Approximer l'aire sous une courbes par des méthodes Monte Carlo Black and white, simple et en utilisant une loi béta

Ragousandirane RADJASANDIRANE

20/09/2021

```
require(MASS)
```

```
## Loading required package: MASS
```

Fonction à étudier g(x)

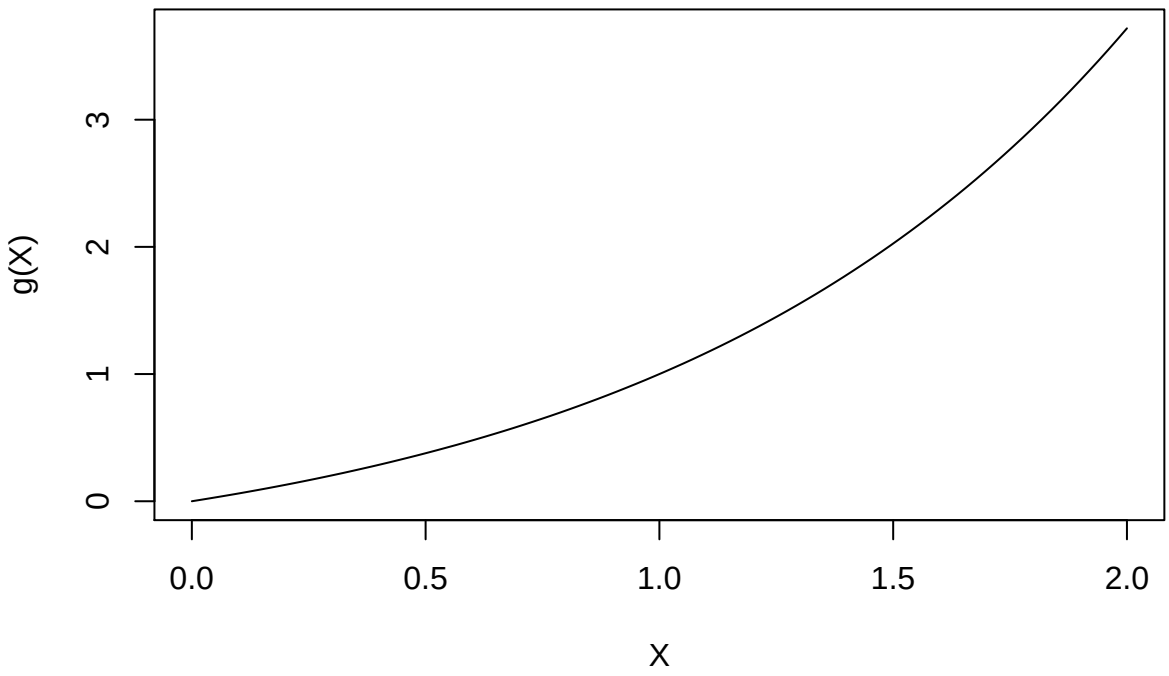
On réimplemente la fonction g de l'énoncé.

```
g <- function(x){
  return ((exp(x) - 1 )/(exp(1) - 1))
}
```

Visualisation de la fonction g

On choisit des valeurs de X allant de 0 à 2 pour visualiser la courbe de la fonction g.

```
X <- seq(from = 0, to = 2, len = 100)
plot(X,g(X), type = "l")
```



Aire sous la courbe selon l'énoncé et la fonction integrate de R

On récupère les solutions optimales pour les comparer à nos solutions calculées.

solution correspond à la solution donnée par l'énoncé.

solution_R correspond à la solution donnée par la fonction *integrate* de R qui permet de calculer l'aire sous une courbe.

```
solution <- (exp(2) - 3) / (exp(1) - 1)
solution_R <- integrate(g, lower = 0, upper = 2)
print(solution)
```

```
## [1] 2.554328
```

```
print(solution_R)
```

```
## 2.554328 with absolute error < 2.8e-14
```

MC black and white

On implemente la fonction qui va estimer l'aire sous la courbe par la méthode de MC BW.

