yn - 1
    Sn <- cumsum(Xn)
    trajectoires[, j] <- Sn
  }
```

```
  # Ouvre une nouvelle fenêtre graphique
  dev.new()

  # Tracé des trajectoires
  matplot(1:N, trajectoires, type = "l", lty = 1, col = rainbow(num_
     traj),
          xlab = "n", ylab = expression(S[n]),
          main = paste("Trajectoires de Sn pour p =", p), lwd = 2)
  abline(h = 0, col = "black", lwd = 2, lty = 2)
}
```

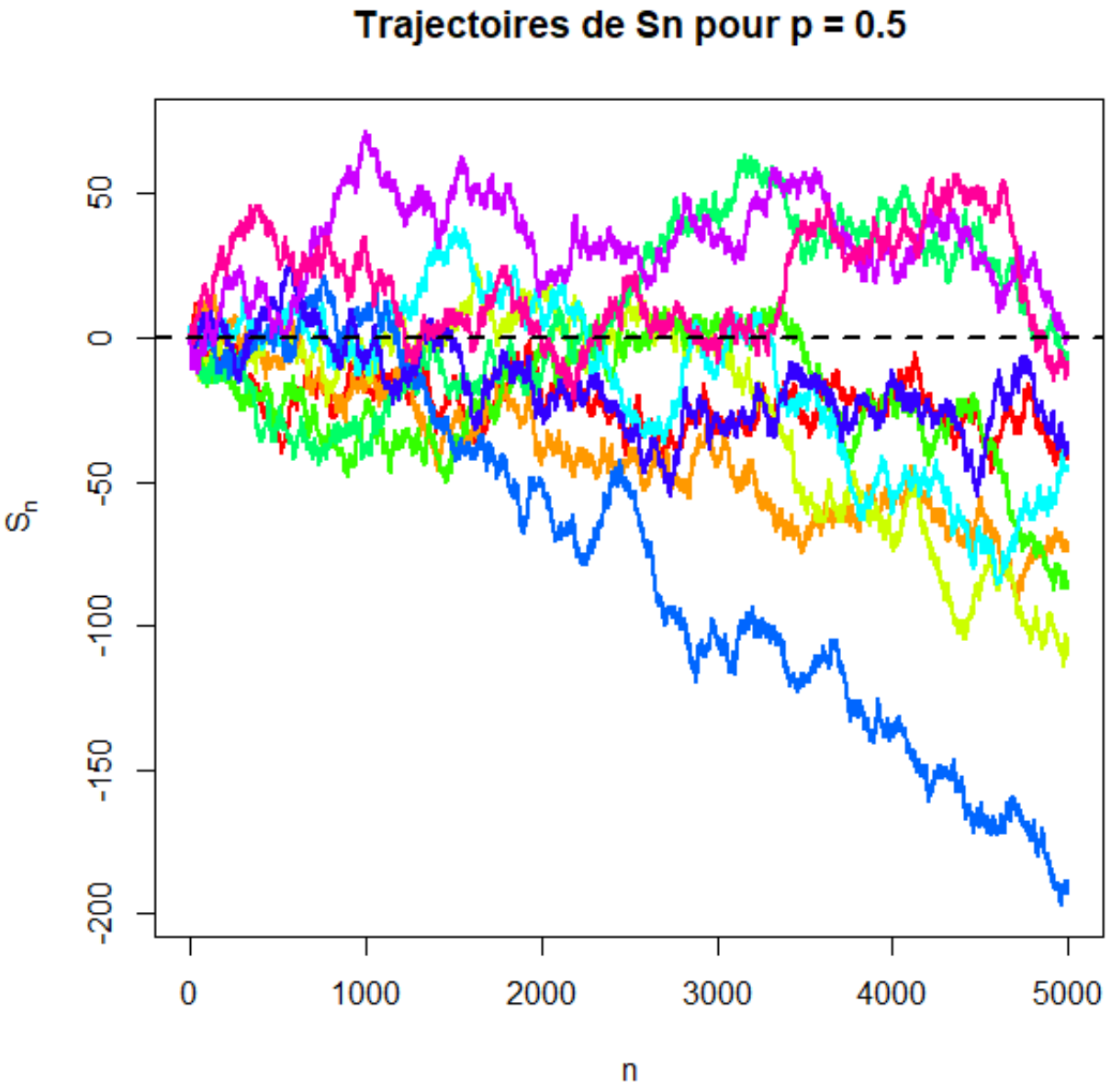


FIGURE 10 – Trajectoire de S_n

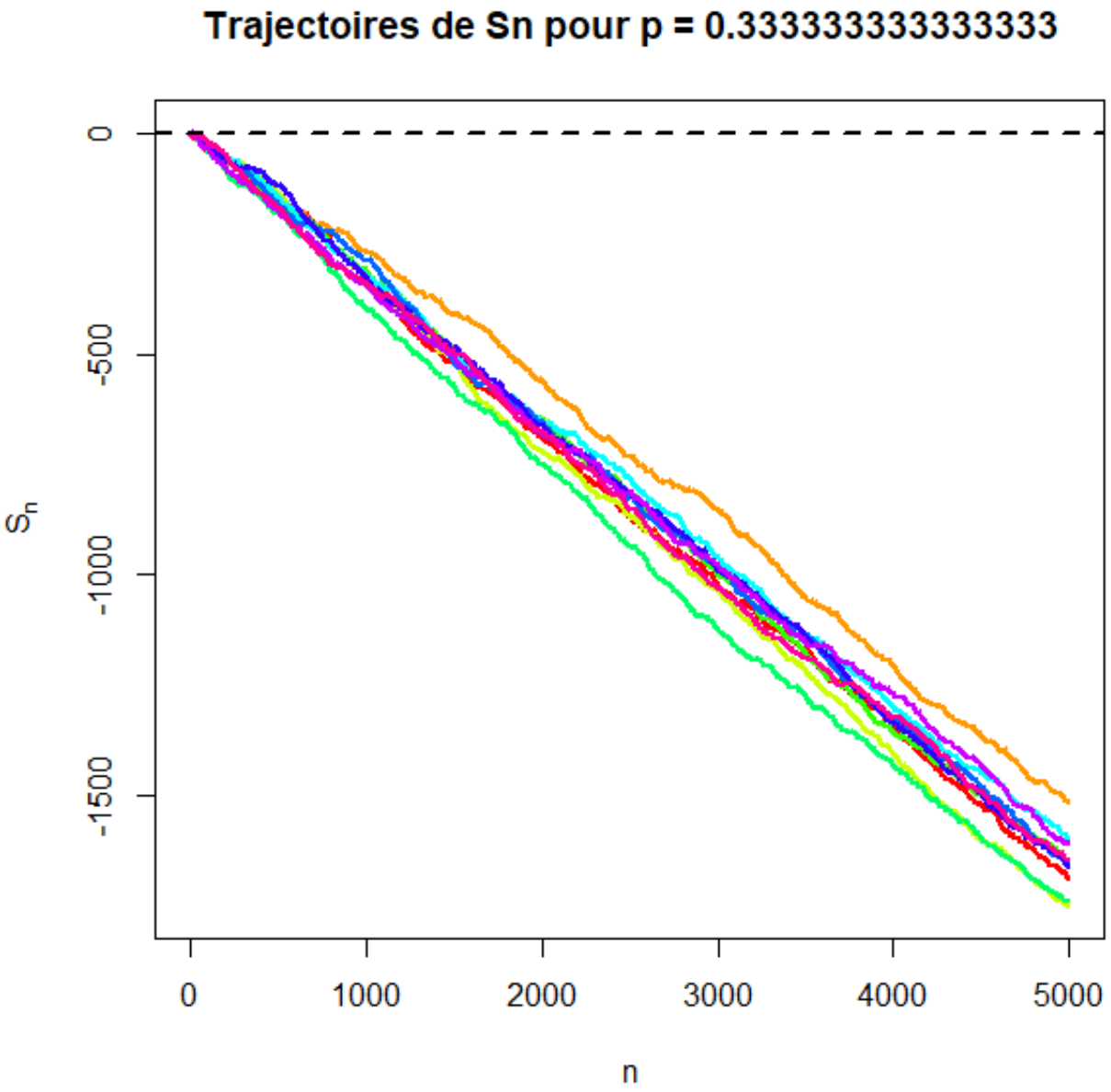


FIGURE 11 – Trajectoire de S_n

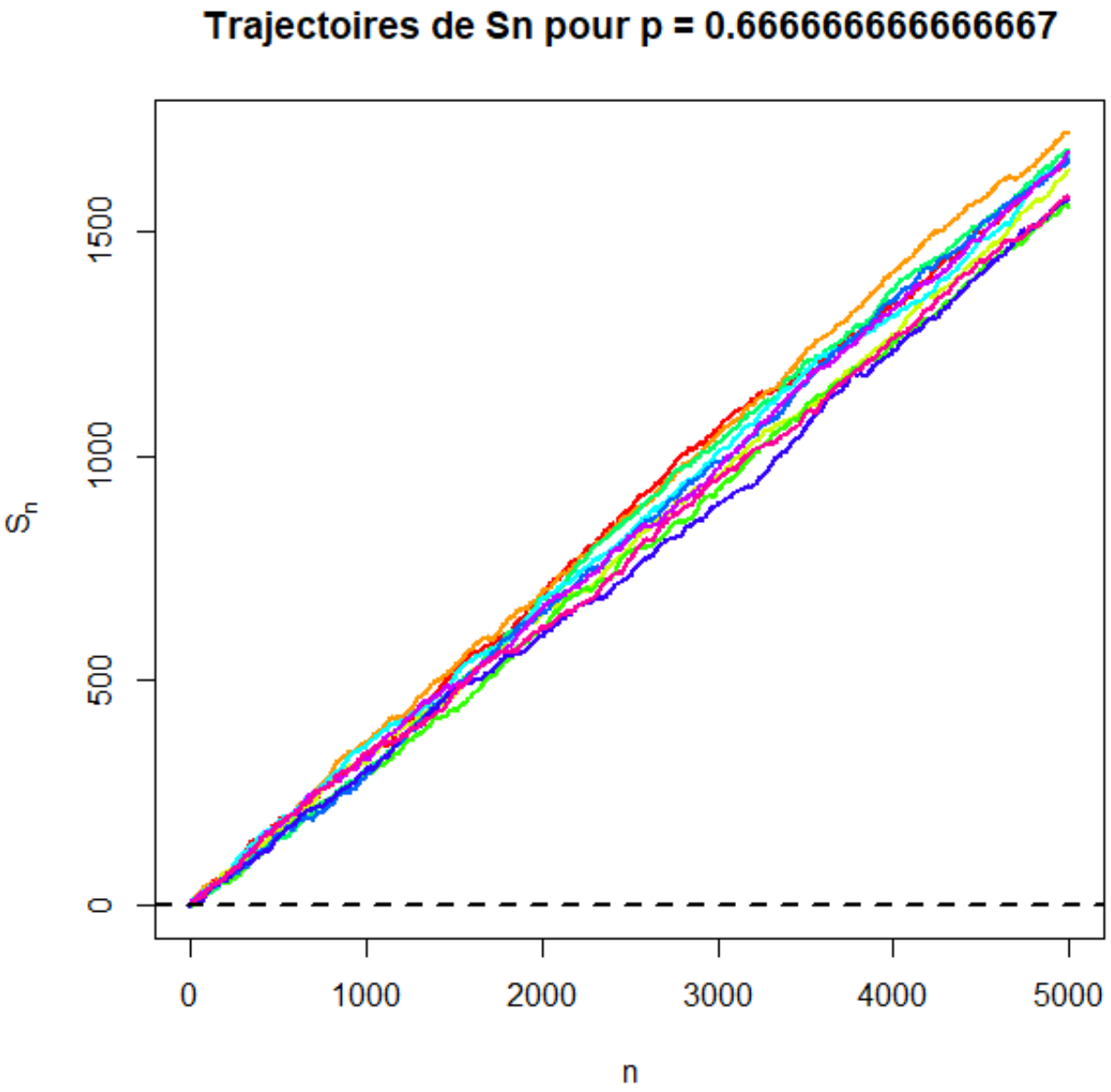


FIGURE 12 – Trajectoire de S_n

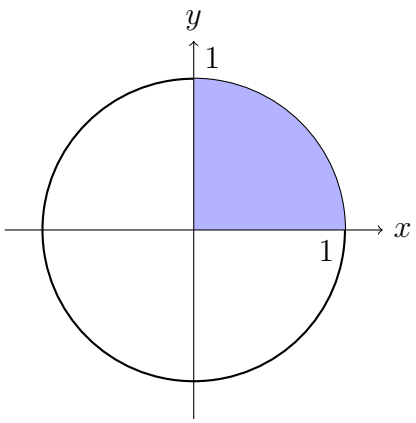
*** Interprétation des graphiques**

Les résultats montrent que :

- Pour $p = \frac{1}{2}$, la marche est **symétrique** et S_n fluctue sans direction privilégiée.
- Pour $p = \frac{1}{3}$, la marche est **biaisée négativement** et S_n diminue en moyenne.
