Cours 5 : Méthode de Monte Carlo

A. Righi

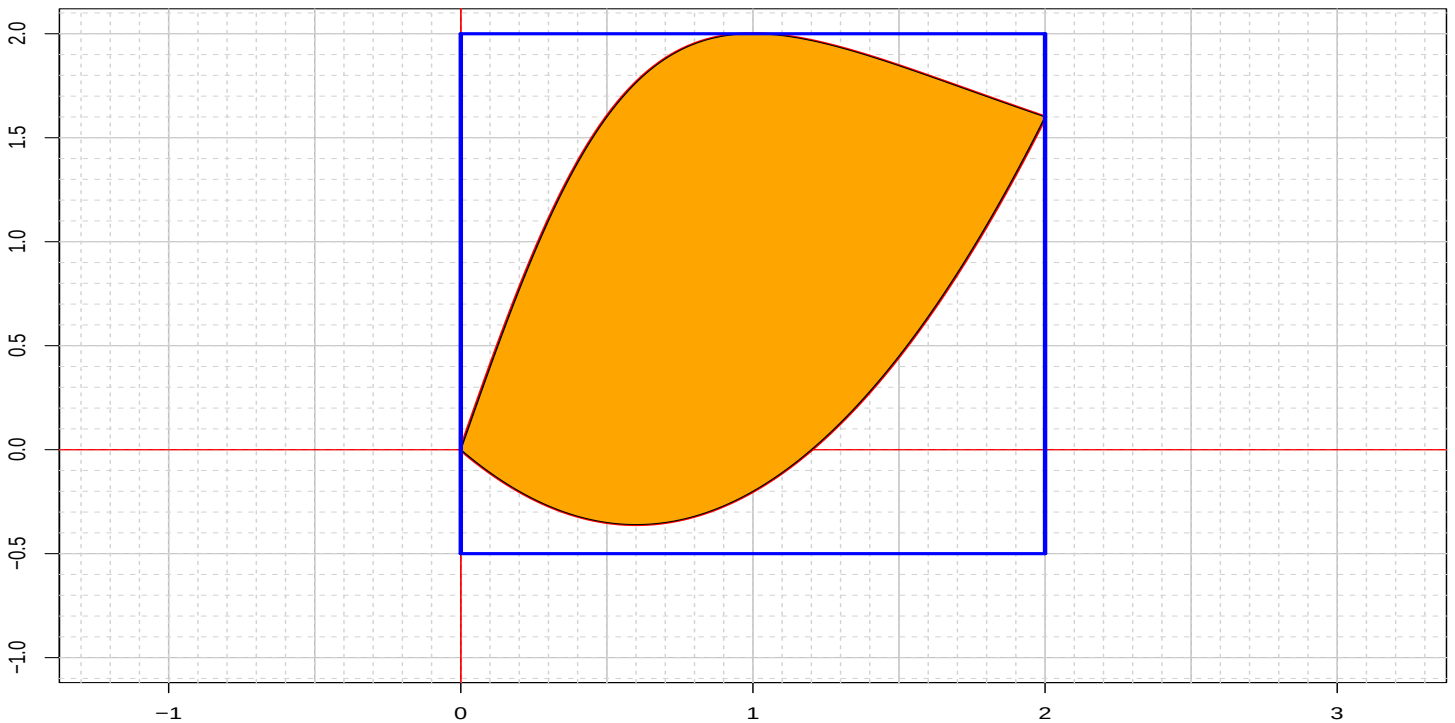
2ème année SC SIW et ISI
Ecole supérieure en informatique, Sidi Bel-Abbès

Année Universitaire 2019-2020

Plan

Vers Monte Carlo

Soit la surface délimitée par les courbes des deux fonctions :
$f(x) = \frac{4x}{x^2+1}$ et $g(x) = x(x-1.2)$.



Simuler la loi uniforme sur la surface colorée, et déduire une méthode pour estimer sa superficie.

Loi des grands nombres

Loi des grands nombres

Soient $X_1, X_2, \ldots, X_n, \ldots$ une suite de variables aléatoires i.i.d telles que $E(X_n) = \mu$. Alors

$$\overline{X}_n = \frac{1}{n}(X_1 + X_2 + \ldots + X_n) \longrightarrow \mu. \qquad \text{quand } n \to \infty.$$

Autrement dit, selon la loi forte des grands nombres, la moyenne empirique est un estimateur convergent de l'espérance.

Corollaire : La fréquence empirique converge vers la probabibilité

Soit A un évenement. Soient $X_1, X_2, \ldots, X_n, \ldots$ une suite i.i.d.

Soit $p = P(X_i \in A)$.

Soit $Y_i = \mathbb{1}_{X_i \in A}$. On a $E(Y_i) = P(X_i \in A) = p$.

Par la loi des grands nombres

$$\overline{Y}_n = \frac{1}{n}(Y_1 + Y_2 + \ldots + Y_n) \longrightarrow E(Y_1), \qquad \text{quand } n \longrightarrow \infty.$$

Donc

Fréquence empirique : probabilité

$$\frac{1}{n}(\mathbb{1}_{X_1 \in A} + \mathbb{1}_{X_2 \in A} + \ldots + \mathbb{1}_{X_n \in A}) \longrightarrow p, \qquad \text{quand } n \longrightarrow \infty.$$

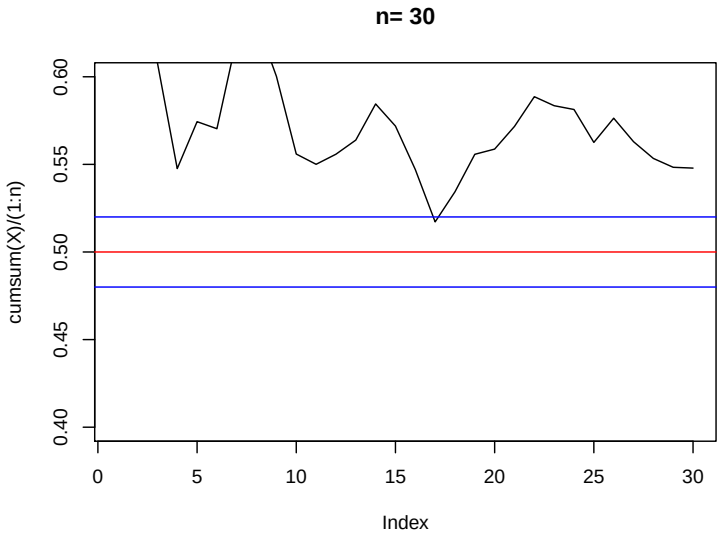
Autrment dit :

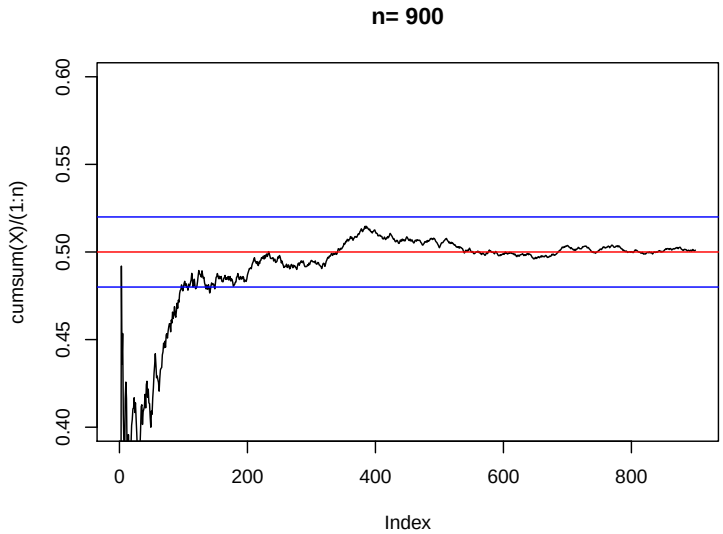
$$\frac{1}{n}\text{Card}\{i \in \{1, 2, \ldots, n\} | X_i(w) \in A\} \longrightarrow p.$$

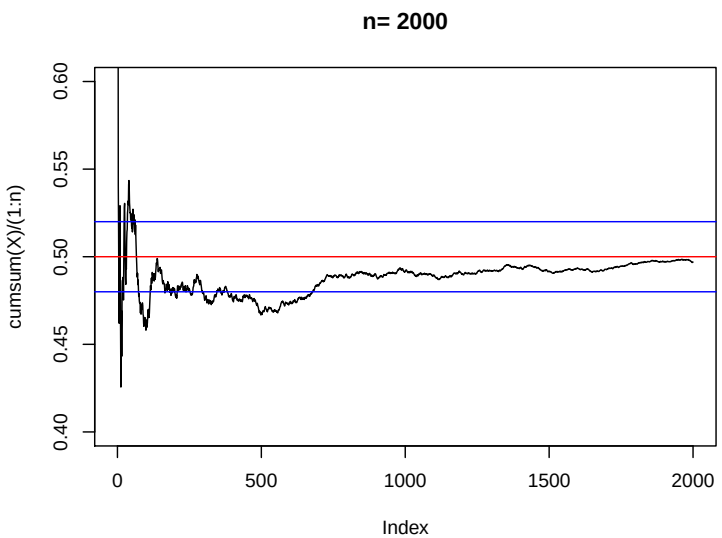
Illustration de la loi des grands nombres pour la loi uniforme

```
par(mfrow=c(2,2))
for(n in c(30,900,2000,45000))
{
X=runif(n)
plot(cumsum(X)/(1:n),type="l",ylim=c(.4,.6),
main=paste("n=",as.character(n)))
abline(h=0.5,col="red")
abline(h=c(0.48,0.52),col="blue")
}
par(mfrow=c(1,1))
```

Illustration de la loi des grands nombres pour la loi uniforme







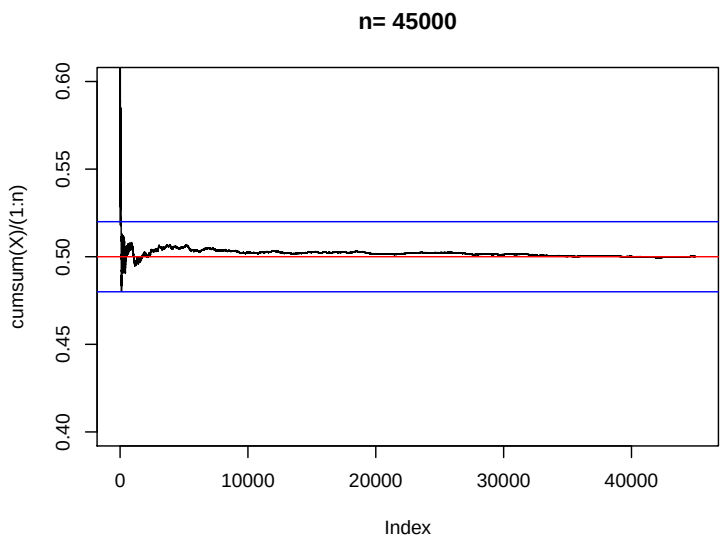
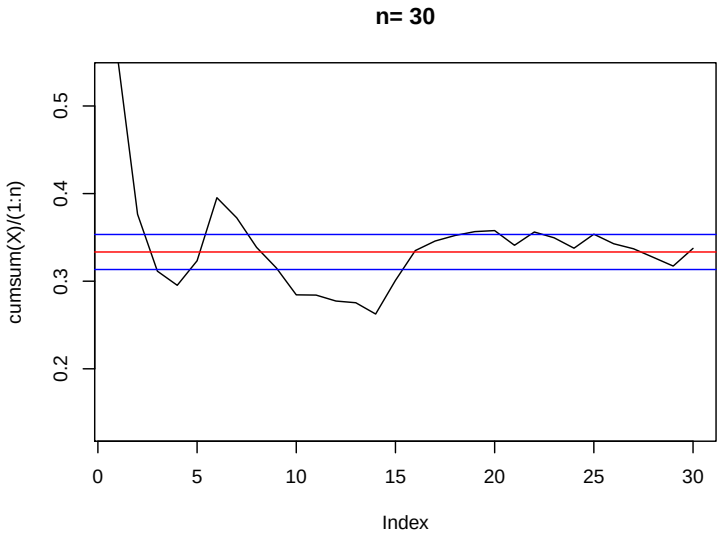
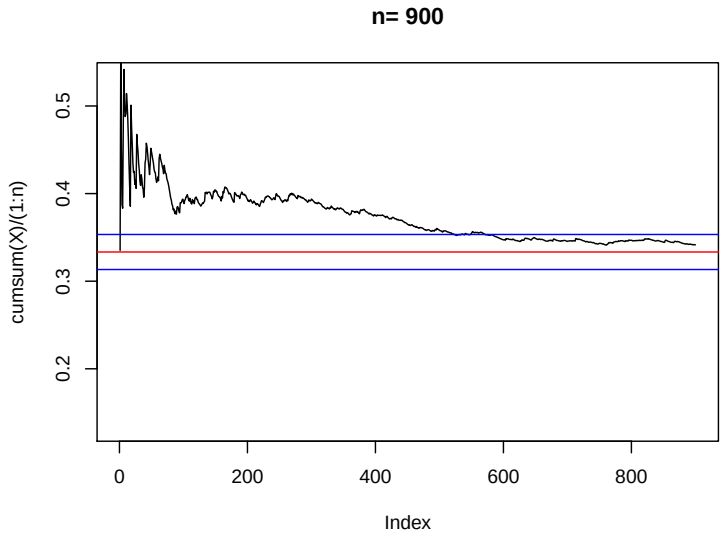


