# Cours 4 : Méthode de rejet, Méthode de décomposition, Simulation de vecteurs aléatoires

A. Righi

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

Année Universitaire 2019-2020

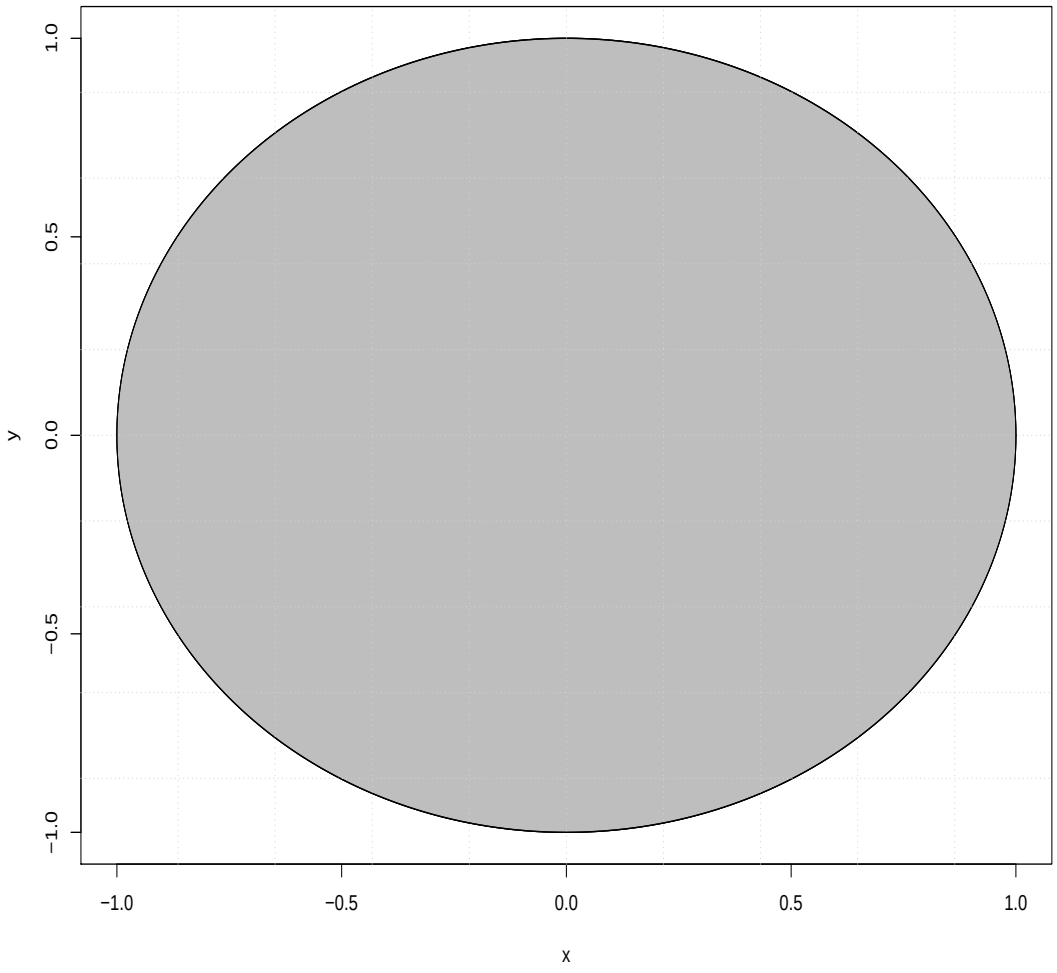
# Plan


1. Introudction
2. Méthode de rejet
3. Simulation d'une densité de proba avec rejet
4. Méthode de décomposition (Mélange)
5. Simulation de vecteurs aléatoires

# Simulation des variables (pseudo)aléatoires

# Méthode de Rejet

Comment simuler la loi uniforme sur le disque de centre $(0,0)$ et de rayon $1$.



## Dessin avec des fonctions de base

```
circle=function(){
theta=seq(0,2*pi,len=1000)
x=r*cos(theta)
y=r*sin(theta)
plot(x,y,xlim=c(-1,1),type="l",col="black")
polygon(x,y,col="gray")
points(x,y,xlim=c(-1,1),type="l",col="black")
}
circle()
grid(10)
```

prog1

## Algorithme ?!

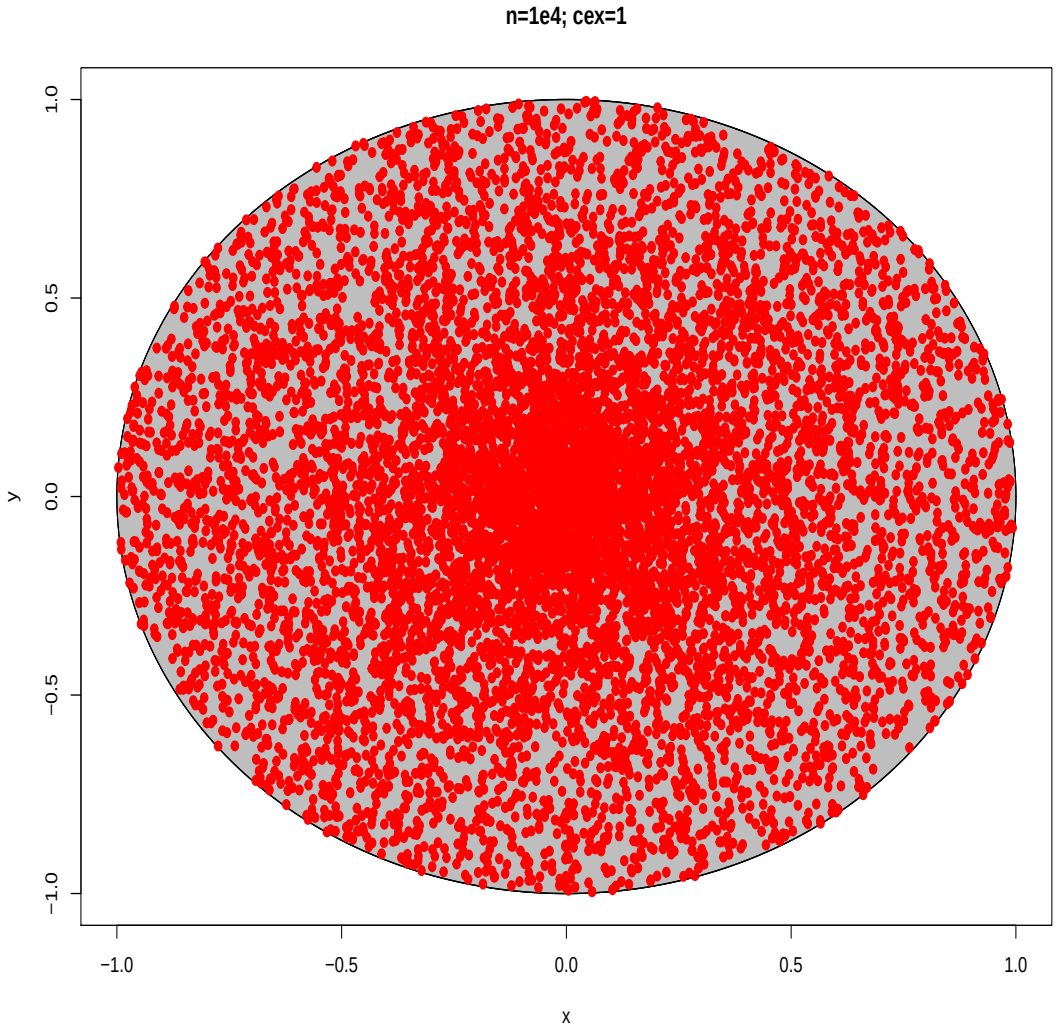
On peut penser comme les points $(x, y)$ du disque unitaire, sont caractérisés par les cordonnées polaires $(r\cos(\theta), r\sin(\theta))$.

L'idée donc est de générer un angle aléatoire entre $0$ et $2\pi$ et un rayon aléatoire entre $0$ et $1$.

```
n=1e4
R=runif(n)
thet=runif(n,0,2*pi)
X=R*cos(thet)
Y=R*sin(thet)
####
circle()
points(X,Y,pch=19,col="red",cex=1)
title("n=1e4; cex=1")
```

prog2

On génére 10000 points avec l'algortihme précedent et on obtient cette figure :



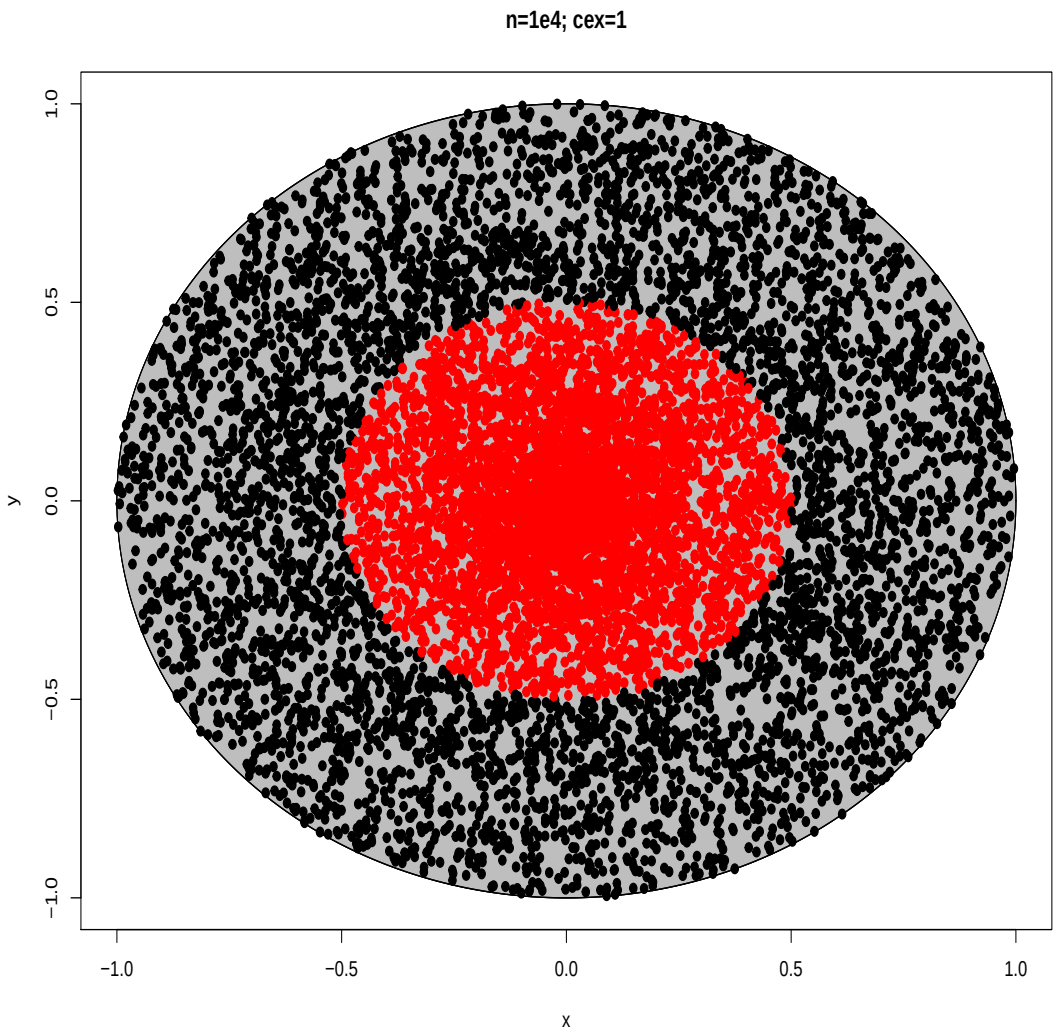

On obtient comme attendu des points aléatoires dans le disque unitaire.

## Distribution uniforme ??

Mais la distribution est-elle uniforme sur le disque ? On va colorier les points dans le disque de rayon $0.5$ en rouge et les autres points en noir.

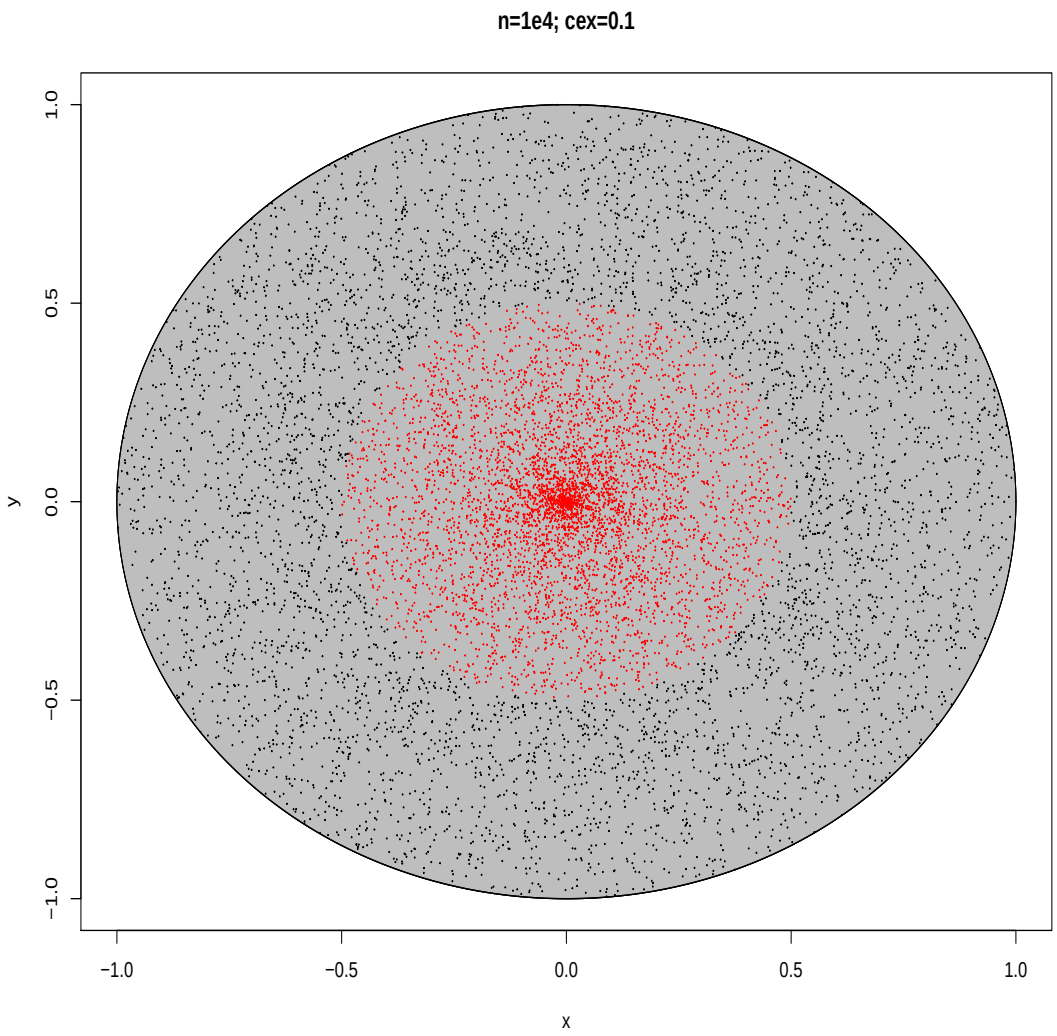
```
circle()
couleur=((X^2+Y^2<0.25)+1)
points(X,Y,pch=19,col=couleur ,cex=1)
sum(X^2+Y^2<0.25)/n
title("n=1e4 ; cex=1")
```

prog3

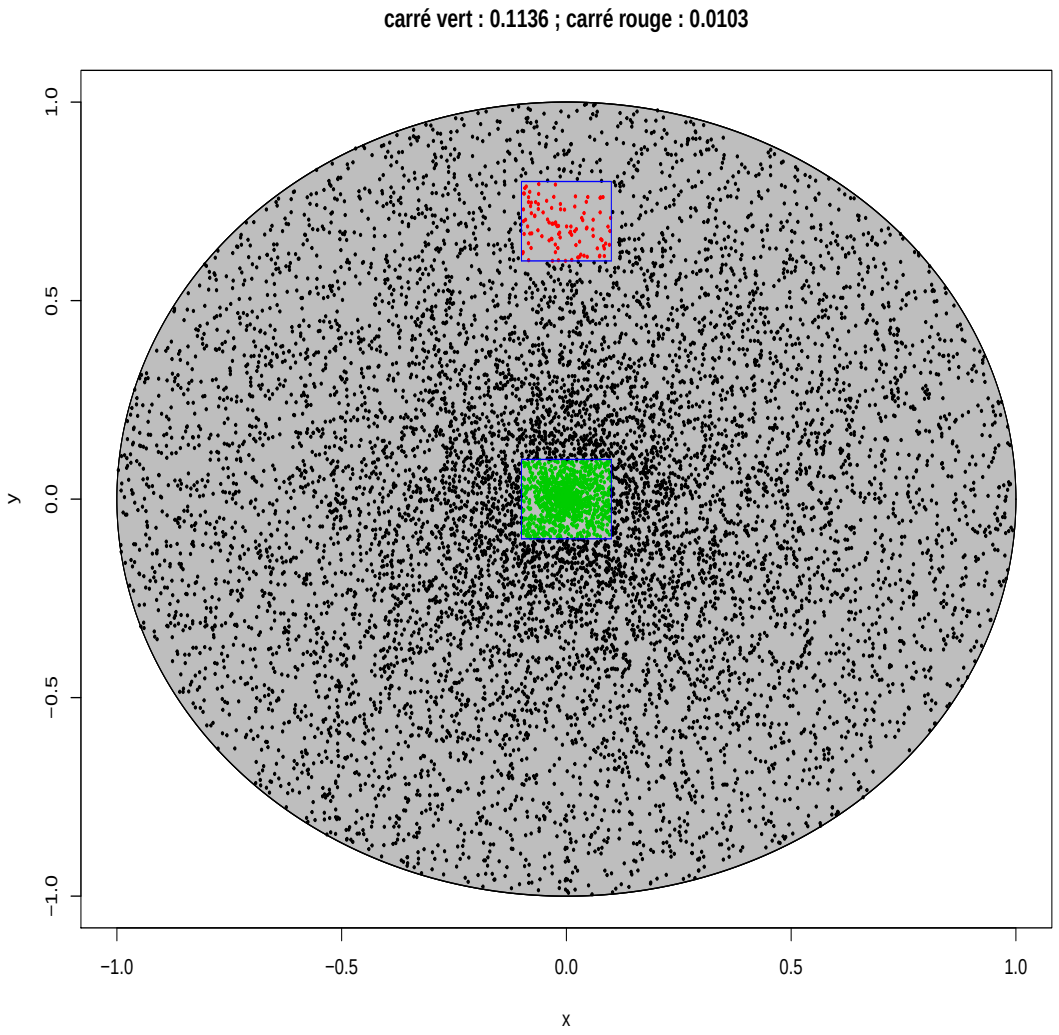



On note qu'il y a plus de concentration dans le disque intérieur.

On refait le dessin en changeant le paramètre "cex".



Une distribution uniforme sur le disque veut dire que pour n'importe carré à l'intérieur de disque avec la même surface aura la même probabilité indépendamment de sa position dans le disque.



