Méthode de Monte Carlo - Projet

Mathias Marciano - Samuel Bensoussan

Préalable

Importons Les librairies que nous allons utiliser :

```
library(microbenchmark)
```

Définissons au préalable quelques fonctions qui nous seront utiles par la suite :

```
#Estimateur de Monte Carlo classique
estim_mc<-function(X, h = identity){
  Y <- h(X)
  sd <- var(Y)
  return (data.frame(value = mean(Y), variance=sd))
}
#Generateur de la loi normale multivariee dans le cas general
rmvnorm<-function(n, mu, sigma){
  z<-matrix(rnorm(length(mu)*n), nrow=length(mu))
  return(mu + t(chol(sigma)) %*% z)
}
```

Exercice 1 *(Déambulation vers la sortie)*

Dans cet exercice on considère la marche aléatoire suivante : $W_m = \begin{cases} 0 & \text{si } m = 0 \\ \sum_{k=1}^{m} X_k & \text{si } m \geq 1 \end{cases}$ avec $(X_n)_{n\in\mathbb{N}}$ une suite de variables i.i.d. de loi $\mathcal{N}(\mu, 1), \mu \in \mathbb{R}^*_-$.

Cette marche aléatoire sera observée pendant une durée aléatoire

$$T = \inf\{m \in \mathbb{N}^* \mid W_m \notin]a, b[\}, \quad \text{avec} \quad a, b \in \mathbb{R} \text{ tels que } a \leq 0 < b.$$

L'objectif de cet exercice est d'estimer la quantité $\delta = \mathbb{P}_\mu [W_T \geq b]$

Calculons $\widehat{\delta}_n$ et $\widehat{\Delta}_n(-\mu)$

Rappel : nous avons vu aux questions 2 et 3 que :

$\hat{\delta}n = \frac{1}{n}\sum_{i=1}^{n} 1\left\{\sum_{j=1}^{T_i} X_{ij} \geqslant b\right\}$ avec $X_{ij} \sim \mathcal{N}(\mu, 1)$

$\hat{\Delta}n(\theta) = \frac{1}{n}\sum_{i=1}^{n} 1\left\{\sum_{j=1}^{T_i} X_{ij} \geqslant b\right\} \prod_{j=1}^{T_i} \frac{f(X_{ij})}{g(X_{ij})}$ avec $X_{ij} \sim \mathcal{N}(\theta, 1)$

avec $W_m^i = \sum_{j=1}^{m} X_{ij}$; $T_i = \inf\left\{m \in \mathbb{N}^* \mid W_m^i \notin \right] a, b[\} $; $f \sim \mathcal{N}(\mu, 1)$ et $g \sim \mathcal{N}(\theta, 1)$

Pour faire l'application numérique de ces estimateurs, il convient tout d'abord de définir les variables globales qui interviennent dans les expressions de ceux-ci :

```
a<--3
b<-5
```

```
N<-10**6
Mu<-(-1)
```

Question 5

L'idée de l'algorihme du calcul de $\hat{\delta}n$ est de créer dans un premier temps une fonction qui nous renvoie n réalisations de la variable aléatoire W_m^i :

```
rgen_W<-function(n, mu, sd=1){
  X<-rnorm(n,mu,sd)
  l<-which(X<b & X>a) #indices de X qui ne respectent pas les conditions du temps d'arret
  indice<-which(X>=b | X<=a) #indices qui respectent les conditions du temps d'arret
  W<-X[indice] #On ajoute a notre vecteur W les "bons" X
  while(length(which(X[l]<b & X[l]>a))>0){
    X<-X[l]+rnorm(l, mu, sd) #La somme des X augmente de 1 terme
    l<-which(X<b & X>a) #indices de X qui ne respectent pas les conditions du temps d'arret
    indice<-which(X>=b | X<=a) #indices qui respectent le temps d'arret
    W<-append(W, X[indice])
  }
  return(W)
}
```

En effet, $\hat{\delta}n$ est une fonction de $(W_m^1, W_m^2, \ldots, W_m^n)$ donc il suffit d'appliquer la fonction de l'estimateur de Monte Carlo classique au vecteur W :

```
#Fonction indicatrice >=b
h1<-function(x){
  return(x>=5)
}
delta_hat<-estim_mc(rgen_W(N, Mu), h1)
```

On obtient le résultat suivant :

```
##          value     variance
## 1 1.099998e-05 1.099987e-05
```

La valeur de $\hat{\delta}n$ est "censé" être proche de 0 car l'événement $\left\{W_m^i \geqslant b\right\}$ est très peu réalisable. Par conséquent, la fonction h1 définie ci-dessus n'est pas nulle avec un probabilité très faible dès lors que $X_{ij} \sim \mathcal{N}(-1, 1)$. Ici en pratique, on a choisi n de sorte que l'événement $\left\{W_m^i \geqslant b\right\}$ se produit plusieurs fois ce qui explique que $\hat{\delta}n$ n'est pas nulle. C'est pourquoi on opte pour la méthode d'échantillonnage préférentiel avec des $X_{ij} \sim \mathcal{N}(1, 1)$. Ainsi, la fonction h1 ne sera pas nulle avec une faible probabilité car l'événement $\left\{W_m^i \geqslant b\right\}$ sera réalisable.