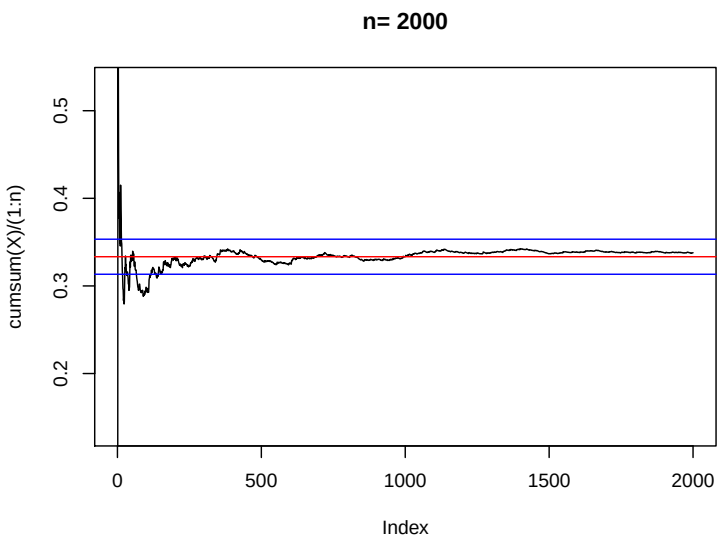
Illustration de la loi des grands nombres pour la loi exponentielle

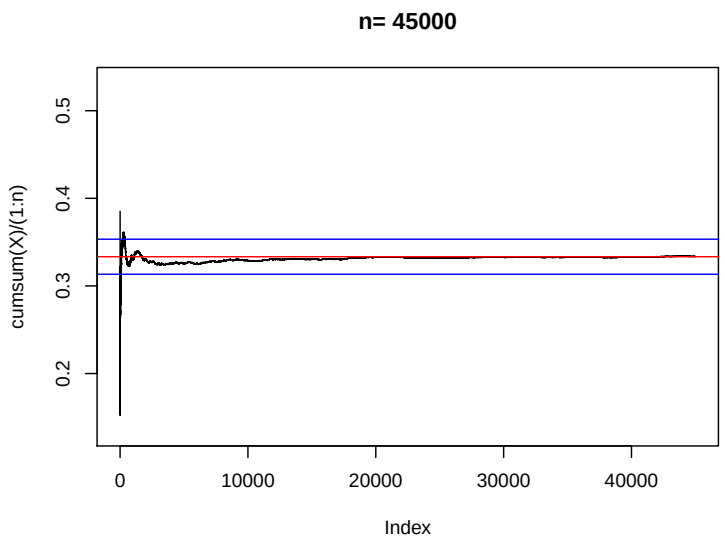
```
par(mfrow=c(2,2))
for(n in c(30,900,2000,45000))
{
X=rexp(n,3)
plot(cumsum(X)/(1:n),type="l",ylim=c(1/3-.2,
1/3+.2),
main=paste("n=",as.character(n)))
abline(h=1/3,col="red")
abline(h=c(1/3-0.02,1/3+0.02),col="blue")
}
par(mfrow=c(1,1))
```

Illustration de la loi des grands nombres pour la loi exponentielle









Importance

- Elle permet d'interpréter la probabilité comme fréquence de réalisation.
- Elle justifie l'estimation en statistique inférentielle.
- Elle justifie en particulier les techniques de sondage aléatoire.
- Elle est cruciale pour les compagnies d'assurances.
- Elle justifie les méthodes de simulations (Monte Carlo)

Mais remarquons qu'elle ne donne pas une vitesse de convergence.

Théorème central limite

Les lois de probabilité gaussiennes apparaissent comme des lois limite dans des situations où on additionne des v.a.r. indépendantes de carrés intégrables et de variance petite devant celle de la somme.

Théorème central limite

Soit (X_n) une suite de v.a.r. i.i.d. de carrés intégrables. d'espérance commune μ et de variance commune σ^2. Alors

$$\frac{S_n - n\mu}{\sqrt{n\sigma^2}} \to \mathcal{N}(0,1)$$

ou de manière équivalente

$$\frac{\bar{X}_n - \mu}{\sqrt{\frac{\sigma^2}{n}}} \to \mathcal{N}(0,1)$$

Pratiquement, ce théorème nous dit comment se comporte la variable aléatoire $\bar{X}_n$, à n fixé et grand :

$$\bar{X}_n \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right)$$

Illustration du théorème central limite : Loi binomiale

```
n=300
nb=1e5
Xbar=numeric(nb)
for(k in 1:nb){
X=rbinom(n,10,0.5)
Xbar[k]=mean(X)
}
hist(Xbar,freq=FALSE,nclass=70,col="gray")
```

Illustration du théorème central limite : Loi binomiale

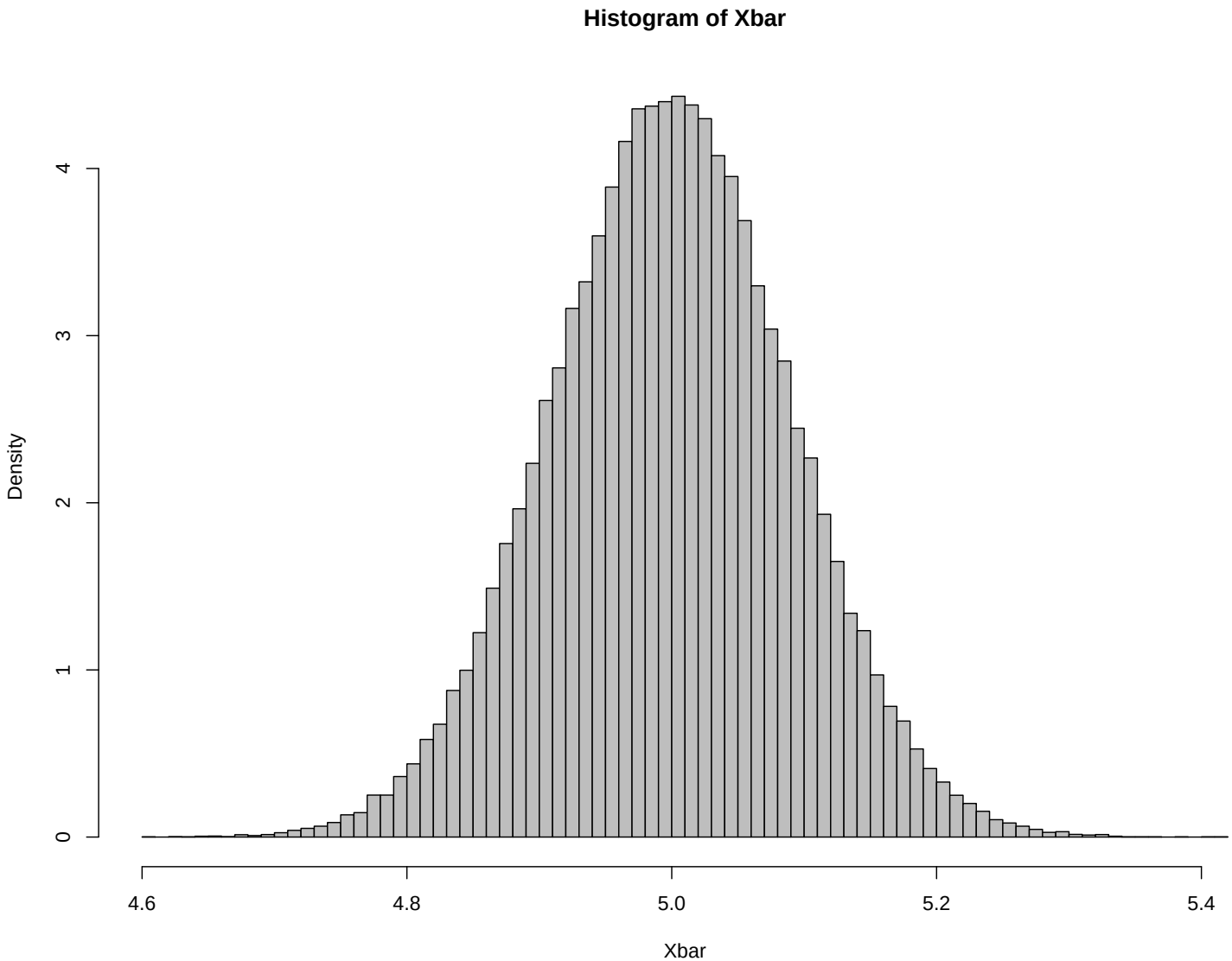
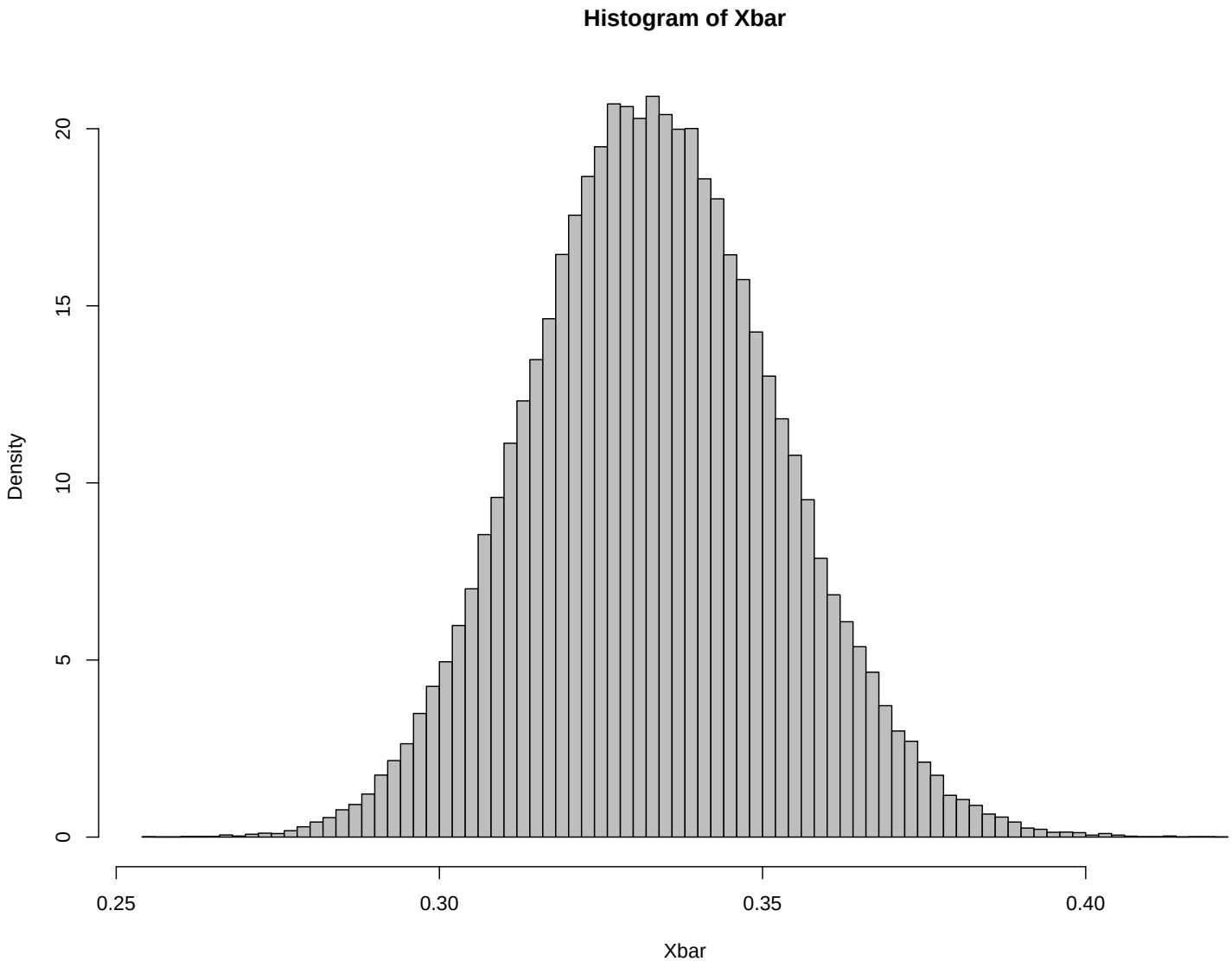


