Simulation

Jérémie Barde

28 February 2023

Résumé

Le présent document contient des exemples de codes R sur la mise en oeuvre de la méthode de simulation Monte-Carlo et de ses applications.

Table des matières

1 Théorème de la fonction quantile **2**

2 Application à d'autres lois **2**

3 Fonction R **2**

4 Méthode Monte Carlo **3**

4.1 Fonction de répartition empirique 3

4.2 Espérance, variance et autres quantitées 4

4.3 VaR 5

4.4 TVaR 5

4.5 Convolution 6

4.6 Lois de mélange 8

1 Théorème de la fonction quantile

Le théorème de la fonction quantile est utile pour la simulation.

$$F_X^{-1}(U) \overset{\mathcal{D}}{=} X$$

Il permet de simuler des réalisations d'une v.a. X. On commence par simuler des réalisations d'une loi $U(0,1)$. On utilise `runif` pour simuler arbitrairement 5 réalisations.

```
runif(5)
```

```
## [1] 0.4244738 0.8845045 0.3436027 0.3351920 0.4535396
```

On remarque que, si l'on relance la ligne de code, les valeurs changent. C'est normal, puisque l'on souhaite obtenir des réalisations aléatoires. D'ailleurs, il est possible de mettre un `set.seed` initialement pour toujours avoir les mêmes valeurs retournées.

```
set.seed(2022)
runif(5)
```

```
## [1] 0.8159777 0.6472593 0.1203286 0.5438002 0.1847300
```

Si l'on relance le code, les valeurs retournées sont toujours les mêmes.

Important : il faut bien comprendre qu'avec la simulation, on obtient des réalisations, soit des valeurs dans le domaine de la loi utilisée et non la densité de la v.a.

2 Application à d'autres lois

On a déjà simulé 5 réalisations d'une loi uniforme et on veut maintenant simuler 5 réalisations d'une loi exponentielle $\beta = 0.1$. Il faut donc inverser la fonction de répartition de la loi exponentielle et l'évaluer à U.

$$F_X^{-1}(U) = -\frac{1}{\beta}\ln(1-U), \text{ où } U \sim Unif(0,1)$$

```
b <- 0.1
U <- runif(5)
qexp(U, b)
```

```
## [1] 10.1002698  0.7720399  0.4288238  4.6253894 14.1573513
```

On obtient bien 5 réalisations d'une loi exponentielle $\beta = 0.1$.

3 Fonction R

Évidemment, il existe déjà, en R, des fonctions pour simuler plusieurs lois de probabilité. Pour la loi exponentielle, il suffit de faire `rexp`. Bien sûr, il est important de comprendre la méthodologie derrière. Si l'on reprend l'exemple de tantôt :

```
rexp(5, b)
```

```
## [1] 63.3865190  0.3879355  2.1895291 24.3607199  3.9532025
```

On peut faire la même chose avec les autres lois déjà programmées dans R. Par exemple, la loi Gamma ne s'inverse pas à la main, on peut utiliser `rgamma`.

4 Méthode Monte Carlo

4.1 Fonction de répartition empirique

Une fois qu'on a simulé des réalisations, on veut pouvoir travailler avec celle-là. On commence par trouver la fonction de répartition empirique.

$$F_X^{(m)}(x) \simeq \frac{1}{m} \sum_{j=1}^{m} 1_{\{X^{(j)} \leq x\}}$$

4.1.1 Exemple

On a 1000 réalisations qui proviennent d'une $Gamma(2, 5)$ et on veut trouver la fonction de répartition empirique.

```
par <- c(2, 0.5)
n <- 1000
X <- rgamma(n, par[1], par[2])
k <- 10

# Première façon
Fn <- function(x) mean(X < x)

# dexième façon
Fm <- ecdf(X)

table <- cbind(Fn(k), Fm(k), pgamma(k, par[1], par[2]))
colnames(table) <- c("Fn(10)", "Fm(10)", "FX(10)")
table
```