Rappel : $\hat{\Delta}n(-\mu) = \frac{1}{n}\sum_{i=1}^{n} 1\left\{\sum_{j=1}^{T_i} X_{ij} \geqslant b\right\} \exp\left(2\mu \sum_{y=1}^{T_i} X_{ij}\right)$

On definit alors une fonction auxiliaire g qui nous permet d'utiliser la fonction estim_mc pour calculer $\hat{\Delta}n(-\mu)$:

```
g<-function(x, mu=-1){
  return((x>=b)*exp(2*mu*x))
}
big_delta_hat<-estim_mc(rgen_W(N, -Mu), g)
```

qui renvoie comme résultat :

```
##          value     variance
## 1 1.452596e-05 1.613475e-10
```

Le rapport des variances vaut alors :

```
## [1] 68175
```

La méthode d'échantillonnage préférentiel est donc une très bonne méthode de réduction de variance car on voit que le rapport ci-dessus est très important, de l'ordre de 10^4.

L'efficacité relative de $\hat{\Delta}n(-\mu)$ par rapport à $\hat{\delta}n$ est donné par :

$R\left(\hat{\delta_n}, \hat{\Delta_n}(-\mu)\right) = \frac{C_1}{C_2} \times \frac{\text{Var}\left[\hat{\delta_n}\right]}{\text{Var}\left[\hat{\Delta_n}(-\mu)\right]}$ avec C_1 le coût de calcul de $\hat{\delta}n$ et C_2 le coût de calcul de $\hat{\Delta}n(-\mu)$
Calculons maintenant le coût de calcul de chaque méthode :

```
#Cout de calcul des méthodes
tme = microbenchmark(estim_mc(rgen_W(N, Mu), h1), estim_mc(rgen_W(N, -Mu), g))

C1<-mean(tme$time[tme$expr=="estim_mc(rgen_W(N, Mu), h1)"]) #Cout de calcul pour delta_hat
C2<-mean(tme$time[tme$expr=="estim_mc(rgen_W(N, -Mu), g)"]) #Cout de calcul pour big_delta_hat
efficacite_relative<-C1*(delta_hat$variance)/(C2*big_delta_hat$variance)
```

L'efficacité relative est de l'ordre de 10^4 :

```
## [1] 48028.03
```

On remarque que le rapport des coûts de calcul vaut :

```
## [1] 0.7044816
```

Ce rapport influe très peu sur l'efficacité relative : on comprend dès lors que c'est le rapport des variances qui va déterminer cette efficacité. Conclusion : $\hat{\Delta}n(-\mu)$ est un **meilleur estimateur** que $\hat{\delta}n$ en particulier car la méthode d'échantillonnage préférentiel est une très bonne méthode de réduction de variance dans ce problème.

Exercice 2 *(Recettes de financier)*

Partie 1

Dans cette partie, on reconsidère la marche aléatoire $(W_n)_{n\in\mathbb{N}}$ introduite dans l'Exercice 1 à la différence que $T \in \mathbb{N}^*$ est fixé. L'objectif ici est déterminer la quantité :

$\delta = \mathbb{E}\left[\max\left\{\lambda \exp\left(-\sigma W_T\right) - K, 0\right\}\right], \quad \text{avec} \quad \lambda, \sigma, K \in \mathbb{R}_+^*$

Définissons les variables qui vont nous être utile pour les applications numériques de cette partie :

```
Mu = -1
t = 91
lambda = 30.63
Sd = 0.01
K = 72.36
N = 10**5
```

Question 1-b)

Rappel : dans la question 1-a), nous avons montré que $W_T \sim \mathcal{N}(\mu T, T)$ et que l'estimateur de Monte Carlo classique de δ est donné par : $\hat{\delta}n = \frac{1}{n}\sum_{i=1}^{n} \max\left\{\lambda \exp\left(-\sigma Y_i\right) - K, 0\right\}$ avec $Y_i \overset{iid}{\sim} \mathcal{N}(\mu T, T)$ On définit la fonction $h2(x) = \max\left\{\lambda \exp\left(-\sigma x\right) - K, 0\right\}$ qui va nous servir pour calculer $\hat{\delta}n$ à l'aide de *estim_mc* :

```
h2<-function(x){
  return(pmax(lambda*exp(-Sd*x)-K,0))
}
```

On utilise la fonction *pmax* et non pas *max* car on va manipuler des vecteurs. De plus, étant donné que $h2(X) = \max\left\{\lambda \exp\left(-\sigma X\right) - K, 0\right\}$ est de carré intégrable et que $\hat{\delta}n$ est un estimateur sans biais

de δ, d'après le cours on a montré que l'erreur quadratique moyenne relativement à δ est donné par : $\text{MSE}\left(\hat{\delta}n, \delta\right) = \text{Var}\,\frac{[h2(Y_1)]}{n} = \text{Var}\left[\hat{\delta}_n\right]$ avec $Y_1 \sim \mathcal{N}(\mu T, T)$

On peut dès lors modifier notre fonction *estim_mc* et renvoyer comme donnée supplémentaire le $\text{MSE}\left(\hat{\delta}n, \delta\right)$:

```
#Estimateur de Monte Carlo classique
estim_mc<-function(X, h = identity){
  Y <- h(X)
  sd <- var(Y)
  return (data.frame(value = mean(Y), variance=sd, MSE=sd/length(X)))
}
```

Ce qui nous renvoie finalement :

```
##      value variance         MSE
## 1 5.310132  32.8893 0.000328893
```

avec un MSE de l'ordre de 3×10^{-4}

Question 2-b)