## Conclusion : La distribution n'est pas uniforme

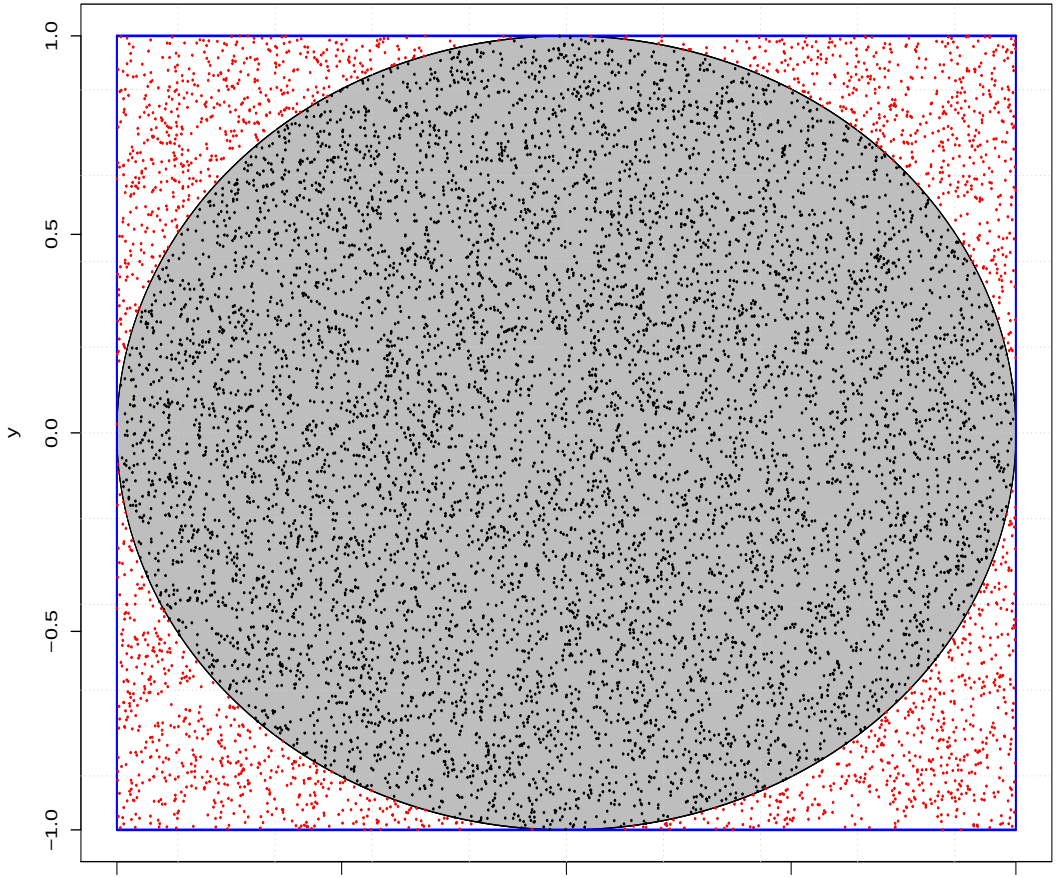
```
circle()
couleur=1+1*((X>-0.1&X<0.1)&(Y>0.6&Y<0.8))+
2*((X>-0.1&X<0.1)&(Y>-0.1&Y<0.1))
points(X,Y,pch=19,col=couleur,cex=0.3)
c1=sum((X>-0.1&X<0.1)&(Y>0.6&Y<0.8))/n
c2=sum((X>-0.1&X<0.1)&(Y>-0.1&Y<0.1))/n
title(paste("carré vert :",as.character(c2),
"; carré rouge :", as.character(c1)))
```

prog4

## Acceptation Rejet.

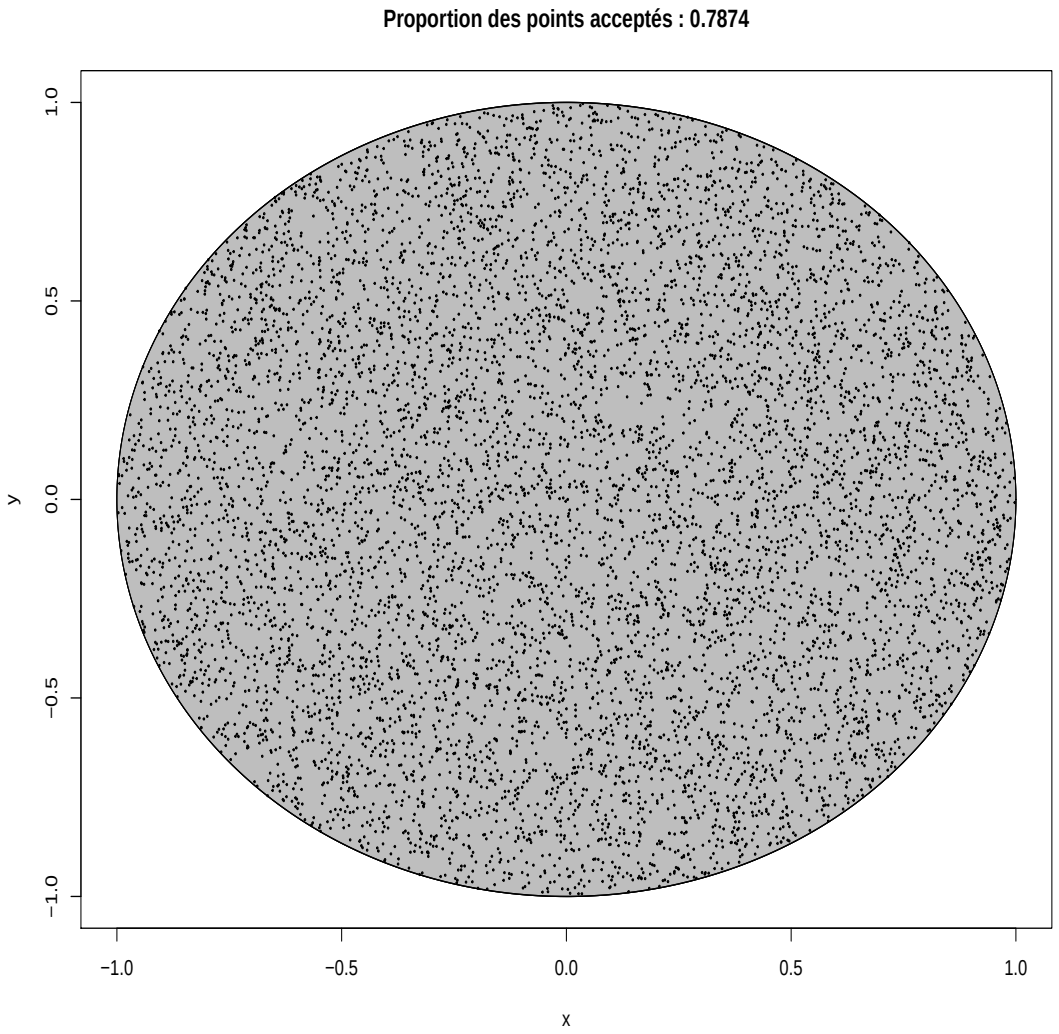
Une méthode qui est juste, serait de générer des points sur le carré qui contient le disque unitaire et rejeter les points en dehors du disque.

Ce qui revient à accepter seulement les points à l'intérieur du disque.



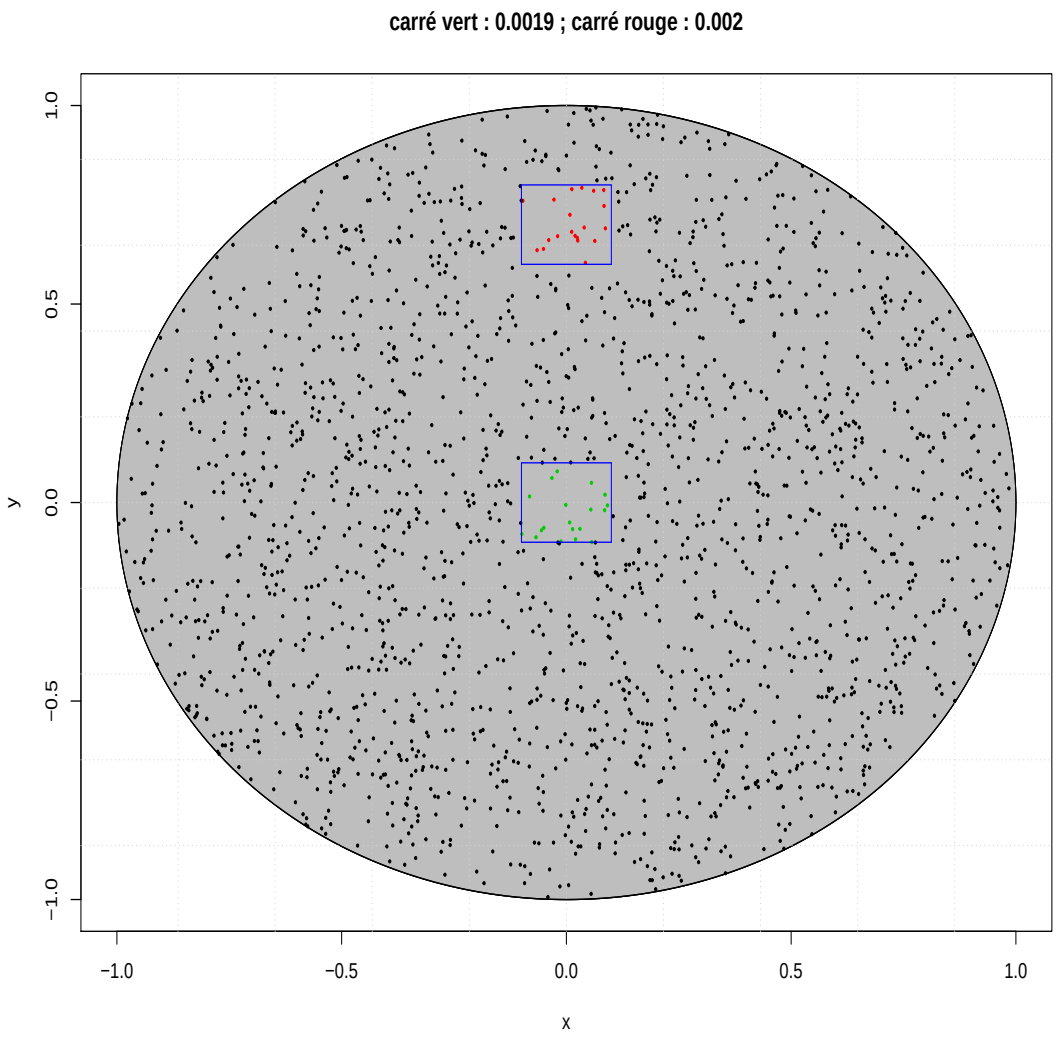
## Acceptation Rejet

```
circle()
grid(10)
segments(-1,-1,-1,1,col="blue",lwd=2)
segments(-1,1,1,1,col="blue",lwd=2)
segments(1,1,1,-1,col="blue",lwd=2)
segments(1,-1,-1,-1,col="blue",lwd=2)
n=1e4
U=runif(n,-1,1)
V=runif(n,-1,1)
Acc=(U^2+V^2<1)
X=U[Acc]
Y=V[Acc]
couleur=(2-Acc)
points(U,V,col=couleur,cex=0.2)
```




Notons que le pourcentage attendu est la surface du disque divisé sur la surface du carré. on a par ailleurs $\pi/4 = 0,785398$.

Refaisons la même chose avec le précédent algorithme.




On voit que la distribution est uniforme.

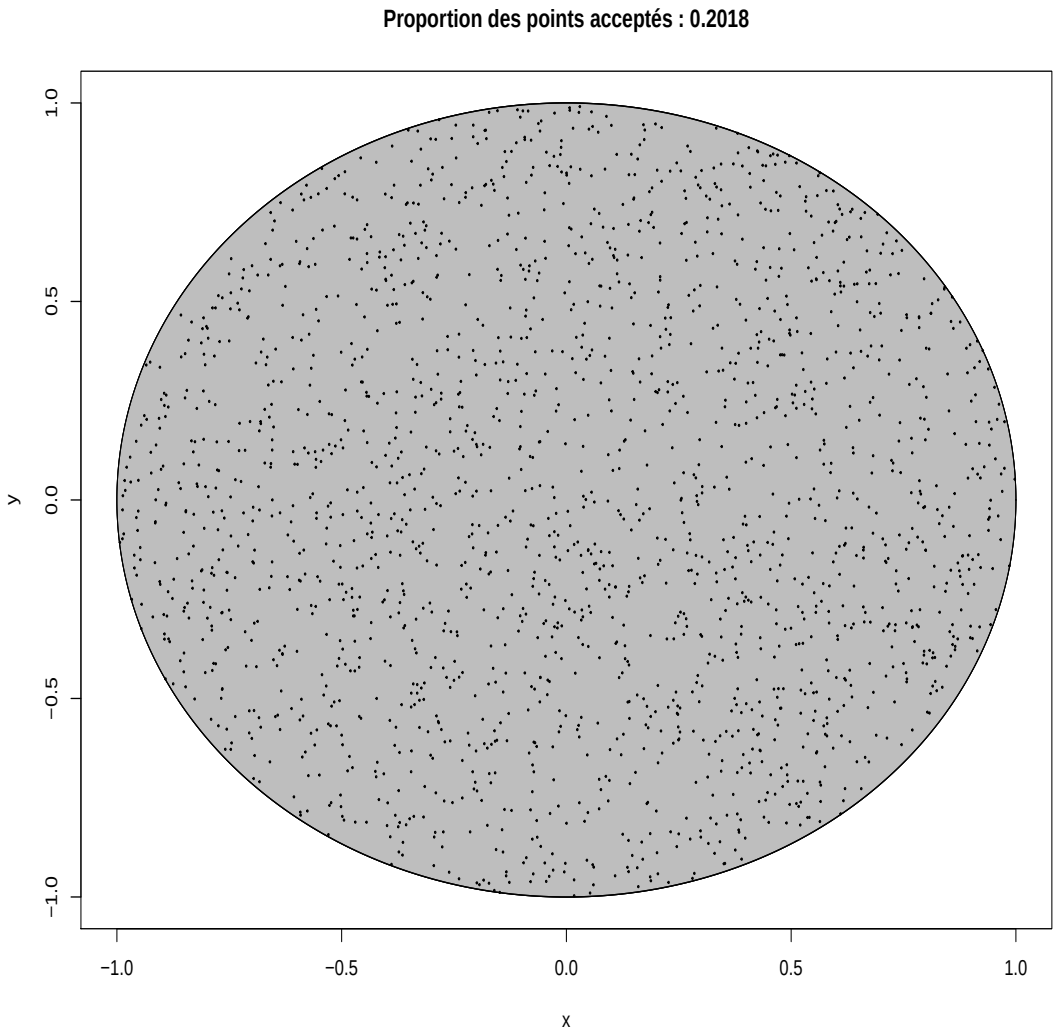
C'est juste si on fait le programme suivant :

```
n=1e4
U=runif(n,-2,2)
V=runif(n,-2,2)
Acc=(U^2+V^2<1)
X=U[Acc]
Y=V[Acc]
couleur=(2-Acc)
points(U,V,col=couleur,cex=0.2)
```

prog6

On génère des points sur un carré plus grand. Mais l'inconvénient est qu'on va avoir un pourcentage élevé de rejet.

On a intérêt à prendre le carré le plus petit qui contient le disque pour minimiser le taux de rejet.


Proportion des points acceptés : 0.2018
1.0
0.5
0.0
−0.5
−1.0
y
−1.0
−0.5
0.0
0.5
1.0
x

## Méthode de rejet pour les lois uniformes

On suppose que l'on sait simuler par l'algorithme $\mathcal{A}$ une variable aléatoire de loi uniforme sur un ensemble borélien $D \subset \mathbb{R}^d$ (par exemple le carré $\left]-1,+1\right[^2$) et que l'on veut simuler une variable aléatoire de loi uniforme sur un sous-ensemble borélien $C \subset D$. L'algorithme

Faire $\mathrm{X} \leftarrow \mathcal{A}$

Tantque $C$ faux

Fin

Retourner $\mathrm{X}$

donne une simulation de la loi uniforme sur $C$.

En effet, soit $(X_n, n \geq 1)$ une suite de variables aléatoires indépendantes de loi uniforme sur $D$ et $\tau = \inf\{n \geq 1 : X_n \in C\}$ ; l'algorithme précédent retourne la variable aléatoire $X_\tau$ telle que pour tout sous-ensemble borélien $B \subset C$

$$\begin{aligned}
P(X_\tau \in B) &= \sum_{k=1}^{\infty} P(\tau = k, X_\tau \in B) \\
&= \sum_{k=1}^{\infty} P(X_1 \notin C, \ldots X_{k-1} \notin C, X_k \in C, X_k \in B) \\
&= \sum_{k=1}^{\infty} P(X_1 \notin C)^{k-1} P(X_k \in B) \\
&= \sum_{k=1}^{\infty} \left(1 - \frac{|C|}{|D|}\right)^{k-1} \frac{|B|}{|D|} = \frac{|B|}{|C|}
\end{aligned}$$

Ainsi, pour simuler une loi uniforme sur le disque unité, on procède comme suit :

## Version Avec boucle

```
rand.disque=function(n,x0=0,y0=0,r=1) {
X=numeric(n)
Y=numeric(n)
for(i in 1:n) {
U=runif(1,-1,1); V=runif(1,-1,1);
while(U^2+V^2>1) {
U=runif(1,-1,1); V=runif(1,-1,1);
}
X[i]=U; Y[i]=V
}
return(cbind(r*X+x0,r*Y+y0))
}
plot(rand.disque(1e4,1,1,2))
```

prog7

## Version sans boucle

```
rand2.disque=function(n,x0=0,y0=0,r=1) {
m=n*1.5
U=runif(m,-1,1)
V=runif(m,-1,1)
Acc=(U^2+V^2<1)
X=U[Acc]
Y=V[Acc]
return(cbind(x0+r*X[1:n],y0+r*Y[1:n]))
}
plot(rand2.disque(1e4,1,1,2),cex=0.5)
```

prog8

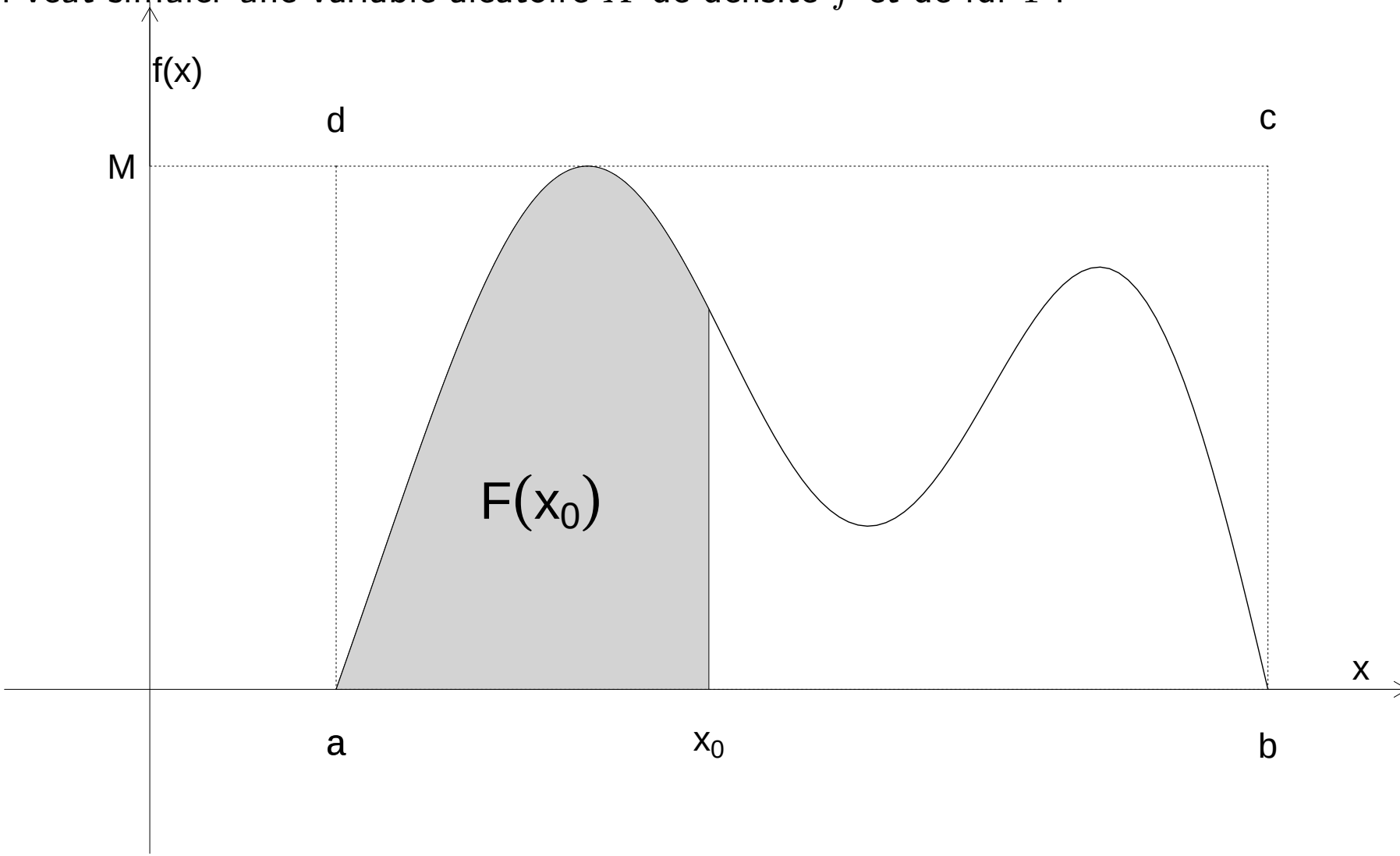
Pour être presque sûr d'avoir $n$ points acceptés on génére $1.5n$. Le nombre $1.5$ n'est pas optimal.