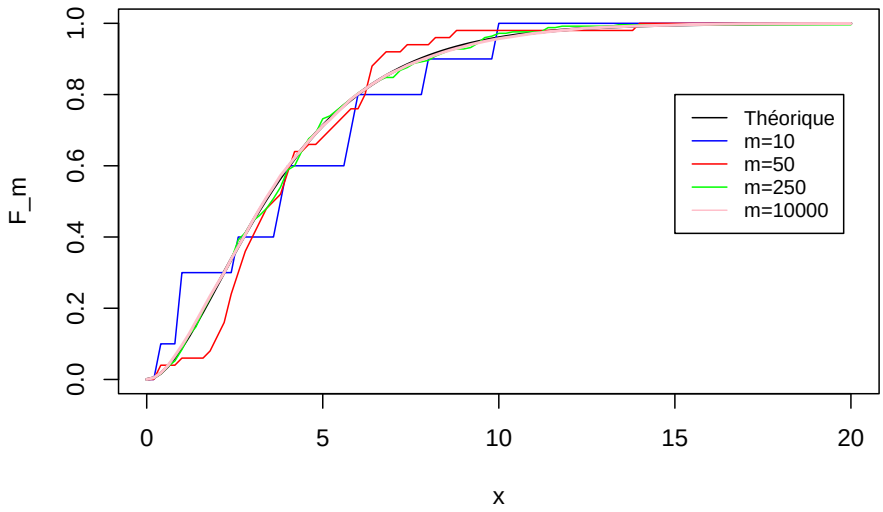
```
##      Fn(10) Fm(10)    FX(10)
## [1,]  0.965  0.965 0.9595723
```

4.1.2 Représentation graphique

Regardons un graphique de la fonction de répartition empirique quand m augmente.

```
set.seed(2022)
data <- sapply(c(10, 50, 250, 10000), function(k) rgamma(k, par[1], par[2]))
F10 <- ecdf(data[[1]])
F50 <- ecdf(data[[2]])
F250 <- ecdf(data[[3]])
F10000 <- ecdf(data[[4]])

curve(pgamma(x, par[1], par[2]), ylim = c(0, 1), xlim = c(0, 20), lwd = 2, ylab = "F_m")
curve(F10, xlim = c(0, 20), lwd = 1, col = "blue", add = T)
curve(F50, xlim = c(0, 20), lwd = 1, col = "red", add = T)
curve(F250, xlim = c(0, 20), lwd = 1, col = "green", add = T)
curve(F10000, xlim = c(0, 20), lwd = 2, col = "pink", add = T)
legend(15, 0.8, legend = c("Théorique", "m=10", "m=50", "m=250", "m=10000"), col = c("black",
    "blue", "red", "green", "pink"), lty = 1:1, cex = 0.8)
```



On constate que plus m augmente, plus on est près de la fonction de répartition théorique. C'est ce qu'on s'attendait à voir.

4.2 Espérance, variance et autres quantitées

On veut trouver différentes quantités reliées aux réalisations qu'on a pour estimer les quantités réelles.

$$\bar{X} = \frac{1}{m}\sum_{j=1}^{m} x_j, \qquad S^2 = \frac{1}{m-1}\sum_{j=1}^{m}(x_j - \bar{X})^2$$

$$\hat{\gamma} = \frac{1}{m}\sum_{j=1}^{m}\left(\frac{(x_j - \bar{X})}{\sqrt{S^2}}\right)^3, \quad \widehat{\pi_X(d)} = \frac{1}{m}\sum_{j=1}^{m}\max(x_j - d, 0)$$

4.2.1 Exemple

On a 500 000 réalisations d'une loi de poisson $\lambda = 3$. On veut trouver l'espérance, la variance, le skewness, $E[X \times 1\{x \leq 2\}]$ et $\pi_X(2)$ empirique.

```
set.seed(2022)
m <- 5e+05
Y <- rpois(m, 3)
d <- 2
# Esperance et variance
esp <- mean(Y)
var <- var(Y)
# Skewness (package moments)
ske <- 1/m * sum(((Y - esp)/(sd(Y)))^3)
ske <- skewness(Y)
# Espérance tronquée
espT <- sum(Y[Y <= 2])/m
espT <- mean(Y * I(Y <= 2))
# Stop-loss
sl <- 1/m * sum(pmax(Y - d, 0))
round(cbind(esp, var, ske, espT, sl), 3)
```

```
##      esp  var   ske  espT    sl
## [1,]   3 2.99 0.576 0.599 1.248
```

4.3 VaR

Il est également possible d'approximer la VaR.

$$VaR_\kappa(x) \simeq F_X^{(m)-1}(\kappa) = \inf\{X^{(j)}, j = 1, 2, \ldots, m; F_X(X^{(j)}) \geq \kappa\}$$