Rappel : on a utilisé la transformation $A(x) = 2\mu T - x$ pour laisser la loi ν invariante et on a obtenu un nouvel estimateur par la méthode de la variable antithétique donné par : $\hat{\delta_n}(A) = \frac{1}{n}\sum_{i=1}^{n} \frac{h_2(Y_i)+h_2 \circ A(Y_i)}{2}$

De même que dans l'exercice 1, on code une fonction auxiliaire h3 pour utiliser la fonction *estim_mc* et éviter de surcharger notre code :

```
A<-function(x){
  return(2*Mu*t - x)
}
h3<-function(x){
  return((h2(x)+h2(A(x)))/2)
}
```

Ainsi, $\hat{\delta_n}(A)$ est donné par :

```
d_n_hat_A<-estim_mc(rnorm(N, Mu*t, sqrt(t)), h3)
```

qui renvoie :

```
##      value variance        MSE
## 1 5.320895   4.3606 4.3606e-05
```

La variance est le MSE sont ici plus faible que précèdemment ce qui est cohérent car la méthode de la variable antithétique est une méthode de réduction de variance.

Question 3

De la même manière que l'exercice 1, l'efficacité relative de $\hat{\delta}n(A)$ par rapport à $\hat{\delta}n$ est donné par :

$R\left(\hat{\delta_n}, \hat{\delta_n}(A)\right) = \frac{C_1}{C_2} \times \frac{\text{Var}\left[\hat{\delta_n}\right]}{\text{Var}\left[\hat{\delta_n}(A)\right]}$ avec C_1 le coût de calcul de $\hat{\delta}n$ et C_2 le coût de calcul de $\hat{\delta}n(A)$

Calculons maintenant le coût de calcul de chaque méthode :

```
#Cout de calcul des méthodes
tme = microbenchmark(estim_mc((rnorm(N, Mu*t, sqrt(t))), h2),
                     estim_mc(rnorm(N, Mu*t, sqrt(t)), h3))
#Cout de calcul pour d_n
C1<-mean(tme$time[tme$expr=="estim_mc((rnorm(N, Mu * t, sqrt(t))), h2)"])
```

```
#Cout de calcul pour d_n_hat_A
C2<-mean(tme$time[tme$expr=="estim_mc(rnorm(N, Mu * t, sqrt(t)), h3)"])

efficacite_relative<-C1*(d_n$variance)/(C2*d_n_hat_A$variance)
```

Temps de calcul de $\hat{\delta}n$ (première entrée) et $\hat{\delta}n(A)$ (deuxième entrée) :

```
## Unit: milliseconds
##                                         expr      min       lq     mean   median
##  estim_mc((rnorm(N, Mu * t, sqrt(t))), h2) 12.50409 13.77168 16.36599 14.68261
##    estim_mc(rnorm(N, Mu * t, sqrt(t)), h3) 15.33932 17.78422 21.45535 18.96555
##        uq      max neval
##  17.04613 47.01820   100
##  22.14686 46.90812   100
```

L'efficacité relative vaut :

```
## [1] 5.753276
```

On remarque que le rapport des coûts de calcul vaut :

```
## [1] 0.7627931
```

De la même façon que l'exercice 1, le rapport des variances qui est de :

```
## [1] 7.54238
```

est dominant par rapport au terme des coûts dans l'efficacité relative car il vaut 10 fois ce dernier. Conclusion : l'efficacité relative étant supérieur à 1, $\hat{\delta}n(A)$ est un **meilleur estimateur** que $\hat{\delta}n$. Ici encore, c'est grâce au fait qu'on a utilisé la méthode de la variable antithétique qui permet de réduire la variance du nouvel estimateur.

Partie 2

Question 3

Soit $\mathbf{X} = (X_1, \ldots, X_d)$ un vecteur gaussien de $\mathbb{R}^d$. L'objectif de cette partie est d'estimer la quantité : $p = \mathbb{E}\left[\max\left\{\frac{1}{d}\sum_{i=1}^{d}\exp(\sigma X_i) - K, 0\right\}\right], \quad \text{avec} \quad \sigma, K \in \mathbb{R}_+^*$

Partie 2

Soit $\mathbf{X} = (X_1, \ldots, X_d)$ un vecteur gaussien de $\mathbb{R}^d$. L'objectif de cette partie est d'estimer la quantité : $p = \mathbb{E}\left[\max\left\{\frac{1}{d}\sum_{i=1}^{d}\exp(\sigma X_i) - K, 0\right\}\right], \quad \text{avec} \quad \sigma, K \in \mathbb{R}_+^*$

Question 4-b)

Rappel : dans la question 4-a) nous avons montré que l'estimateur de Monte Carlo classique est donné par J :

$\widehat{J}_n = \frac{1}{n}\sum_{i=1}^{n}\max\{\lambda_1\exp(\sigma_1 X_{1i}) + \lambda_2\exp(\sigma_2 X_{2i}) - K, 0\}$ avec $(X_i = (X_{1i}, X_{2i}))_{i\in[1,n]}$ suite de variables aléatoires iid suivant la loi de X.