Illustration du théorème central limite : Loi exponentielle

```
n=300
nb=1e5
Xbar=numeric(nb)
for(k in 1:nb){
X=rexp(n,3)
Xbar[k]=mean(X)
}
hist(Xbar,freq=FALSE,nclass=70,col="gray")
```

Illustration du théorème central limite : Loi exponentielle



Méthodes de Monte-Carlo

Simulation aléatoire : Monte Carlo

- De manière générale, la simulation permet d'étudier et expérimenter un système donné dont on connaît les interactions complexes, de mesurer les effets de certains changements dans les interactions sur le comportement du système, d'expérimenter de nouvelle situations.
- Lorsque dans la simulation intervient un élément aléatoire, on parle de simulation aléatoire.

Simulation aléatoire : Monte Carlo

Les exemples d'application sont très variés, citons par exemple :

[•] la simulation de files d'attente, de réseaux,

[•] la simulation de portefeuilles d'actifs en finance,

[•] la comparaison d'estimateurs en statistique,

[•] la recherche d'état stationnaire en physique, en économie.

Simulation aléatoire : Monte Carlo

- Remarquons de plus que si l'on cherche une représentation fidèle des phénomènes observés, on est rapidement confronté à des difficultés dues aux calculs non explicites.
- Les techniques de simulation vont nous permettre d'approcher numériquement ces calculs.

Principe

les méthodes de Monte-Carlo qui ont pour essence l'utilisation d'expériences répétées pour évaluer une quantité, résoudre un système déterministe.

Ces méthodes peuvent servir pour :

- le calcul d'intégrale,
- la résolution d'équations aux dérivées partielles,
- la résolution de système linéaire,
- la résolution de problèmes d'optimisation (algorithme du recuit simulé).

Avantage

- domaine d'application très vaste.
- peu d'hypothèses de mise en oeuvre.
- facile à implémenter

Intégration par Monte-Carlo

La méthode de Monte-Carlo consiste à écrire l'intégrale

Intégration par Monte-Carlo

La méthode de Monte-Carlo consiste à écrire l'intégrale

$$I = \int_0^1 g(x)dx$$

Intégration par Monte-Carlo

La méthode de Monte-Carlo consiste à écrire l'intégrale

$$I = \int_0^1 g(x)dx$$

sous la forme

Intégration par Monte-Carlo

La méthode de Monte-Carlo consiste à écrire l'intégrale

$$I = \int_0^1 g(x)dx$$

sous la forme

$$I = E[g(\mathbf{U})]$$

où $\mathbf{U}$ est une variable suivant une loi uniforme sur $[0, 1]$

Intégration par Monte-Carlo

Si $(\mathbf{U}_i)_{i\in\mathbf{N}}$ est une suite de variables aléatoires indépendantes et de loi uniforme sur [0,1], On utilise la Loi des grands nombres

$$\frac{1}{n}\sum_{i=1}^{n} g(\mathbf{U}_i) \to E[g(\mathbf{U})] \qquad p.s$$

si $u_1, u_2, \ldots, u_n$ sont des nombres tirés au hasard dans [0,1].

Intégration par Monte-Carlo

Si $(\mathbf{U}_i)_{i\in\mathbf{N}}$ est une suite de variables aléatoires indépendantes et de loi uniforme sur [0,1], On utilise la Loi des grands nombres

$$\frac{1}{n}\sum_{i=1}^{n} g(\mathbf{U}_i) \to E[g(\mathbf{U})] \qquad p.s$$

si $u_1, u_2, \ldots, u_n$ sont des nombres tirés au hasard dans [0,1].

$$\frac{1}{n}(g(u_1)+g(u_2)+\cdots+g(u_n))$$

est une approximation de $\int_0^1 g(x)dx.$

Soit

$$I = \int_0^1 cos(x)dx.$$

On connaît la valeur de cette intégrale par le calcul des primitives.

$$I = \sin(1) - \sin(0) = \sin(1).$$

Nous allons essayer l'intégration par Monte Carlo sur cet exemple simple.

Exemple : $\int_0^1 \cos(x)$

```
g=function(x) cos(x)
integrate(g,0,1)$value
sin(1) ## 0.841471
## Monte Carlo
n=1e3
s=0 ;
for( i in 1:n){
u=runif(1)
s=s+g(u)
}
s/n
```

Trois executions de ce programme a donné trois résultats :