**Question.** Quel est le facteur optimal si on veut être sûr à $99.9\%$ d'avoir au minimum $n$ points acceptés ?

# Simulation d'une densité de proba avec rejet

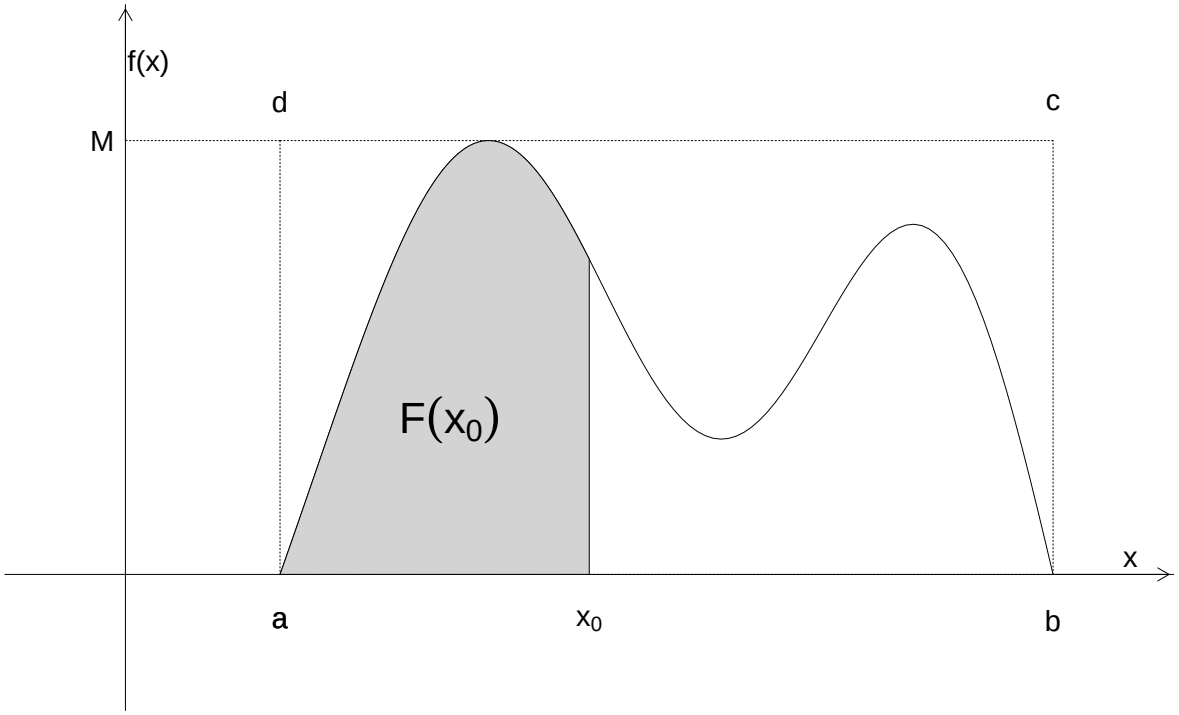
## Principe

On veut simuler une variable aléatoire $X$ de densité $f$ et de fdr $F$.



## Hypothèses

1. $f$ est à support compact, i.e que $f$ est nulle en dehors d'un intervalle $[a, b]$.
2. Il existe un majorant $M$ de $f(x)$ : $\forall x \in [a, b]\ \ f(x) \leq M$.

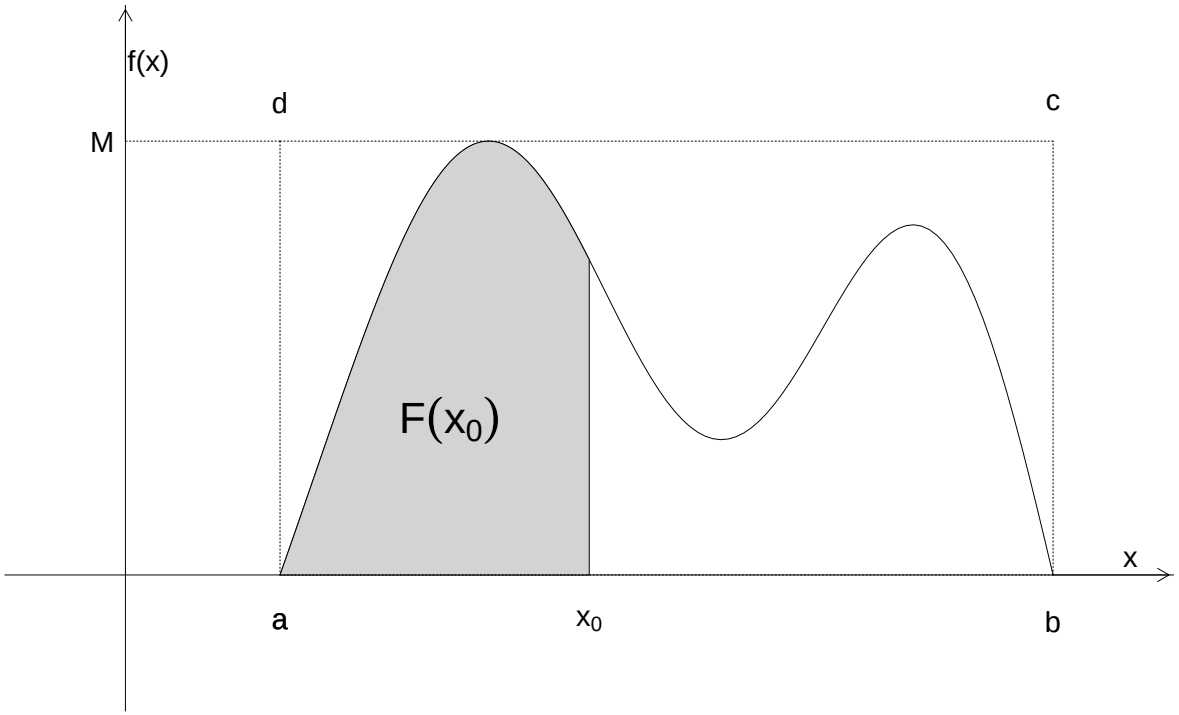



-surface du rectangle $(abcd) = M(b-a)$

-surface sous la courbe $= 1$ (puisque c'est une densité)

-surface de la zone hachurée $= F(x_0)$.

# Méthode




**a.** On tire un point $A = (X, Y)$ uniformément distribué sur le rectangle (abcd) :

On tire $U_1 \longrightarrow X = a + (b - a)U_1$ On tire $U_2 \longrightarrow Y = MU_2$

**b.** Si $Y \leq f(X)$, on garde $X$ sinon on tire un autre point.

**c.** On réitère jusqu'à ce que la condition $Y \leq f(X)$ soit remplie.

**Théorème**

*La variable $X$ ainsi obtenue a une densité $f$ et une fonction de répartition $F$.*

**Démonstration** : On a

$$\begin{aligned}
P\{X \leq x_0\} &= P\{X \leq x_0 |\ on\ \ gardeA\} \\
&= (X,Y)\} = P\{X \leq x_0 | Y \leq f(X)\} \\
&= \frac{P\{X \leq x_0, Y \leq f(X)\}}{P\{Y \leq f(X)\}} \\
P\{X \leq x_0, Y \leq f(X)\} &= \frac{\text{surface de la zone hachurée}}{\text{surface du rectangle (abcd)}} = \frac{F(x_0)}{M(b-a)} \\
P\{Y \leq f(x)\} &= \frac{\text{surface sous la courbe}}{\text{surface du rectangle (abcd)}} = \frac{1}{M(b-a)}
\end{aligned}$$

et donc

$$P\{X \leq x_0\} = \frac{F(x_0)}{M(b-a)} * \frac{M(b-a)}{1} = F(x_0)$$

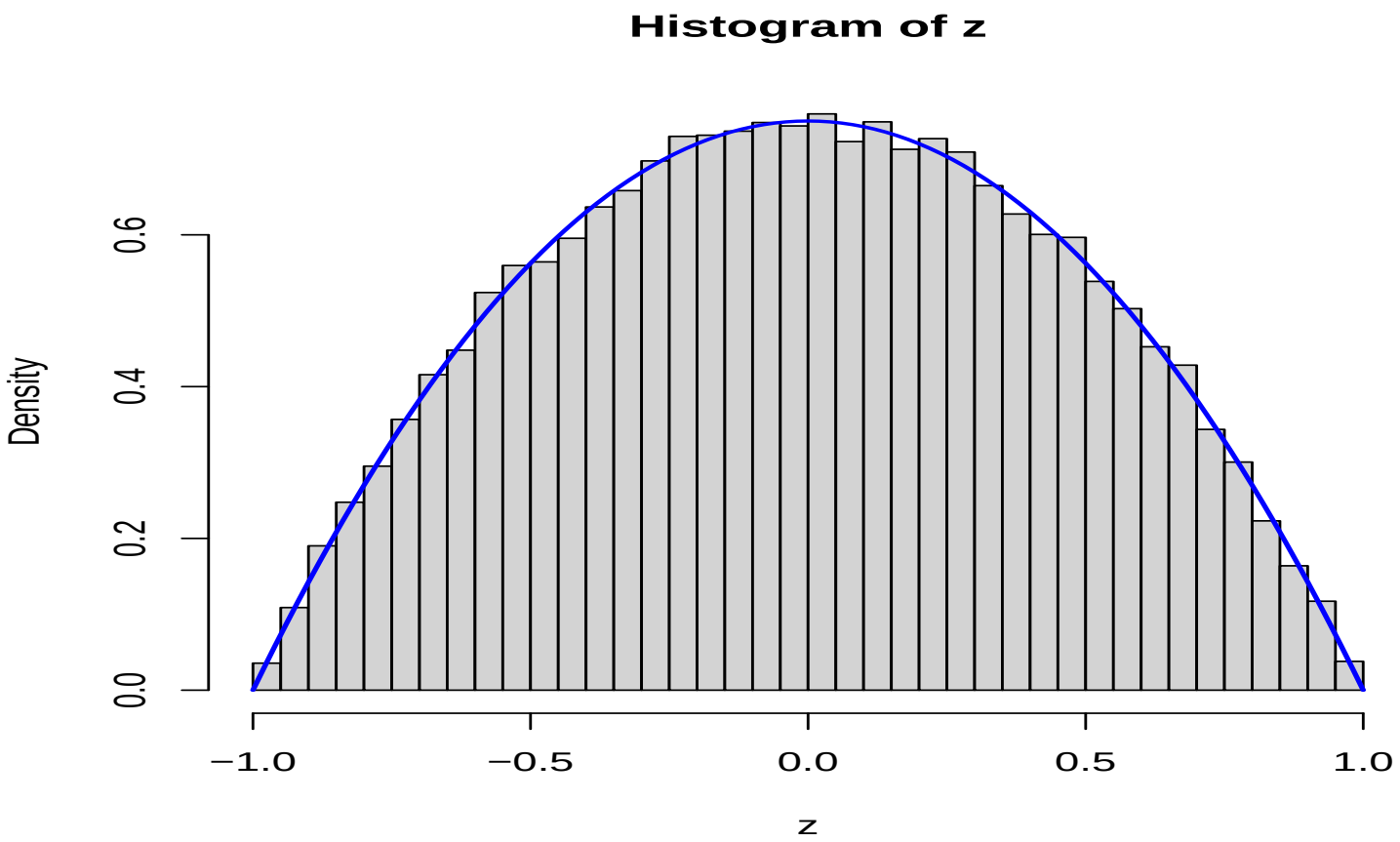
## Choix de la constante $M$

**Remarque** : Il est de plus important de noter qu'on choisit la constante M la plus petite possible pour minimiser le nombre de rejets : plus la majoration est grossière, plus il faut de tirages pour obtenir une valeur acceptable.

## Simuler la densité : $f(x) = \frac{3}{4}(1-x^2)\mathbf{1}_{[-1,1]}(x)$

**Objectif**

Simuler la densité : $f(x) = \frac{3}{4}(1-x^2)\mathbf{1}_{[-1,1]}(x)$



## Méthode d'inversion $f(x) = \frac{3}{4}(1-x^2)\mathbf{1}_{[-1,1]}(x)$

Le calcul de la fonction de répartition est simple. Pour $x \in ]-1,1[$:

$$F(x) = \frac{3}{4}\left[t - \frac{1}{3}t^3\right]_{-1}^{x} = \frac{1}{2} + \frac{3}{4}x - \frac{1}{4}x^3$$

Pour l'inverse de $F$ on doit résoudre une équation de troisième degré.

.....

On va simuler $f$ par la méthode de rejet.

- La fonction $f$ est à support compact.
- Elle est bornée.
- Le meilleur choix de $M$ est $\frac{3}{4}$.

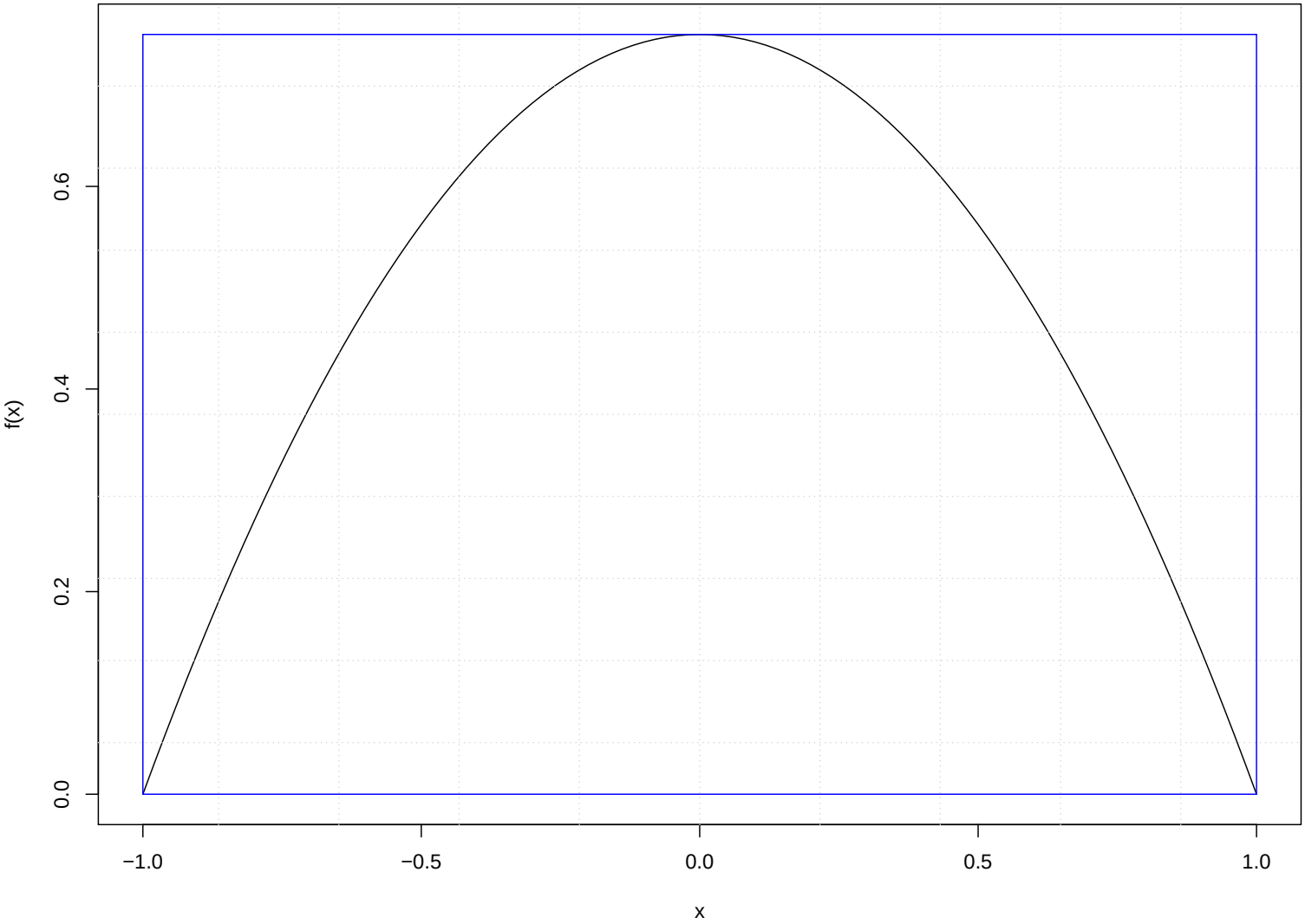
En effet on peut le constater via le graphe.

Si on fait la dérivée : $f'(x) = \frac{-3}{2}x$. La dérivée s'annule dans le point $0$ et $f''(x) = \frac{-3}{2}$.

Donc le maximum sur $]-1,1[$ est atteint dans le point $0$.