Définissons les variables globales qui vont intervenir dans notre code :

```
K = 67.8
Mu = c(0,0)
ro = -0.72
l1 = 32
l2 = 35.9
sd1 = 0.5
sd2 = 0.3
```

```
N = 10**4
sigma=matrix(c(1,ro,ro,1), nrow=2)
```

Codons dans R la fonction $h_4(X_1, X_2) = \max\{\lambda_1 \exp(\sigma_1 X_1) + \lambda_2 \exp(\sigma_2 X_{2i}) - K, 0\}$ associé à l'estimateur de Monte Carlo classique de J :

```
h4<-function(x1,x2){
  return(pmax(l1*exp(-sd1*x1)+l2*exp(-sd2*x2)-K, 0))
}
```

L'estimateur de Monte Carlo classique est donné par le code suivant :

```
#On renvoie par defaut un intervalle de confiance a un niveau de 90%
J_n_hat = function(n, alpha=0.1){
  #On genere n vecteurs gaussiens de R2 sous forme de matrice
  X<-rmvnorm(n, Mu, sigma)
  X1<-X[1,] #correspond aux X1i
  X2<-X[2,] #correspond aux X2i
  A<-pmax(l1*exp(-sd1*X1)+ l2*exp(-sd2*X2)-K,0)
  J_hat<-mean(A)
  sd<-var(A)
  q<-qnorm(1 - alpha/2) * sqrt(sd/n)
  return(data.frame(value=J_hat, variance=sd, ic_inf=J_hat - q, ic_sup=J_hat + q))
}
```

La forme de l'intervalle de confiance a été démontré en cours à l'aide du théorème centrale limite et du théorème de Slutsky. Il est de la forme :

$$\left[\widehat{J}_n - q_{1-\frac{\alpha}{2}}\sqrt{\frac{\widehat{\sigma}_n}{n}}; \widehat{J}_n + q_{1-\frac{\alpha}{2}}\sqrt{\frac{\widehat{\sigma}_n}{n}}\right]$$

avec $\widehat{\sigma}_n$ la variance empirique de $h_4(X_1, X_2)$.

Résultats obtenus pour l'estimateur et l'intervalle de confiance :

```
##      value variance   ic_inf   ic_sup
## 1 8.685556 169.3722 8.471489 8.899622
```

Avec un temps de calcul de :

```
## Unit: milliseconds
##        expr      min       lq     mean   median       uq      max neval
##  J_n_hat(N) 3.024898 3.607804 4.276718 3.841977 4.357862 15.04341   100
```

La variance de cet estimateur est particulièrement élevée c'est pourquoi on va recourir à une méthode de réduction de variance.

Question 5-b)

On a montré précèdemment que : $\mathbb{E}[\lambda_1 \exp(-\sigma_1 x_1) + \lambda_2 \exp(-\sigma_2 x_2)] = \lambda_1 \exp(\sigma_1^2/2) + \lambda_2 \exp(\sigma_2^2/2)$ La méthode de contrôle consiste à choisir une fonction h_0 telle que $\mathbb{E}[h_0(\mathbf{X})] = m$ soit facile à calculer et $\mathrm{Var}[h_0(\mathbf{X})] < +\infty$. On nous incite dans l'énoncé à travers les calculs d'espérance faits précèdemment à prendre comme fonction : $h_0(X_1, X_2) = \lambda_1 \exp(-\sigma_1 X_1) + \lambda_2 \exp(-\sigma_2 X_2) - K$ qui est de variance finie. On a ainsi par linéarité de l'espérance : $m = \lambda_1 \exp(\sigma_1^2/2) + \lambda_2 \exp(\sigma_2^2/2) - K$

Ajoutons-les à notre code :

```
h0 <- function(x1,x2){
  return(l1*exp(-sd1*x1)+l2*exp(-sd2*x2)-K)
```

```
}
m<-l1*exp((sd1**2)/2)+l2*exp((sd2**2)/2)-K
```

L'estimateur de J par la méthode de la variable de contrôle est donné par :

$\widehat{J}_n(b) = \frac{1}{n}\sum_{k=1}^{n}\{h(\mathbf{X}_k) - b[h_0(\mathbf{X}_k) - m]\}$

Etape 1 : déterminer le b optimal

Nous allons déterminer le b optimal en appliquant la **Stratégie n°1** du cours qui est la période de chauffe : on génére une suite $(\mathbf{X}_n)_{n\geq 1}$ de variables aléatoires iid suivant la loi de X, on utilise les l premiers termes de cette suite pour déterminer le b optimal puis enfin on utilise les n-l termes restants de la suite pour estimer J. Pour déterminer le nombre l de termes à choisir, on va faire varier l de 1 à N tout en estimant le b optimal à l'aide la formule :

$\widehat{b}_\ell^\star = \frac{\sum_{k=1}^{\ell}(h_0(\mathbf{X}_k)-m)\left(h(\mathbf{X}_k)-\bar{h}_\ell\right)}{\sum_{k=1}^{\ell}(h_0(\mathbf{X}_k)-m)^2}$

Le l sera déterminé dès lors que la valeur de $\widehat{b}_\ell^\star$ se stabilise.