- Pour $p = \frac{2}{3}$, la marche est **biaisée positivement** et S_n augmente en moyenne.

3.1.11 Calcul Approché de π

*** Représentation d'un cercle de rayon 1**



On veut déterminer **l'aire** de la partie en bleu ce qui revient à trouver l'aire d'un quart de cercle de rayon qui vaut $\frac{\pi}{4}$.

On peut donc écrire que :

$$A_D = \frac{\pi}{4}$$

3.1.12 Processus et Loi de S_n

On considère :

$$X_n = 1_D(X_n)$$

(a) Quelle est la loi de Y_n ?

Nous définissons la variable X_n comme suit :

$$Y_n = \begin{cases} 1, & \text{si } X_n \in D, \\ 0, & \text{si } X_n \notin D. \end{cases}$$

Ainsi, la variable aléatoire Y_n suit une loi de **Bernoulli** de paramètre $\mathbb{P}(X_n \in D)$:

$$Y_n \sim \mathcal{B}(\mathbb{P}(X_n \in D))$$

où :

$$\mathbb{P}(Y_n = 1) = \mathbb{P}(Y_n \in D) = \pi.$$

(b) Démonstration de la convergence

Nous voulons montrer que la moyenne empirique :

$$\overline{Y}_N = \frac{1}{N} \sum_{n=1}^{N} Y_n$$