On reprend l'exemple 4.1.1 et on veut trouver la $VaR_{0.99}(X)$.

```
k <- 0.99
# Première façon
Empirique1 <- sort(X)[k * n]
# Deuxième façon
Empirique2 <- quantile(X, k)
Theorique <- qgamma(k, par[1], par[2])
cbind(Empirique1, Empirique2, Theorique)
```

```
##     Empirique1 Empirique2 Theorique
## 99%   12.76622   12.76848   13.2767
```

On n'est pas si loin de la vraie valeur, mais on pourrait être plus proche si on augmentait le nombre d'observations. On refait le calcul avec $m = 1000000$.

```
m <- 1e+06
k <- 0.99
X <- rgamma(m, par[1], par[2])
# Première façon
Empirique1 <- sort(X)[k * m]
# Deuxième façon
Empirique2 <- quantile(X, k)
Theorique <- qgamma(k, par[1], par[2])
cbind(Empirique1, Empirique2, Theorique)
```

```
##     Empirique1 Empirique2 Theorique
## 99%   13.24856   13.24857   13.2767
```

On est déjà beaucoup plus proche.

4.4 TVaR

La TVaR n'est pas toujours un calcul très évident, il peut être utile de savoir l'approximer.

$$TVaR_\kappa(X) \simeq \frac{1}{1-\kappa}\left(\frac{1}{m}\sum_{j=j_0+1}^{m} X^{(j)} + X^{(j_0)}(F^{(m)}(X^{(j_0)}) - \kappa)\right)$$

Où $X^{(j_0)} = F^{(m)-1}(x)$.

4.4.1 Exemple : continue

On a $X \sim Pareto(3, 10)$ et on veut approximer la $TVaR_{0.9}(X)$.

```
set.seed(2022)
m <- 1e+06
a <- 3
l <- 10
k <- 0.9
X <- rpareto(m, a, l)
Fm <- ecdf(X)
VaR <- quantile(X, k)[[1]]
```

```
## Calcul TVaR

# Première façon : espérance tronquée
F1 <- 1/(1 - k) * (mean(X * I(X > VaR)) + VaR * (Fm(VaR) - k))
# Deuxième façon : stop-loss
F2 <- VaR + 1/m * sum(pmax(X - VaR, 0))/(1 - k)
# Troisème façon : rapide si loi continue
F3 <- mean(X[X > VaR])

cbind(F1, F2, F3)
```

```
##            F1       F2       F3
## [1,] 22.28727 22.28727 22.28727
```

Ici, on peut remarquer que la deuxième portion de la première façon donne 0.

4.4.2 Exemple : discret

On a $X \sim Bin(20, 0.35)$ et on veut approximer la $TVaR_{0.9}(N)$.

```
set.seed(2022)
m <- 1e+06
n <- 20
p <- 0.35
k <- 0.9
N <- rbinom(m, n, p)
Fm <- ecdf(N)
VaR <- quantile(N, k)[[1]]
## Calcul TVaR

# Première façon : espérance tronquée
F1 <- 1/(1 - k) * (mean(N * I(N > VaR)) + VaR * (Fm(VaR) - k))
# Deuxième façon : stop-loss
F2 <- VaR + 1/m * sum(pmax(N - VaR, 0))/(1 - k)
# Troisième façon : Définition
F3 <- mean(sort(N)[(0.9 * m + 1):m])

cbind(F1, F2, F3)
```

```
##            F1       F2       F3
## [1,] 10.80204 10.80204 10.80204
```

Ici, on ne peut pas enlever la deuxième portion de la première façon, car elle ne donne pas nécessairement 0.

4.5 Convolution

Il est aussi possible d'approximer des produits de convolution.