0.8401454 ; 0.8404826 ; 0.8458725

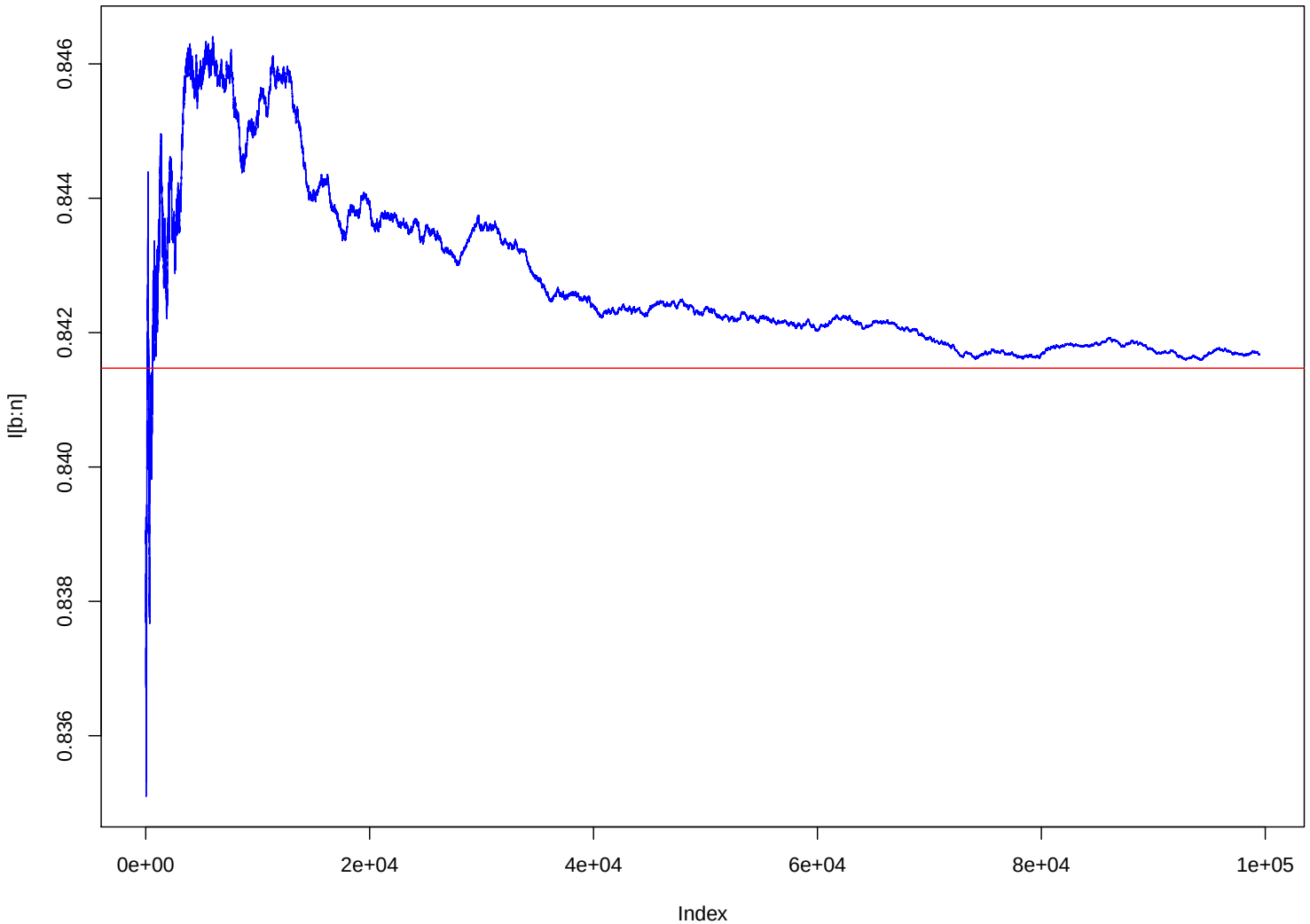
Exemple : sans boucle

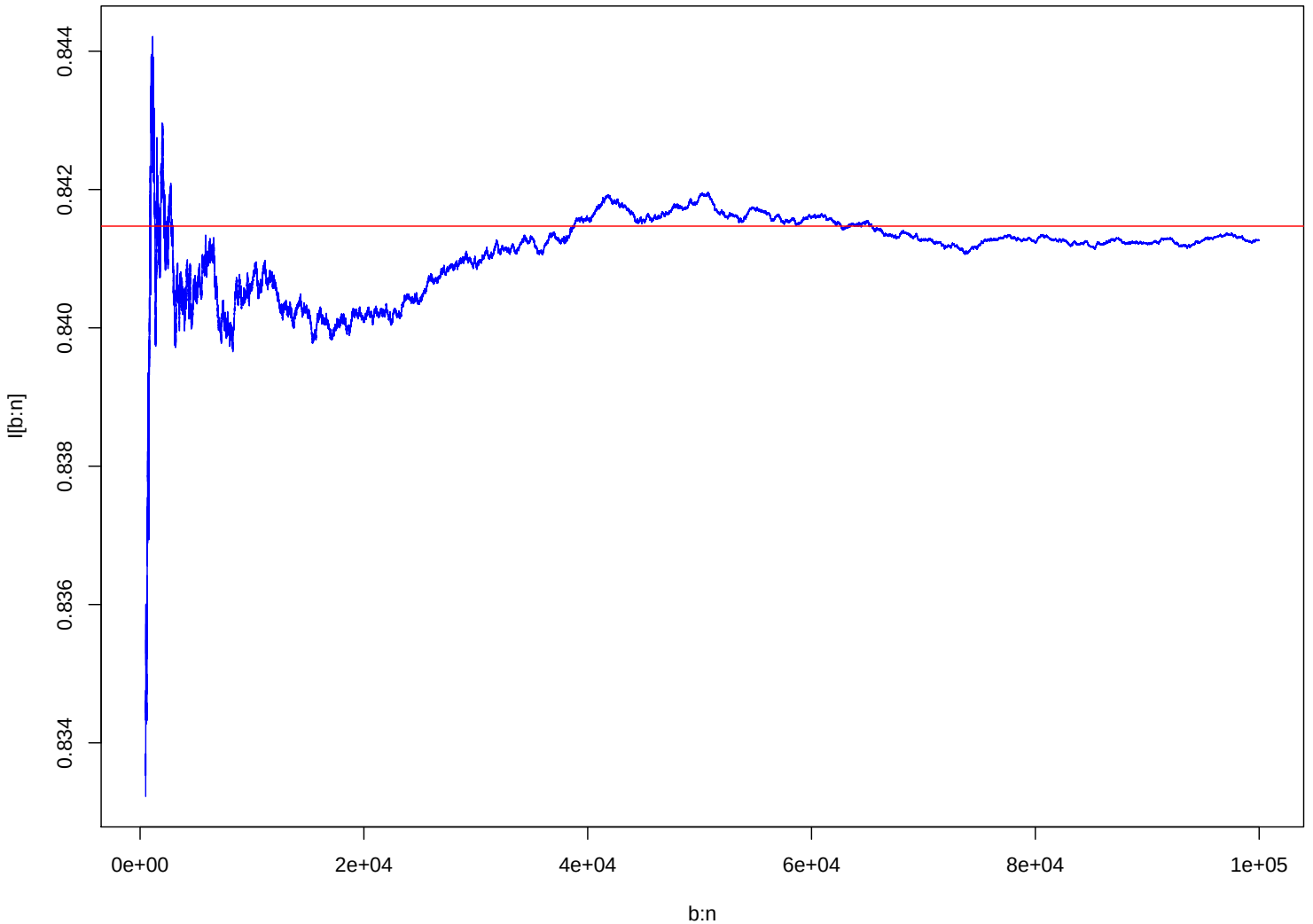
Programme

```
1 g=function(x) cos(x)
2 n=1e5
3 u=runif(n)
4 x=g(u)
5 mean(x)
```

Illustration de la Convergence

```
g=function(x) cos(x)
n=1e6
u=runif(n)
x=g(u)
mean(x)
l=(cumsum(x)/1:n)
b=500
plot(b:n,l[b:n], type="l",col="blue")
abline(h=sin(1),col="red")
```


I[b:n]
0.836
0.838
0.840
0.842
0.844
0.846
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
Index


I[b:n]
0.834
0.836
0.838
0.840
0.842
0.844
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
b:n

Intégration par Monte-Carlo

Prenons l'exemple plus général d'une intégrale du type

$$I = \int_{\mathbb{R}^d} g(x) f(x) dx$$

où $f(x) \geq 0$ et $\int_{\mathbb{R}^d} f(x)dx = 1$

Alors X est une v.a de densité f par rapport à la mesure de Lebesgue sur $\mathbb{R}^d$.

Toujours par la loi des grands nombres, si $(X_i)_{i \in N}$ est une suite de variables indépendantes sur $\mathbb{R}^d$ de loi de densité f.

$$\frac{1}{n} \sum_{i=1}^{n} g(X_i) \longrightarrow E[g(X)] \qquad p.s$$

Intégration par Monte-Carlo

Si $(x_1, x_2, \cdots, x_n)$ est une réalisation de $(X_1, \cdots, X_n)$

alors

$$\frac{1}{n}(g(x_1) + g(x_2) + \cdots + g(x_n))$$

sera une approximation de $\mathbf{I} = \int_{\mathbb{R}^d} g(x) f(x) dx$

Exemple

On veut approximer l'intégrale suivante par Monte Carlo :

$$I = \int\int \frac{1}{2\pi}\cos(x^2+y^2)e^{-\frac{x^2+y^2}{2}}dxdy$$

$f(x,y) = \frac{1}{2\pi}e^{-\frac{x^2+y^2}{2}}$.
$g(x,y) = \cos(x^2+y^2)$

$$f(x,y) = f_X(x)f_Y(y) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}\frac{1}{\sqrt{2\pi}}e^{-\frac{y^2}{2}}$$

Programme

```
g=function(x1,x2) cos(x1^2+x2^2)
n=1e5
X1=rnorm(n)
X2=rnorm(n)
Z=g(X1,X2)
mean(Z)
```

Vitesse de convergence : Monte-Carlo

Quel est la vitesse de convergence dans la méthode de Monte-Carlo ?

TCL

D'après le Théorème centrale limite , si $\sigma^2 = Var[g(X_1)]$:

$$\frac{\sqrt{n}}{\sigma}\left[\frac{1}{n}\sum_{i=1}^{n} g(X_i) - \mathbb{E}(g(X_1))\right] \text{converge en loi vers G}$$

où G est une variable aléatoire gaussienne centrée de variance 1.

Intervalle de confiance : Monte-Carlo

Par conséquent pour n suffisamment grand, l'écart(l'erreur)
$[\epsilon_n = \frac{1}{n}\sum_{i=1}^{n} g(X_i) - E[g(X_1)]$ satisfait

$$\mathbf{P}(|\frac{\sqrt{n}\epsilon_n}{\sigma}| \leq 1,96) \sim \mathbf{P}(|G| \leq 1,96) = 0,95$$

et donc $|\epsilon_n|$ est une quasi-certitude de 95 inférieur à $\sigma\frac{1,96}{\sqrt{n}}$.
Par suite un intervalle de confiance pour $E[g(X_1)]$ au niveau $0,95$ est

$$\left[\frac{1}{n}\sum_{i=1}^{n} g(X_i) - \sigma\frac{1,96}{\sqrt{n}} \quad , \quad \frac{1}{n}\sum_{i=1}^{n} g(X_i) + \sigma\frac{1,96}{\sqrt{n}}\right]$$

Intervalle de confiance : Monte-Carlo

En général σ ne sera pas calculable et on l'approximera par une méthode de Monte-Carlo puisque

$$\frac{1}{n}\sum_{i=1}^{n} g(X_i)^2 - (\frac{1}{n}\sum_{i=1}^{n} g(X_i))^2 \longrightarrow \sigma^2 \qquad p.s.$$

Ce calcul sera mené en même temps que celui de l'espérance.

Intervalle de confiance

$$\left[\frac{1}{n}\sum_{i=1}^{n} g(X_i) - \hat{\sigma}\frac{1,96}{\sqrt{n}} \quad , \quad \frac{1}{n}\sum_{i=1}^{n} g(X_i) + \hat{\sigma}\frac{1,96}{\sqrt{n}}\right]$$

Exemple : Intervalle de confiance

Programme

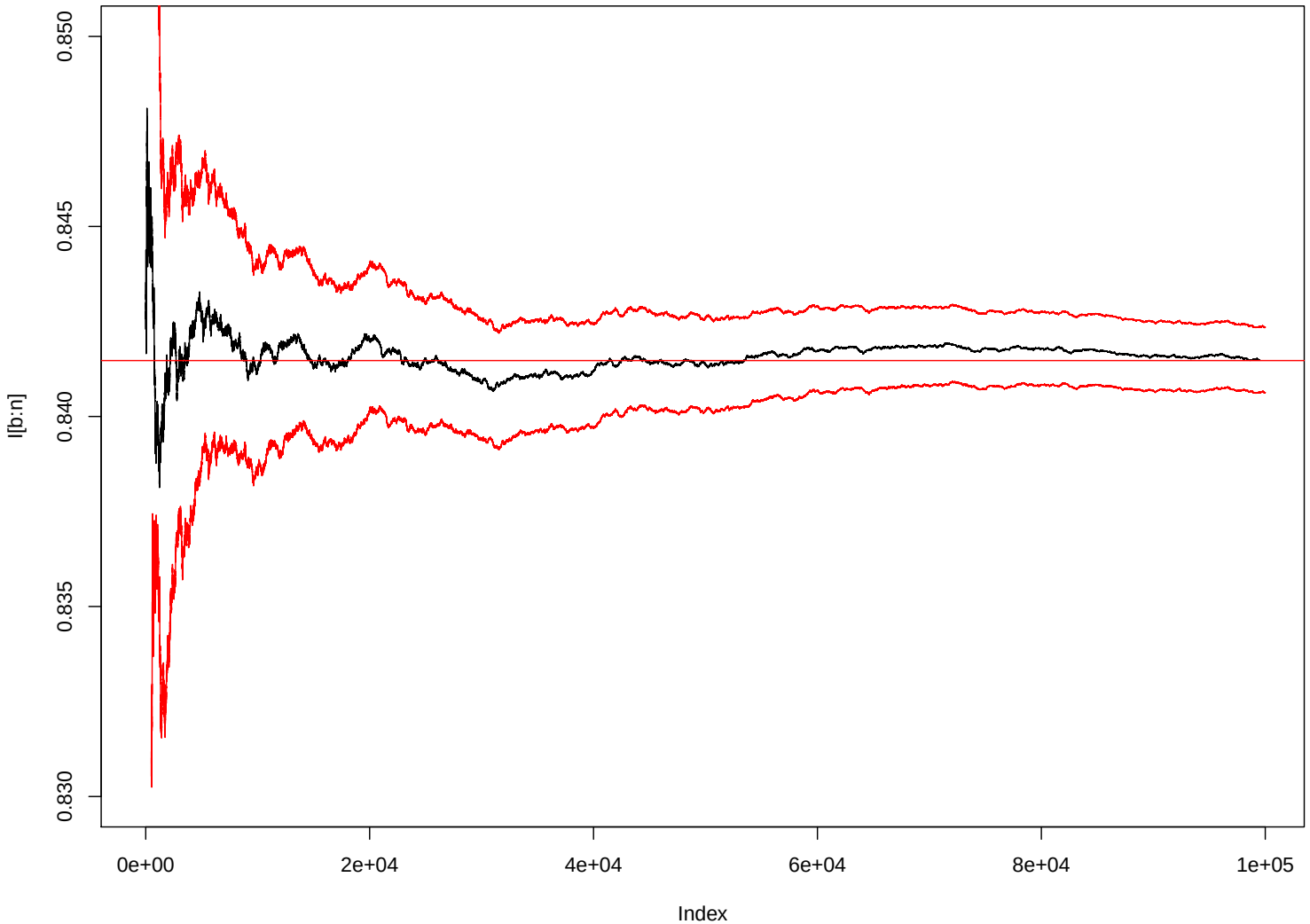
```
g=function(x) cos(x)
n=1e4 ; u=runif(n) ; x=g(u)
sigma=sd(x)
Ichap=mean(x)
binf=Ichap -1.96*sigma/sqrt(n)
bsup=Ichap+1.96*sigma/sqrt(n)
Ichap ;binf ;bsup
```

Résultat

```
>Ichap
[1] 0.8411504
> binf
[1] 0.8384422
> bsup
[1] 0.8438586
```

Illustration de la convergnce par l'intervalle de confiance

```
g=function(x) cos(x)
n=1e5
u=runif(n)
x=g(u)
I=(cumsum(x)/1:n)
sd=sqrt(cumsum(x^2)/(1:n)-I^2)
binf=I-sd*1.96/sqrt(1:n)
bsup=I+sd*1.96/sqrt(1:n)
b=500
plot(I[b:n], type="l",col="black",ylim=c(0.83,0.85))
points(b:n,binf[b:n],type="l",col="red")
points(b:n,bsup[b:n],type="l",col="red")
abline(h=sin(1),col="red")
```


I[b:n]
0.850
0.845
0.840
0.835
0.830
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
Index

Monte Carlo et les grandes dimensions

Soit

$$I = \int_0^1 \cdots\cdots \int_0^1 \cos\left(x_1^2 + x_2^2 + \ldots + x_m^2\right) dx_1 dx_2 \ldots dx_m$$

```
g=function(x) cos(2*sum(x^2))
nb=1e6 ; m=10
z=numeric(nb)
for(i in 1:nb){ x=runif(m) ; z[i]=g(x) }
(mean(z)) ; (sd(z))
(binf=mean(z)-sd(z)*1.96/sqrt(nb))
(bsup=mean(z)+sd(z)*1.96/sqrt(nb))
```

Avantages de Monte-Carlo

La vitesse de convergence est donc de l'ordre de $\frac{1}{\sqrt{n}}$.