On tire aléatoirement des points (V,U) et on détermine si ces points sont sous la courbe en vérifiant que V <= g(U).

On compte le nombre de succès (ns) qui revient à compter le nombre de points qui sont effectivement sous la courbe.

On peut, avec ce nombre de succès, approximer l'aire sous la courbe par rapport à l'aire du carré (formé à l'aide du majorant de la fonction et l'abscisse de ce majorant (ici le majorant est g(2))) en utilisant la formule : **I <- m(b-a).(ns/n)**

```
estim_I <- function(n) {
  b <- 2
  a <- 0
  m <- g(2)
  v <- runif(n,min = a, max = m)
  u <- runif(n,min = a, max = b)
  points <- sum(ifelse(v <= g(u),1,0))
  I <- m*(b-a)*points/n
  return(I)
}
```

On obtient une valeur proche de la valeur optimale (= 2.5543284) avec 100000 points

```
estim_I(100000)
```

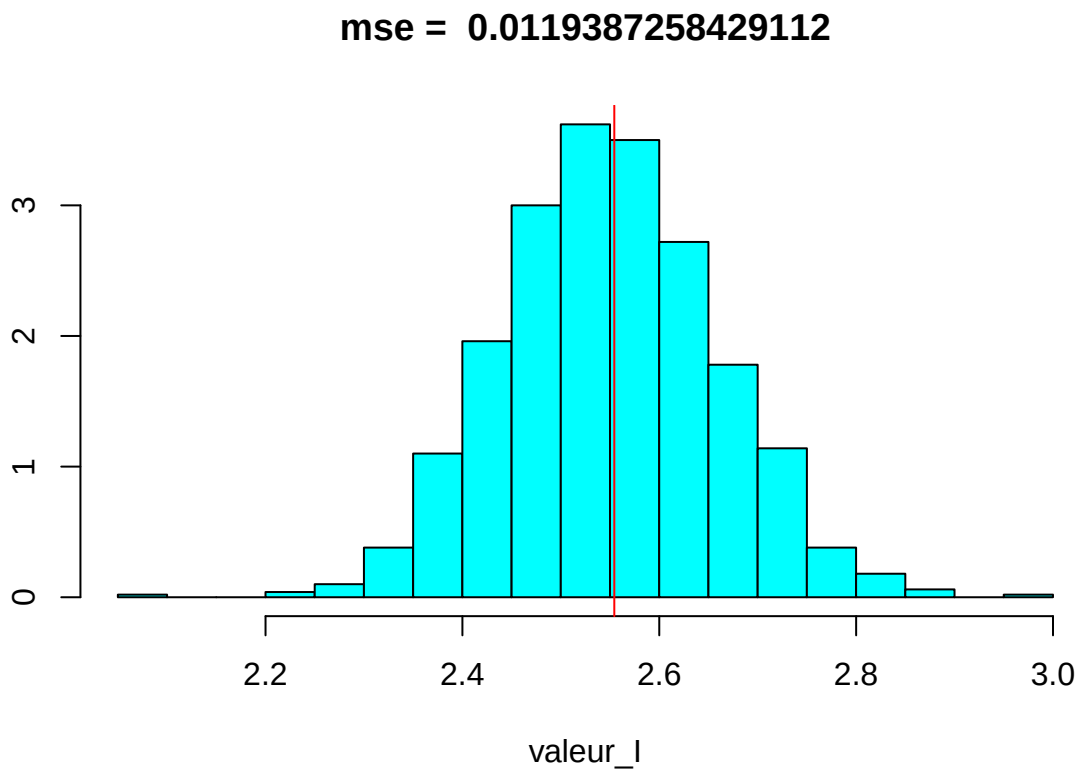
```
## [1] 2.553567
```

MSE de l'estimateur MC BW

```
mse_I <- function(n,solution_I,nrep){
  valeur_I <- c()
  for (run in 1:nrep) {
  valeur_I <- c(valeur_I, estim_I(n))
  }
  MSE <- mean((valeur_I - solution_I)**2)
  truehist(valeur_I)
  abline(v = solution_I, col = "red")
  title(main = paste("mse = ",MSE))
}
```

