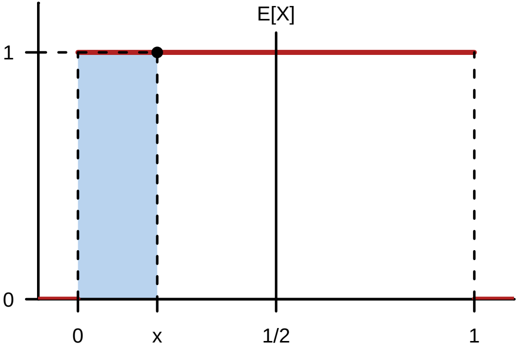
Simulation en Biologie - Simul-Rnd-Var

Bruno Toupance

Université de Paris

SEP 2021

La base de tout : générateur uniforme sur [0, 1]



Générateurs congruentiels

Génération d'une suite de nombres par une formule tout à fait déterministe de manière à obtenir une suite qui semble aléatoire (indépendance et distribution uniforme dans l'intervalle de variation).

$$X_i = (aX_{i-1} + b) \text{ modulo } (m + 1)$$

avec a, b, m des entiers positifs.

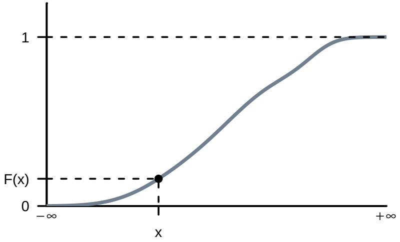
La suite (X_i) est de période m pour des valeurs de (a, b) correctement choisies

Se transforme en générateur sur $]0, 1[$ par division par $(m + 2)$

Méthode de transformation

Méthode de transformation

X : variable aléatoire continue de loi définie par sa fonction de répartition F_X



Si F_X est strictement croissante, la fonction réciproque de F_X existe, est **continue** et **croissante** : F_X^{-1}

C'est la fonction quantile $Q_X(p)$

Méthode de transformation

U : variable aléatoire uniforme sur $[0, 1]$

alors :

$$Y = F_X^{-1}(U) = Q_X(U)$$

suit la loi de X

Démonstration :

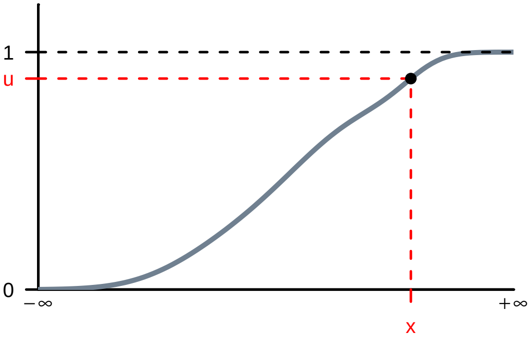
$$F_Y(y) = \mathbb{P}[Y < y] = \mathbb{P}\left[F_X^{-1}(U) < y\right] = \mathbb{P}[U < F_X(y)]$$

$$F_Y(y) = F_X(y)$$

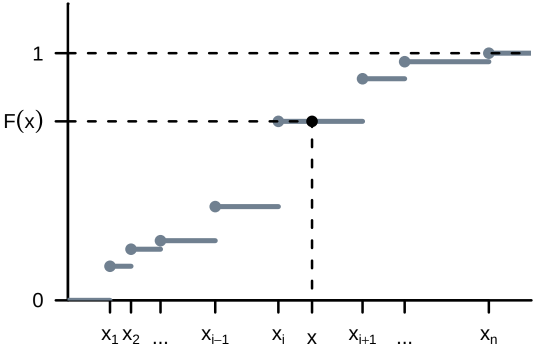
Difficulté : connaître la fonction réciproque

- Simuler U selon une loi uniforme $\mathcal{U}([0, 1])$
- Valeur simulée : u
- Prendre :

$$x = F_X^{-1}(u)$$



Par définition la fonction de répartition F_X n'est pas continue...



Méthode de transformation : Cas des lois discrètes

Dans la plupart des cas, il suffit de procéder par itération :

- Simuler U selon une loi uniforme $\mathcal{U}([0, 1])$
- Valeur simulée : u
- Construire $F(x)$ pas-à-pas :

$$F(x_i) = F(x_{i-1}) + \mathbb{P}[X = x_i]$$

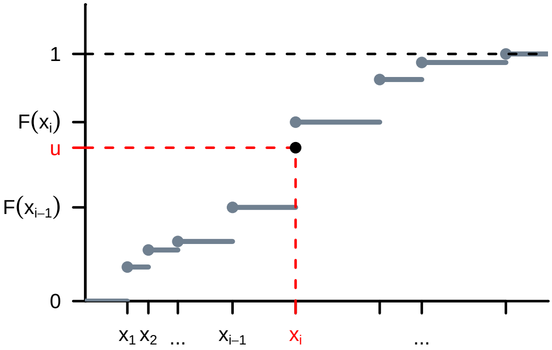
avec :

$$F(x_0) = 0$$

- Prendre x_i si :

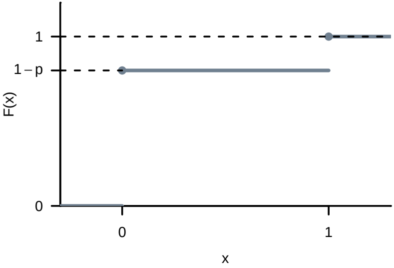
$$F(x_{i-1}) < u \leq F(x_i)$$

$$\mathbb{P}\left[F(x_{i-1}) < U \leq F(x_i)\right] = F(x_i) - F(x_{i-1}) = \mathbb{P}\left[X = x_i\right]$$



On définit : $F(x_0) = 0$

Loi de Bernoulli de paramètre p :