Avantages de Monte-Carlo

La vitesse de convergence est donc de l'ordre de $\frac{1}{\sqrt{n}}$.

Avantages

Cette vitesse peut paraître faible en petite dimension, mais présente l'avantage

Avantages de Monte-Carlo

La vitesse de convergence est donc de l'ordre de $\frac{1}{\sqrt{n}}$.

Avantages

Cette vitesse peut paraître faible en petite dimension, mais présente l'avantage

- d'être insensible à la dimension

Avantages de Monte-Carlo

La vitesse de convergence est donc de l'ordre de $\frac{1}{\sqrt{n}}$.

Avantages

Cette vitesse peut paraître faible en petite dimension, mais présente l'avantage

- d'être insensible à la dimension
- de ne pas dépendre de la régularité de la fonction g à intégrer, pourvu que $g(X_1)$ soit de carré intégrable.

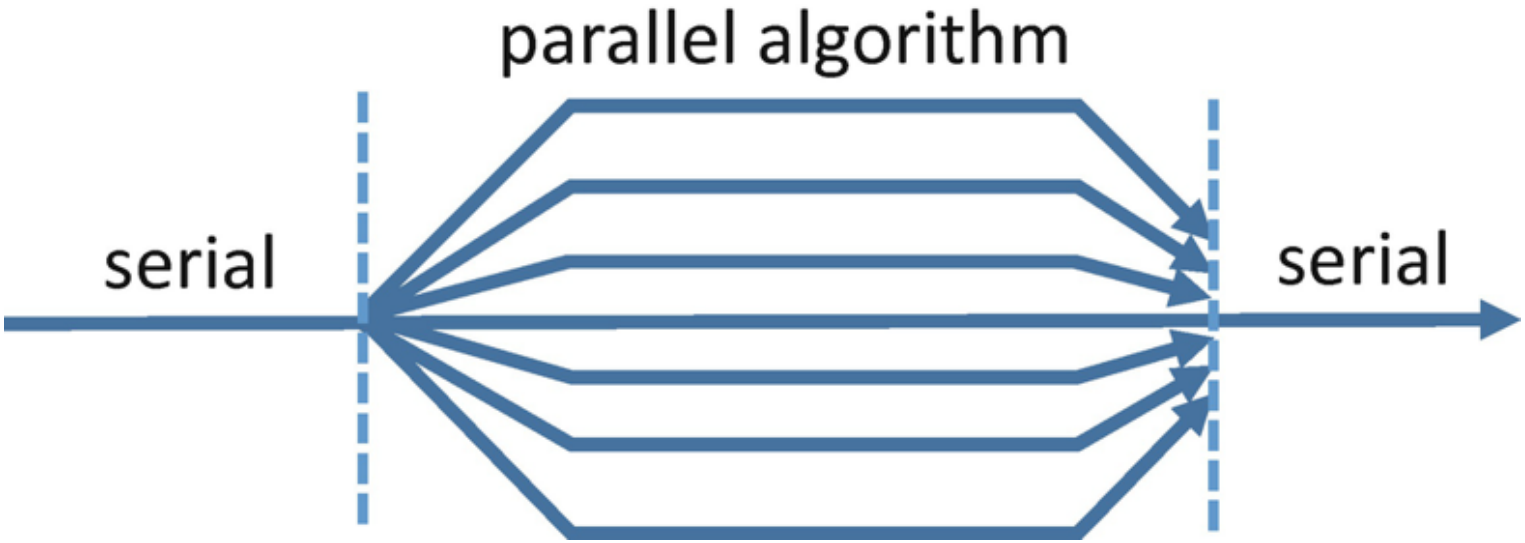
n peut être très grand !

- Notons que la possibilité de produire une quantité presque infinie de variables aléatoires distribuées selon une distribution donnée ouvre l'accès à l'utilisation de résultats fréquentistes et asymptotiques beaucoup plus facilement que dans le cas de l'inférence usuelle, où la taille de l'échantillon est souvent fixée.
- On peut donc appliquer la Loi des Grands Nombres ou le Théorème de la Limite Centrale, puisqu'ils autorisent tous deux une évaluation de la convergence des méthodes de simulation (équivalente aux bornes déterministes utilisées par les approches numériques).

Monte Carlo est parallélisable

La méthode de Monte Carlo pour l'intégration est parallélisable. Cela permet de pousser la précision vers des niveaux très élévés.

Exercice



Supposons qu'on veut calculer $I = \int_0^1 g(x)dx$ avec un programme parallèle.

Nous faisons K calculs en parallèles.

Chaque processus i utilise n_i variables.

Soit m_i, σ_i les résultats partiels de chaque processus.

Donner l'intervalle de confiance que donnera le programme prinicpal.

Précision de la méthode

La précision de la méthode est donnée par :

$$1.96\frac{\sigma}{\sqrt{n}}$$

Nous pouvons augmenter la précision :

- En diminuant la constante 1.96 ; Mais c'est liée à la confiance. Diminuer la constante 1.96 diminue la confiance de l'intervalle.
- En augmantant le n autant que possible et nécessaire. ça dépend de la capacité en matériel, mémoire. Mais ça va augmenter le temps de calcul de la méthode.

Supposons que le niveau de confiance est fixé, et on a n fixé.
La précision d'une méthode est proportionnel à l'écart-type(racine de la variance) de la méthode σ.
Peut-on améliorer la variance ?

Techniques de réduction de variance

Comment réduire la variance σ^2 ?

Il y a plusieurs techniques de réduction de variance

- Échantillonnage préférentiel ou fonction d'importance
- Variable de contrôle
- Variables antithétiques
- Méthode de stratification

Réduction de variance : Introduction

L'objectif est d'estimer une intégrale I par méthodes de Monte-Carlo de la façon la plus précise possible. Soit $I = \mathbb{E}(Z)$, avec Z de carré intégrable. Lorsque l'on considère deux estimateurs par méthodes de Monte-Carlo de I, pour une même taille d'échantillon, l'estimateur le plus précis est celui dont la variance est la plus petite. Plus précisément, supposons que

$$I = \mathbb{E}(X) = \mathbb{E}(Y)$$

avec X et Y deux variable aléatoire de carré intégrable. Considérons $X_1, ..., X_n$ des variables aléatoires i.i.d. de même loi que X ainsi que $Y_1, ..., Y_n$ des variables aléatoires i.i.d. de même loi que Y. On peut alors estimer I par

$$\widehat{I_n} = \frac{1}{n}\sum_{i=1}^{n} X_i \quad \text{et par} \quad \widehat{\theta_n} = \frac{1}{n}\sum_{i=1}^{n} Y_i.$$

D'après le cours précédent, pour des échantillons de même taille n, l'estimateur le plus précis est celui dont la variance est la plus faible. Étant donné que

$$Var(\widehat{I_n}) = \frac{Var(X)}{\sqrt{n}} \quad \text{et} \quad Var(\widehat{\theta_n}) = \frac{Var(Y)}{\sqrt{n}},$$

comparer les variances de ces estimateurs revient à comparer les variances de X et de Y. Lorsque l'on utilise une méthode de Monte-Carlo, on a donc intérêt à écrire I sous la forme $I = \mathbb{E}(Z)$, avec Z de plus petite variance possible.

Attention au temps de calcul

Remarque

Cependant, pour comparer les deux méthodes d'estimation, il faut aussi tenir compte des temps de calcul pour estimer I.
Notons t_1 *(respectivement* t_2*) le temps de calcul pour évaluer* I *par* $\widehat{I_n}$ *(respectivement* $\widehat{\theta_n}$*),*
alors l'estimateur $\widehat{\theta_n}$ *est plus* ***efficace*** *que* $\widehat{I_n}$ *si*

$$\varepsilon = \frac{t_1\, Var(\widehat{I_n})}{t_2\, Var(\widehat{\theta_n})} > 1.$$

Si on ne connaît pas les variances alors on utilise l'estimateur sans biais de la variance, pour définir l'efficacité d'une méthode par rapport à une autre.

Exemple

Soit l'intégrale suivante

$$\int_0^1 cos(x)e^{-2x}.$$

On peut l'écrire par exemple sous deux formes :

Forme1 :

$$I = \int \mathbb{1}_{[0,1]}(x)cos(x)e^{-2x} = \int f_X(x)g(x)dx = E(g(X))$$

où $X \sim \mathcal{U}[0,1]$ et $g(x) = cos(x)e^{-2x}$.

Forme2 :

$$I = \int 2e^{-2x}\,\mathbb{1}_{[0,\infty[}(x)\,\mathbb{1}_{[0,1]}(x)\frac{1}{2}cos(x) = \int f_X(x)g(x)dx = E(g(X))$$

où $X \sim \mathsf{Exp}(2)$ et $g(x) = \mathbb{1}_{[0,1]}(x)\frac{1}{2}cos(x)$.

Méthode 2

```
## Method 2
g=function(x) (1/2)*cos(x)*(x>0)*(x<1)
n=1e5
x=rexp(n,2)
z=g(x)
mean(z)
sd(z)
```

```
> mean(z)
[1] 0.3940008
> sd(z)
[1] 0.2644128
```

Méthode 1

```
## Method 1
g=function(x) exp(-2*x)*cos(x)
n=1e5
x=runif(n)
z=g(x)
mean(z)
sd(z)
```

```
mean(z)
[1] 0.3933105
> sd(z)
[1] 0.1645997
>
```

Temps de calcul

```
## Method 1
g=function(x) exp(-2*x)*cos(x)
start=Sys.time()
n=1e6 ;x=runif(n) ;z=g(x)
t1=as.numeric(Sys.time()-start); v1=var(z)
## Method 2
g=function(x) (1/2)*cos(x)*(x>0)*(x<1)
start=Sys.time()
n=1e6 ; x=rexp(n,2) ;z=g(x)
t2=as.numeric(Sys.time()-start);v2=var(z)
(comp=(t2*v2)/(t1*v1))
```

```
> comp
[1] 0.3767271
```

Échantillonnage préférentiel ou Fonction d'importance

- Le principe de la méthode d'échantillonnage moyen consiste à écrire l'intégrale I sous la forme

$$I = \int_U g(x)dx \quad = \quad \int_U f_X(x)\frac{g(x)}{f_X(x)}dx = \mathbb{E}\left[\frac{g(X)}{f_X(X)}\right],$$

avec X une variable aléatoire de densité f_X et on pose $g(x) = 0, \forall x \notin U$.

- On suppose avoir choisi la densité f_X de sorte que $Z = \dfrac{g(X)}{f_X(X)} \in L^2(\Omega)$.
- Evidement nous avons un degré de liberté : le choix de la fonction f_X.
- Afin de minimiser la variance de l'estimateur $\widehat{\theta_N}$ par échantillonnage moyen, il nous faut trouver une densité f_X de sorte que la variance de Z soit la plus petite possible.