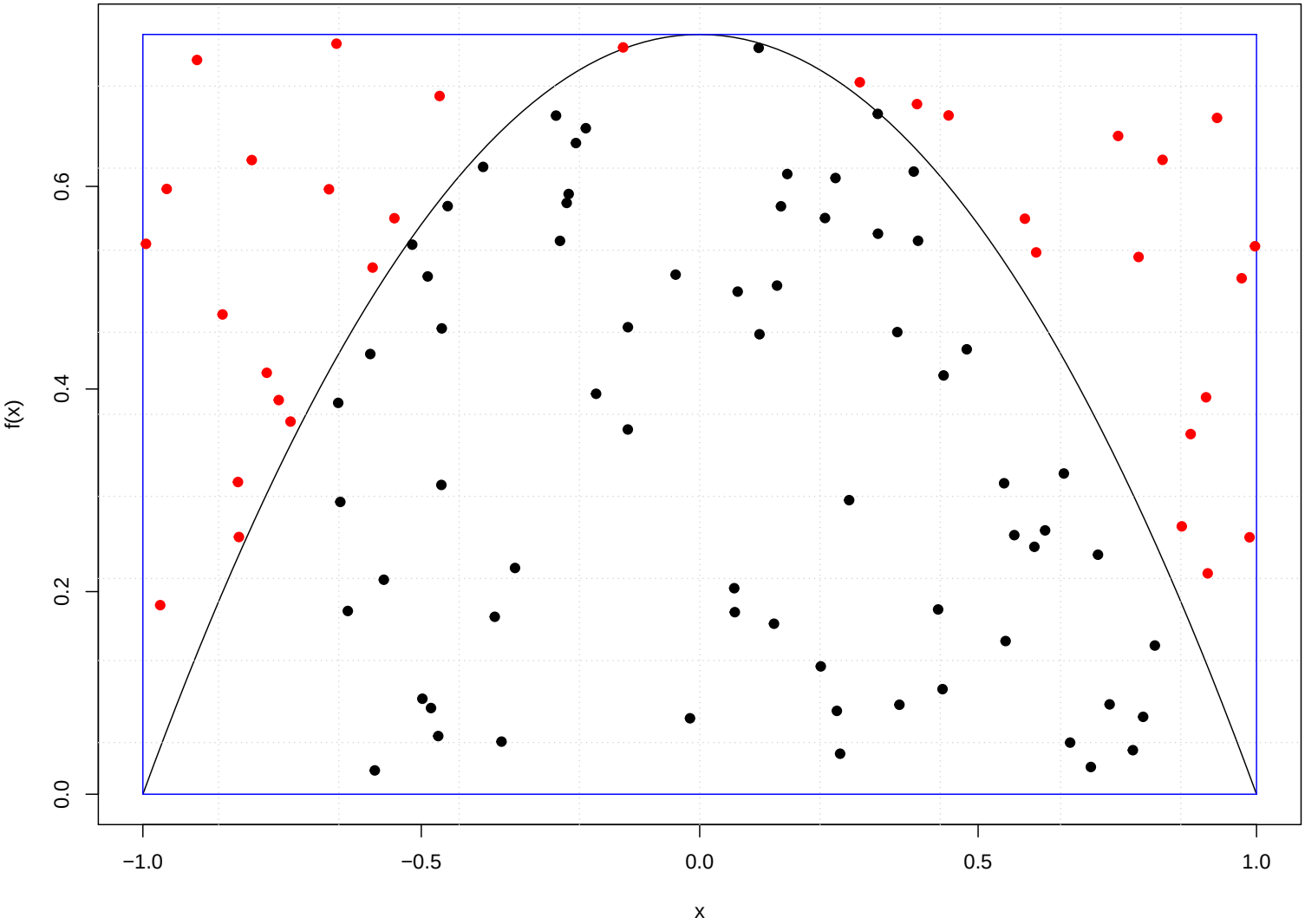
Alors

$$\forall x \in ]-1,1[ : f(x) \leq f(0) = \frac{3}{4}$$


0.0
0.2
0.4
0.6
f(x)
−1.0
−0.5
0.0
0.5
1.0
x

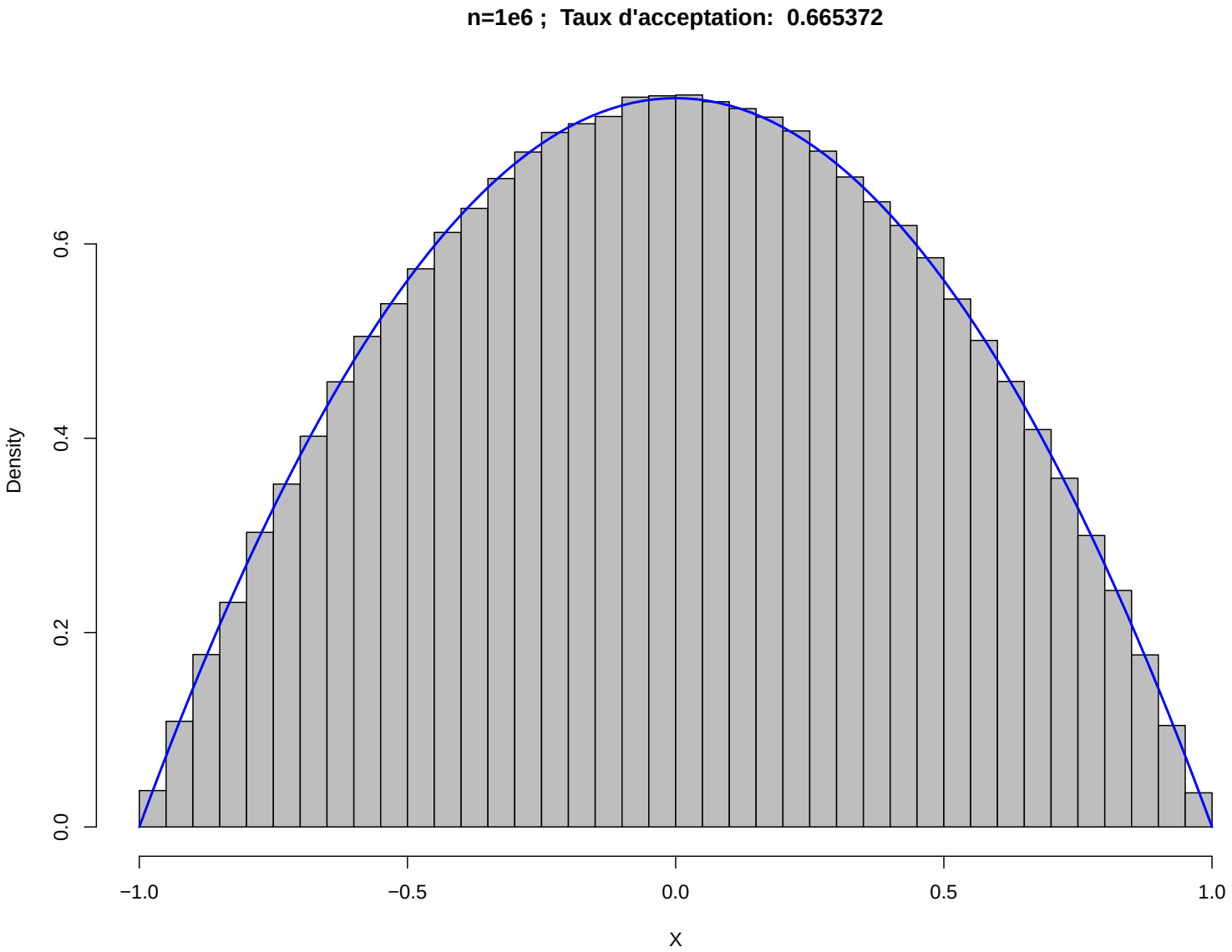
```
f=function(x)  (3/4)*(1-x^2)
curve(f,-1,1)
grid(10)
segments(-1,3/4,1,3/4,col="blue")
segments(-1,3/4,-1,0,col="blue")
segments(-1,0,1,0,col="blue")
segments(1,0,1,3/4,col="blue")
n=100;a=-1;b=1;M=3/4;
U1=runif(n);  X=(b-a)*U1+a
U2=runif(n);Y=M*U2
Acc=Y<=f(X)
col=2-Acc
points(X,Y,pch=19,col=col,cex=1)
```

prog9


f(x)
0.0
0.2
0.4
0.6
−1.0
−0.5
0.0
0.5
1.0
x

```
n=1e6 ;  a=-1;b=1; M=3/4;
U1=runif(n) ;  X=(b-a)*U1+a
U2=runif(n) ;Y=M*U2
Acc=Y<=f(X)
X=X[Acc]
hist(X,nclass=35,freq=FALSE,col="gray",main="")
curve(f,add=TRUE,col="blue",lwd=2)
title(paste("n=1e6  ;  Taux d'acceptation : ",
as.character(sum(Acc)/n)))
```

prog10

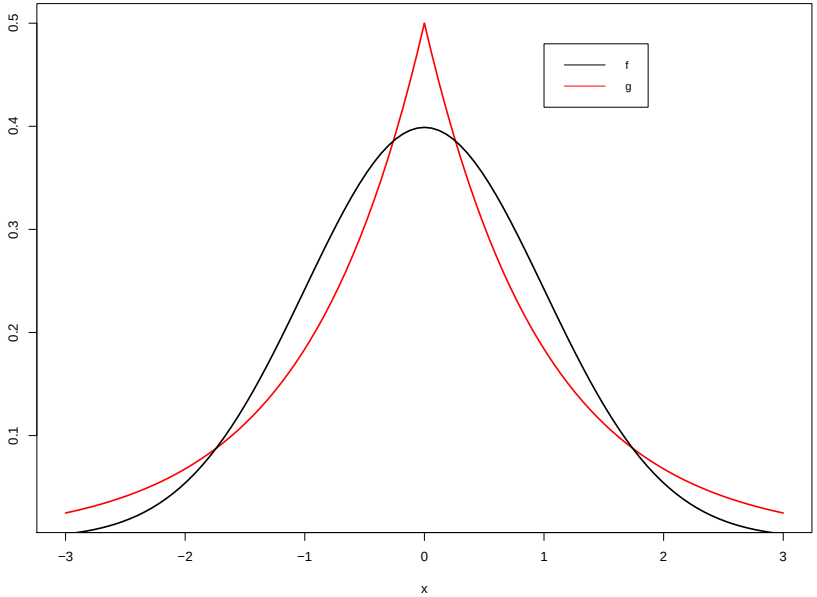

n=1e6 ; Taux d'acceptation: 0.665372
Density
0.0
0.2
0.4
0.6
−1.0
−0.5
0.0
0.5
1.0
X

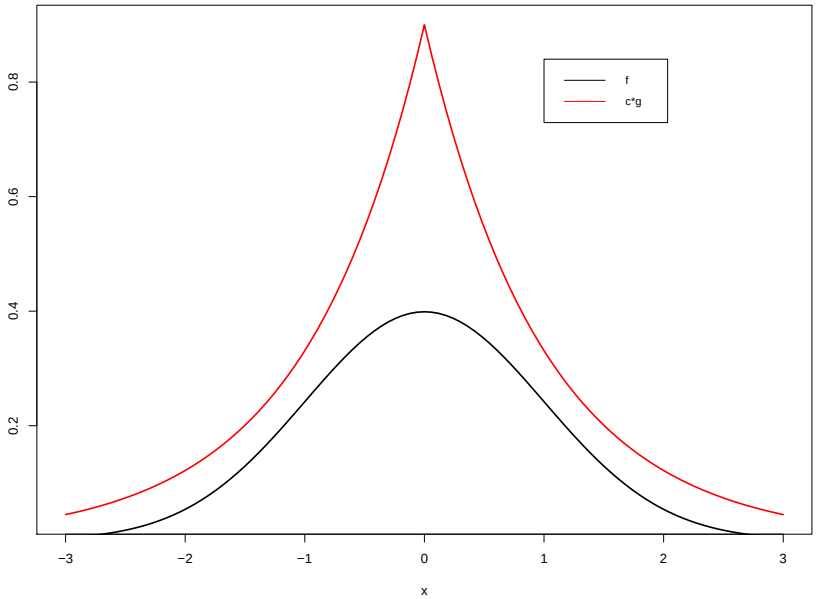
## Méthode de rejet générale

- On cherche à échantillonner la densité de probabilité $f$ sur $\mathbb{R}^d$.
- On suppose qu'on peut générer une suite i.i.d $(Y_n)_{n\geq 1}$ de vecteurs aléatoires de $\mathbb{R}^d$ admettant une densité de probabilité $g$ qui vérifie :

$$\text{pour tout } x \in \mathbb{R}^d,\ \frac{f(x)}{g(x)} \leq c$$

  pour une constante $c$ (nécessairement supérieure ou égale à $1$).
- On dispose également d'une suite i.i.d $(U_n)_{n\geq 1} \sim \mathcal{U}[0,1]$.


f
g
0.1
0.2
0.3
0.4
0.5
−3
−2
−1
0
1
2
3
x


f
c*g
0.2
0.4
0.6
0.8
−3
−2
−1
0
1
2
3
x

# Théroème de rejet

Soient $(Y_n)_{n\in\mathbb{N}}$ une suite i.i.d ayant comme densité $g$.

Soient $(U_n)$ une suite i.i.d $\sim \mathcal{U}[0,1]$

Les deux suites sont indépendantes.

## Théroème de rejet

Soit $\tau$ le premier indice $n \geq 1$ tel que $U_n \leq \dfrac{f(Y_n)}{cg(Y_n)}$

et posons $Z = Y_\tau$. Alors :

**(a)** $Z$ admet la densité de probabilité $f$, et :

**(b)** $E[\tau] = c$.

## Démonstration.

On a

$$P(Z \leq x) = P(Y_\tau \leq x) = \sum_{n \geq 1} \mathbb{P}(\tau = n, Y_n \leq x).$$

Soit $A_k$ l'évènement $\{U_k > \dfrac{f(Y_k)}{cg(Y_k)}\}$.

On a :

$$\begin{aligned} \mathbb{P}(\tau = n, Y_n \leq x) &= \mathbb{P}(A_1, \ldots, A_{n-1}, \bar{A}_n, Y_n \leq x) \\ &= \mathbb{P}(A_1) \ldots \mathbb{P}(A_{n-1}) \mathbb{P}(\bar{A}_n, Y_n \leq x) \end{aligned}$$

$$
\begin{aligned}
\mathbb{P}(\bar{A}_k\,, Y_k \leq x) &= \int_{\mathbb{R}^d} \mathbb{P}(U_k \leq \frac{f(y)}{cg(y)})\mathbf{1}_{\{y \leq x\}} g(y)dy \\
&= \int_{-\infty}^{x} \frac{f(y)}{cg(y)} g(y)dy \\
&= \int_{-\infty}^{x} \frac{f(y)}{c} dy \\
&= \frac{1}{c}\int_{-\infty}^{x} f(y)dy
\end{aligned}
$$

$$\begin{aligned}\mathbb{P}(Z \leq x) &= \sum_{n\geq 1}(1-\frac{1}{c})^{n-1}\frac{1}{c}\int_{-\infty}^{x} f(y)dy \\ &= \int_{-\infty}^{x} f(y)dy.\end{aligned}$$

**(b).** Aussi, en utilisant les calculs ci-dessus,

$$\begin{aligned}\mathbb{P}(\tau = n) &= \mathbb{P}(A_1, \ldots, A_{n-1}, \bar{A}_n) \\ &= \mathbb{P}(A_1)\ldots\mathbb{P}(A_{n-1})\mathbb{P}(\bar{A}_n) \\ &= \left(1-\frac{1}{c}\right)^{n-1}\frac{1}{c},\end{aligned}$$

d'où il suit que $\tau$ suit une loi géométrique de paramètre $\frac{1}{c}$ et donc $E(\tau) = c$.

## Shcéma général de la méthode de rejet

On voudrait simuler une variable aléatoire réelle $X$ de densité de probabilité $f$.

On suppose

- qu'il existe une autre densité de probabilité $g$ telle que le ratio $\frac{f}{g}$ soit borné, disons par $c$ ($f \leq cg$ ),
- qu'on sache simuler $Y$ de densité $g$ .

La version basique de la méthode de rejet prend la forme suivante :

**Algorithme (Méthode de rejet)**

1. Générer $Y \to g$, $U \to \mathbb{U}_{[0,1]}$.
2. Accepter $X = Y$ si $U \leq f(Y)/cg(Y)$ ;
3. Revenir au $1$ sinon.

# Méthode de rejet : Simulation de la loi normale

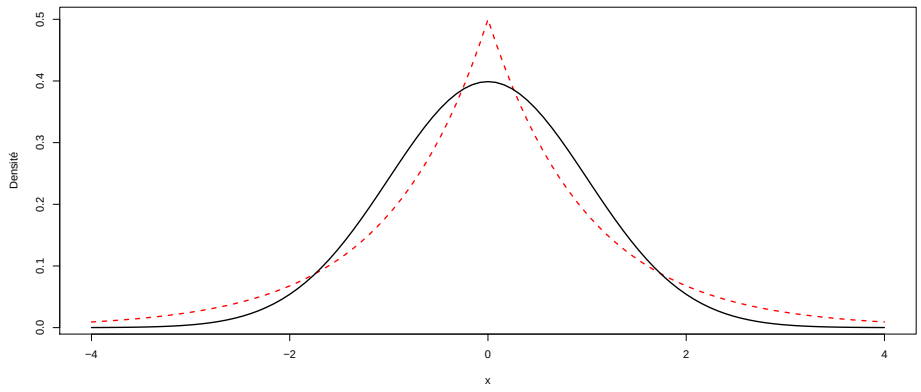



**Figure –** $f$ : la densité normale. $g$ : la densité double exponentielle.

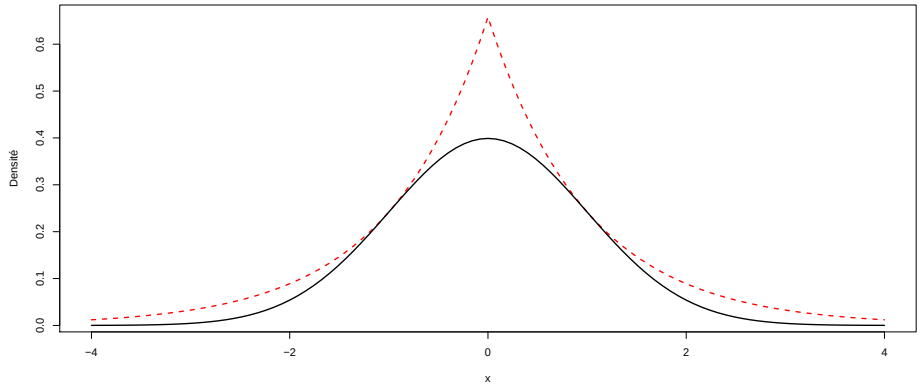
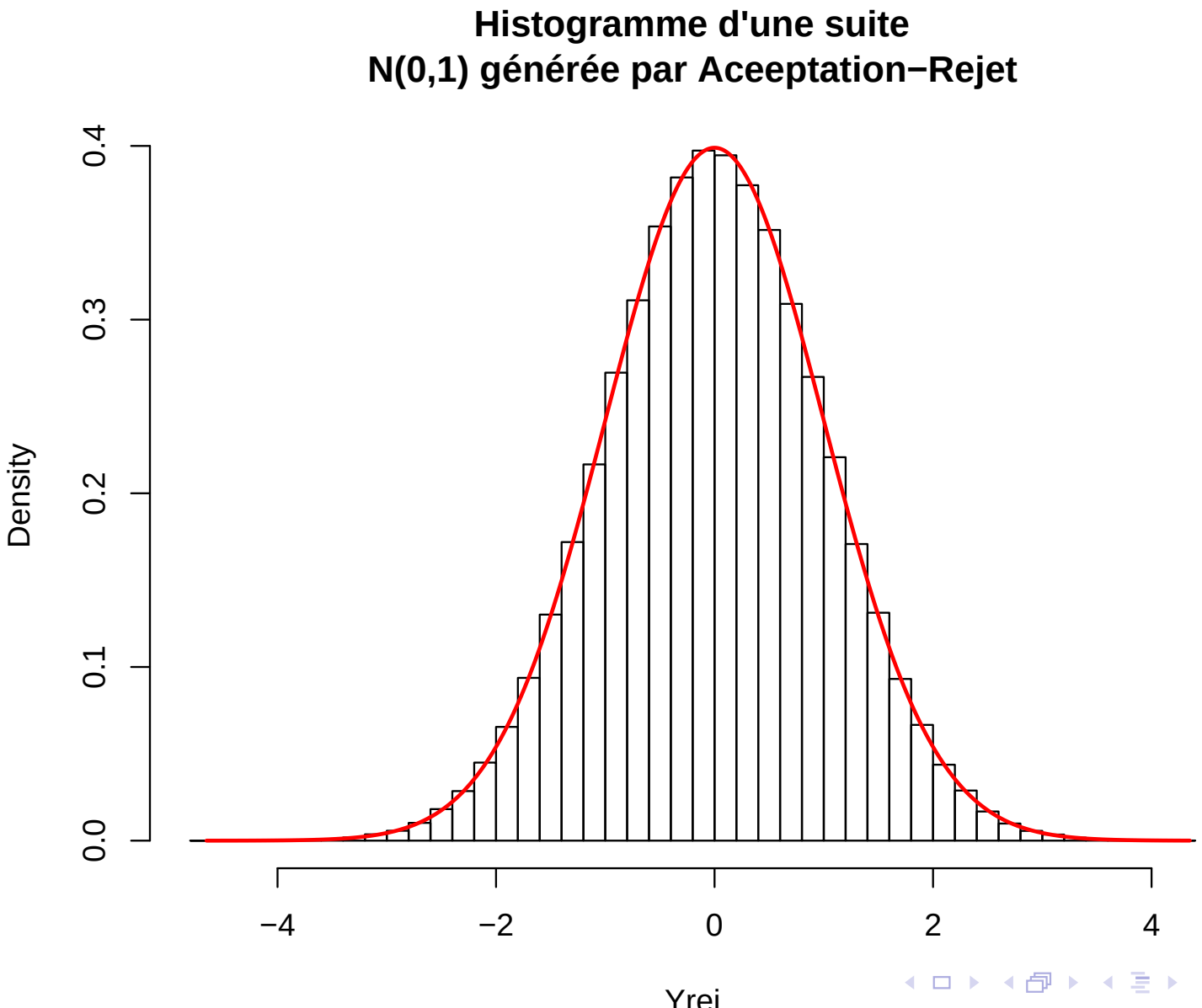



**Figure –** $f$ : la densité normale. $g$ : la densité double exponentielle. $c = \sqrt{\frac{2e}{\pi}}$(optimal)

# Méthode de rejet : Simulation de la loi normale (Voir TP)



## Le $c$ optimal

On remarque que le taux d'acceptation est égale à $\frac{1}{c}$.

Pour générer une variable de densité $f$, on doit générer en moyenne $c$ variables suivant la densité instrumentale $g$. ($E(\tau) = c$.)

On a intérêt à choisir le plus petit $c$ possible.