Loi de Bernoulli

- Simuler U selon une loi uniforme $\mathcal{U}([0, 1])$

- Valeur simulée : u

- Prendre :

$$0 \text{ si } u < 1 - p \qquad \text{et} \qquad 1 \text{ si } u \geq 1 - p$$

ou

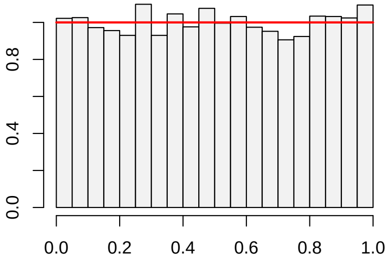
$$1 \text{ si } u < p \qquad \text{et} \qquad 0 \text{ si } u \geq p$$

Loi de Bernoulli - Implémentation R

- Simulation d'un *n*-échantillon uniforme sur [0, 1]

```
n <- 10000
u <- runif(n)
truehist(u)
head(u)
```

[1] 0.98696722 0.05314978 0.49168854 0.64380867
0.01060348 0.50767618

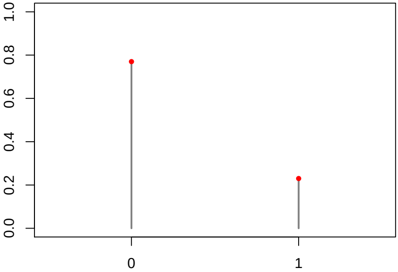


Loi de Bernoulli - Implémentation R

- Transformation : Bernoulli de paramètre ($p = 0.23$)

```
p <- 0.23
x <- rep(0, times = n)
x[u < p] <- 1
plot(table(x) / n, type ="h")
head(x)
```

```
[1] 0 1 0 0 1 0
```



Loi de Bernoulli - Application

Dans tous les cas où on doit choisir au hasard entre deux possibilités :

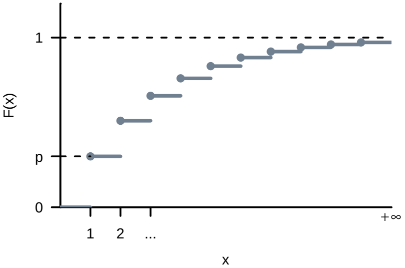
- Faire "ceci" avec une probabilité 0.3
- Faire "cela" avec une probabilité $1 - 0.3 = 0.7$

Solution :

- Tirer **un** nombre aléatoire u uniforme sur $[0, 1]$
- Décider ce qu'on fait :
  - Si $u < 0.3$, faire "ceci"
  - Sinon, faire "cela"

Loi géométrique

Loi géométrique de paramètre p :



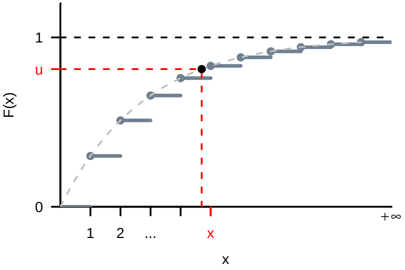
$$F(x) = 1 - (1-p)^{\lfloor x \rfloor}$$

N.B. : la notation $\lfloor \quad \rfloor$ désigne la valeur entière "plancher" (EN : *floor*)

En continu :

$$F(x) = 1 - (1-p)^x \qquad \text{et} \qquad F^{-1}(y) = \frac{\ln(1-y)}{\ln(1-p)}$$

Loi géométrique



- Simuler U selon une loi uniforme $\mathcal{U}([0, 1])$
- Valeur simulée : u
- Prendre :

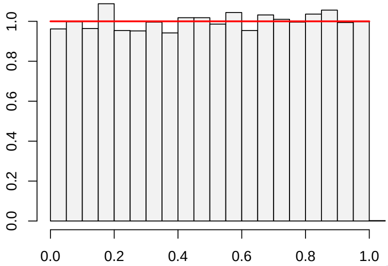
$$\left\lfloor \frac{\ln(1-u)}{\ln(1-p)} \right\rfloor + 1$$

Loi géométrique - Implémentation R

- Simulation d'un n-échantillon uniforme sur [0, 1]

```
n <- 10000
u <- runif(n)
truehist(u)
head(u)
```

[1] 0.5462990 0.7740723 0.9836019 0.7098483 0.9345680 0.2424564

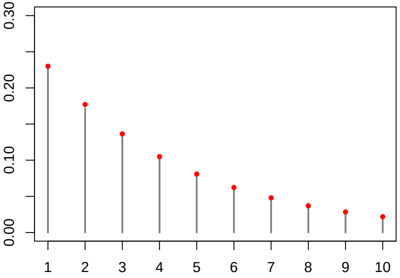


Loi géométrique - Implémentation R

- Transformation : Géométrique de paramètre ($p = 0.23$)

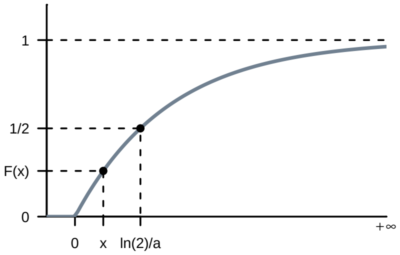
```
p <- 0.23
x <- floor(log(1 - u) / log(1 - p)) + 1
plot(table(x) / n, type = "h")
head(x)
```

[1] 4 6 16 5 11 2



Loi exponentielle

X variable aléatoire suivant une loi exponentielle $\mathcal{E}(a)$:



$$F(x) = 1 - \exp(-ax)$$

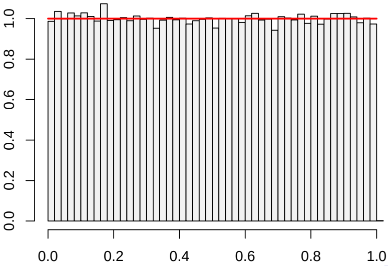
Fonction réciproque (quantile) :

$$F^{-1}(y) = -\frac{1}{a} \ln(1 - y)$$

- Simulation d'un n-échantillon uniforme sur [0, 1]

```
n <- 100000
u <- runif(n)
truehist(u)
head(u)
```

