Exercices (Fibonacci+Monte Carlo)

Exercice 1: Séquence de Fibonacci

Dans cet exercice, vous devez implémenter une version modifiée de la séquence de Fibonacci, avec les règles suivantes :
1-La séquence commence avec deux termes initiaux (par exemple : f0 et f1) où ils sont inférieurs à n.
2-Chaque terme suivant est la somme des deux termes précédents, mais modulo n i.e.,

$$f[i] = (f[i-1] + f[i-2]) \bmod n.$$

3-La longueur totale de la séquence est n.

Exercice 2 : Estimation de l'aire d'un cercle avec la méthode de Monte Carlo

Dans cet exercice, vous écrirez un programme R pour estimer l'aire d'un cercle et la valeur de π (pi) en utilisant la méthode de Monte Carlo. Votre programme devra également afficher un graphique des points et du cercle.

Étapes à suivre :

1. Permettez à l'utilisateur d'entrer le nombre de points (n) et le rayon du cercle (r).
2. Calculez :
    1. L'aire estimée du cercle.
    2. La valeur estimée de π.
3. Affichez les résultats calculés.
4. Tracez un graphique :

    1. Les points à l'intérieur du cercle doivent être en **vert**.
    2. Les points à l'extérieur du cercle doivent être en **rouge**.
    3. Le cercle doit être dessiné en **bleu**.

5. Ajoutez une légende expliquant les éléments graphiques.

Exercice 3 : Estimation d'une aire sous une courbe avec Monte Carlo

1. Écrivez un programme en R pour estimer l'aire sous la courbe de $f(x)=-x^2+4$ sur l'intervalle $x \in [-2,2]$ en utilisant la méthode Monte Carlo.
2. Affichez le résultat et tracez un graphique montrant la courbe et les points générés.

Solution Ex1:

```
fibonacci_mod <- function(f0, f1, n) {
if (n < 2) {
stop("n doit être au moins 2.") # Assurez-vous que n >= 2
}

if (f0 >= n || f1 >= n) {
stop("f0 et f1 doivent être inférieurs à n.") # Assurez-vous que f0 < n et f1 < n
}

if (n == 2) {
return(c(f0, f1)) # Renvoie directement [f0, f1] si n = 2
}

# Allocation de mémoire efficace
fib_additional <- numeric(n - 2) # Stocker uniquement les termes supplémentaires

# Calculer les termes supplémentaires
fib_additional[1] <- f0
fib_additional[2] <- f1
for (i in 3:n ) {
fib_additional[i] <- (fib_additional[i - 1] + fib_additional[i - 2]) %% n
}

# Renvoie la séquence complète
return(fib_additional)
}

# Exemple d'utilisation :
f0 <- 0
f1 <- 2
n <- 10
print(fibonacci_mod(f0, f1, n))
#ou
result <- fibonacci_mod(f0, f1, n)

# Affichage du résultat avec un message clair
cat("Les pseudo-nombres du générateur de Fibonacci sont :", result, "\n")
```

Solution EX2:

```
# Paramètres
n <- 500  # Nombre de points
r <- 2    # Rayon
# ou nous utilisons
 n <- as.integer(readline(prompt = "Entrer le nombre de points (n) : "))
 r <- as.numeric(readline(prompt = "Entrer le rayon (r) : "))

start_time <- Sys.time()    # Début du chronométrage

# Générer des points aléatoires
x <- runif(n, -r, r)  # Points aléatoires dans l'intervalle [-r, r]
y <- runif(n, -r, r)  # Points aléatoires dans l'intervalle [-r, r]

inside_count <- 0  # Initialiser le compteur

for (i in 1:n) {
  if (x[i]^2 + y[i]^2 <= r^2) {  # Vérifier si le point est à l'intérieur du cercle
    inside_count <- inside_count + 1
  }
}

# Estimation de l'aire du cercle
area_estimated <- (inside_count / n) * (2 * r)^2  # Estimation de l'aire par Monte Carlo

pi_estimated <- 4 * (inside_count / n) # Estimation de pi

area_real <- pi * r^2  # Aire réelle du cercle

end_time <- Sys.time() # Fin du chronométrage
time_taken <- end_time - start_time

# Affichage des résultats
cat(sprintf("Aire estimée du cercle : %.4f\n", area_estimated))
cat(sprintf("Aire réelle du cercle : %.4f\n", area_real))
cat(sprintf("Pi estimé : %.4f\n", pi_estimated))
cat(sprintf("Pi réel : %.4f\n", pi))
cat(sprintf("Temps écoulé : %.4f secondes\n", as.numeric(time_taken)))

# Tracé
plot( x, y, col = ifelse((x^2 + y^2 <= r^2), "green", "red"), pch = 16,  xlab = "x", ylab = "y", main =
"Simulation Monte Carlo : Cercle dans un carré")
symbols(0, 0, circles = r, add = TRUE, inches = FALSE, lwd = 2, col = "blue")  # Limite du cercle
legend( "topright", legend = c("Points à l'intérieur", "Points à l'extérieur", "Limite du cercle"),
```

col = c("green", "red", "blue"), pch = c(16, 16, NA), lty = c(NA, NA, 1), bty = "n")
#inches = FALSE: Cela indique que les dimensions du cercle sont spécifiées dans les unités du système de coordonnées actuel (par exemple, les unités des axes), et non en pouces sur l'écran.