4.5.1 Exemple cas exponentielle

On a $X_i \sim Exp(\lambda),\ i = 1, 2, \ldots, n$ et $S = \sum_{i=1}^{n} X_i$.

$$\begin{aligned} S &= -\frac{1}{\lambda}\ln(1-U_1) + \cdots + \frac{1}{\lambda}\ln(1-U_n) \\ &= -\frac{1}{\lambda}(\ln(1-U_1) + \cdots + \ln(1-U_n)) \\ &= -\frac{1}{\lambda}\ln((1-U_1) \times \cdots \times (1-U_n)) \\ &= -\frac{1}{\lambda}\ln(U_1 \times \cdots \times U_n) \end{aligned}$$

Ce qui devient une technique pour simuler une loi Erlang manuellement. On peut l'essayer :

```
m <- 1e+05
l <- 0.1
U1 <- runif(m)
U2 <- runif(m)
S <- -1/0.1 * log(U1 * U2)
EspEmp <- mean(S)
EspThe <- mgamma(1, 2, l)
cbind(EspEmp, EspThe)
```

```
##        EspEmp EspThe
## [1,] 19.97155     20
```

4.5.2 Exemple : loi bernouilli

On veut approximer la convolution de 10 v.a bernouilli, $S = \sum_{j=1}^{10} I_j$:

```
set.seed(2022)
m <- 1e+06
p <- 0.75
I <- sapply(1:10, function(i) rbinom(m, 1, p))

S <- rowSums(I)

EspEmp <- mean(S)
EspThe <- sum(dbinom(0:10, 10, p) * 0:10)
cbind(EspEmp, EspThe)
```

```
##        EspEmp EspThe
## [1,] 7.501304    7.5
```

On obtient bel et bien que $S \sim Bin(10, 0.75)$.

4.5.3 Exemple : loi Erlang et pareto

La simulation devient très pratique quand on veut faire le produit de convolution entre deux lois où l'on ne connaît pas le résultat. Par exemple, on a $X \sim Erl(3, 0.5)$, $Y \sim Pareto(3, 10)$ et $S = X + Y$. Voici l'intégrale qu'il faudrait résoudre pour trouver $F_S(25)$.

$$\begin{aligned} F_S(25) &= \int_0^{25} F_Y(k) f_X(s-k) dk \\ &= \int_0^{25} \left(1 - \left(\frac{10}{10+k}\right)^3\right) \cdot \frac{0.5^3}{\Gamma(3)}(25-k)^2 e^{-0.5(25-k)} dk \end{aligned}$$

Ce n'est pas une intégrale plaisante à résoudre. On obtiendrait que $F_S(25) = 0.9532$, trouvé à l'aide de (https://www.desmos.com). Voyons avec la simulation :

```
m <- 1e+06
X <- rgamma(m, 3, 0.5)
Y <- rpareto(m, 3, 10)
S <- X + Y

Fm <- ecdf(S)
Fm(25)
```

```
## [1] 0.95343
```

On obtient une valeur qui est tout de même très proche.

4.6 Lois de mélange

Regardons maintenant deux exemples de simulation avec une loi de mélange.

4.6.1 Exemple : paramêtre discret

On a $S|\Lambda \sim Po(\Lambda)$ et

$$S|\Lambda \sim Po(\Lambda) \quad \Pr(\Lambda = \lambda) = \begin{cases} \frac{5}{7} & ,\lambda = 4 \\ \frac{2}{7} & ,\lambda = 9 \end{cases}$$

On commence par simuler m réalisations de Λ et ensuite on simule m réalisations de $S|\Lambda$.

```
m <- 1e+06
lam <- sample(c(4, 9), m, replace = TRUE, prob = c(5/7, 2/7))
Slam <- rpois(m, lam)

Empirique <- mean(Slam)
Theorique <- 5/7 * 4 + 2/7 * 9
round(cbind(Empirique, Theorique), 4)
```

```
##      Empirique Theorique
## [1,]    5.4289    5.4286
```

4.6.2 Exemple : Exponentielle/Gamma

On a

$$S|\Lambda \sim Exp(\Lambda), \quad \Lambda \sim Gamma(2, 0.5)$$

Même principe que pour le premier exemple.

```
m <- 1e+05
a <- 2
b <- 6
lam <- rgamma(m, a, b)
Slam <- rexp(m, lam)

Empirique <- mean(Slam)
Theorique <- b/(a - 1)  # Pareto
round(cbind(Empirique, Theorique), 4)
```

```
##      Empirique Theorique
## [1,]    5.9764         6
```

On peut se convaincre graphiquement que le mélange donne une loi pareto.

```
Fm <- ecdf(Slam)
curve(ppareto(x, a, b), lwd = 4, col = "red", ylim = c(0, 1), xlim = c(0, 60))
curve(Fm, add = T, lwd = 2, col = "blue")
legend(45, 0.8, legend = c("Pareto", "Mélange"), col = c("red", "blue"), lty = 1:1,
    cex = 0.8)
```