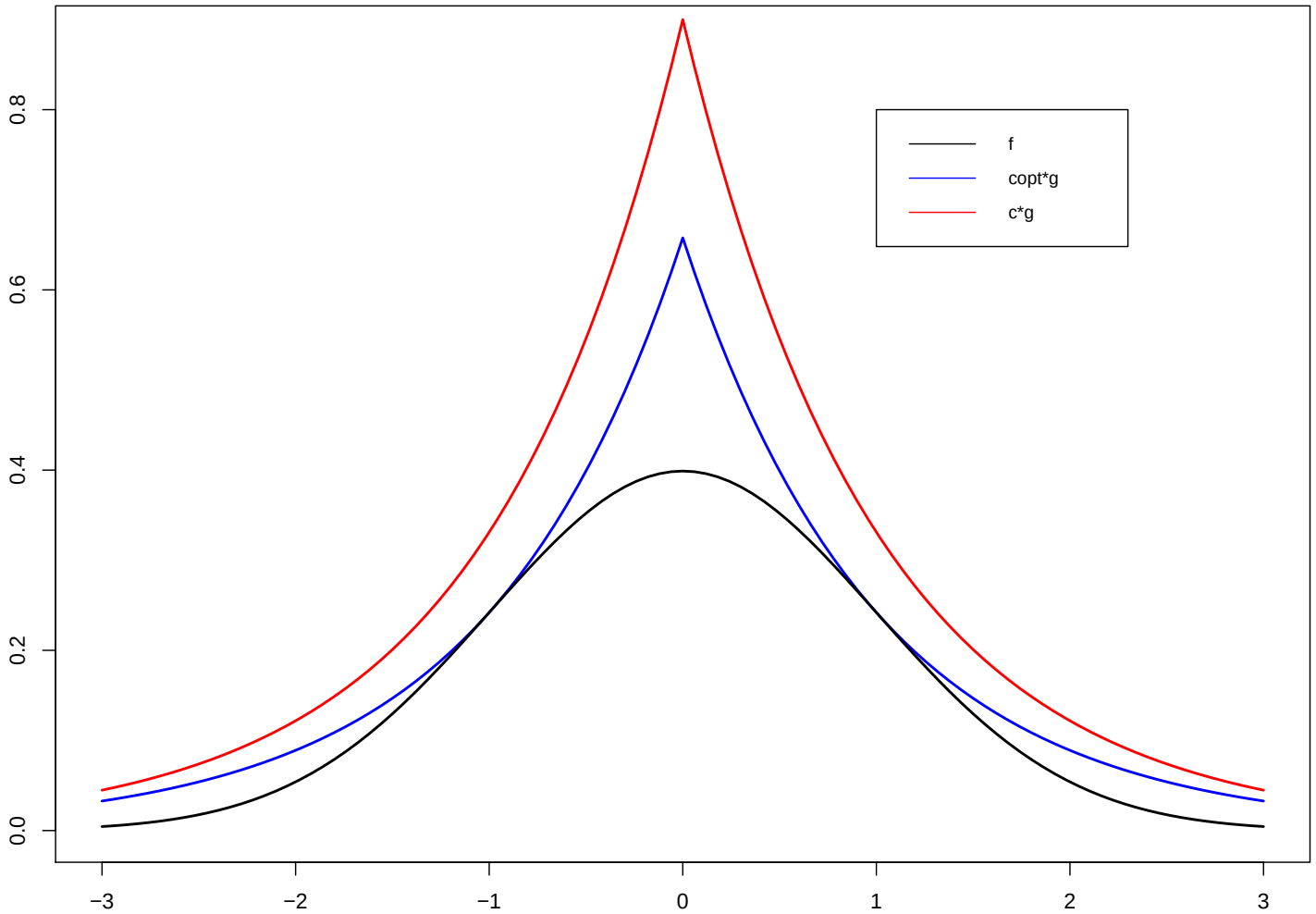
Comme $c$ doit vérifier la condition $c \geq \frac{f(x)}{g(x)}$. C'est à dire que $c$ doit être un majorant de $\frac{f(x)}{g(x)}$.

Alors le choix optimal est

$$\boxed{c_{\textbf{opt}} = \sup_x \frac{f(x)}{g(x)}}$$

# Le $c$ optimal

$$c_{\textbf{opt}} = \sup_x \frac{f(x)}{g(x)}$$



## Exemple : $f(x) \propto e^{-x^3}$

Soit

$$f(x) = \frac{e^{-x^3}}{Z} \mathbf{1}_{x \geq 1}$$

avec

$$Z = \int_1^{+\infty} e^{-x^3} dx$$

**(la constante $Z$ n'est pas connue mais nous n'en aurons pas besoin).**

Soit $g(x) = \frac{\mathbf{1}_{x \geq 1}}{x^2}$.

Les fonctions $f$ et $g$ sont des densités de probabilité.

Il est facile de simuler suivant la loi de densité $g$ en utilisant la technique d'inversion de la fonction de répartition.

## Exemple : $f(x) \propto e^{-x^3}$

Nous commençons par étudier la fonction $h : x \geq 1 \mapsto x^2 e^{-x^3}$.

Nous avons $h'(x) = \left(2x - 3x^4\right) e^{-x^3} \leq 0$ pour tout $x \geq 1$.

Donc, pour tout $x \geq 1, h(x) \leq h(1) = e^{-1}$

Nous en déduisons (pour $x \geq 1$)

$$f(x) \leq \frac{1}{eZ} g(x)$$

Nous posons $c = 1/(eZ)$. Soit

$$\alpha(x) = \frac{f(x)}{cg(x)} = x^2 e^{-x^3+1}$$

Nous ne connaissons pas $Z$ mais nous pouvons calculer $\alpha(x)$ pour tout $x \geq 1$ donc nous pouvons utiliser l'algorithme de simulation par rejet pour simuler suivant la densité $f$.

La densité instrumentale $g$ est facile à simuler. Nous allons utiliser la méthode d'iversion.

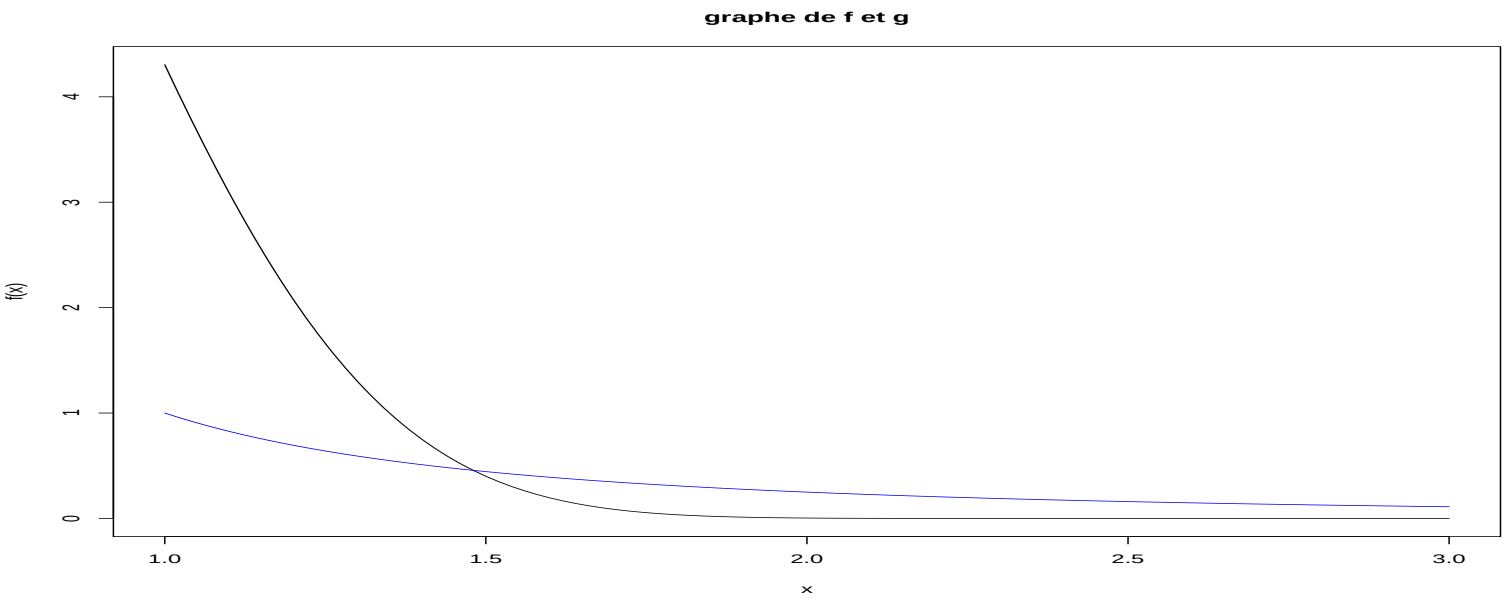
par inversion de la fonction de répartition :

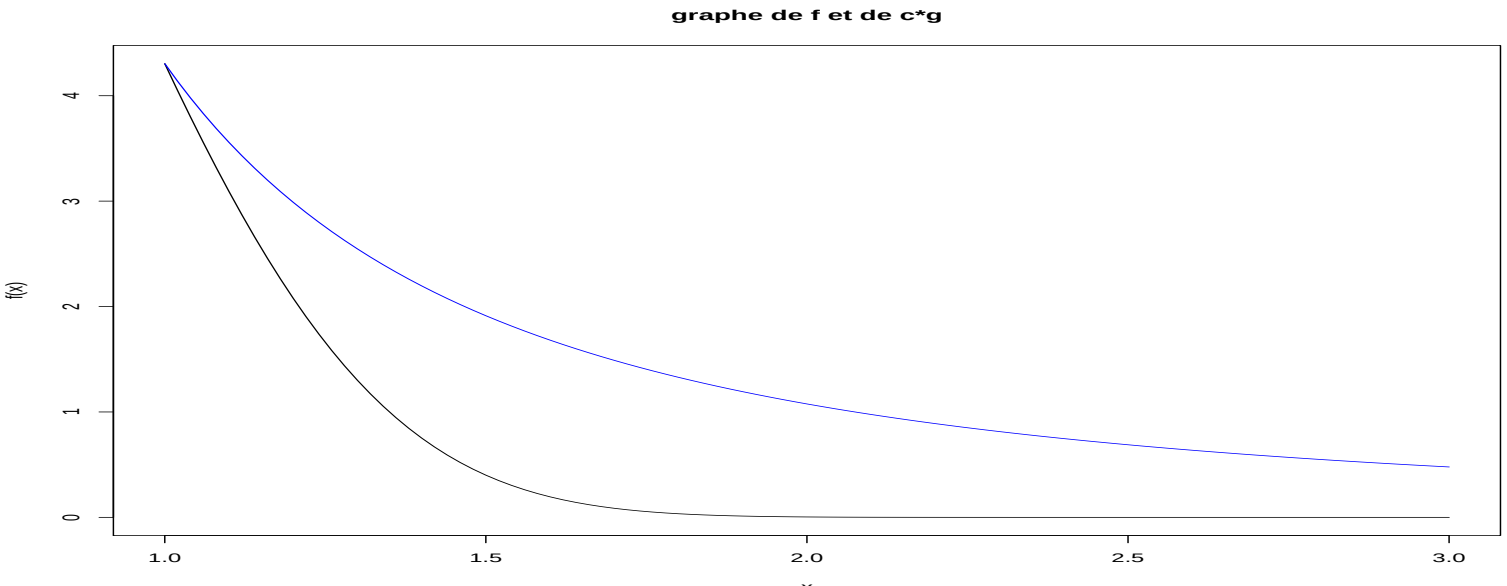
$$\begin{aligned}
\text{pour } x \geq 1, G(x) &= \int_1^x g(u)du \\
&= \left[-\frac{1}{u}\right]_1^x \\
&= 1 - \frac{1}{x} \\
\text{pour } u \in [0;1], G^{-1}(u) &= \frac{1}{1-u}
\end{aligned}$$

## Graphes : Programmes

```
m=function(x) exp(-x^3)
Z=integrate(m,1,Inf)$value
f=function(x) m(x)/Z
g=function(x) x^(-2)
curve(f,1,3)
curve(g,add=T, col="blue")
title("graphe de f et g")
### Recherche du c optimal
h=function(x) f(x)/g(x)
curve(h,1,3)
c=optimise(h,lower=1,upper=10,maximum=TRUE)$objective
cg=function(x) c*g(x)
curve(f,1,3) ;curve(cg,add=T, col="blue")
title("graphe de f et de c*g")
```

# Graphes


graphe de f et g
f(x)
0
1
2
3
4
1.0
1.5
2.0
2.5
3.0
x

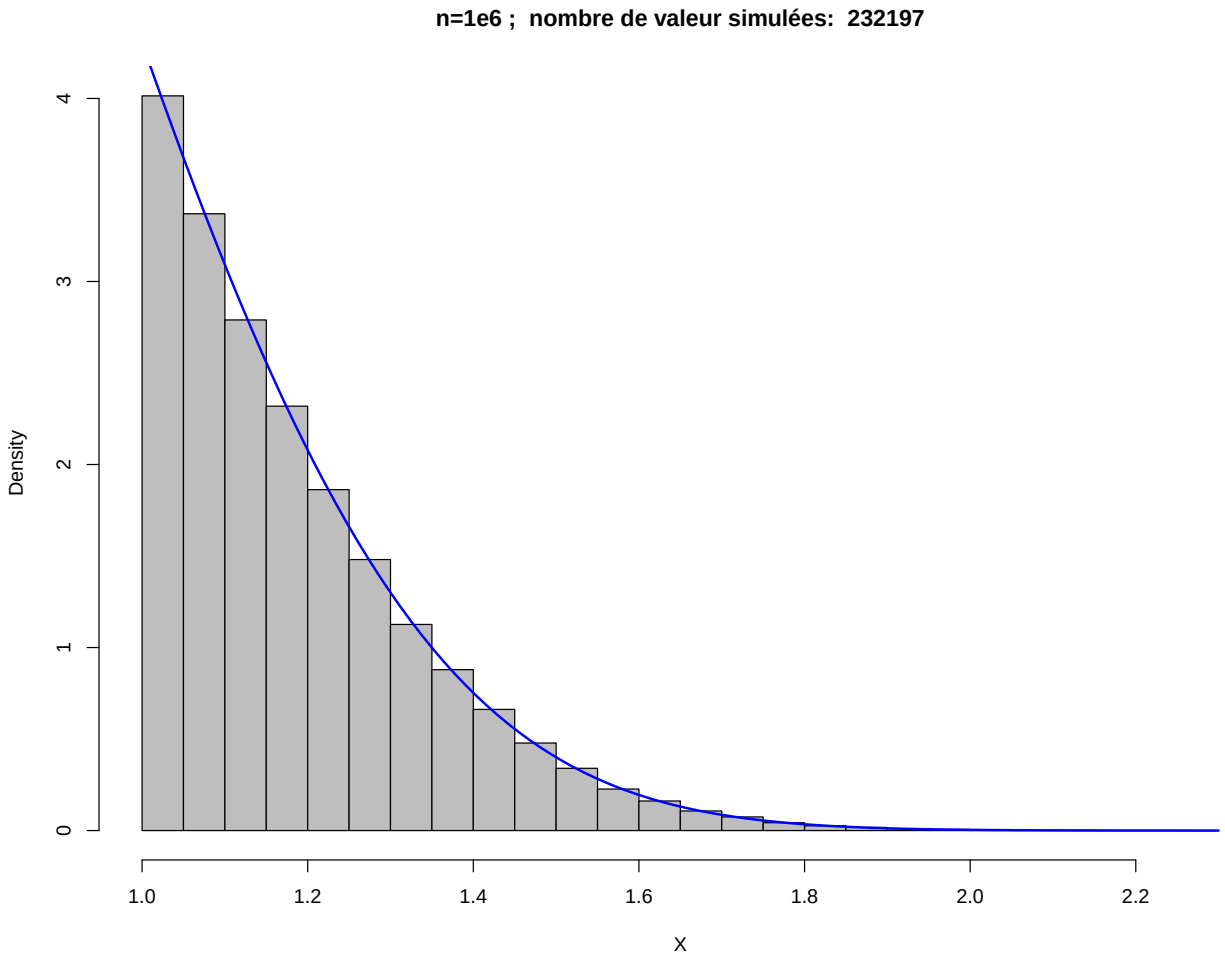

graphe de f et de c*g
f(x)
0
1
2
3
4
1.0
1.5
2.0
2.5
3.0

## Avec calcul la constante de normalisation :Programme

```
n=1e6
V=runif(n)
X=1/(1-V)
U=runif(n)
Acc=(c*g(X)*U <f(X))
X=X[Acc]
hist(X,freq=FALSE,col="gray",nclass=35,main="")
curve(f,add=T,col="blue",lwd=2)
title(paste("n=1e6 ;  nombre de valeur simulées : ",
as.character(sum(Acc))))
```

prog12

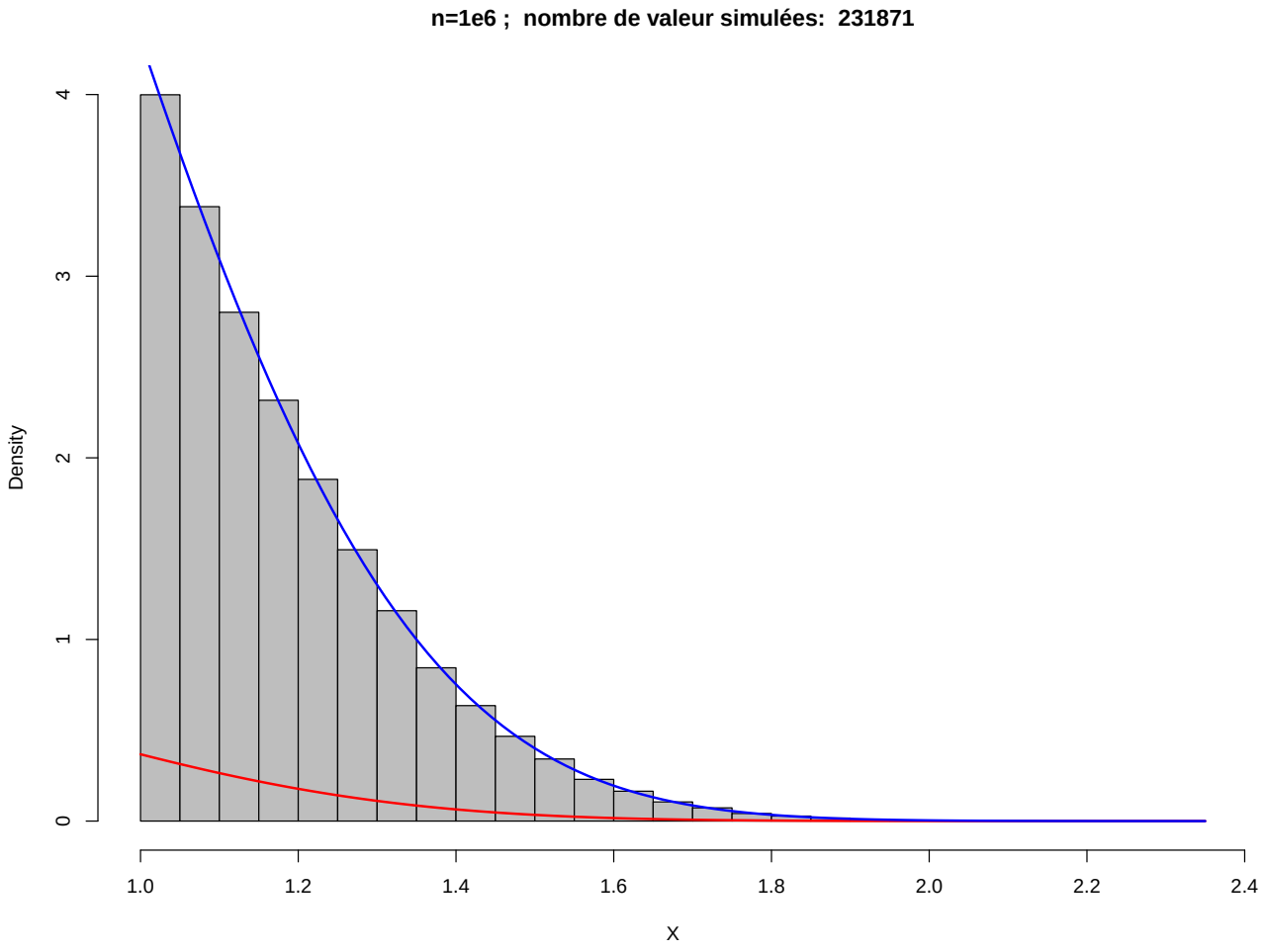
# Avec caclul de la constante de normalisation :Graphes



## Sans calculer la constante de normalisation

```
f=function(x) exp(-x^3) ; g=function(x) x^(-2)
h=function(x) f(x)/g(x)
curve(h,1,3)
c=optimise(h,lower=1,upper=10,maximum=TRUE)$objective
n=1e6 ;V=runif(n) ; X=1/(1-V)
U=runif(n); Acc=(c*g(X)*U <f(X))
X=X[Acc]
hist(X,freq=FALSE,col="gray",nclass=35,main="")
curve(f,add=T,col="red",lwd=2)
Z=integrate(f,1,Inf)$value
f2=function(x) f(x)/Z
curve(f2,add=T,col="blue",lwd=2)
title(paste("n=1e6 ; nombre de valeur simulées : ",
as.character(sum(Acc))))
```

# Sans calculer la constante de normalisation



## Avantages de la méthode de rejet

**Avantages**

1. On n'a pas besoin de calculer ni la fonction de répartition $F$ ni $F^{-1}$
2. On n'a pas besoin de calculer les constantes de normalisations.