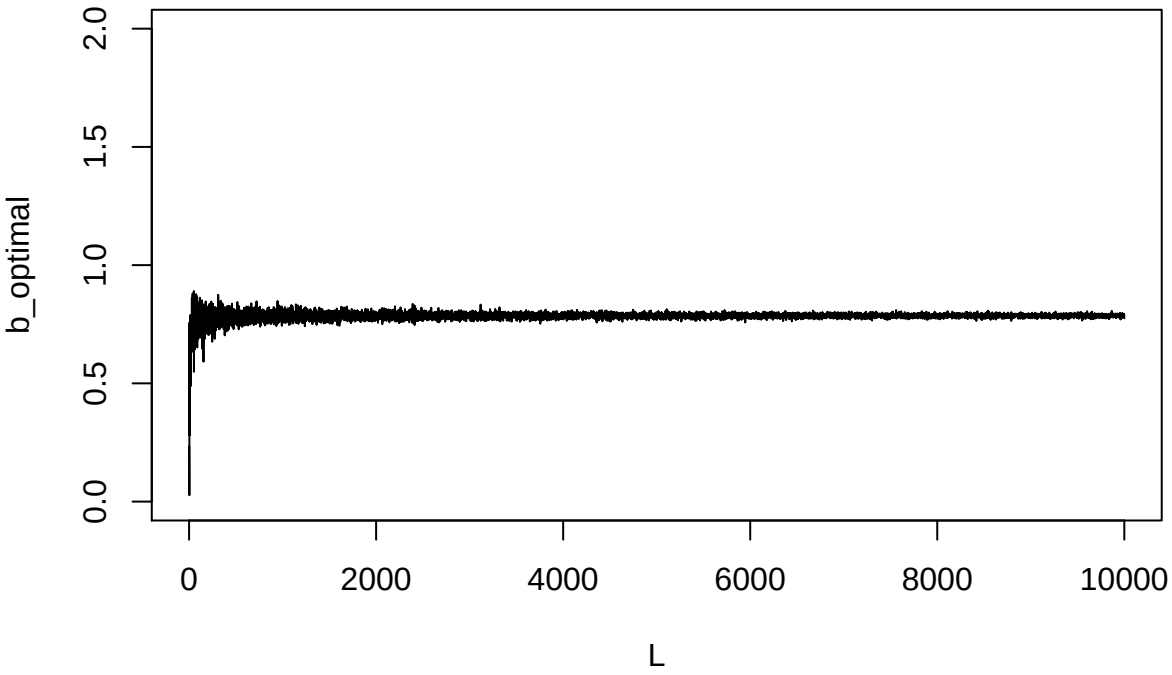
Etape 2 : calculer l'estimateur

Nous avons maintenant toutes les outils pour coder l'estimateur efficacement :

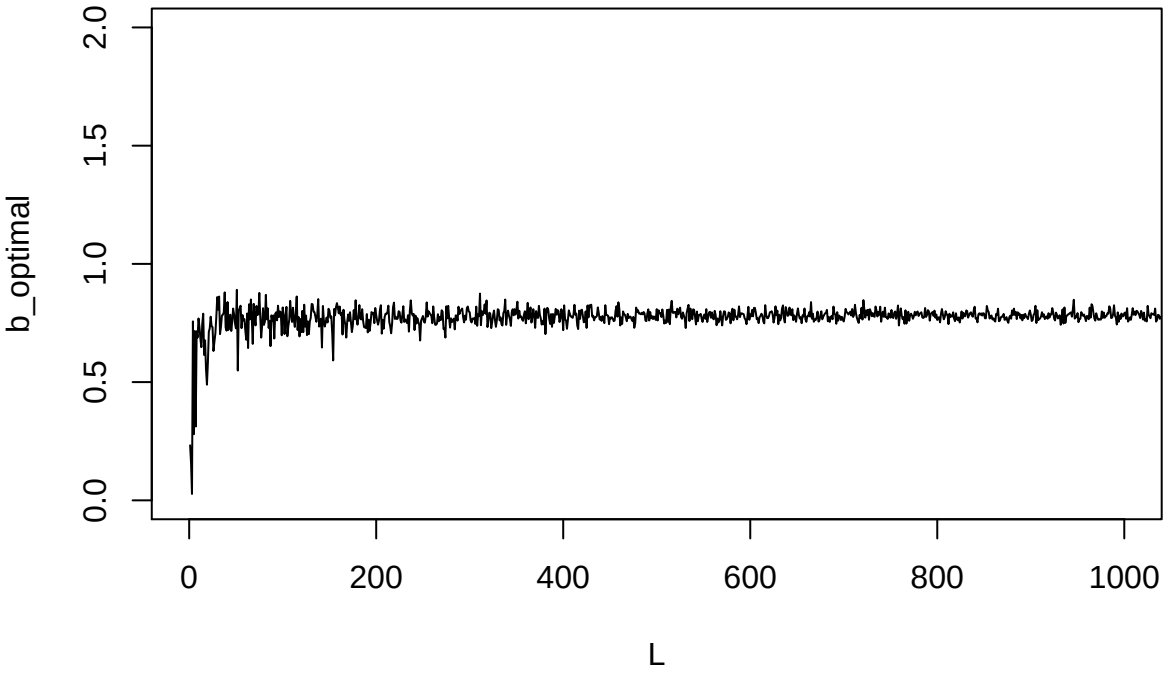
```
b_opti_l <- function(n){
  #On stocke les valeurs de b optimal pour chaque l dans un vecteur
  B<-numeric(n)
  for(l in 1:n){
    X<-rmvnorm(l, Mu, sigma)
    #On restranscrit exactement la formule enonce precedemment
    h_l<-mean(h4(X[1,],X[2,]))
    numerateur<-(h4(X[1,],X[2,])-m)*(h0(X[1,],X[2,])-h_l)
    numerateur<-sum(numerateur)
    denominateur<-sum((h0(X[1,],X[2,])-m)**2)
    B[l]<-numerateur/denominateur
  }
  return(B)
}
```

Puis on trace les valeurs de $\widehat{b}_\ell^\star$ en fonction de l :

```
B<-b_opti_l(N)
plot(B, type="l", xlab="L", ylab="b_optimal", ylim=c(0,2))
```