[1] 0.56790351 0.05098637 0.47663190 0.39775107 0.89804282 0.82005418

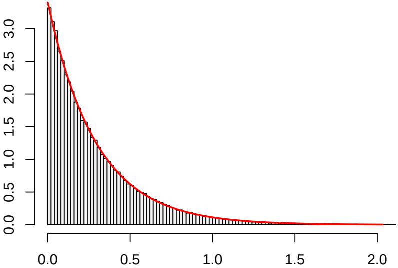


Loi exponentielle - Implémentation R

- Transformation : $X = F^{-1}(U)$

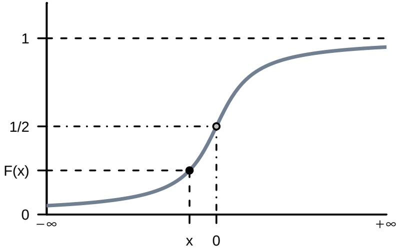
```
a <- 3.4
x <- -log(1 - u) / a
truehist(x)
head(x)
```

[1] 0.2467960 0.0153918 0.1904324 0.1491425 0.6715301 0.5044410



Loi de Cauchy

X variable aléatoire suivant une loi de Cauchy de paramètres :



$$F(x) = \frac{1}{\pi} \arctan(x) + \frac{1}{2}$$

Fonction réciproque (quantile) :

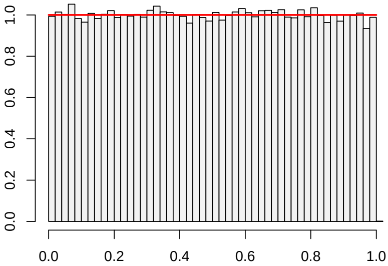
$$F^{-1}(y) = \tan\left(\pi\left(y - \frac{1}{2}\right)\right)$$

Loi de Cauchy - Implémentation R

- Simulation d'un *n*-échantillon uniforme sur [0, 1]

```
n <- 100000
u <- runif(n)
truehist(u)
head(u)
```

[1] 0.3677855 0.8883514 0.1003319 0.8282950 0.3155977 0.0346401

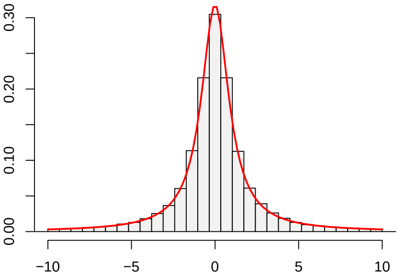


Loi de Cauchy - Implémentation R

- Transformation : $X = F^{-1}(U)$

```
x <- tan(pi * (u - 1/2))
truehist(x)
head(x)
```

[1] -0.4410235 2.7331097 -3.0668002 1.6704213 -0.6541927 -9.1527537



Loi Normale - Box-Muller

On sait que si :

- U_1 suit une loi uniforme sur $[0, 1]$
- U_2 suit une loi uniforme sur $[0, 1]$
- U_1 et U_2 indépendantes

alors :

- $X_1 = \sqrt{-2 \ln U_1} \cos(2\pi U_2)$ suit une loi normale $(0, 1)$
- $X_2 = \sqrt{-2 \ln U_1} \sin(2\pi U_2)$ suit une loi normale $(0, 1)$
- X_1 et X_2 sont indépendantes

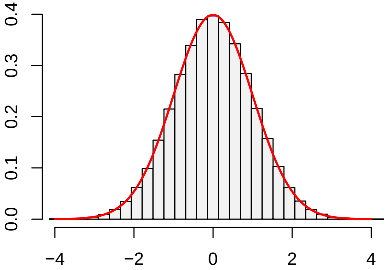
Loi Normale- Box-Muller - Implémentation R

- Simulation de deux *n*-échantillons uniforme sur [0, 1]

```
n <- 100000
u1 <- runif(n)
u2 <- runif(n)
```

- Transformation

```
x1 <- sqrt(-2 * log(u1)) * cos(2 * pi * u2))
truehist(x1)
```



Méthodes *ad hoc*

Méthodes *ad hoc*

Loi de Cauchy

Loi de Cauchy $(x_0 = 0, a = 1)$:

On sait que si :

X_1 suit une loi Normale $(0, 1)$
X_2 suit une loi Normale $(0, 1)$
X_1 et X_2 indépendantes