**Attention**

Il faut faire attention à deux choses :

1. Le taux d'acceptaion ne doit pas être très faible.
2. La fonction instrumentale $g$ doit être relativement simple à simuler.

## La méthode de rejet est valable pour les lois discrètes

L'algorithme ci-dessus est encore valable si la v.a. $X$ à simuler a une densité $f$ par rapport à une mesure $\mu$ quelconque et que cette densité est majorée par $kg$ où $g$ est la densité par rapport à $\mu$ d'une variable $Y$ facile à simuler. Ceci se traduit par (pour tout $A$ )

$$\mathbb{P}(X \in A) = \int_A f(x)\mu(dx) \leq \int_A kg(x)\mu(dx) = k\mathbb{P}(Y \in A)$$

Si la v.a. $X$ a une loi portée par un ensemble discret $E$, on peut choisir pour $\mu$ la mesure de comptage sur $E$ et la méthode de rejet est aussi valable pour les lois discrètes, $f(x)$ étant dans ce cas égale à $\mathbb{P}(X = x)$

## Simulation d'une loi discrète finie par rejet

Reprenons l'exercice qui nous demande de simuler une loi discrète de support $a = (a_1, \ldots, a_m)$ fini avec des probabilités $p = (p_1, \ldots, p_m)$.

La loi à simuler est $f(a_k) = p_k$.

Ici la loi instrumentale $g$ est l'uniforme discrète. $g(a_k) = \frac{1}{m}$

Le $c$ optimal est

$$\sup_k \frac{f(a_k)}{g(a_k)} = \sup_k (mp_k) = m \sup_k p_k.$$

Ça nous donne le programme suivant :

# Programme pour une loi discrète à support fini

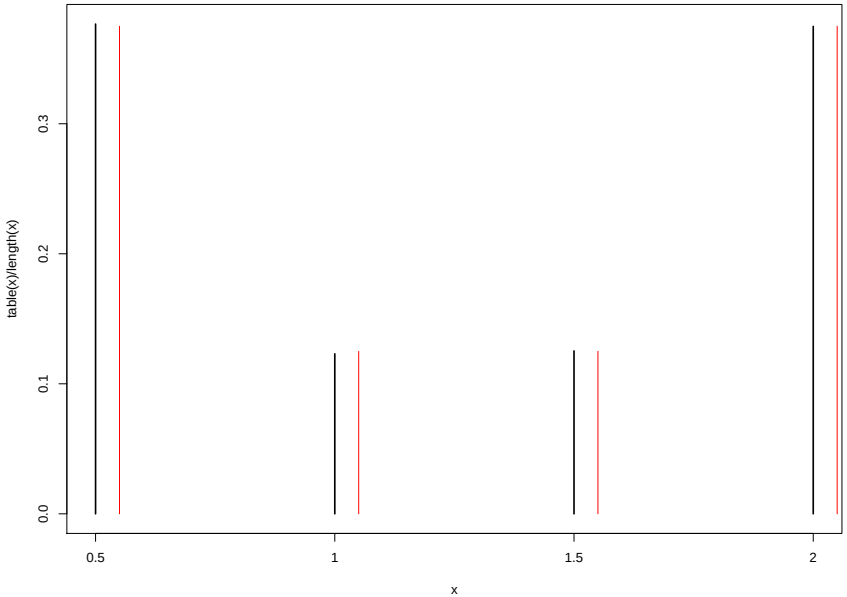
```
rsamp2=function(n,a,p){
m=length(a)
c=max(m*p)
N=n*c*1.3
X=floor(m*runif(N))+1
U=runif(N)
Acc=p[X]>(U*c/m)
X=X[Acc]
X=X[1:n]
return(a[X])
}
```

prog14

# Programme pour une loi discrète à support fini

```
p=c(3/8,1/8,1/8,3/8)
a=c(0.5,1,1.5,2)
x=rsamp2(10000,a,p)
plot(table(x)/length(x))
points(a+0.05,p,col="red",type="h")
```

prog15



# Méthode de comparaison pour une loi discrète à support fini

```
rsamp<-function(n,a,p){
m=length(a)
P=rep(0,m)
P[1]=p[1]
for(k in 2:m) P[k]=P[k-1]+p[k]
U=runif(n)
X=rep(0,n)
for(i in 1:n) {
for(k in 1:m) {   if(U[i]<=P[k]) {X[i]=a[k];      break;}
}
}
return(X)
}
```

prog16

## Temps d'execution

```
m=150
p=(runif(m,0,1))
p=p/sum(p)
a=1:m
start=Sys.time()
x=rsamp2(1e5,a,p)
end=Sys.time()
temps.rej=end-start
start=Sys.time()
x=rsamp(1e5,a,p)
end=Sys.time()
temps.comp=end-start
as.numeric(temps.comp)/as.numeric(temps.rej)
###11.59094
```

## Vérification par un histogramme

Quand on simule une variable aléatoire réelle à l'aide d'une des méthodes, on peut vérifier empiriquement que la loi simulée est bien celle que l'on voulait. Soient $X_1, X_2, \ldots$ des v.a.r. i.i.d. de loi de densité $f$ (avec une dérivée $f'$ vérifiant $\left\|f'\right\|_\infty < C$ pour une certaine constante $C$ ). Pour tout $a < b$

$$\frac{1}{n(b-a)} \sum_{i=1}^{n} \mathbf{1}_{[a;b]}(X_i) \underbrace{\text{p.s.}}_{n \to +\infty} \frac{\mathbb{P}(X_1 \in [a;b])}{b-a}$$

et

$$\frac{\mathbb{P}(X_1 \in [a;b])}{b-a} = \frac{1}{b-a} \int_a^b f(t)dt$$

## Vérification par un histogramme

Donc

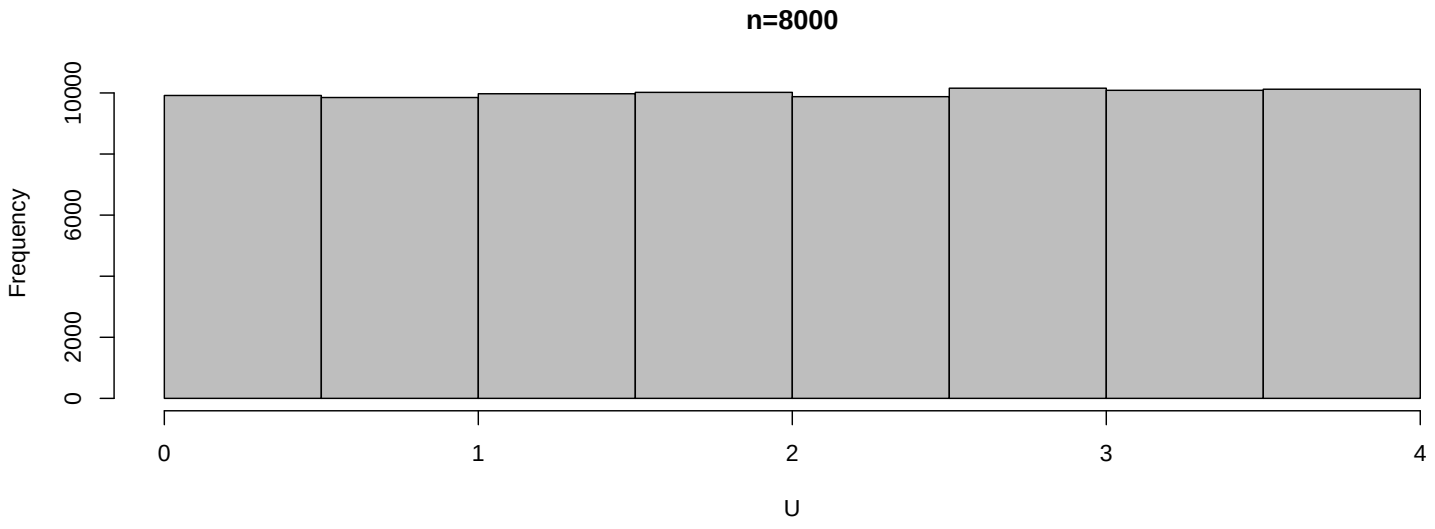
$$\begin{aligned}\left|\frac{\mathbb{P}(X_1 \in [a;b])}{b-a} - f(a)\right| &= \left|\frac{1}{b-a}\int_a^b f(t) - f(a)dt\right| \\ &\leq \frac{1}{b-a}\int_a^b |f(t) - f(a)|dt \\ &\leq \frac{1}{b-a}\int_a^b C|t-a|dt \\ &= \frac{C(b-a)}{2} \underset{b\backslash a}{\longrightarrow} 0\end{aligned}$$

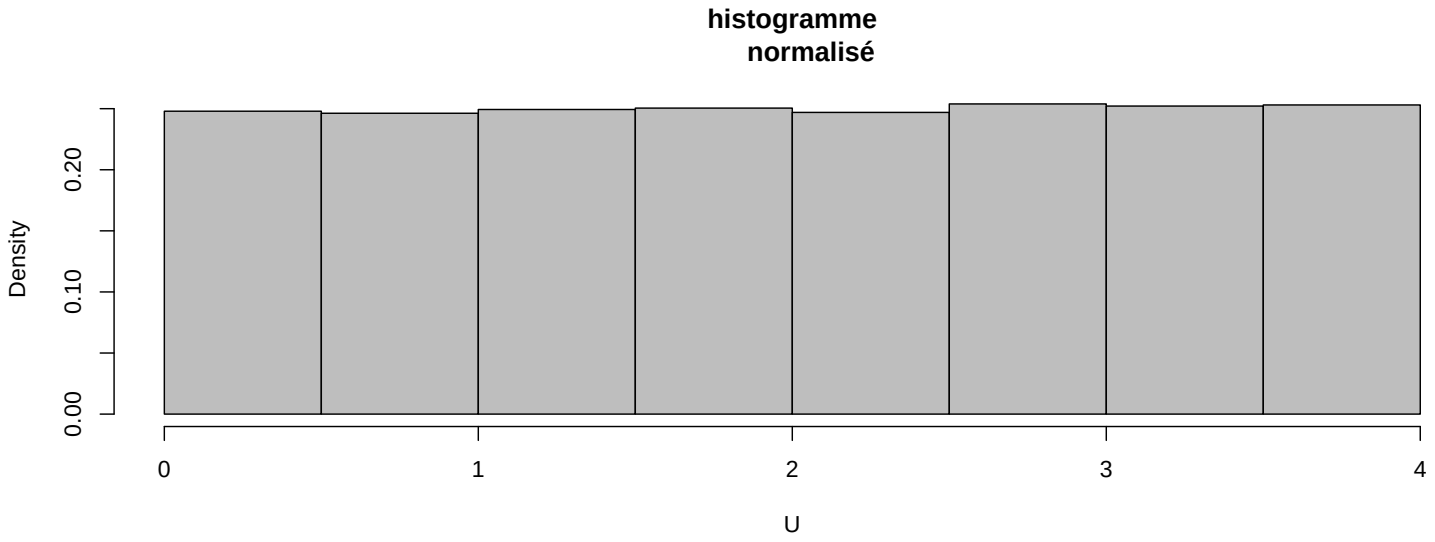
Donc l'histogramme (correctement renormalisé) des $n$ valeurs simulées $X_1, X_2, \ldots, X_n$ est proche de la densité $f$.

Avec le langage R, pour obtenir un histogramme normalisé, on met l'argument proba à la valeur TRUE.

```
par(mfrow=c(2,1))
U=runif(8e4,0,4)
hist(U,nclass=8,col="gray",main=("n=8000"))
hist(U,nclass=8,col="gray",prob=TRUE,main="histogramme
normalisé")
```

prog18


n=8000
Frequency
0
2000
6000
10000
0
1
2
3
4
U


histogramme
normalisé
Density
0.00
0.10
0.20
0
1
2
3
4
U

# Méthode de décomposition

**Exemple simple : Cas continu**

- Soit $f = \frac{1}{3} f_1 + \frac{2}{3} f_2$. Donc $F = \frac{1}{3} F_1 + \frac{2}{3} F_2$.
- Soit $X_1 \sim f_1$ et $X_2 \sim f_2$.
- Soit $N \sim B(1/3)$ : Les variables $X_1, X_2, N$ sont indépedantes.

On Pose

$$X = X_1 \text{ si } N = 1 \text{ et } X = X_2 \text{ si } N = 0.$$

Trouvons $F_X$.

$$\begin{aligned} F_X(x) &= P(X \leq x) \\ &= P(X \leq x, N = 1) + P(X \leq x, N = 0) \\ &= P(X_1 \leq x) P(N = 1) + P(X_2 \leq x) P(N = 0) \\ &= \frac{1}{3} F_1(x) + \frac{2}{3} F_2(x) \end{aligned}$$

# Méthode de décomposition

Soit $X$ une variable aléatoire de type mélange. Sa densité se met sous la forme :

$$f = \sum_{n \geq 0} p_n f_n$$

où $(f_n)_{n\geq 0}$ est une famille de densités de probabilité (i.e. $f_n(x) \geq 0$ et $\int_R f_n(x)dx = 1$ ) et $(p_n)_{n\geq 0}$ définit une loi de probabilité i.e. $p_n \geq 0$ et $\sum_{n\geq 0} p_n = 1$. On a le théorème suivant :

**Théorème : Mélange**

Soit $(X_n)_{n\geq 0}$ une famille de variables aléatoires tq $X_n \sim f_n$.
Soit $N$ une variable aléatoire à valeurs entières de loi $(p_n)_{n\geq 0}$ indépendante de la suite $(X_n)_{n\geq 0}$.
Alors la variable aléatoire $X_N$ a pour densité $f$

La preuve de ce résultat est facile. Elle est similaire à l'exemple traité ci-dessus.

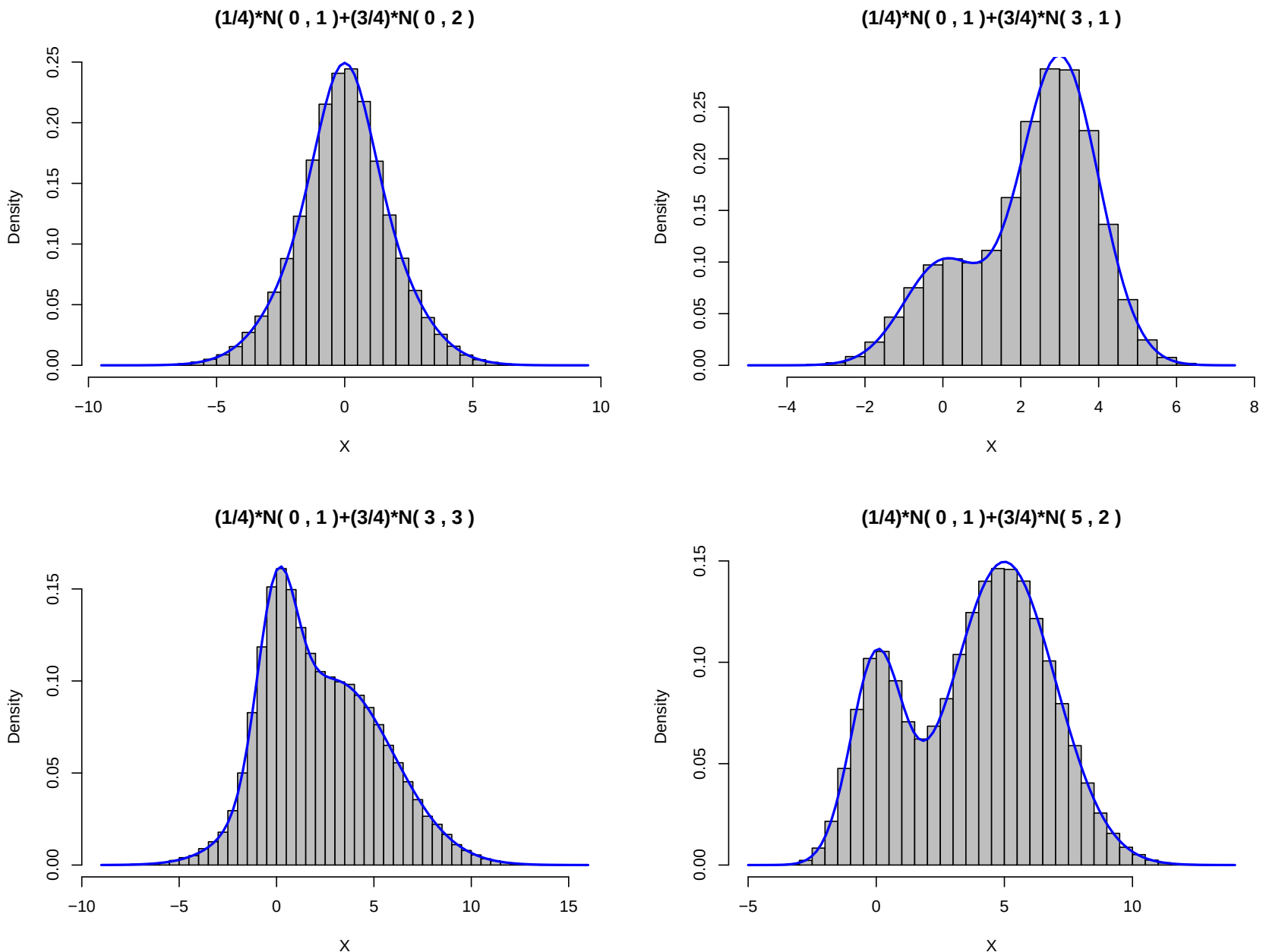
## Exemple

1. A l'aide de ce théorème, simuler un mélange de lois gaussiennes : par exemple, une variable aléatoire de densité

$$f(x) = \frac{1}{4}\frac{e^{-x^2}}{\sqrt{2\pi}} + \frac{3}{4}\frac{e^{-(x-5)^2}}{\sqrt{2\pi}}$$

2. Vérifier que la distribution des points obtenus est bien celle attendue en traçant, sur un même graphique, histogramme des points ainsi que la densité attendue. On observera l'aspect bimodal de la distribution.

```
p1=1/4; p2=3/4;m1=0;sigma1=1;m2=5;sigma2=2;
f=function(x) p1*dnorm(x,m1,sigma1)+p2*dnorm(x,m2,
    sigma2)
### Avec boucle
n=1e4;X=numeric(n)
for(j in 1:n) {
U=runif(1)
if(U<p1) X[j]=rnorm(1,m1, sigma1) else
X[j]=rnorm(1,m2, sigma2)}
## sans boulce
n=1e5;X1=rnorm(n,m1,sigma1);X2=rnorm(n,m2,sigma2)
U=runif(n);X=X1*(U<p1)+X2*(U>p1)
hist(X,nclass=40,freq=FALSE,col="gray",main="")
curve(f,add=T,col="blue",lwd=2)
title(main= paste("(1/4)*N(",as.character(m1),",",as.
    character(sigma1),")+(3/4)*N(",as.character(m2),","
    ,as.character(sigma2),")"))
```


(1/4)*N( 0 , 1 )+(3/4)*N( 0 , 2 )
Density
X
(1/4)*N( 0 , 1 )+(3/4)*N( 3 , 1 )
Density
X
(1/4)*N( 0 , 1 )+(3/4)*N( 3 , 3 )
Density
X
(1/4)*N( 0 , 1 )+(3/4)*N( 5 , 2 )
Density
X

On souhaite maintenant généraliser la méthode précédente à des densités de la forme

$$f = \sum_{n \geq 0} q_n g_n$$

où $(g_n)_{n\geq 0}$ n'est plus nécessairement une famille de densités de probabilité (i.e. $g_n$ satisfait $g_n(x) \geq 0$ mais $\int_R g_n(x)dx \neq 1$ ) et $(q_n)_{n\geq 0}$ satisfont uniquement $q_n \geq 0$. On conserve le fait que $\int_R f(x)dx = 1$.
Dans ce cas, on se ramène à l'écriture précédente en écrivant la décomposition :

$$f = \sum_{n \geq 0} q_n \left( \int_R g_n(x)dx \right) \frac{g_n}{\left(\int_R g_n(x)dx\right)}$$

On est donc dans la situation de lois de type mélange en posant
$p_n = q_n \left(\int_R g_n(x)dx\right)$ et $f_n = \frac{g_n}{\left(\int_R g_n(x)dx\right)}$

## Simulation de vecteurs aléatoires

- Si les composantes du vecteur sont indépendantes, il suffit de simuler chaque composante.
- On peut simuler une composante, puis des lois conditionnelles successives.
- Ainsi, si $(X,Y)$ désigne un couple de variables aléatoires réelles ou vectorielles de densité $f(x,y)$, la densité de $X$ est $f_X(x) = \int f(x,y)dy$ et la densité conditionnelle de $Y$ sachant $X = x$ est $f(y|x) = \frac{f(x,y)}{f_X(x)}$

- Pour simuler le couple $(X, Y)$, on simule d'abord $X$ de densité $f_X$, puis ayant obtenu la valeur réelle (ou vectorielle) $x$, on simule $Y$ de densité $f(.|x)$ indépendamment de $X$.
  En procédant ainsi pas à pas, on peut simuler des vecteurs aléatoires de dimension quelconque.
- on peut enfin utiliser les changements de variables, comme on l'a fait dans la méthode de Box-Muller pour couple de variables gaussiennes $\mathcal{N}(0, 1)$ indépendantes.