converge en probabilité vers $\frac{\pi}{4}$ en utilisant l'inégalité de Tchebychev.

*** Rappel de l'inégalité de Tchebychev**

L'inégalité de Tchebychev affirme que pour toute variable aléatoire X avec variance finie et pour tout $\epsilon > 0$:

$$P(|X - E[X]| \geq \epsilon) \leq \frac{\mathrm{Var}(X)}{\epsilon^2}.$$

Nous allons appliquer cette inégalité à $\overline{Y}_N$.

*** Espérance et variance de Y_n**

Les Y_n sont des variables indicatrices de l'événement $X_1^2 + X_2^2 \leq 1$, donc :
- **Espérance** :

$$E[Y_n] = P(X_1^2 + X_2^2 \leq 1) = \frac{\pi}{4}.$$

- **Variance** : Pour une variable indicatrice, la variance est donnée par :

$$\mathrm{Var}(Y_n) = E[Y_n](1 - E[Y_n]) = \frac{\pi}{4}\left(1 - \frac{\pi}{4}\right).$$

* Variance de $\overline{Y}_N$

Puisque les Y_n sont *i.i.d.*, la variance de la moyenne empirique est :

$$\mathrm{Var}(\overline{Y}_N) = \frac{\mathrm{Var}(Y_n)}{N} = \frac{1}{N}\left(\frac{\pi}{4}\left(1 - \frac{\pi}{4}\right)\right).$$