#lwd = 2: Cela signifie "Line Width" (épaisseur de la ligne). Une valeur de 2 rend la ligne du cercle deux fois plus épaisse que la valeur par défaut. Une valeur plus élevée augmente l'épaisseur, et une valeur plus faible la diminue.

---

pch = c(16, 16, NA), lty = c(NA, NA, 1), bty = "n"

Ces paramètres sont utilisés dans la fonction legend :

pch = c(16, 16, NA) /////////

-*cex= détermine la taille des points.

-* pch désigne le type de symbole utilisé pour représenter les points dans la légende.

-* La valeur 16 correspond à un cercle plein (par défaut utilisé pour les points).

-* La valeur NA signifie qu'aucun symbole ne sera affiché à cet endroit (par exemple, pour une ligne ou une zone sans symbole).

lty = c(NA, NA, 1)

-*lty (Line Type) spécifie le style de la ligne.

-*Les deux premières valeurs NA indiquent qu'aucune ligne ne sera tracée pour les deux premiers éléments de la légende (correspondant aux points).

-*La valeur 1 correspond à une ligne pleine (solide) pour le dernier élément de la légende (ici, la bordure du cercle).

-*lty = 2 : Ligne en tirets (---).////lty = 3 : Ligne pointillée (····).
lty = 4 : Ligne alternant des tirets et des points (- · - ·). …etc

bty = "n"

-*bty (Box Type) contrôle la boîte entourant la légende.

-*La valeur "n" signifie "no box" : aucune bordure ne sera tracée autour de la légende.

-*Les autres options incluent "o" (par défaut, une boîte rectangulaire) et d'autres variantes (par exemple, "l" pour une bordure sur un côté).

2 ème Méthode pour exercice de cercle

```
Monte_cercle <-function (n,r){

# Générer des points aléatoires
x <- runif(n, -r, r)  # Points aléatoires dans l'intervalle [-r, r]
y <- runif(n, -r, r)  # Points aléatoires dans l'intervalle [-r, r]

inside_count <- 0  # Initialiser le compteur

for (i in 1:n) {
  if (x[i]^2 + y[i]^2 <= r^2) {  # Vérifier si le point est à l'intérieur du cercle
    inside_count <- inside_count + 1
  }
}

# Estimation de l'aire du cercle
area_estimated <- (inside_count / n) * (2 * r)^2  # Estimation de l'aire par Monte Carlo

pi_estimated <- 4 * (inside_count / n) # Estimation de pi

area_real <- pi * r^2  # Aire réelle du cercle

# Affichage des résultats
cat(sprintf("Aire estimée du cercle : %.4f\n", area_estimated))
cat(sprintf("Aire réelle du cercle : %.4f\n", area_real))
cat(sprintf("Pi estimé : %.4f\n", pi_estimated))
cat(sprintf("Pi réel : %.4f\n", pi))

# Tracé
plot( x, y, col = ifelse((x^2 + y^2 <= r^2), "green", "red"), pch = 16,  xlab = "x", ylab = "y", main =
"Simulation Monte Carlo : Cercle dans un carré")
symbols(0, 0, circles = r, add = TRUE, inches = FALSE, lwd = 2, col = "blue")  # Limite du cercle
legend( "topright", legend = c("Points à l'intérieur", "Points à l'extérieur", "Limite du cercle"),
  col = c("green", "red", "blue"), pch = c(16, 16, NA), lty = c(NA, NA, 1), bty = "n")
return(c(pi_estimated,area_estimated))
}

Monte_cercle(1000,2)
```

Solution exercice 3

```
# Fonction pour estimer l'aire sous la courbe en utilisant la méthode Monte Carlo
S <- function(N) {
 n <- 0  # Compteur pour les points sous la courbe

 # Vecteurs pour stocker les coordonnées des points aléatoires
 x_pts <- runif(N, min = -2, max = 2) # Générer un x aléatoire dans [-2, 2]
 y_pts <- runif(N, min = 0, max = 4) # Générer un y aléatoire dans [0, 4]

 for (i in 1:N) {

   # Vérifier si le point est sous ou sur la courbe
  if (y_pts[i] <= (-x_pts[i]^2 + 4)) {  # Condition pour la fonction f(x) = -x^2 + 4
    n <- n + 1  # Incrémenter le compteur si le point est sous la courbe
  }
 }

 # Calculer l'aire estimée
 S <- 16 * n / N  # L'aire totale du rectangle est 16 (largeur 4 x hauteur 4)

 # Affichage de l'aire estimée
 cat("La surface estimée est : ", S, "\n")

 # Tracé du graphique
 plot(x_pts, y_pts, pch = 16,
     col = ifelse(y_pts <= (-x_pts^2 + 4), "green", "blue"), cex = 0.6,
     xlab = "x", ylab = "y", main = "Estimation Monte Carlo de l'aire sous la courbe",
     xlim = c(-2, 2), ylim = c(0, 4))
 curve(-x^2 + 4, from = -2, to = 2, col = "red", lwd = 1, add = TRUE)
 legend("topright", legend = c("Points sous la courbe", "Points hors de la courbe", "Courbe f(x)"),
       col = c("green", "blue", "red"), pch = c(16, 16, NA), lty = c(NA, NA, 1), bty = "n")

 return(S)  # Retourner l'aire estimée
}
S(10000)
#ou
# Utilisation de la fonction
#N <- 10000  # Nombre de points aléatoires
#result <- S(N)
#cat("Aire estimée :", result, "\n")
```