

Mini-projet : Estimation d'aires par la methode de Monte Carlo

1 Méthode de Monte Carlo pour approximer π

1.1 Estimation de π

1.1.1 Estimation de $\frac{\pi}{4}$

L'aire totale du disque de rayon $R = 1$ est :

$$\text{Aire du disque} = \pi R^2 = \pi$$

L'aire du quart de disque est donc :

$$\sigma = \frac{\pi R^2}{4} = \frac{\pi}{4}$$

L'aire du carré unitaire qui contient ce quart de disque est :

$$S = R^2 = 1$$

La probabilité qu'un point M appartienne à ce quart de disque est le rapport de l'aire du quart de disque à l'aire du carré :

$$P = \frac{\text{Aire du quart de disque}}{\text{Aire du carré}} = \frac{\sigma}{S} = \frac{\frac{\pi R^2}{4}}{R^2} = \frac{\frac{\pi}{4}}{1} \approx \frac{\pi}{4}$$

1.1.2 Application avec R

Premièrement on veut déterminer si un point de coordonnées (x, y) appartient à un cercle de centre $(0, 0)$ et de rayon 1.

Un point (x, y) appartient au cercle si la distance entre ce point et le centre du cercle est inférieure ou égale au rayon R . Cette distance est calculée à l'aide de la formule de la distance :

$$\sqrt{(x - a)^2 + (y - b)^2} \leq R$$

Dans notre cas, $a = 0$ et $b = 0$, nous mettons au carré les deux côtés pour éviter de calculer la racine carrée :

$$x^2 + y^2 \leq R^2$$

Finalement, un point (x, y) appartient au cercle si :

$$x^2 + y^2 \leq 1$$

Voici une fonction R qui permet d'estimer π en utilisant la méthode de Monte Carlo :

```
mc.pi <- function(n) {  
  # Génère n coordonnées x et y aléatoires uniformément dans [0,  
    1]  
  x <- runif(n, 0, 1)  
  y <- runif(n, 0, 1)  
  
  # Compte le nombre de points à l'intérieur du quart de cercle de  
    rayon 1  
  # et estime Pi à partir de la proportion  
  estimation_pi <- 4 * (sum(x^2 + y^2 <= 1) / n)  
  
  return(estimation_pi)  
}
```

1.2 Simulations avec $n=10*j$, pour $j=1:p$

```
t <- 50  # Nombre d'estimations par colonne  
p <- 7   # Nombre de colonnes  
  
# Initialiser la matrice PIE  
PIE <- matrix(0, nrow = t, ncol = p)  
  
# Remplir la matrice avec des estimations de pi  
for (j in 1:p) {  
  n <- 10^j  # Calculer n comme 10 à la puissance j  
  for (i in 1:t) {  
    PIE[i, j] <- mc.pi(n)  # Estimer pi et remplir la matrice  
  }  
}
```