Proposition

La fonction f_X minimisant la variance de Z est définie par

$$f_X(x) = \frac{|g(x)|}{\int_U |g(x)|dx}, x \in \mathbb{R}^d.$$

Démonstration. Nous avons que

$$Var(Z) = \int \frac{g^2(X)}{f_X(x)} dx - I^2.$$

D'après l'inégalité de Jensen,

$$\int \frac{g^2(x)}{f_X(x)} dx = \mathbb{E}(\frac{g^2(X)}{f_X^2(X)}) \geq \left[\mathbb{E}\left(\frac{|g(X)|}{f_X(X)}\right)\right]^2 = \left(\int |g(x)|dx\right)^2.$$

Par conséquent, quel que soit le choix de f_X,

$$Var(Z) \geq \left(\int |g(x)|dx\right)^2 - I^2.$$

Alors, la variance de Z est minimale si l'on choisit la fonction f_X définie par

$$f_X(x) = \frac{|g(x)|}{\int |g(x)|dx}, x \in \mathbb{R}^d.$$

En effet, pour ce choix, $Var(Z) = \left(\int |g(x)|dx\right)^2 - I^2$.

Il est utopique de penser utiliser la densité f_X définie par

$$f_X(x) = \frac{|g(x)|}{\int_U |g(x)|dx}, x \in \mathbb{R}^d,$$

En effet, l'intégrale $I = \int g(x)dx$ n'étant pas connue, il y a peu de chances que l'on connaisse la valeur de l'intégrale $\int |g(x)|dx$.

En particulier, si g est positive, la variance de $Z = g(x)/f_X(x)$ est minimisée par le choix

$$f_X(x) = \frac{g(x)}{\int_U g(x)dx} x \in \mathbb{R}^d,$$

choix qui dépend de la quantité à estimer I.

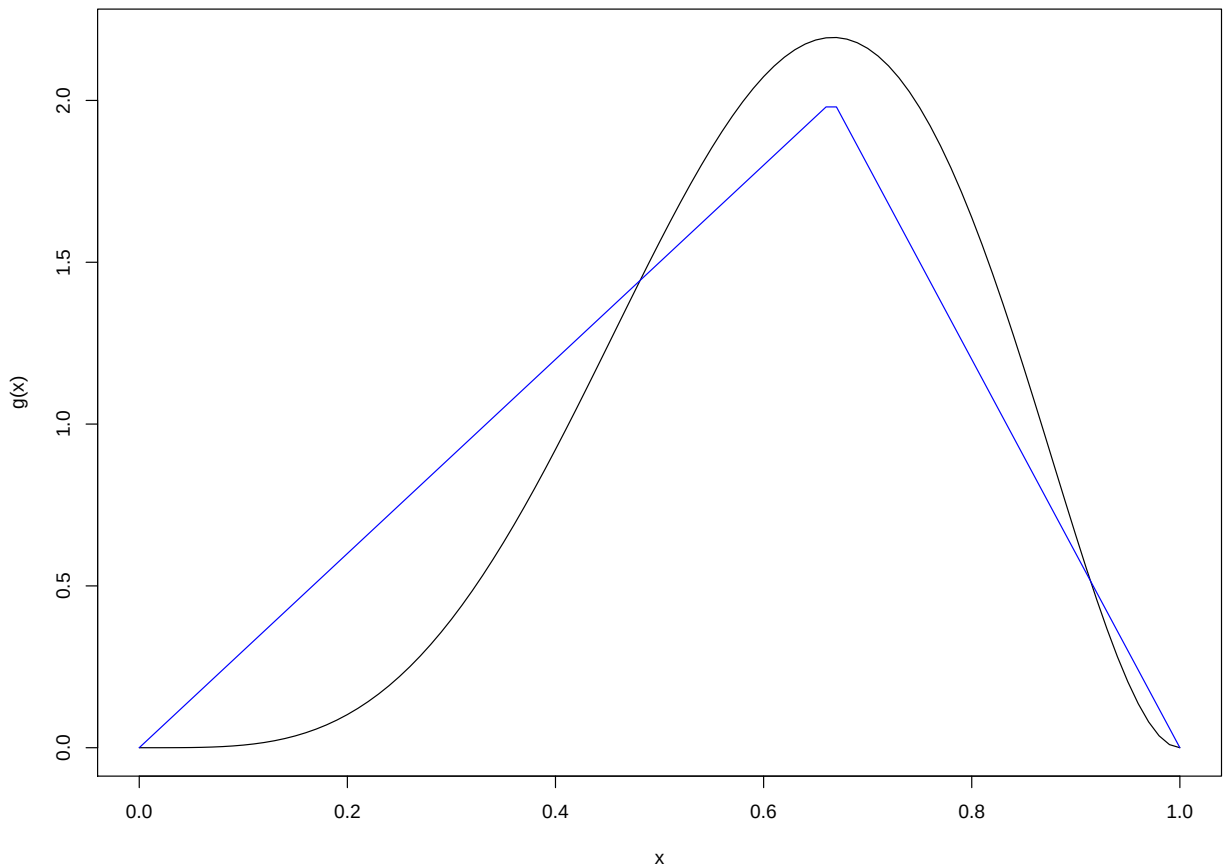
Notez que dans ce cas, $Var(Z) = 0$.

On peut cependant en déduire une méthode générale. En pratique, on cherche à trouver une fonction "simple", polynômiale par exemple, relativement proche de $|g(x)|$ qui sera renormalisée (de façon à obtenir une densité f_X).

Soit

$$I = \int_0^1 x^4 (1-x)^2$$

S'inspirant de la forme de cette fonction, on peut remplacer la loi uniforme standard par la loi triangulaire $\mathcal{T}(a = 0, c = 2/3, b = 1)$.



Elle ressemble à un triangle basé sur le segment $[0;1]$ et culminant en $(2/3;2)$.

Sa densité est

$$f^*(x) = \begin{cases} 3x & x \in [0;2/3] \\ 6(1-x) & x \in [2/3;1] \end{cases}$$

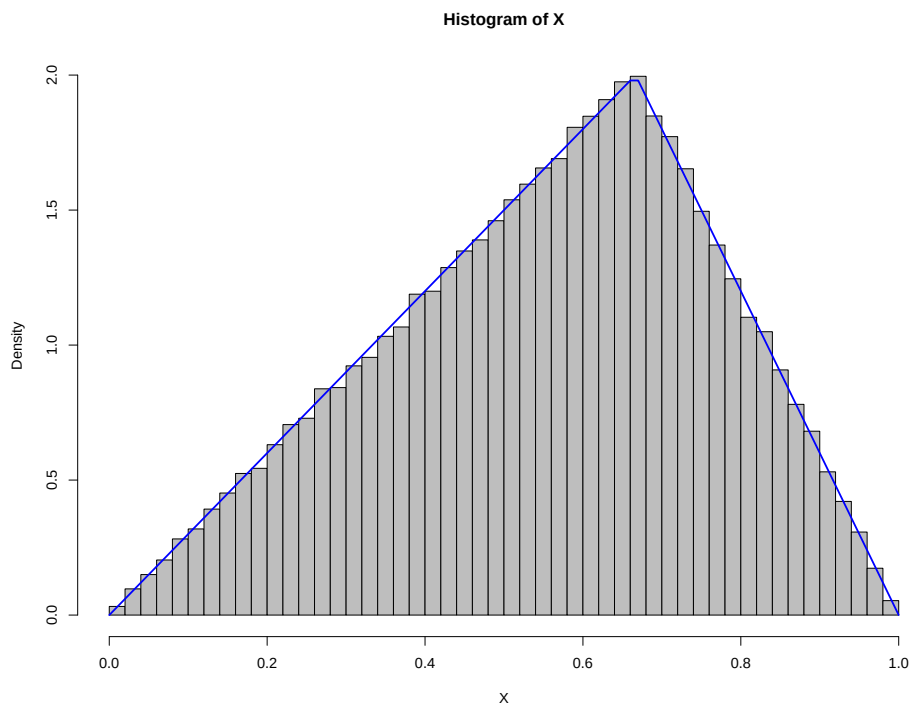
Méthode d'inversion

$$F^*(x) = \begin{cases} \frac{3}{2}x^2 & x \in [0;2/3] \\ -2+6(x-\frac{x^2}{2}) & x \in [2/3;1] \end{cases}$$

$$F^*(x) = \begin{cases} \frac{3}{2}x^2 & x \in [0;2/3] \\ -2+6(x-\frac{x^2}{2}) & x \in [2/3;1] \end{cases} \Rightarrow F^{*-1}(y) = \begin{cases} \sqrt{\frac{2}{3}y} & y \in [0;2/3] \\ 1+\frac{\sqrt{3-3y}}{3} & y \in [2/3;1] \end{cases}$$

Pour trouver la fonction inverse de la deuxième expression, on a résolue l'équation quadratique : $3x^2 - 6x + y + 2 = 0$

$$\Delta = 9 - 3(y+2) = 3 - 3y; \quad x1 = 1 - \frac{\sqrt{3-3y}}{3}; \quad x2 = 1 + \frac{\sqrt{3-3y}}{3};$$



Comparaison de différentes méthodes

Monte Carlo brute

```
### Monte Carlo brute
g=function(x) 100*x^4*(1-x)^2
n=1e5;
X=runif(n) ;
Z=g(X) ;
mean(Z);
(sd1=sd(Z))
```

Comparaison de différentes méthodes

Ech preferentiel

```
### Ech preferentiel
## Simulation de f
f=function(x) (x<2/3)*3*x+(x>2/3)*6*(1-x)
Finv=function(y) (y<2/3)*sqrt((2/3)*y)+
(y>=2/3)*(1-sqrt(3-3*y)/3)
U=runif(n);
X=Finv(U)
### Monte Carlo
gf=function(x) g(x)/f(x)
W=gf(X);
mean(W);
(sd2=sd(W))
```

Comparaison de différentes méthodes

```
## Rejet
U=runif(n,0,1); V=runif(n,0,g(2/3));Acc=(V<=g(U));
(prop=sum(Acc)/n);(est=prop*g(2/3))
(sd3=sqrt(prop*(1-prop))*g(2/3))
```

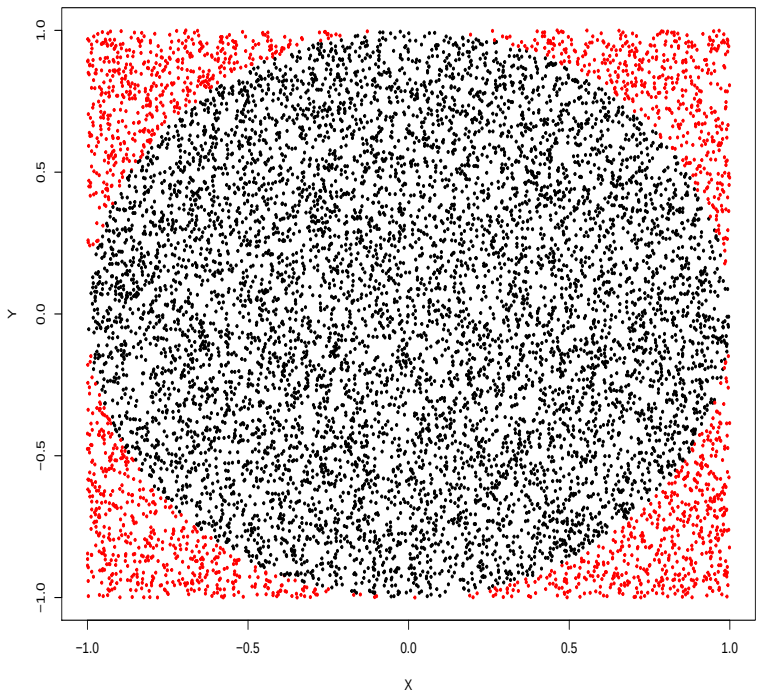
```
c(sd1,sd2,sd3)
[1] 0.8053485 0.3675663 1.0871554
c(sd1^2,sd2^2,sd3^2)
[1] 0.6485863 0.1351050 1.1819069
```

On note qu'avec un échantillonnage préférentiel une réduction de la variance d'un rappport 1/5 par rapport à la méthode de Monte Carlo brute.