Faisons un zoom :



On observe que $\widehat{b}_\ell^\star$ se stabilise en oscillant autour de 0.8 à environ $l = 200$: c'est donc cette valeur que nous allons choisir.

```
#Fonction qui renvoie le b_optimal en utilisant le l-echantillon associe
b_opti <- function(n, X){
  m<-l1*exp((sd1**2)/2)+l2*exp((sd2**2)/2)-K
  h_l<-mean(h4(X[1,],X[2,]))
  numerateur<-sum((h4(X[1,],X[2,])-m)*(h0(X[1,],X[2,])-h_l))
  denominateur<-sum((h0(X[1,],X[2,])-m)**2)
  return(numerateur/denominateur)
```

```
}

J_hat_b<-function(n, alpha=0.1){
  X<-rmvnorm(n, Mu, sigma)
  m<-l1*exp((sd1**2)/2)+l2*exp((sd2**2)/2)-K
  l<-200 #Nombre de termes utilises pour estimer b_optimal
  b<-b_opti(l, X[,1:l]) #On utilise les l premiers termes de Xn pour generer b
  X<-X[,(l+1):n] #On garde ensuite seulement les n-l termes pour Jn(b)
  J<-h4(X[1,],X[2,])-b*(h0(X[1,],X[2,])-m)
  J_hat<-mean(J)
  sd<-var(J)
  q<-qnorm(1 - alpha/2) * sqrt(sd/n)
  return(data.frame(value=J_hat, variance=sd, ic_inf=J_hat - q, ic_sup=J_hat + q))
}
```

Les résultats obtenus sont les suivants :

```
##      value variance   ic_inf   ic_sup
## 1 8.639284 12.46601 8.581209 8.697359
```

Avec un temps de calcul de :

```
## Unit: milliseconds
##         expr      min       lq     mean   median       uq     max neval
##  J_hat_b(N) 3.823575 4.476311 5.186968 4.655388 5.130752 22.4556   100
```

Question 6

De la même manière que les exercices précèdemment, l'efficacité relative de $\widehat{J_n}(b)$ par rapport à $\widehat{J_n}$ est donné par :

$R\left(\widehat{J_n}, \widehat{J_n}(b)\right) = \frac{C_1}{C_2} \times \frac{\text{Var}\left[\widehat{J_n}\right]}{\text{Var}\left[\widehat{J_n}(b)\right]}$ avec C_1 le coût de calcul de $\widehat{J_n}$ et C_2 le coût de calcul de $\widehat{J_n}(b)$

Calculons maintenant le coût de calcul de chaque méthode :

```
#Cout de calcul des méthodes
tme = microbenchmark(J_n_hat(N), J_hat_b(N))
#Cout de calcul pour d_n
C1<-mean(tme$time[tme$expr=="J_n_hat(N)"])
#Cout de calcul pour d_n_hat_A
C2<-mean(tme$time[tme$expr=="J_hat_b(N)"])

efficacite_relative<-C1*(J_hat$variance)/(C2*J_n_hat_b$variance)
```

Temps de calcul de $\widehat{J_n}$ (première entrée) et $\widehat{J_n}(b)$ (deuxième entrée) :

```
## Unit: milliseconds
##         expr      min       lq     mean   median       uq      max neval
##  J_n_hat(N) 3.522240 3.638377 4.394573 3.865646 4.530057 18.98517   100
##  J_hat_b(N) 4.143654 4.463516 5.719438 4.800347 5.630826 22.26502   100
```

L'efficacité relative vaut :

```
## [1] 10.43945
```

On remarque que le rapport des coûts de calcul vaut :

```
## [1] 0.7683575
```

De la même façon que l'exercice 1, le rapport des variances qui est de :

```
## [1] 13.58671
```

est dominant par rapport au terme des coûts dans l'efficacité relative car il vaut environ 20 fois ce dernier.

Conclusion : l'efficacité relative étant supérieur à 1, $\widehat{J}_n(b)$ est un **meilleur estimateur** que $\widehat{J}_n$. Ici encore, utiliser la méthode de la variable de contrôle comme méthode de réduction de variance est un bon compromis pour estimer J par rapport à l'estimateur de Monte Carlo classique

Partie 3

Les variables globales suivantes seront utiles pour les applications numériques de cette partie :

```
d<-10
Sd<-0.5
K<-1.17
N<-10**5
L<-40
```

Dans cette dernière partie, nous voulons estimer :

$p = \mathbb{E}\left[\max\left\{\frac{1}{d}\sum_{i=1}^{d}\exp\left(\sigma X_i\right) - K, 0\right\}\right]$ avec $\sigma, K \in \mathbb{R}_+^*$ et $\mathbf{X} = (X_1, \ldots, X_d)$ un vecteur gaussien standard de $\mathbb{R}^d$.

Question 7

Soit $X_j = (X_{1j}, \ldots, X_{dj})$ avec $j \in [1; n]$ un vecteur gaussien standard de $\mathbb{R}^d$. L'estimateur de Monte Carlo classique de p est donnée par :

$\widehat{p_n} = \frac{1}{n}\sum_{j=1}^{n}\max\left\{\frac{1}{d}\sum_{i=1}^{d}\exp\left(\sigma X_{ij}\right) - K, 0\right\}$

L'idée de l'algorithme de $\widehat{p_n}$ est de générer une matrice $n \times d$ où chaque ligne de cette dernière correspond à un vecteur gaussien standard de $\mathbb{R}^d$. Ensuite, il nous suffit d'appliquer $exp(\sigma x)$ à la matrice concernée et d'obtenir la moyenne de chacun de ces lignes.