- On suppose que la densité de la loi que l'on veut simuler est

$$f(x) = \int g(x,y)dy$$

où $g$ est une fonction borélienne positive.

- Puisque $f$ est une densité, le théorème de Fubini montre que $\int g(x,y)dxdy = \int f(x)dx = 1$, c'est à dire que $g$ est une densité.
- Si $(X,Y)$ désigne un couple de densité $g$, la densité de $Y$ est $g_Y(y) = \int g(x,y)dx$ tandis que la densité conditionnelle de $X$ sachant $Y = y$ est $g(x|y) = \frac{g(x,y)}{g_Y(y)}$.

- Si les densités $g_Y$ et $g(.|y)$ sont aisément simulables, on simule d'abord $Y$ de densité $g_Y$, puis ayant obtenu $y$ on simule la densité conditionnelle $g(\cdot|y)$ indépendamment de $Y$, ce qui fournit une simulation de $X$ de densité $f$.
- De nouveau, les densités considérées ne sont pas nécessairement par rapport à la mesure de Lebesgue, mais peuvent être prises par rapport à une mesure positive $\sigma$ -finie quelconque.

Par exemple, si $n$ est un paramètre strictement positif, pour simuler une densité la densité définie par

$$f(x) = n \int_1^{+\infty} y^{-n} e^{-xy} dy \, \text{sur} [0; +\infty[$$

on a $g(x,y) = ny^{-n} e^{-xy} 1_{[0,+\infty[}(x)]_{[1,+\infty[}(y)$

On en déduit que

$$g_Y(y) = ny^{-(n+1)} 1_{[1,+\infty[}(y)$$

et que pour $y \geq 1, g(x|y) = ye^{-xy} 1_{[0,+\infty[}(x)$

C'est à dire que la loi conditionnelle de $X$ sachant $Y = y$ est exponentielle de paramètre $y$.

La fonction de répartition de la loi de $Y$ est

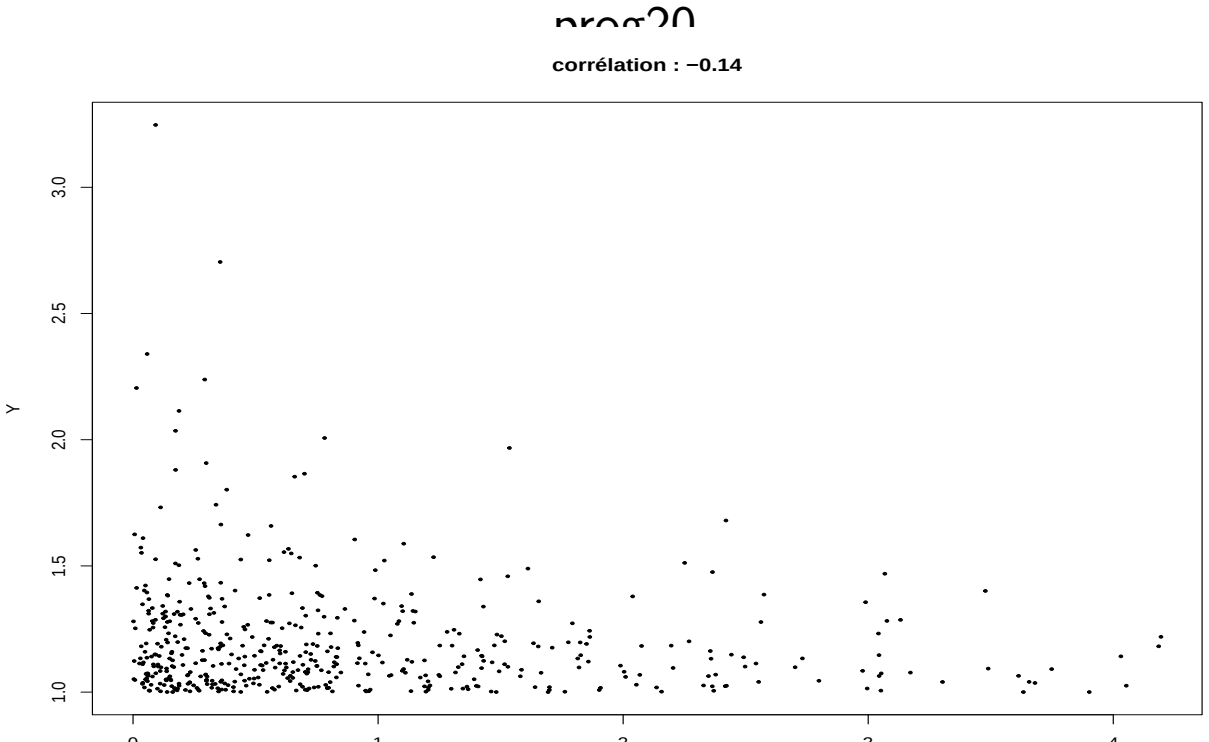
$$F_Y(t) = \left(1 - t^{-n}\right) 1_{[1,+\infty[}(t)$$

$$F_Y^{-1}(u) = (1-u)^{-\frac{1}{n}} 1_{[0,[1}(u)$$

Puisque si $U$ suit une loi $\mathcal{U}([0,1]), 1-U$ suit également une loi $\mathcal{U}([0,1])$, On déduit lalgorithme suivant de simulation de $X$

1. $U \leftarrow \mathcal{U}[0,1]$ ; $Y \leftarrow U^{-1/n}$
2. $V \leftarrow \mathcal{U}[0,1]$ ; $X \leftarrow -\log(V)/Y$.

```
n=6;N=500
Y<-runif(N)^(-1/n)
X<-log(runif(N))/Y
plot(X,Y,pch=19,cex=0.4)
cor=round(cor(X,Y),2)
title(main=paste("corrélation :",as.character(cor)))
hist(X,nclass=40,freq=FALSE,col="gray")
```



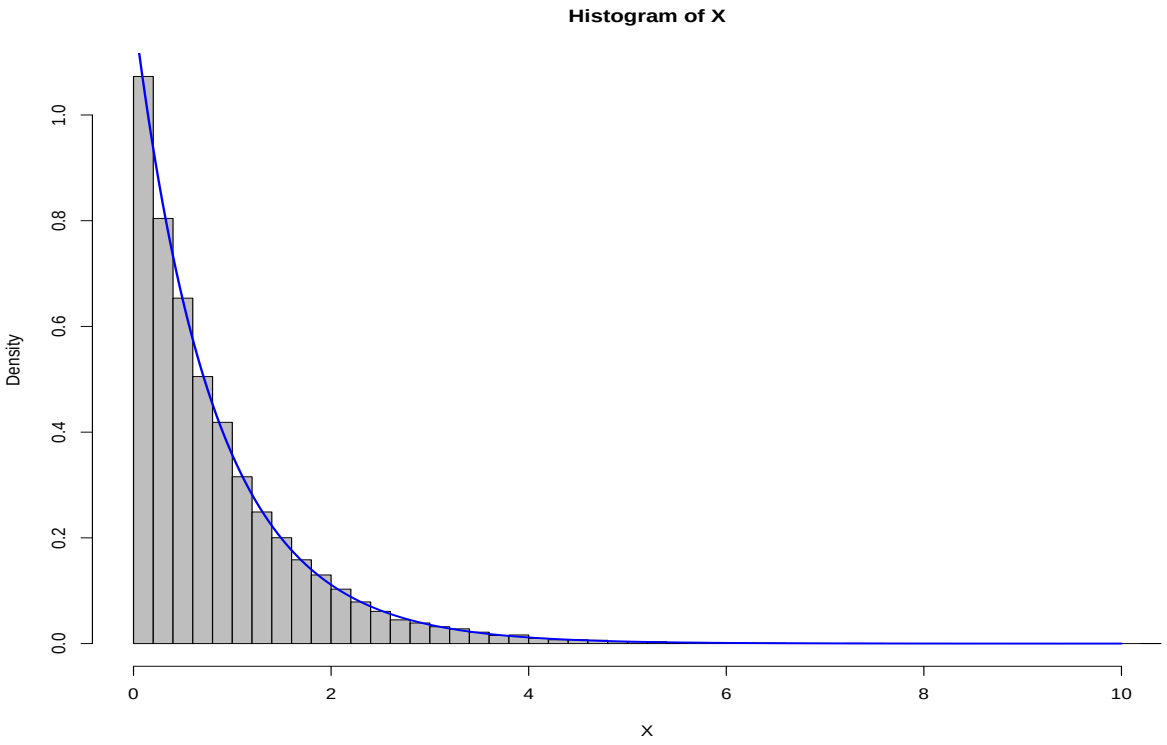
```
n=6
N=50000
f=function(x){
g=function(y) n*y^(-n)*exp(-x*y)
integrate(g,1,Inf)$value
}

Y<-runif(N)^(-1/n)
X<-log(runif(N))/Y
xx=seq(0,10,len=10000)
yy=sapply(xx,f)
points(xx,yy,type="l",lwd=2,col="blue")
```

prog20

Pour cet exemple, on s'intéresse à simuler $f(x)$, et non la densité $g(x, y)$.

On vérifie avec l'histogramme que $X$ suit bien la loi de densité $f$.



## Vecteurs gaussiens

- Pour simuler un vecteur gaussien $X = (X_1, \cdots, X_d)$ d'espérance $m = (m_1, \cdots, m_d)$ et de matrice de covariance $\Sigma$, il suffit de simuler un vecteur gaussien centré $Y = (Y_1, \cdots, Y_d)$ de matrice de covariance $\Sigma$ et de poser $X_i = Y_i + m_i$ pour $i = 1, \cdots, d$;
- dans la suite on suppose donc que $m = 0$.
- Si la matrice de covariance $\Sigma$ est diagonale, c'est à dire si les composantes du vecteur gaussien sont indépendantes, il suffit de simuler successivement $d$ variables aléatoires gaussiennes réelles.

Pour le cas général :

- Puisque $\Sigma$ est symétrique de type positif (puisque pour tout vecteur $v \in \mathbb{R}, \langle v, \Sigma v \rangle = \mathrm{Var}(\langle v, X \rangle \geq 0)$, un résultat classique d'algèbre linéaire permet d'écrire la décomposition de Cholesky de $\Sigma$, c'est à dire de trouver une matrice $A$ triangulaire inférieure telle que $\Sigma = AA^*$.
- Il suffit alors de simuler un vecteur gaussien $Z = (Z_1, \cdots, Z_d)$ dont les composantes sont gaussiennes $\mathcal{N}(0,1)$ indépendantes, puis de calculer $X = AZ$
- Le vecteur $X$ est clairement gaussien, centré de matrice de covariance $\Sigma$.

## Décomposition de Cholesky

Rappelons que lorsque la matrice $\Sigma = \left(S_{i,j}, 1 \leq i,j \leq d\right)$ est définie positive, la décomposition de Cholesky de $\Sigma$ (qui est disponible dans de nombreuses bibliothèques de programmes) est calculée de la façon suivante :

$$a_{1,1} = \sqrt{S_{1,1}}, \quad a_{i,1} = \frac{S_{1,i}}{a_{1,1}} \quad \text{pour} \quad 2 \leq i \leq d$$

pour $i$ croissant de $2$ à $d$ : $a_{i,i} = \sqrt{S_{i,i} - \sum_{1 \leq k \leq i-1} \left|a_{i,k}\right|^2}$

pour $i < j \leq d$

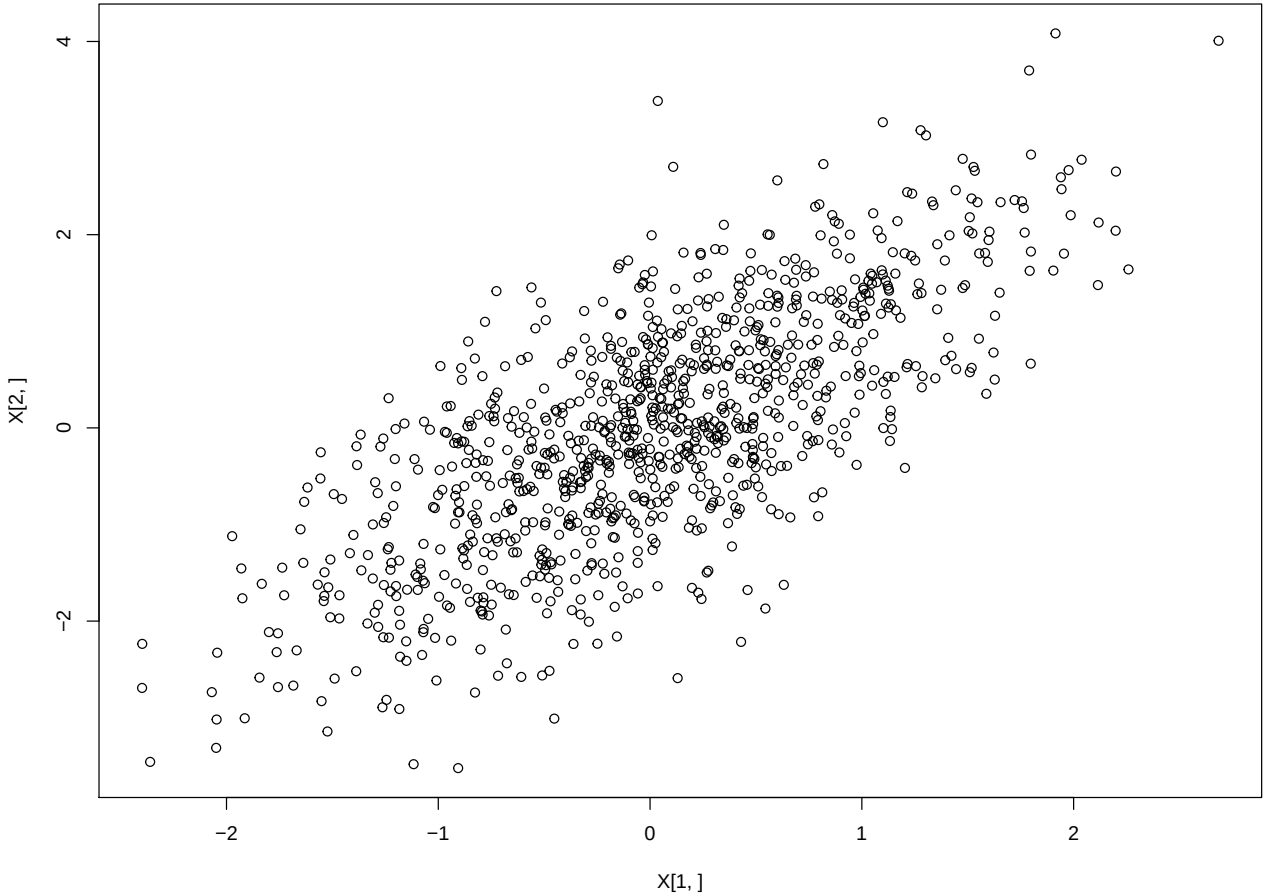
$$a_{j,i} = \frac{S_{i,j} - \sum_{k=1}^{i-1} a_{i,k} a_{j,k}}{a_{i,i}}, a_{i,j} = 0$$

## Programme

```
Sigma=matrix(c(11/16,7*sqrt(3)/16,7*sqrt(3)/16,25/16)
      ,2,2)
A=t(chol(Sigma))
tcrossprod(A,A)
Sigma
A
n=1000
Z1=rnorm(n)
Z2=rnorm(n)
Z=cbind(Z1,Z2)
X=crossprod(A,Z)
plot(X[1,],X[2,])
cov(t(X))
Sigma
```

```
> rowMeans(X)
[1]  0.012850377 -0.006955678
> cov(t(X))
[,1]     [,2]
[1,] 0.677808 0.737685
[2,] 0.737685 1.509316
> Sigma
[,1]      [,2]
[1,] 0.6875000 0.7577722
[2,] 0.7577722 1.5625000
>
```

prog22


X[2, ]
X[1, ]
4
2
0
−2
−2
−1
0
1
2

## Cas particulier : Vecteur gaussien dimension 2

Dans le cas particulier $d = 2$,

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix},$$

on a

$$A = \begin{pmatrix} \sigma_1 & 0 \\ \rho\sigma_2 & \sigma_2\sqrt{1-\rho^2} \end{pmatrix}$$

Donc si $Z_1$ et $Z_2$ sont des variables aléatoires gaussiennes $\mathcal{N}(0,1)$ indépendantes,

$$X_1 = m_1 + \sigma_1 Z_1, X_2 = m_2 + \sigma_2\left(\rho Z_1 + \sqrt{1-\rho^2} Z_2\right),$$

Le vecteur $X = (X_1, X_2)$ est gaussien de vecteur espérance $m = (m_1, m_2)$ et de matrice de covariance $\Sigma$

## Exercice

Soit la densité $f$ définie par

$$f(x) = \frac{2}{7}e^{-2x}\mathbf{1}_{[0,\infty[}(x) + \frac{1}{7}\mathbf{1}_{[-1,1]}(x) + \frac{8}{7}\frac{1}{\sqrt{2\pi}}\exp(-2(x-3)^2).$$

- Expliquer comment on peut simuler cette densité ( générer **une** v-a suivant $f$) ?

Dans cet exercice, on dispose seulement d'une fonction **runif()** qui génèrent des variables uniformes sur $[0,1]$. On peut écrire : générer $U \sim \mathcal{U}[0,1]$. Toute autre loi, sa simulation doit être basée sur des uniformes $\mathcal{U}[0,1]$.