*** Application de l'inégalité de Tchebychev**

On applique maintenant l'inégalité de Tchebychev :

$$P(|\overline{Y}_N - \frac{\pi}{4}| \geq \epsilon) \leq \frac{\mathrm{Var}(\overline{Y}_N)}{\epsilon^2}.$$

En remplaçant par la variance calculée :

$$P(|\overline{Y}_N - \frac{\pi}{4}| \geq \epsilon) \leq \frac{\frac{1}{N}\left(\frac{\pi}{4}\left(1-\frac{\pi}{4}\right)\right)}{\epsilon^2}.$$

Ce qui donne :

$$P(|\overline{Y}_N - \frac{\pi}{4}| \geq \epsilon) \leq \frac{\frac{\pi}{4}\left(1-\frac{\pi}{4}\right)}{N\epsilon^2}.$$

*** Conclusion : Convergence en probabilité**

On observe que lorsque $N \to \infty$, le côté droit de l'inégalité tend vers 0 pour tout $\epsilon > 0$. Cela signifie que :

$$P(|\overline{Y}_N - \frac{\pi}{4}| \geq \epsilon) \to 0 \quad \text{quand } N \to \infty.$$

Ce qui correspond à la définition de la **convergence en probabilité** vers $\pi/4$:

$$\overline{Y}_N \xrightarrow{\mathbb{P}} \frac{\pi}{4}.$$

(c) Approximation de π