Introduisons une fonction auxiliaire h5 présente dans l'estimateur de Monte Carlo classique :

```
h5<-function(x){
  return(pmax(x-K,0))
}
```

Il en découle le code suivant pour estimer $\widehat{p_n}$:

```
p_hat<-function(n){
  X<-matrix(rnorm(n*d), nrow=n)
  X<-exp(Sd*X)
  #Renvoie un vecteur qui contient la moyenne de chaque ligne de la matrice
  X<-rowMeans(X)
  return(estim_mc(X, h5))
}
p_n_hat<-p_hat(N)
```

Voici le résultat obtenu :

```
##        value variance        MSE
## 1 0.05977526 0.012431 1.2431e-07
```

Avec un temps de :

```
## Unit: milliseconds
##       expr      min       lq     mean   median       uq      max neval
##  p_hat(N) 112.6857 119.4692 125.8145 122.0034 125.8976 177.5804   100
```

On note la faible variance de l'estimateur de Monte Carlo classique par rapport à ce qu'on a vu précèdemment.

Question 10-b)

Rappel : dans la question 10-a), nous avons pris la partition suivante qui va nous servir de strates :

$$D_1 =]-\infty, q_{1/L}]\,; D_L =]\,q_{\frac{L-1}{L}}; +\infty[\text{ et } \quad \forall i \in [2; L-1], \quad D_i =]\,q_{\frac{i-1}{L}}; q_{\frac{i}{L}}]$$

avec q_i quantile d'ordre i de la loi normale centrée réduite.

Dans cette partie, nous sommes dans le cas d'un estimateur stratifié avec allocation proportionnelle car pour $\ell \in [1, L], \mathbb{P}[< \mathbf{X}, u > \in D_\ell] = 1/L$ donc le nombre de tirage de la loi conditionnelle $L\,(< \mathbf{X}, u > | < \mathbf{X}, u > \in D_k)$ pour chaque $k = 1, \ldots, K$ sera identique avec une valeur de $\frac{n}{L}$. Ainsi, notre estimateur stratifié est de la forme :

$\widehat{p}_n\left(\frac{n}{L}, \ldots, \frac{n}{L}\right)(u) = \sum_{l=1}^{L} \frac{1}{L} \sum_{i=1}^{n/L} h_6\left(\mathbf{Y}_i^{(l)}, u\right)$ avec $\mathbf{Y}_i^{(l)}$ une réalisation de la loi $L\,(< \mathbf{X}, u > | < \mathbf{X}, u > \in D_l)$ et $h_6(x, u) = \max\left\{\frac{1}{d}\sum_{i=1}^{d} \exp\left[\sigma\left(x u_i + Z_i - < \mathbf{Z}, u > u_i\right)\right] - K, 0\right\}$ avec Z vecteur gaussien standard de dimension d.

Etape 1 : simuler $\mathbf{Y}_i^{(l)}$

Pour simuler $\mathbf{Y}_i^{(l)}$, nous allons appliquer une formule du cours qui invoque l'inverse généralisé et la fonction de répartition de notre loi normale centrée réduite. Dans notre cas, cela donne :

$\mathbf{Y}_i^{(l)} = F^{\leftarrow}\left[F\left(d_l\right) + U\left\{F\left(d_{l+1}\right) - F\left(d_l\right)\right\}\right], l = 1, \ldots, L$ avec $d_1 = -\infty; \quad d_{L+1} = +\infty; \forall l = 1, \ldots L: \quad d_\ell = \frac{q_{l-1}}{L}$ et $U \sim U([0, 1])$

Ce qui donne dans notre cas : $\mathbf{Y}_i^{(l)} = F^{\leftarrow}\left[d_l + U\left\{d_{l+1} - d_l\right\}\right]$ car F est continue

Ce qui implique : $\mathbf{Y}_i^{(l)} = F^{\leftarrow}\left[\frac{l-1}{L} + U\left\{\frac{1}{L}\right\}\right]$

Codons ainsi une fonction qui nous renvoie un vecteur contenant n réalisations de $\mathbf{Y}^{(l)}$:

```
rgen_Y<-function(n, l){
  U<-runif(n)
  Y<-qnorm((l-1+U)/L)
  return(Y)
}
```

Etape 2 : générer $\widehat{p}_n$

La plus grande difficulté de cette partie consiste à générer $\widehat{p}_n$ avec un code efficace qui n'utilise qu'un seul for. La première solution "naïve" que nous avions était de code la fonction h_6 de façon vectorielle comme voici :

```
h6_faux<-function(Y_l, u, Z){
  Z_scalaire_u<-Z%*%u
  X<-mean(exp(sd*(Y_l*u + Z - Z_scalaire_u[1] * u)))
  return(pmax(X-K, 0))
}
```

Puis pour une strate fixé l de générer $\frac{n}{L}$ réalisations de $\mathbf{Y}^{(l)}$ et d'appliquer la fonction h_6 à ce vecteur de réalisations pour pouvoir estimer la partie de la formule de l'estimateur $\sum_{i=1}^{n/L} h_6\left(\mathbf{Y}_i^{(l)}, u\right)$. Cependant, un problème ce soulève car si $\mathbf{Y}^{(l)}$ devient un vecteur et non plus un scalaire dans la fonction h_6, le terme $Y_l u$

devient alors une opération entre deux vecteurs en particulier la multiplication composante par composante qui n'est pas ce que nous souhaitons.