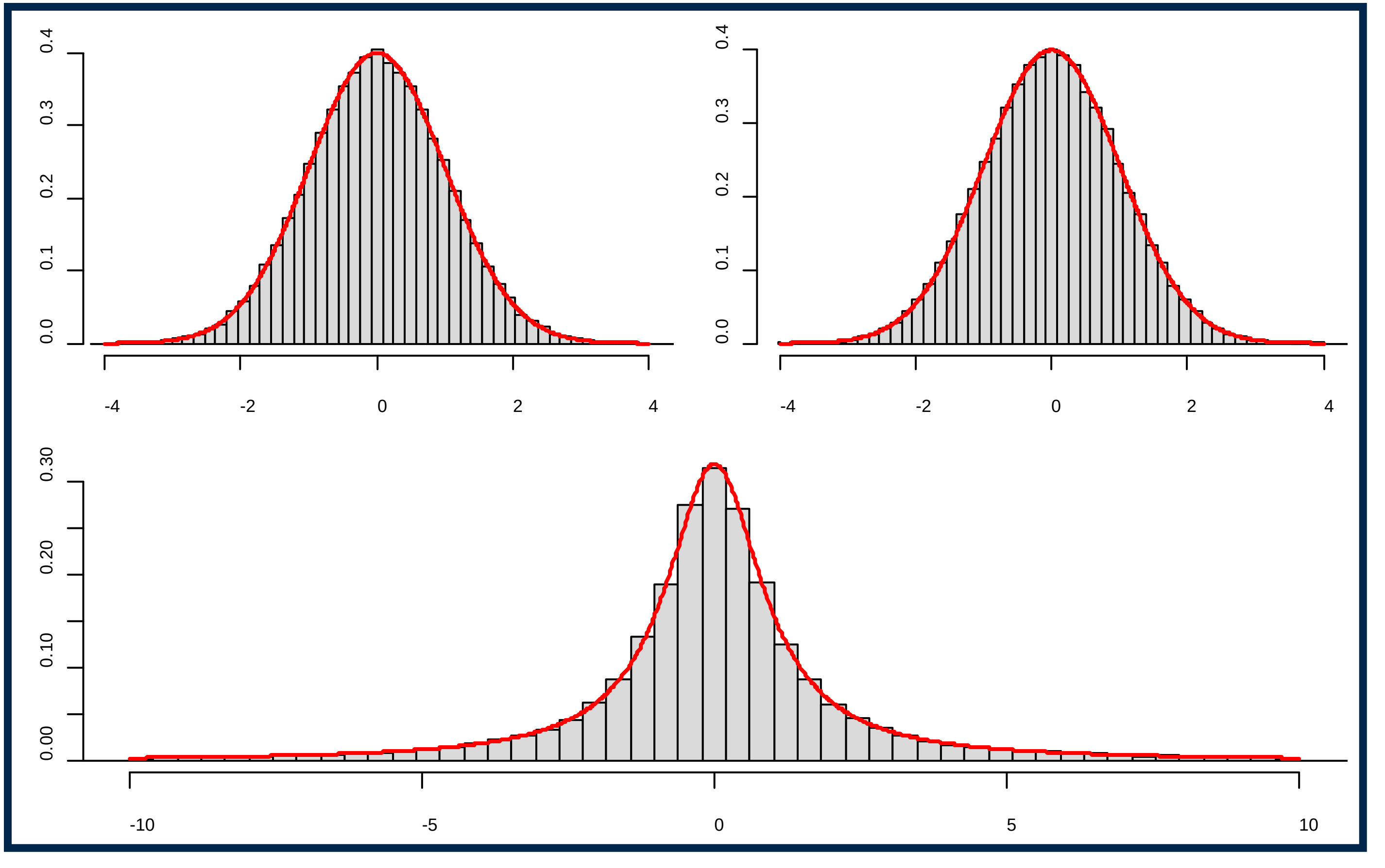
Alors :

X_1 / X_2 suit une loi de Cauchy $(0, 1)$

Méthodes *ad hoc*

Loi de Cauchy – Implémentation R

```
n <- 100000; x1 <- rnorm(n); x2 <- rnorm(n)
truehist(x1 / x2)
```



Méthodes *ad hoc*

Loi de χ^2

Loi de $\chi^2\,(n)$:

On sait que si :

$X_1, X_2, \ldots, X_n$ sont n v.a.r. i.i.d. Normale $(0, 1)$

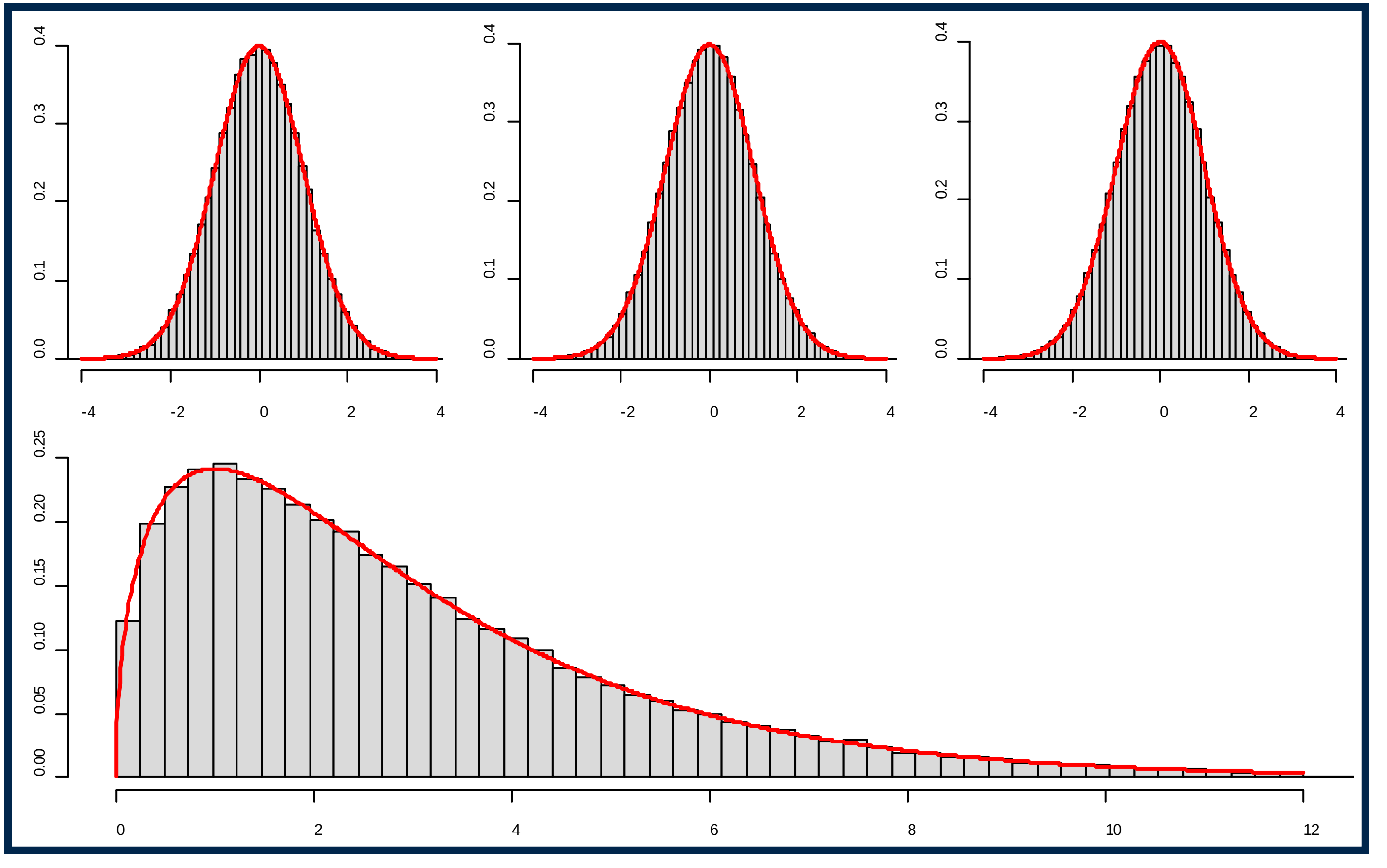
Alors :