```
# Définir le nombre de points simulés
N <- 10000

# Générer des points (X, Y) uniformément dans [0,1]    [0,1]
X <- runif(N, 0, 1)
Y <- runif(N, 0, 1)

# Vérifier si les points sont dans le quart de disque (X^2 + Y^2
   1)
inside_circle <- (X^2 + Y^2 <= 1)

# Calculer la proportion des points     l int érieur du quart de
   disque
Y_bar <- mean(inside_circle)

# Estimer
pi_estime <- 4 * Y_bar

# Afficher le résultat
print(pi_estime)
```

On obtient une approximation de π qui est **3.1444**

3.1.13 Détermination de y en fonction de x

Nous considérons un point M de coordonnées (x, y) appartenant au quart de disque C. L'équation du cercle unité est :

$$x^2 + y^2 = 1$$

En isolant y, on obtient :

$$y = \pm\sqrt{1 - x^2}$$

Comme nous nous plaçons dans **le quart supérieur** du disque, nous avons :

$$y = \sqrt{1 - x^2}, \quad \text{avec } y \in [0, 1].$$

3.1.14 a) Calcul de $E[Y_n]$

$$E[Y_n] = E[\varphi(X_n)]$$

$$= \int_0^1 \varphi(x) f(x)\, dx = \int_0^1 \sqrt{1 - x^2}\, dx$$

En utilisant une intégration bien connue :

$$\int_0^1 \sqrt{1 - x^2}\, dx = \frac{\pi}{4}$$

Ainsi :

$$E[Y_n] = \frac{\pi}{4}$$

et la variance est donnée par :

$$\mathrm{Var}(Y_n) = E[Y_n^2] - (E[Y_n])^2$$

b) Calcul de $E[Y_n^2]$

$$E[Y_n^2] = E[\ell^2(X_n)]$$

$$= \int_0^1 \ell^2(x) f(x)\, dx$$

$$= \int_0^1 (1 - x^2)\, dx$$

Calcul de l'intégrale :

$$\int_0^1 (1 - x^2)\, dx = \left[x - \frac{x^3}{3}\right]_0^1$$

$$= (1 - \frac{1}{3}) - (0 - 0) = \frac{2}{3}$$

Ainsi :

$$E[Y_n^2] = \frac{2}{3}$$

Donc la variance est :

$$\mathrm{Var}(Y_n) = \frac{2}{3} - \left(\frac{\pi}{4}\right)^2$$

c) Convergence en probabilité
On a

$$P(|\overline{Y}_N - \frac{\pi}{4}| \geq \epsilon) \leq \frac{\mathrm{Var}(\overline{Y}_N)}{\epsilon^2}.$$

Ce qui revient à écrire

$$P(|\overline{Y}_N - \frac{\pi}{4}| \geq \epsilon) \leq \frac{\frac{2}{3} - \left(\frac{\pi}{4}\right)}{N\epsilon^2}.$$

*** Conclusion : Convergence en probabilité**

On observe que lorsque $N \to \infty$, le côté droit de l'inégalité tend vers 0 pour tout $\epsilon > 0$. Cela signifie que :

$$P(|\overline{Y}_N - \frac{\pi}{4}| \geq \epsilon) \to 0 \quad \text{quand } N \to \infty.$$

Ce qui correspond à la définition de la **convergence en probabilité** vers $\pi/4$

d)Approximation de π

```
# Définir le nombre de points aléatoires
N <- 10000

# Générer N valeurs aléatoires X uniformément dans [0,1]
X <- runif(N, 0, 1)

# Calculer Y = sqrt(1 - X^2)
Y <- sqrt(1 - X^2)

# Calculer la moyenne empirique de Y
Y_bar <- mean(Y)

# Estimation de
pi_estime <- 4 * Y_bar

# Afficher le résultat
print(pi_estime)
```

On obtient l'estimation de π qui est **3.125898**.

4 Conclusion

En définitive, ce projet m'a permis de comprendre les sujets traités dans cette Ue me donnant ainsi une capacité à appliquer ces connaissances dans d'autres domaines si le besoin se présente. Convaincu que ce master m'a appris énormément des choses me permettant de me lancer dans les directions qui me seront présentées comme opportunités dans un futur proche ou lointain