Estimation de π par Monte Carlo

On estime la surface de disque unitaire par Monte Carlo.

```
n=1e4
X=runif(n,-1,1);
Y=runif(n,-1,1)
Acc=(X^2+Y^2 < 1 )
col=2-Acc
plot(X,Y,pch=19,
col=col ,cex=0.42)
```



```
n=1e5
X=runif(n,-1,1)
Y=runif(n,-1,1)
Acc=(X^2+Y^2 < 1 )
prop=sum(Acc)/n
pbinf=prop-1.96*sqrt(prop*(1-prop))/sqrt(n)
pbsup=prop+1.96*sqrt(prop*(1-prop))/sqrt(n)
```

```
> (estim=4*prop)
[1] 3.14876
> (binf=4*pbinf)
[1] 3.138613
> (bsup=4*pbsup)
[1] 3.158907
```

Remarque

Estimer une proportion est un cas particulier de l'estimation d'une espérance.

```
pbinf2=mean(Acc)-1.96*sd(Acc)/sqrt(n)
pbsup2=prop+1.96*sd(Acc)/sqrt(n)
(estim=4*mean(Acc))
(binf=4*pbinf2)
(bsup=4*pbsup2)
```

```
> (estim=4*mean(Acc))
[1] 3.14876
> (binf=4*pbinf2)
[1] 3.138613
> (bsup=4*pbsup2)
[1] 3.158907
```

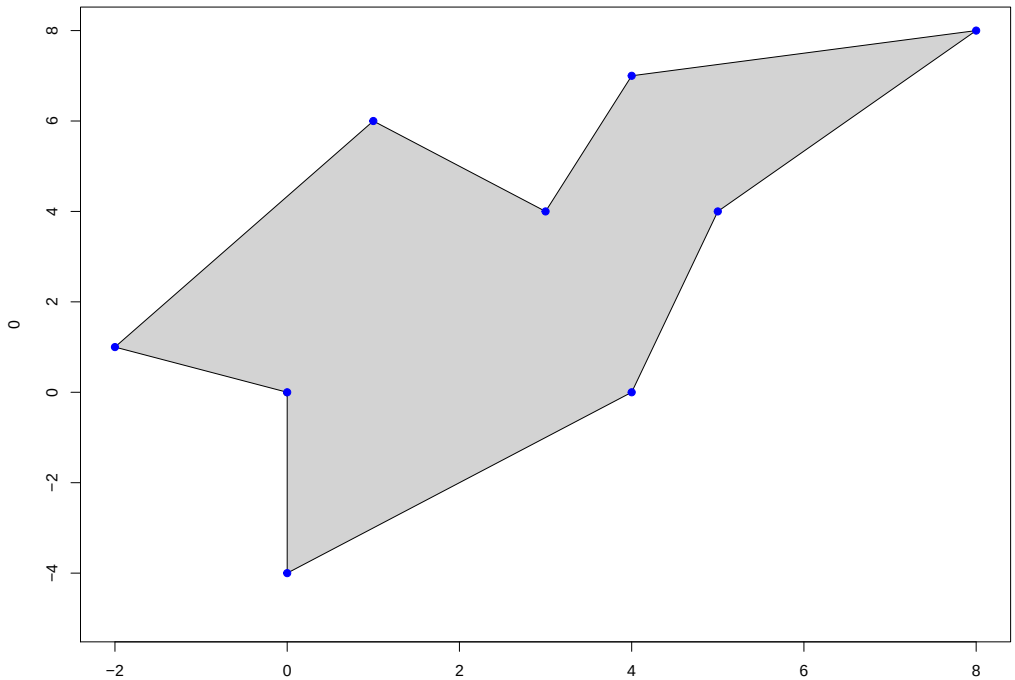
Estimer le volume d'une boule de dimension m

```
n=1e6 ; m=3
X=runif(n*m)
M=matrix(X, ncol=m)
x=1*apply(M^2,1,sum)<1
prop=mean(x)
```

```
> Vol ; 4*pi/3
[1] 4.18508
[1] 4.18879
> (binf=2^m*(prop-1.96*sqrt(prop*(1-prop)/n)))
[1] 4.177248
> (bsup=2^m*(prop+1.96*sqrt(prop*(1-prop)/n)))
[1] 4.192912
```

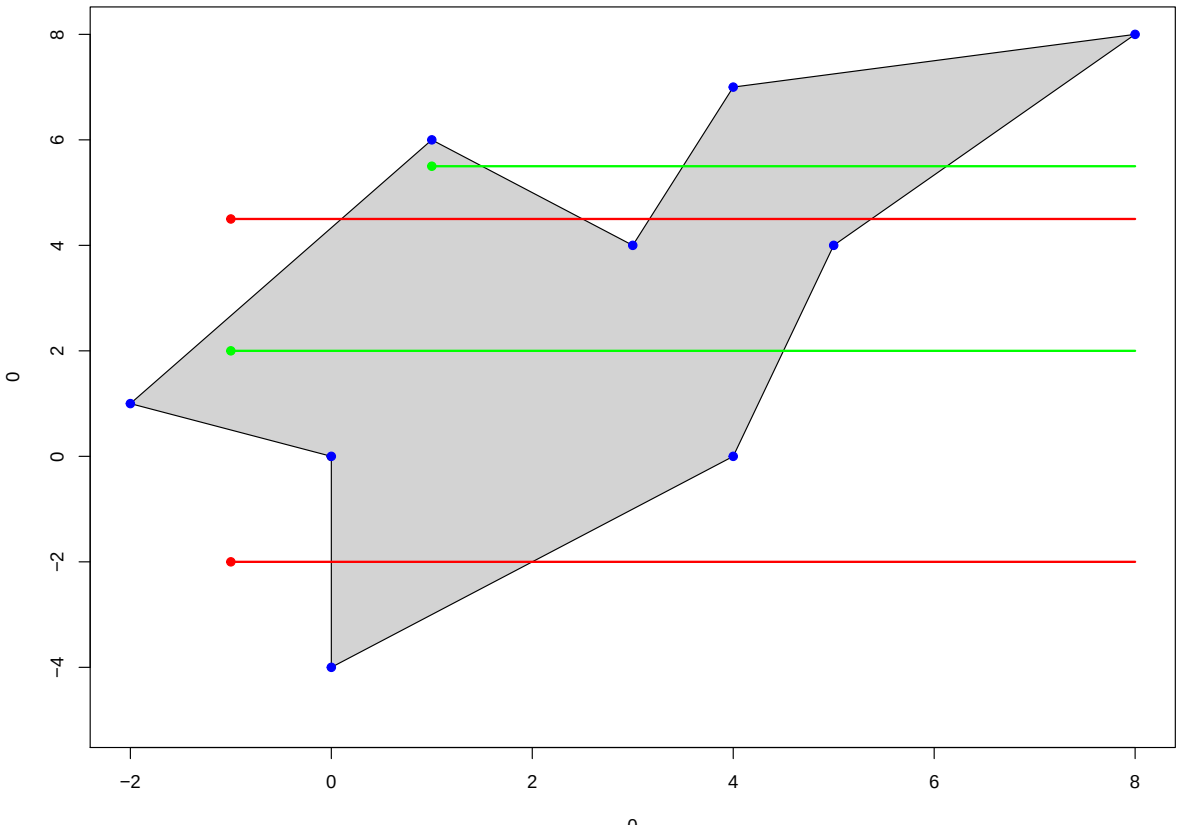
Estimer la surface de ce polygone par Monte-Carlo !

```r
x=c(0,4,5,8,4,3,1,-2,0)
y=c(-4,0,4,8,7,4,6,1,0)
plot(0,0,xlim=c(-2,8),ylim=c(-5,8))
polygon(x,y,col="lightgray")
points(x,y,pch=19,col="blue")
```



Indication

Pour décider si un point appartient à un polygone, on pourrait utiliser cette technique : *Comptez les intersections entre le segment vert(rouge) et les segments du polygone. Si le nombre d'intersections est impair, alors le point est dans le polygone, sinon il est dehors.*



Optimisation : Méthode de Gradient

Objectif

Minimiser une fonction différentiable $f(x)$ sur un espace $\mathbb{E}$.

Algorithme du gradient

On se donne un point itéré initial $x_0 \in \mathbb{E}$ et un seuil de tolérance $\varepsilon > 0$.

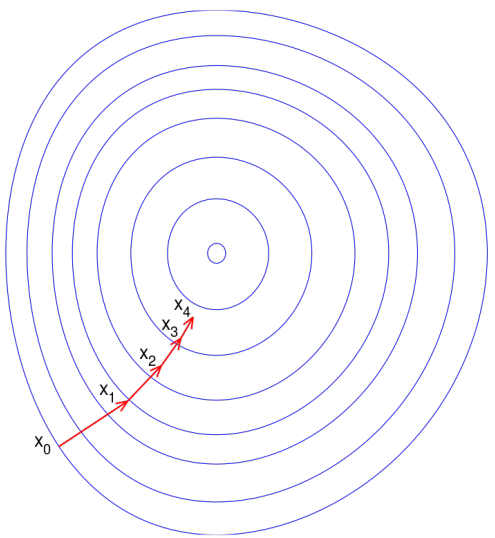
L'algorithme du gradient définit une suite d'itérés $x_1, x_2, \ldots \in \mathbb{E}$, jusqu'à ce qu'un test d'arrêt soit satisfait. Il passe de x_k à x_{k+1} par les étapes suivantes.

1. Simulation : calcul de $\nabla f(x_k)$.
2. Test d'arrêt : si $\|\nabla f(x_k)\| \leqslant \varepsilon$, arrêt.
3. Calcul du pas $\alpha_k > 0$.
4. Nouvel itéré : $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$.

Gradient à Pas fixe et Gradient à pas optimal

L'étape 3 : Si le choix est fixe : $\alpha_k = \alpha$ l'algorithme est dit à pas fixe.

Si la recheche est faite par une optimisation (uni-dimensionnel) sur f en x_k le long de la direction $-\nabla f(x_k)$, l'algorithme est à pas optimal.



Prinicipe

- Cet algorithme structurellement très simple repose sur le fait que, dans le voisinage d'un point x, la fonction f décroît le plus fortement dans la direction opposée à celle du gradient de f en x, à savoir dans la direction $-\nabla f(x)$.
- De manière plus précise, cette affirmation exprime en termes suggestifs le fait que, si $\nabla f(x) \neq 0$, la solution du problème d'optimisation

$$\min_{\|d\|=1} \langle \nabla f(x), d \rangle$$

est la direction $d = -\nabla f(x)/\|\nabla f(x)\|$, orientée donc vers l'opposé du gradient

À la fois l'estimation statistique et l'apprentissage automatique s'intéressent au problème de la minimisation d'une fonction objectif qui a la forme d'une somme : :

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

,

où le paramètre x^* qui minimise $f(x)$.

Chacune des fonctions f_i est généralement associée avec la i-ème observation de l'ensemble des données (utilisées pour l'apprentissage).

Méthode de gradient stochastique

Considérons le problème de minimisation d'une moyenne de fonctions :

$$\min_x \frac{1}{m} \sum_{i=1}^{m} f_i(x)$$

Comme $\nabla \sum_{i=1}^{m} f_i(x) = \sum_{i=1}^{m} \nabla f_i(x)$, L'algorithme de gradiant stochastique va répéter :

$$x^{(k)} = x^{(k-1)} - t_k \cdot \frac{1}{m} \sum_{i=1}^{m} \nabla f_i \left(x^{(k-1)} \right), \quad k = 1, 2, 3, \ldots$$

Si m est trop grand la méthode échouera. Problème de mémoire et de temps d'éxécution.

Par contre , l'algorithme de gradiant stochastique SGD va faire :

$$x^{(k)} = x^{(k-1)} - t_k \cdot \nabla f_{i_k}\left(x^{(k-1)}\right), \quad k = 1, 2, 3, \ldots$$

où $i_k \in \{1, \ldots, m\}$ est un **indice choisi aléatoirement** à l'iteration k

Notons que

$$\mathbb{E}\left[\nabla f_{i_k}(x)\right] = \nabla f(x)$$

On voit que SGD utilise un estimateur sans biais du gradiant à chaque itération.