## Réponse

Il s'agit d'une loi mélange de trois densités(exponentielle, uniforme et normale).

$$f(x) = \frac{1}{7}\ 2e^{-2x}\mathbf{1}_{[0,\infty[}(x) + \frac{2}{7}\ \frac{1}{2}\mathbf{1}_{[-1,1]}(x) + \frac{4}{7}\ \frac{1}{(1/2)\sqrt{2\pi}}\exp\left(-\frac{(x-3)^2}{2*(1/4)}\right)$$

La méthode sera alors :

**Etape 1 :** Génération d'une variable exponentielle de paramètre 2 :

**A.** Générer $U_1 \sim \mathcal{U}[0,1]$. **B.** Poser $X = -\log(1-U_1)/2$.

**Etape 2 :** Génération d'une variable uniforme sur $[-1,1]$

**A.** Générer $U_2 \sim \mathcal{U}[0,1]$. **B.** Poser $Y = 2U_2 - 1$.

**Etape 3 :** Génération d'une variable normale $\mathcal{N}(3,1/4)$

**A.** Générer $U \sim \mathcal{U}[0,1]$. Générer $V \sim \mathcal{U}[0,1]$. **B.** Poser $W = \sqrt{-2\log(U)}\sin(2\pi V)$

**C.** Poser $Z = \frac{1}{2}W + 3$.

## Dernière étape : Mélange

**Etape 4 : Mélange** **A.** Générer $U_3 \sim \mathcal{U}[0,1]$.

**B.** Faire

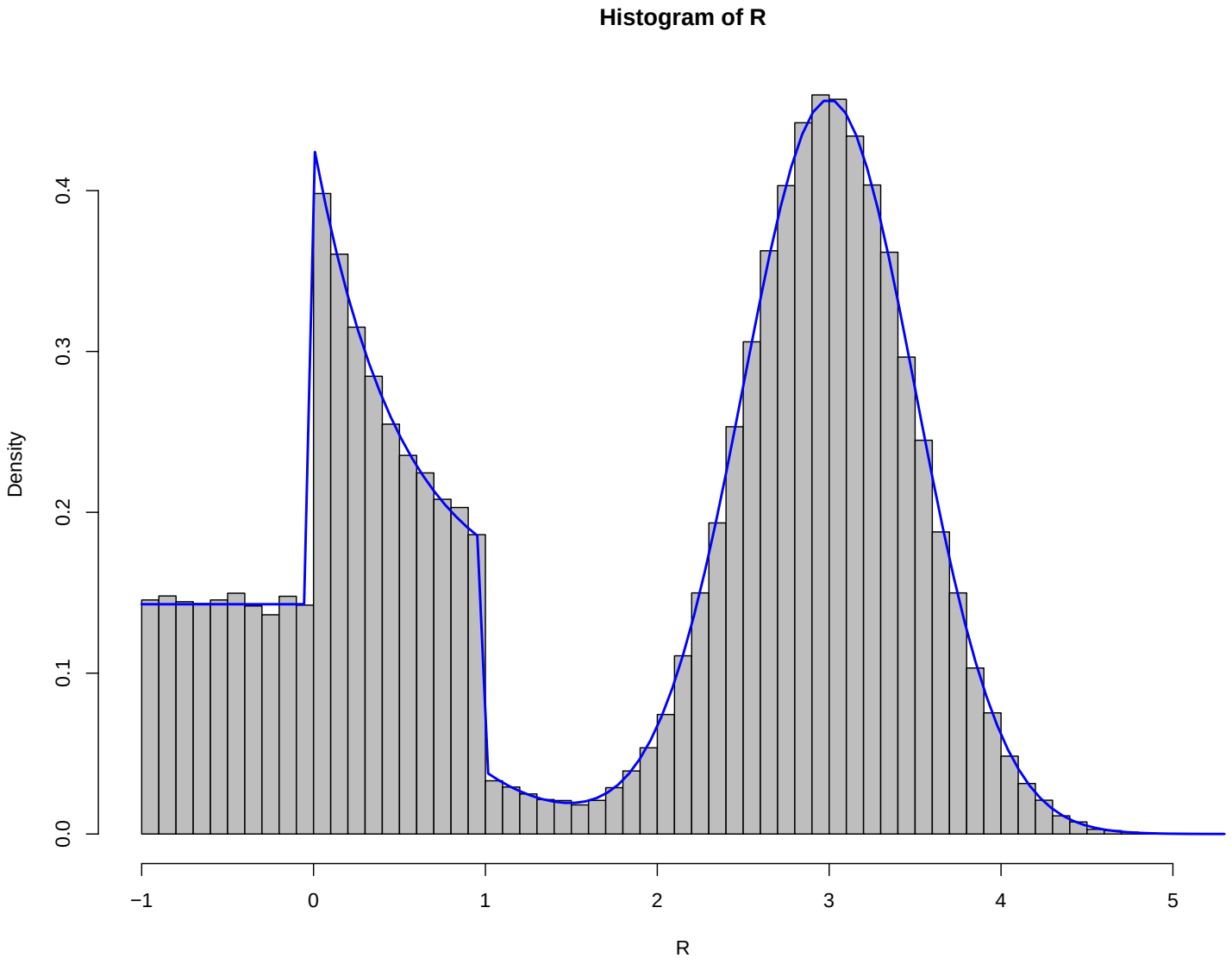
$$\begin{cases} R = X & \text{Si} \quad 0 \leq U_3 \leq \frac{1}{7}. \\ R = Y & \text{Si} \quad \frac{1}{7} < U_3 \leq \frac{3}{7}. \\ R = Z & \text{Si} \quad \frac{3}{7} < U_3 \end{cases}$$

La variable $R$ obtenue sera de densité de $f$.

## Programme

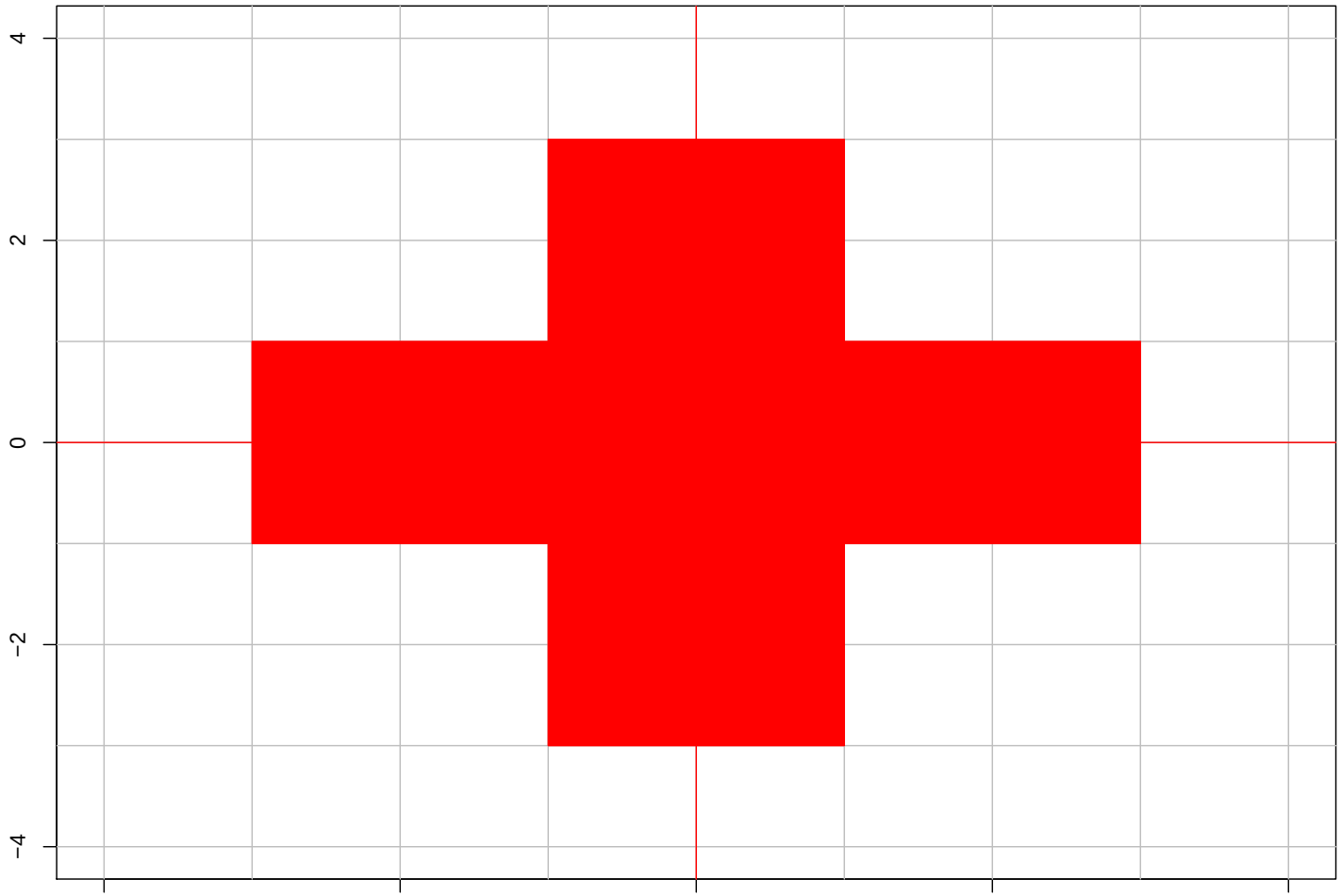
```
f=function(x) {
(1/7)*dexp(x,2)+(2/7)*dunif(x,-1,1)+(4/7)*dnorm(x
    ,3,1/2) }
n=1e5
U1=runif(n); X=-log(U1)/2
U2=runif(n);Y=2*U2-1
U=runif(n); V=runif(n); W=sqrt(-2*log(U))*sin(2*pi*V)
Z=(1/2)*W+3
## Mélange
U3=runif(n)
R=(U3<1/7)*X+(U3>1/7)*(U3<3/7)*Y+(U3>3/7)*Z
hist(R,nclass=70,col="gray",freq=FALSE)
curve(f,add=TRUE,col="blue",lwd=2)
```

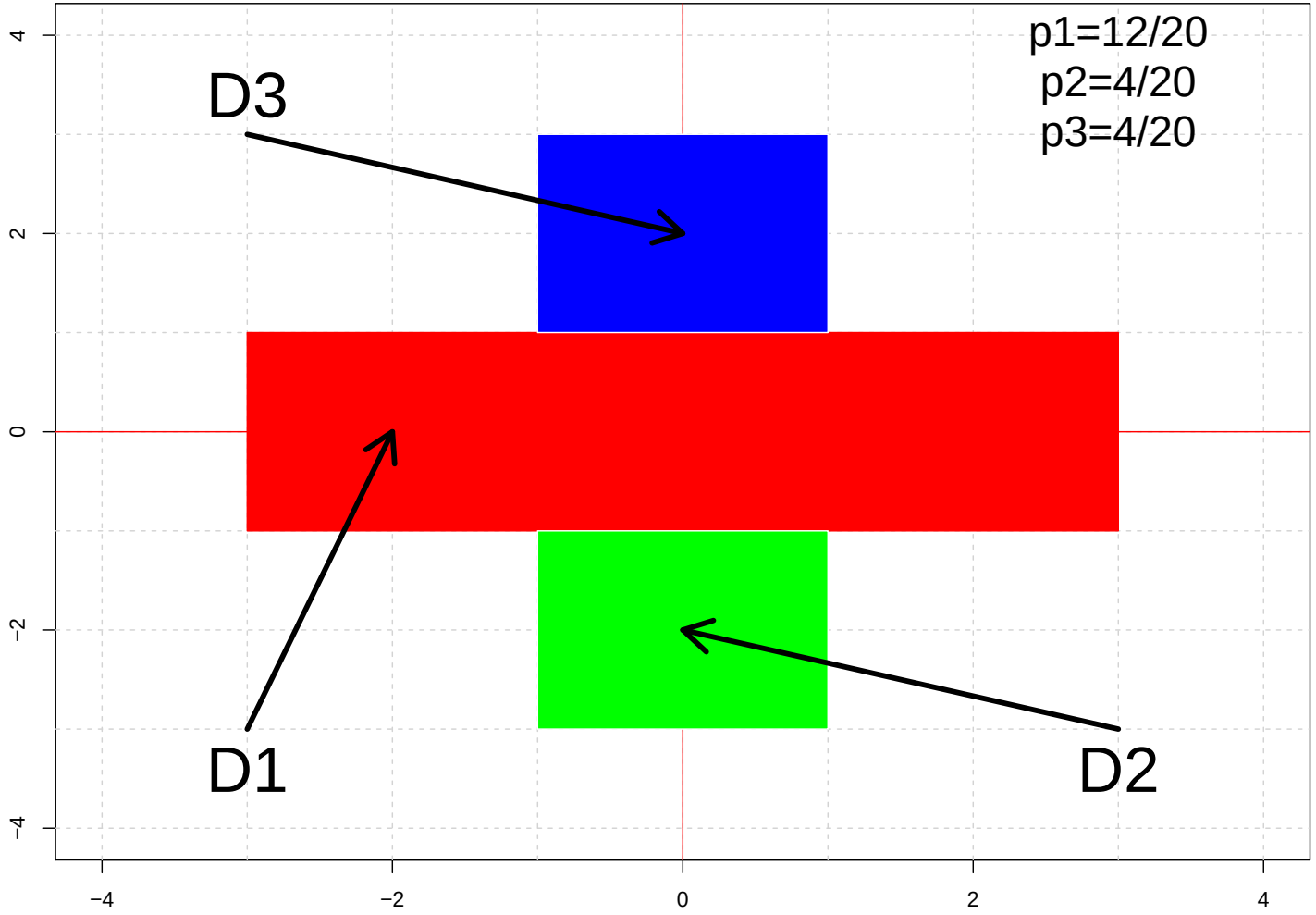
prog22


Histogram of R
Density
0.0
0.1
0.2
0.3
0.4
−1
0
1
2
3
4
5
R

# Méthhode de décomostion : Exercice

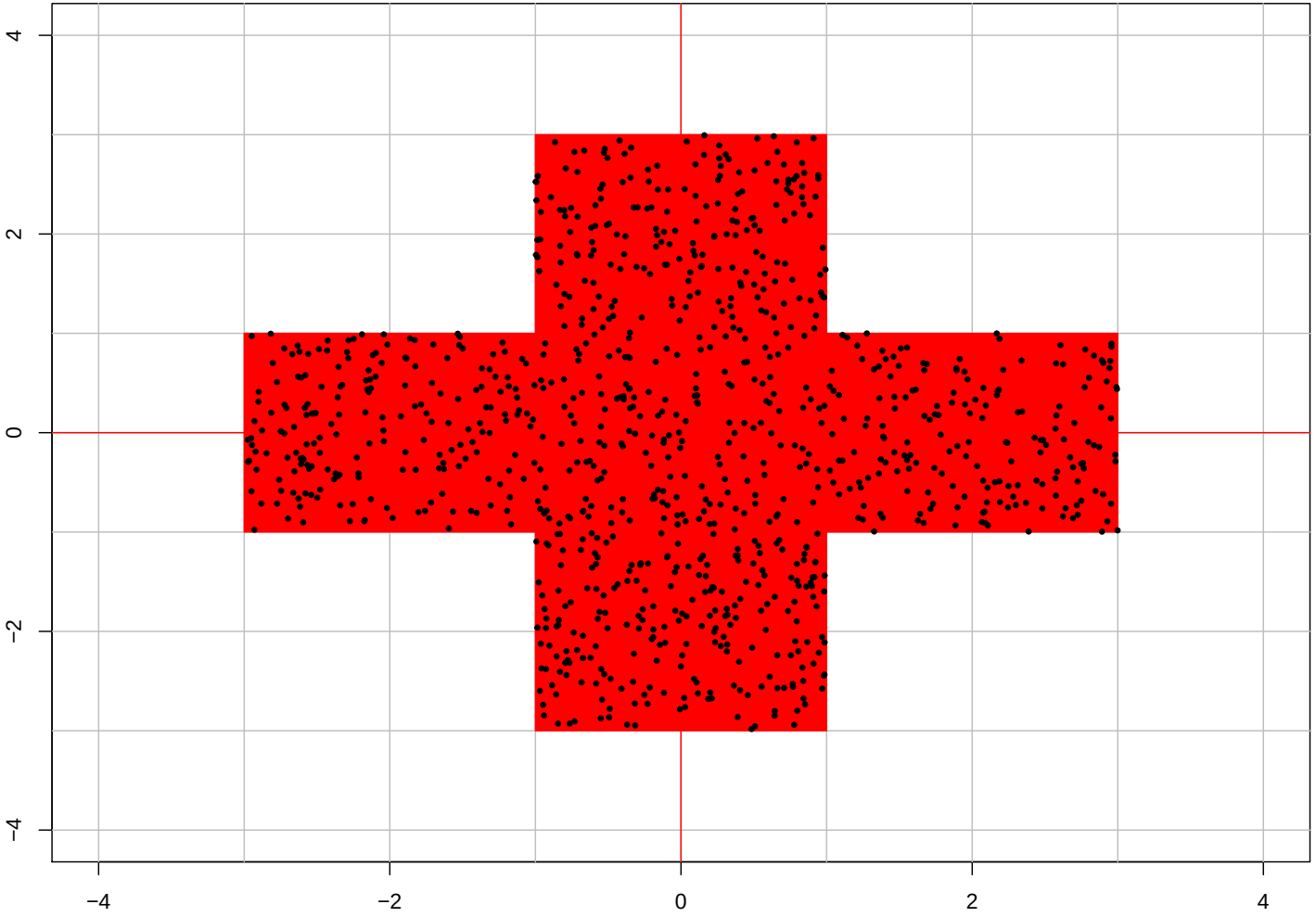
Ecrire le programme R qui permet de simuler la loi uniforme sur la surface rouge.




p1=12/20
p2=4/20
p3=4/20
D3
D1
D2
4
2
0
−2
−4
−4
−2
0
2
4

```
dessinE=function() {
plot(0,0,col="white",xlab="",ylab="",
xlim=c(-4,4),ylim=c(-4,4))
abline(h=-6:6,v=-6:6,col="gray")
abline(h=0,v=0,col="red")
x0=-3 ; y0=-1
xx=x0+6*c(0,0,1,1)
yy=y0+2*c(0,1,1,0)
polygon(x=xx,y=yy,col="red",border="red")
x0=-1 ; y0=-3
xx=x0+2*c(0,0,1,1)
yy=y0+6*c(0,1,1,0)
polygon(x=xx,y=yy,col="red",border="red")
}
dessinE()
```

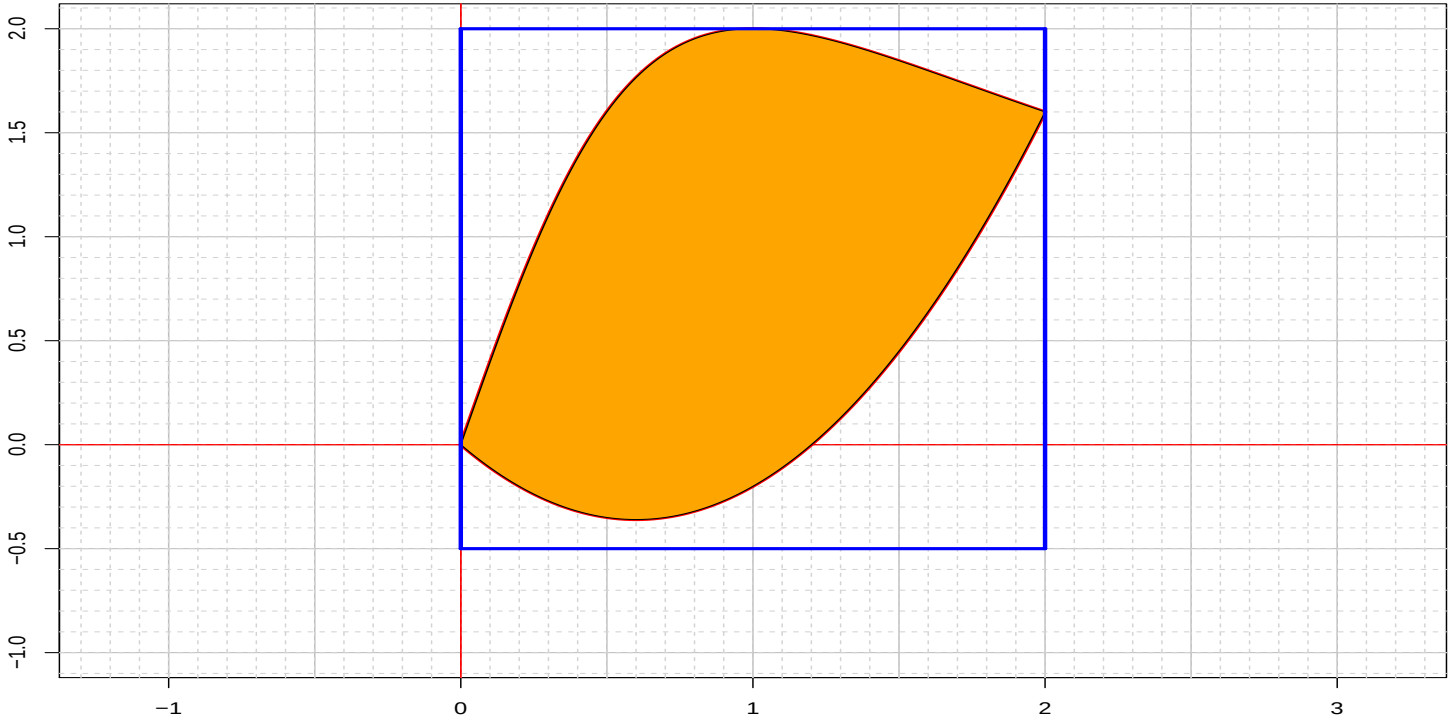
prog23


4
2
0
−2
−4
−4
−2
0
2
4

```
dessinE()
n=1e3  ;   p1=3/5; p2=1/5; p3=1/5
### D1
a=-3;b=3;c=-1;d=1;
X1=(b-a)*runif(n)+a  ; Y1=(d-c)*runif(n)+c
### D2
a=-1;b=1;c=-3;d=-1;
X2=(b-a)*runif(n)+a  ; Y2=(d-c)*runif(n)+c
#### D3
a=-1;b=1;c=1;d=3;
X3=(b-a)*runif(n)+a  ; Y3=(d-c)*runif(n)+c
#### Mélange
U=runif(n)
X=X1*(U<p1)+X2*(U>p1)*(U<p1+p2)+X3*(U>p1+p2)
Y=Y1*(U<p1)+Y2*(U>p1)*(U<p1+p2)+Y3*(U>p1+p2)
points(X,Y,pch=19,cex=0.5)
```

## 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.