$X_1 + X_2 + \ldots + X_n$ suit une loi de $\chi^2\,(n)$

Méthodes *ad hoc*

Loi de χ^2 – Implémentation R

```
n <- 100000; x1 <- rnorm(n); x2 <- rnorm(n); x3 <- rnorm(n)
truehist(x1^2 + x2^2 + x3^2)
```



Méthodes *ad hoc*

Loi de Student

Loi de Student (n) :

On sait que si :

X_1 suit une loi Normale $(0, 1)$
X_2 suit une loi de $\chi^2\ (n)$
X_1 et X_2 indépendantes

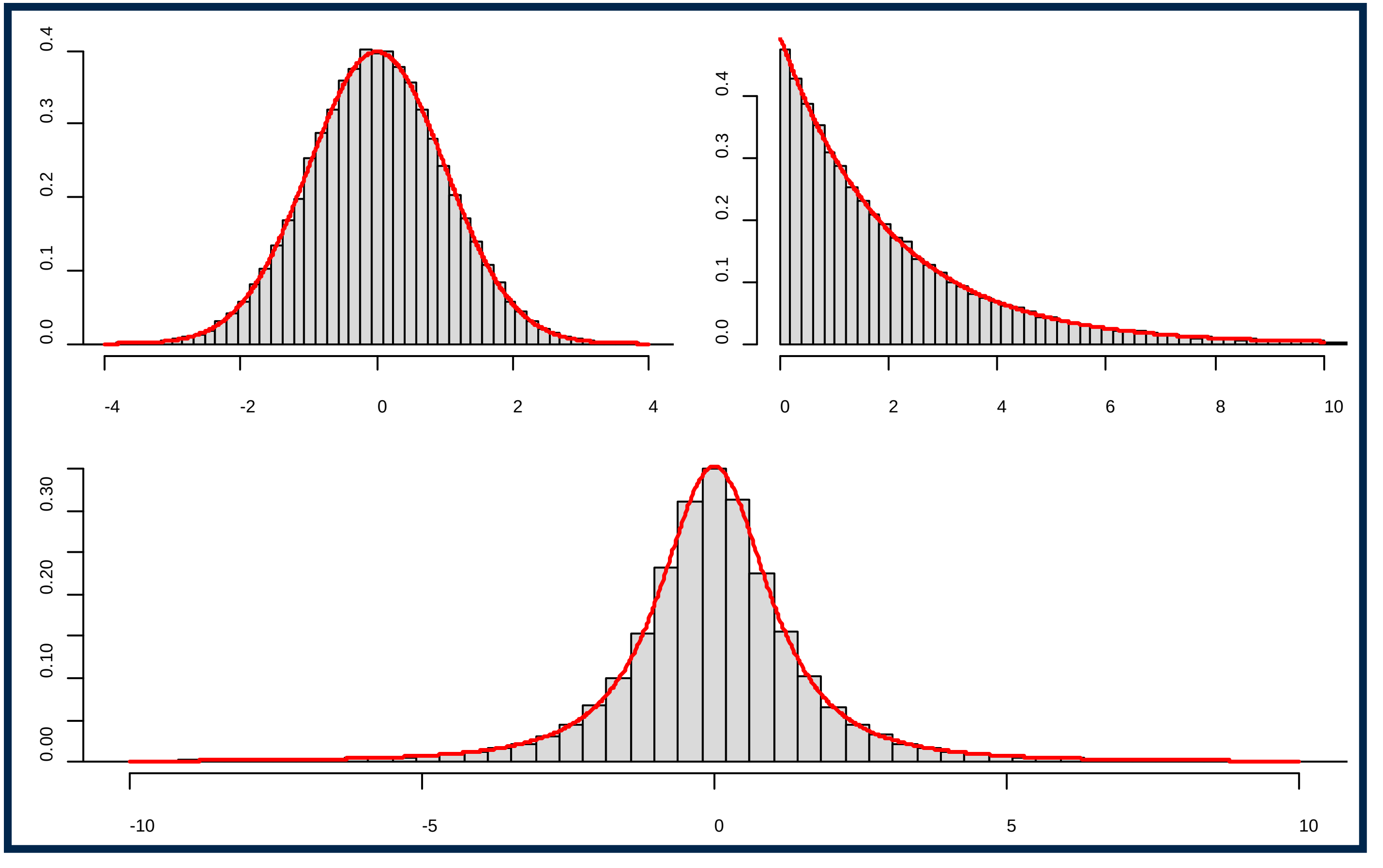
Alors :

X_1 / sqrt(X_2 / n) suit une loi de Student (n)