Il va permettre un gain très important en mémoire et temps d'éxecution.

Utilisation

L'algorithme du gradiant stochastique est très utilisé en

- Apprentissage automatique(SVM, ...)
- Entraînement des réseaux de neurones artificiels
- Régression logistique

Variantes

Il y a plusieurs variantes et extensions

- SGD Sampling

$$x_{k+1} := x_k - \alpha_k \frac{1}{\ell} \sum_{j=1}^{\ell} \nabla f_{I_j^{(k)}}(x_k)$$

$$I_j^{(k)} \sim \mathcal{U}\{1, \ldots, m\}$$

- SGD Moment

$$\Delta x_{k+1} := \alpha_k \nabla f_{i_k}(x_k) + \beta \Delta x_k$$

$$x_{k+1} := x_k - \Delta x_{k+1}$$

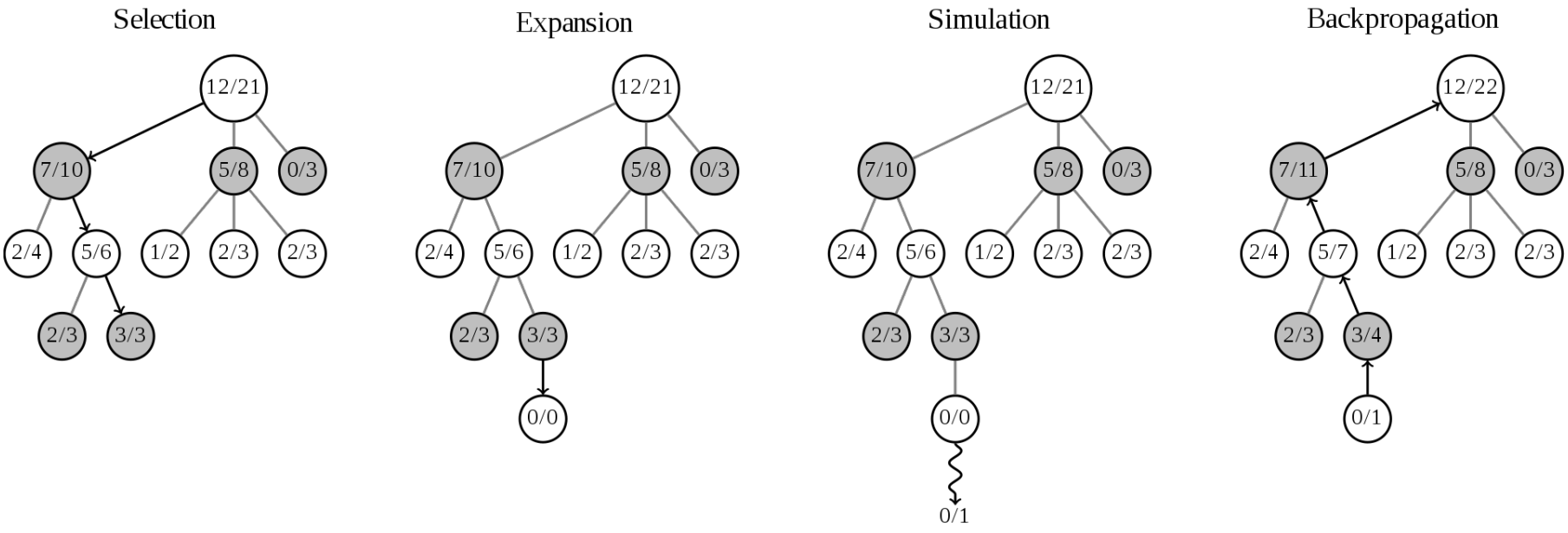
$$i_k \sim \mathcal{U}\{1, \ldots, m\}$$

- SGD moyenné

$$\bar{x} = \frac{1}{n} \sum_{k=0}^{n-1} x_k$$

- ...

MCTS : Monte Carlo ou Monte Carlo tree search



MCTS : La recherche arborescente Monte Carlo

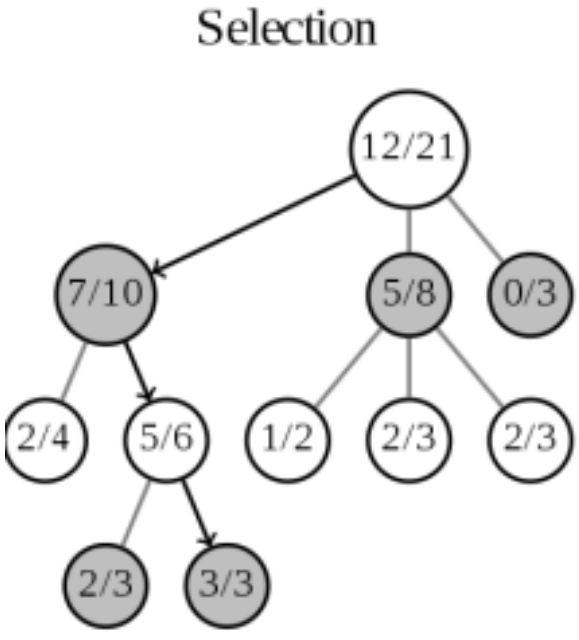
L'algorithme MCTS est un algorithme qui explore l'arbre des possibles.

- La racine est la configuration initiale du jeu.
- Chaque noeud est une configuration et ses enfants sont les configurations suivantes.
- MCTS conserve en mémoire un arbre qui correspond aux noeuds déjà explorés de l'arbre des possibles.
- Une feuille de cet arbre est soit une configuration finale (i.e. on sait si un des joueurs a gagné, ou s'il y a match nul), soit une configuration à partir de laquelle aucune **simulation** n'a encore été lancée.
- Dans chaque noeud, on stocke deux nombres : le nombre de simulations gagnantes, et le nombre total de simulations.

Chaque étape est composée de quatre phases5.

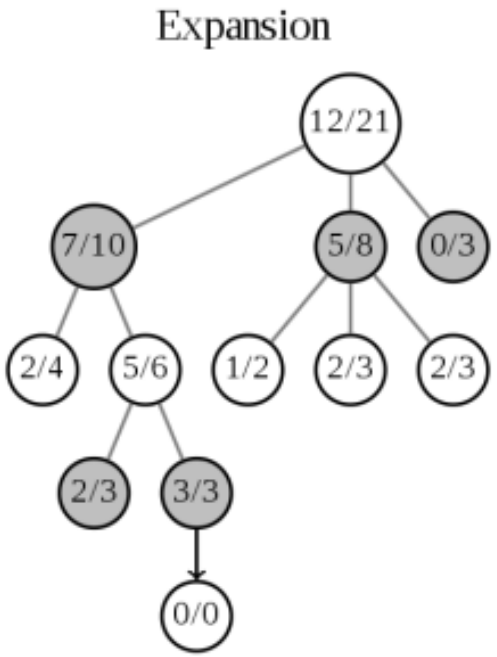
- **Sélection**

  Depuis la racine, on sélectionne successivement des enfants jusqu'à atteindre une feuille.



Dans cette phase, le choix des enfants est guidé par un compromis entre exploitation (aller vers un enfant qui a été prouvé comme prometteur) et exploration (aller visiter un autre enfant, qui a l'air moins prometteur mais qui pourrait l'être davantage). Dans l'exemple donné dans la figure, on choisit la feuille de droite (de profondeur 3).

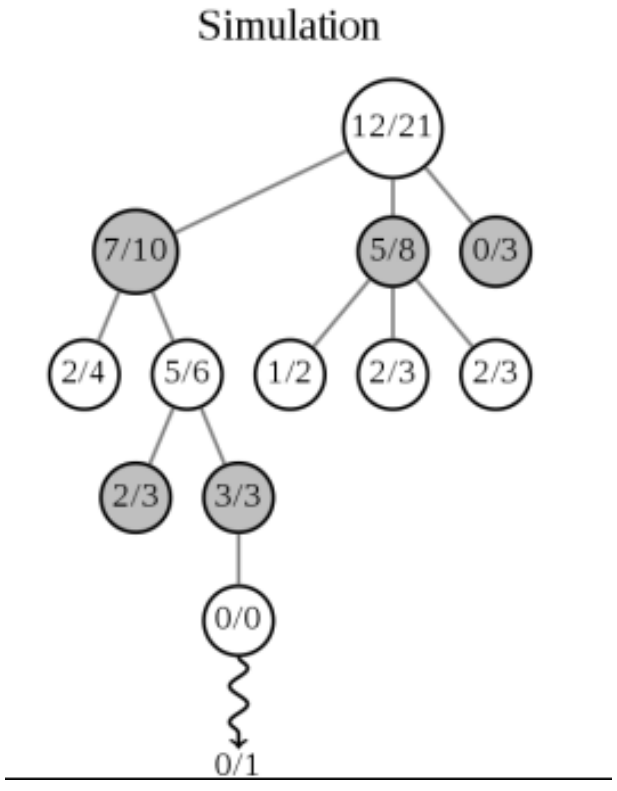
2. Expansion : si cette feuille n'est pas finale, créer un enfant (ou plusieurs) en utilisant les règles du jeu et choisir l'un des enfants.



Sur l'exemple, on ajoute cet enfant, marqué 0/0.

3. Simulation :

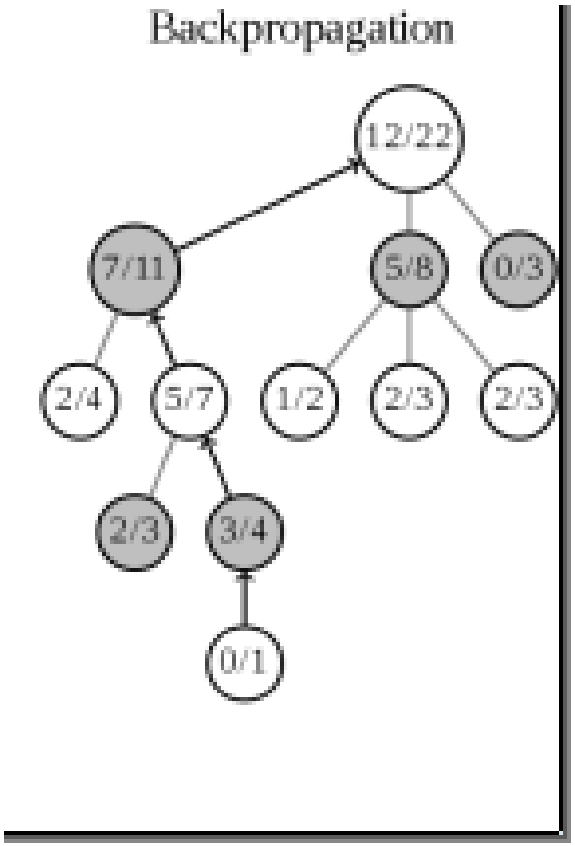
Simuler une exécution d'une partie **au hasard** depuis cet enfant, jusqu'à atteindre une configuration finale.



En effet, on éxecute un grand nombre de parties au hasard à partir de chaque noeud.

4.Rétropropagation :

utiliser le résultat de la partie au hasard et mettre à jour les informations sur la branche en partant du noeud enfant vers la racine.



- Sur l'exemple, la partie était perdante. On incrémente donc uniquement le nombre de simulations totales sur la branche : 0/0 devient 0/1, 3/3 devient 3/4, 5/6 devient 5/7, 7/10 devient 7/11, et 12/21 devient 12/22.
- Si la partie avait été gagnante :0/0 serait devenu 1/1, 3/3 serait devenu 4/4, 5/6 serait devenu 6/7, 7/10 serait devenu 8/11, et 12/21 serait devenu 13/22.

AlphaGo, AlphaZero

Le logiciel AlphaZero utilise Monte Carlo search Tree.

- Deep Learning
- MCTS
- Neuronal Networks

Voir ces liens :

- Alphazero sur www.developpez.com
- Alphago Zero Tutorial sur youtube.com