C'est pourquoi il préférable d'avoir une méthode "matricielle" pour coder l'estimateur stratifié $\widehat{p}_n$. L'idée est d'obtenir le terme avec la somme de la fonction h6 sous forme de matrice comme ceci :

$$\begin{bmatrix} u_1 \\ \vdots \\ u_d \end{bmatrix} \begin{bmatrix} \mathbf{Y}_1^{(l)} & \mathbf{Y}_2^{(l)} & \dots & \mathbf{Y}_{n/L}^{(l)} \end{bmatrix} = \begin{bmatrix} u_1\mathbf{Y}_1^{(l)} & \dots & \dots & \dots & u_1\mathbf{Y}_{n/L}^{(l)} \\ u_2\mathbf{Y}_1^{(l)} & & & & \vdots \\ \vdots & & & & \vdots \\ u_d\mathbf{Y}_1^{(l)} & \dots & \dots & \dots & u_d\mathbf{Y}_{n/L}^{(l)} \end{bmatrix}$$

Puis on additionne chaque terme de cette matrice par une matrice dont chaque terme est la réalisation d'une loi $N(0,1)$ et par le même principe on détermine le terme $< \mathbf{Z}, u > u_i$.

Concernant l'estimation de la variance de la méthode, comme les $\mathbf{Y}^{(l)}$ sont indépendants (entre les strates et à l'intérieur de celles-ci) et identiquement distribuées à l'intérieur d'une strate, nous avons en reprenant les notations du cours :

$\widehat{\sigma}_n^2(q_1,\dots,q_K) = \sum_{l=1}^{L} \frac{p_l^2}{q_l} \frac{1}{n_l-1} \sum_{i=1}^{n_l} \left\{ h\left(\mathbf{Y}_i^{(l)}\right) - \frac{1}{n_l} \sum_{j=1}^{n_l} h\left(\mathbf{Y}_j^{(l)}\right) \right\}^2$

Le terme $\frac{1}{n_l-1} \sum_{i=1}^{n_l} \left\{ h\left(\mathbf{Y}_i^{(l)}\right) - \frac{1}{n_l} \sum_{j=1}^{n_l} h\left(\mathbf{Y}_j^{(l)}\right) \right\}^2$ est facilement déterminé en utilisant la fonction *var* au vecteur $Y^{(l)}$ et le rapport $\frac{p_l^2}{q_l}$ vaut toujours $\frac{1}{L}$ ce qui revient à stocker les variances empiriques de chaque $Y^{(l)}$ dans un vecteur puis de faire la moyenne en utilisant *mean*.

On débouche ainsi au code final :

```
p_hat_strat<-function(n){
  P<-matrix(numeric(n), nrow=L)
  u<-rep(1,d)
  u<-u/sqrt(sum(u**2))
  S<-0
  Variance<-numeric(L)
  for(i in 1:L){
    Y_l<-rgen_Y((n/L),i) #matrice de taille n/L
    Matrice<-u%*%t(Y_l) #Matrice de taille d*n/L
    Z<-matrix(rnorm(d*n/L), nrow=d) #Matrice de taille d*n/L
    Z_scalaire_u<-t(Z)%*%u #Vecteur de taille n/L
    Matrice<-Matrice + Z - u%*%t(Z_scalaire_u)
    Matrice<-exp(Sd*Matrice)
    Moyenne<-colMeans(Matrice) - K #Vecteur de taille n/L
    Vecteur<-pmax(Moyenne, 0)
    S<-S+mean(Vecteur)/L
    #On ajoute la variance de Y(l) à notre vecteur
    Variance[i]<-var(Vecteur)
  }
  return(data.frame(value=S, variance=mean(Variance)))
}
```

On obtient le résultat suivant :

```
##        value    variance
```

```
## 1 0.05957854 0.002309128
```

Question 11

Le temps de notre algorithme est de :

```
## Unit: milliseconds
##            expr      min       lq     mean   median       uq      max neval
##  p_hat_strat(N) 177.0182 181.383 185.3681 184.0143 187.8943 237.2933   100
```

Avec un rapport de coût de calcul de :

```
tme = microbenchmark(p_hat(N), p_hat_strat(N))
C1<-mean(tme$time[tme$expr=="p_hat(N)"]) #Cout de calcul pour p_hat
C2<-mean(tme$time[tme$expr=="p_hat_strat(N)"]) #Cout de calcul pour p_hat_strat
```

```
## [1] 0.7052405
```

Et ainsi une efficacité relative de :

```
## [1] 3.796604
```

Le coût de calcul de l'estimateur stratifié est plus faible que l'estimateur de Monte Carlo classique. De plus, on a un rapport des variances de environ autour de 5 c'est pourquoi l'efficacite relative est plus grande que 1. On en conclut que la méthode de stratification est plus efficace que la méthode de Monte Carlo classique.

Bilan : les méthodes qui ont été présenté dans ce projet sont de bonnes techniques de réduction de variance et sont à priori très efficace pour estimer diverses quantités. On note en particulier la très grande efficacité relative de la méthode d'échantillonnage préférentiel qui était environ de l'ordre de 10^4 ce qui est "énorme" comparé aux autres efficacités relatives que nous avons pu rencontrer dans le second exercice.