Méthodes *ad hoc*

Loi de Student – Implémentation R

```
n <- 100000; x1 <- rnorm(n); x2 <- rchisq(n, 2)
truehist(x1 / sqrt(x2 / 2))
```



Méthodes *ad hoc*

Loi de Beta (α, β)

Loi de Beta (α, β) :

On sait que si :

X_1 suit une loi Gamma (α, 1)

X_1 suit une loi Gamma (β, 1)

X_1 et X_2 indépendantes

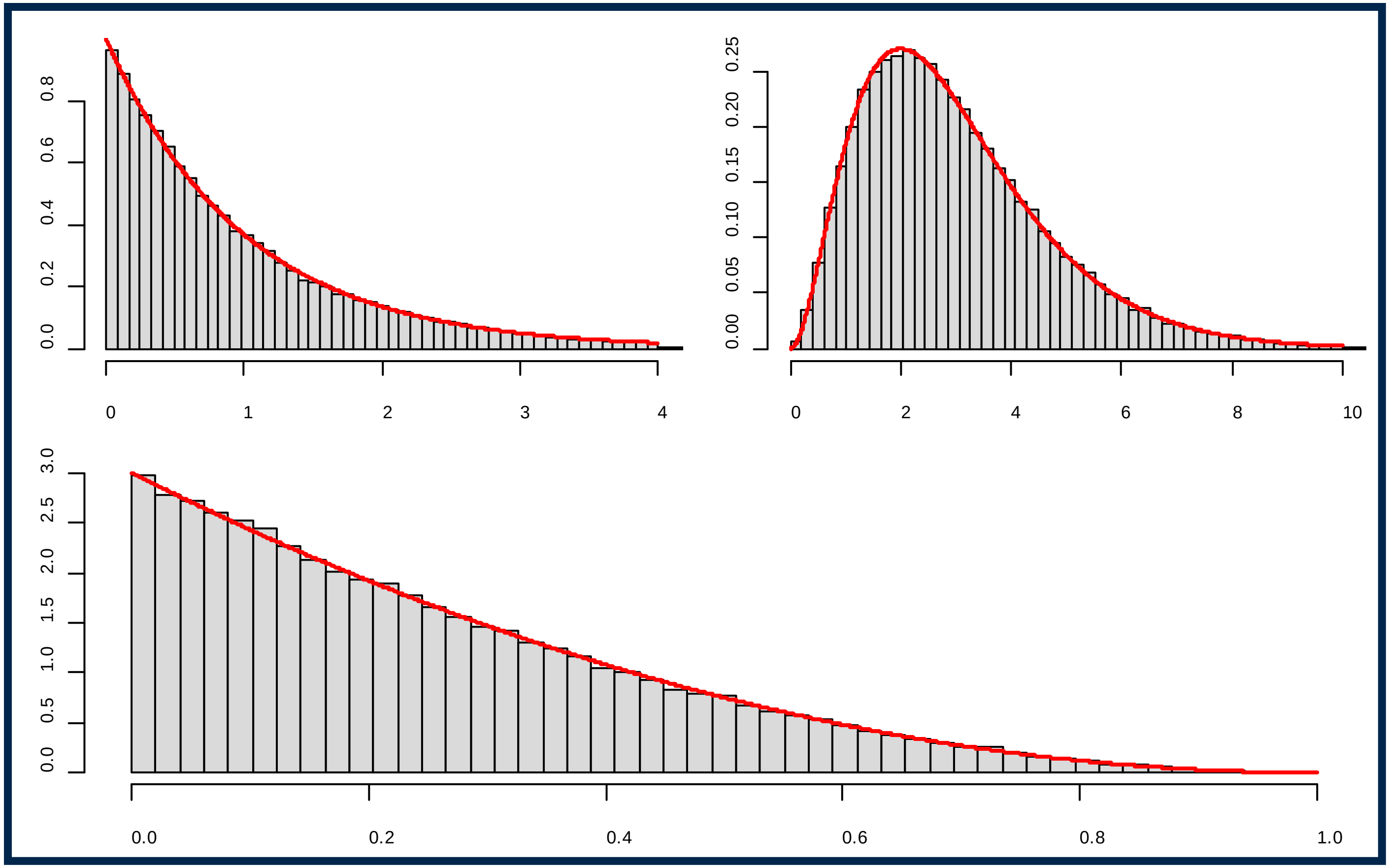
Alors :

$X_1 / (X_1 + X_2)$ suit une loi Beta (α, β)

Méthodes *ad hoc*

Loi Beta – Implémentation R

```
n <- 100000; x1 <- rgamma(n, 1, 1); x2 <- rgamma(n, 3, 1)
truehist(x1 / (x1 + x2))
```



Lois multidimensionnelles

Lois multidimensionnelles

Loi à simuler : $(X_1, X_2, \ldots, X_p) \sim f_{\,1, 2, \ldots, p}$

avec (éventuellement) X_i non indépendante de X_j

Il suffit de remarquer que :

$$f_{\,1, 2, \ldots, p}\,(x_1, x_2, \ldots, x_p) = f_1\,(x_1)\;\; f_{2|1}\,(x_2 \mid x_1) \ldots\, f_{p|1, 2, \ldots, p\text{-}1}\,(x_p \mid x_1, x_2, \ldots, x_{p\text{-}1})$$

Simulation :

- Simuler $X_1 \sim f_1$ Valeur simulée : x_1
- Simuler $X_2 \sim f_{2|X1=x1}$ Valeur simulée : x_2
- etc.