On obtient un MSE d'environ 0.012 et une moyenne des solutions calculées qui semblent être proches de la solution

```
mse_I(1000,solution,1000)
```



Echantillonnage simple

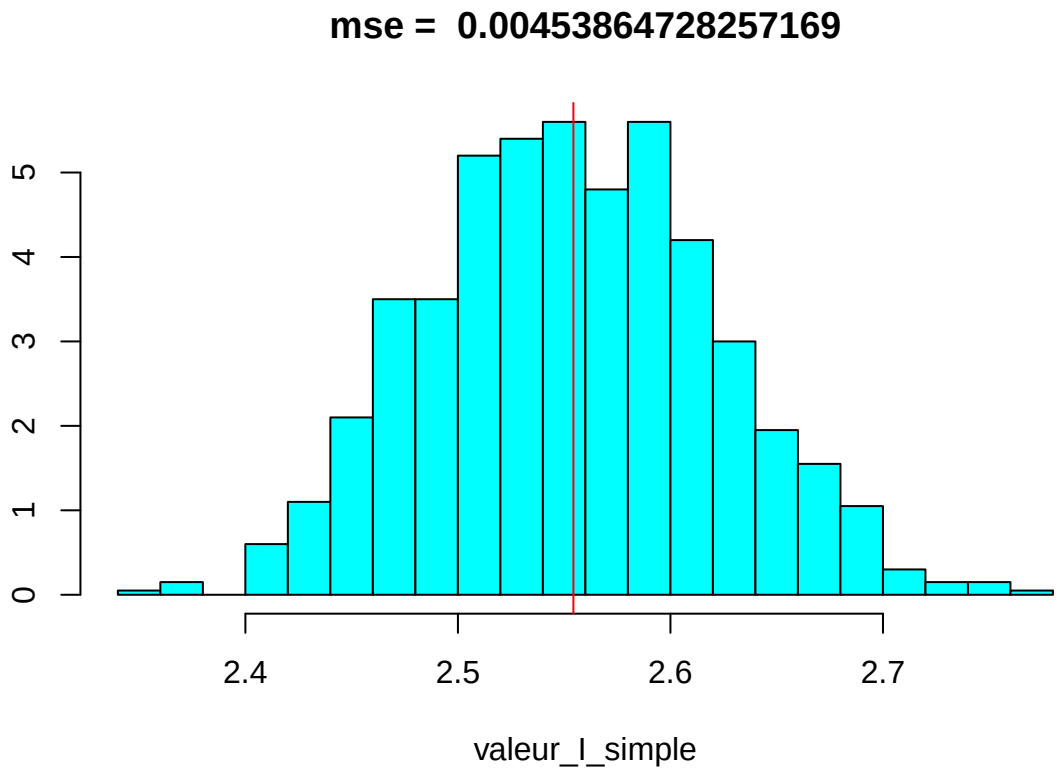
```
estim_I_simple <- function(n){
  b <- 2
  a <- 0


  x <- runif(n,min = a, max = b)

  # I = (b-a)* E[g(X)]
  I_estim = (b - a) * mean(g(x))
  return(I_estim)
}
```

```
estim_I_simple(1000)
```

```
## [1] 2.542057
```

```
mse_I_simple <- function(n,solution_I,nrep){
  valeur_I_simple <- c()
  for (run in 1:nrep) {
  valeur_I_simple <- c(valeur_I_simple, estim_I_simple(n))
  }
  MSE <- mean((valeur_I_simple - solution_I)**2)
  truehist(valeur_I_simple)
  abline(v = solution_I, col = "red")
  title(main = paste("mse = ",MSE))
}
```

```
mse_I_simple(1000,solution,1000)
```



dbeta : fonction de densité de loi beta mais nous on multiplie par 2