Lois multidimensionnelles

Exemple : loi multinomiale

Loi à simuler : $(X_1, X_2, X_3) \sim \text{Multinomiale}(n, p_1, p_2, p_3)$

Simulation :

- Simuler $X_1 \sim \text{Binomiale}\ (n, p_1)$

  Valeur simulée : x_1

- Simuler $X_2 \sim \text{Binomiale}\ (n - x_1, p_2 / (1 - p_1))$

  Valeur simulée : x_2

- Simuler $X_3 \sim \text{Binomiale}\ (n - x_1 - x_2, p_3 / (1 - p_1 - p_2) = 1)$

  Valeur simulée : $x_3 = n - x_1 - x_2$

Lois multidimensionnelles

Exemple : loi bi-normale centrée-réduite (ρ)

Loi à simuler : $(X_1, X_2) \sim$ Bi-Normale centrée-réduite (ρ)

Simulation :

- Simuler $X_1 \sim$ Normale (0,1)

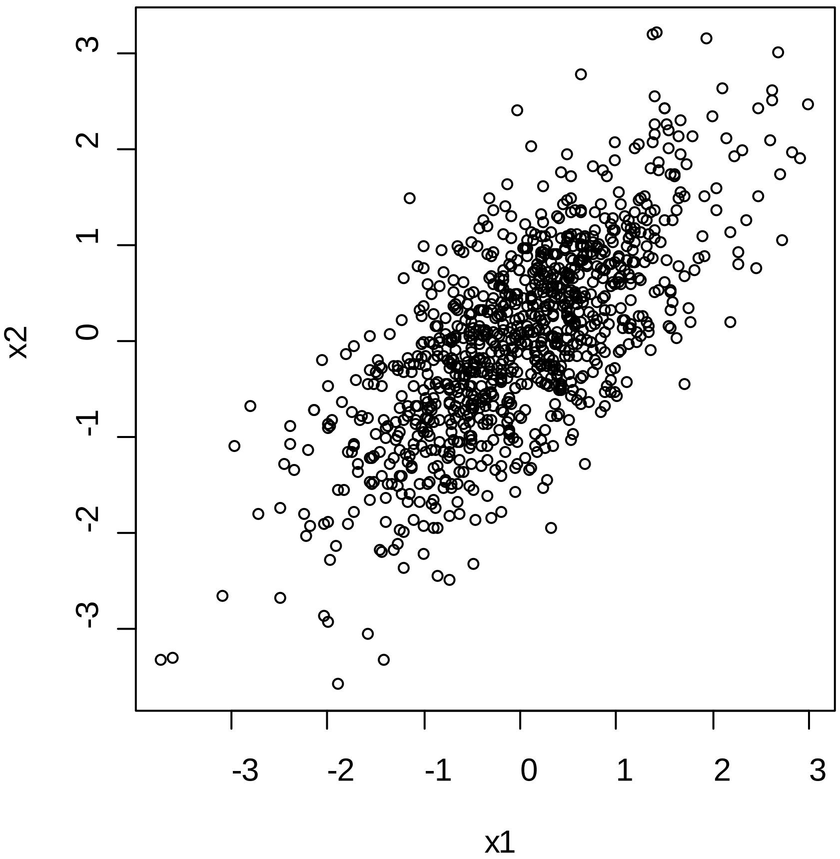
  Valeur simulée : x_1

- Simuler $X_2 \sim$ Normale ($\rho\, x_1$, 1- ρ^2)

  Valeur simulée : x_2

Lois multidimensionnelles

Exemple : loi bi-normale – Implémentation R




```
n <- 1000
rho <- 0.7
x1 <- rnorm(n,0,1)
x2 <- rnorm(n,x1*rho,sqrt(1-rho^2))
plot(x1,x2)
```

Méthode du rejet

Simulations de variables aléatoires

Méthode du rejet

f fonction de densité d'une variable de support D à simuler

g fonction de densité d'une variable X de même support D :

- que l'on sait simuler
- qui vérifie : $f(x) \leq m \cdot g(x)$

- Simuler (i.i.d.): $X_1, X_2, \ldots \sim g$ et $U_1, U_2, \ldots \sim \textit{Uniforme sur } [0,1]$

- Accepter : x_i si $u_i \leq \dfrac{f(x_i)}{m \cdot g(x_i)}$

En d'autres termes la loi conditionnelle de X sachant $U \leq \dfrac{f(X)}{m \cdot g(X)}$ a comme densité f

La probabilité d'acceptation est : $1 / m$

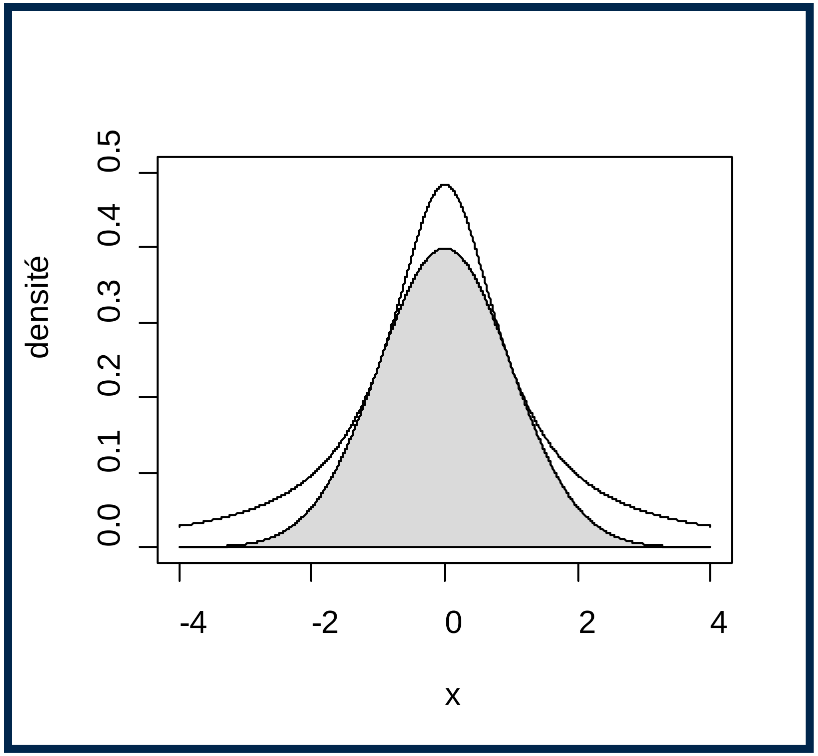
Méthode du rejet

Exemple : loi normale

Loi Normale (0,1) $$f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right)$$

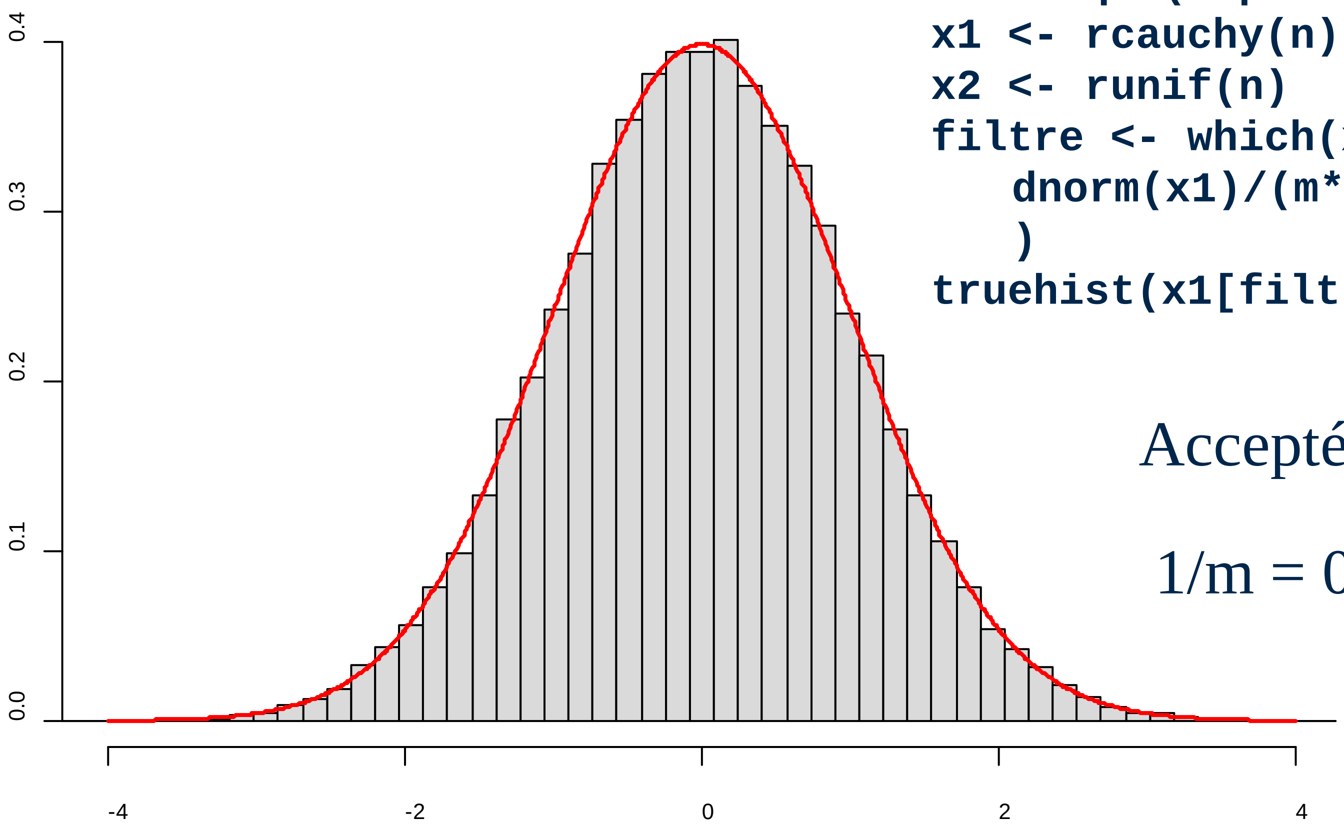
Loi de Cauchy (0, 1) $$g(x) = \frac{1}{\pi\left(1+x^2\right)}$$

$$\frac{f(x)}{g(x)} = \sqrt{\frac{\pi}{2}} \cdot \left(1+x^2\right) \cdot \exp\left(-\frac{1}{2}x^2\right) \leq \sqrt{\frac{2\pi}{e}}$$



Méthode du rejet

Exemple : loi normale – Implémentation R




```
n <- 100000
m <- sqrt(2*pi/exp(1))
x1 <- rcauchy(n)
x2 <- runif(n)
filtre <- which(x2 <=
    dnorm(x1)/(m*dcauchy(x1))
    )
truehist(x1[filtre])
```

Acceptées = 66041

1/m = 0.6577446

Méthode MCMC

Monte-Carlo Markov Chain

Algorithme de Metropolis-Hastings

Le problème :

Générer un échantillon de valeurs de densité cible f

Condition nécessaire :

Savoir calculer $f(x)$ pour tout x

Initialisation : valeur x_0

Loi de proposition : $g(y \mid x)$

Pour chaque x_i, on propose une valeur y_i suivant la loi de proposition

On accepte ou on rejette y_i suivant une règle d'acceptation-rejet :

Si rejet : $x_{i+1} = x_i$

Si acceptation : $x_{i+1} = y_i$

Algorithme de Metropolis-Hastings

Règle d'acceptation-rejet

Acceptation de y_i avec un probabilité :

$$p = \min\left(1, \frac{f(y_i)\, g(x_i \mid y_i)}{f(x_i)\, g(y_i \mid x_i)}\right)$$

La loi de proposition peut être :

- symétrique : $g(y \mid x) = g(x \mid y)$ $\quad$ $p = \min\left(1, \frac{f(y_i)}{f(x_i)}\right)$

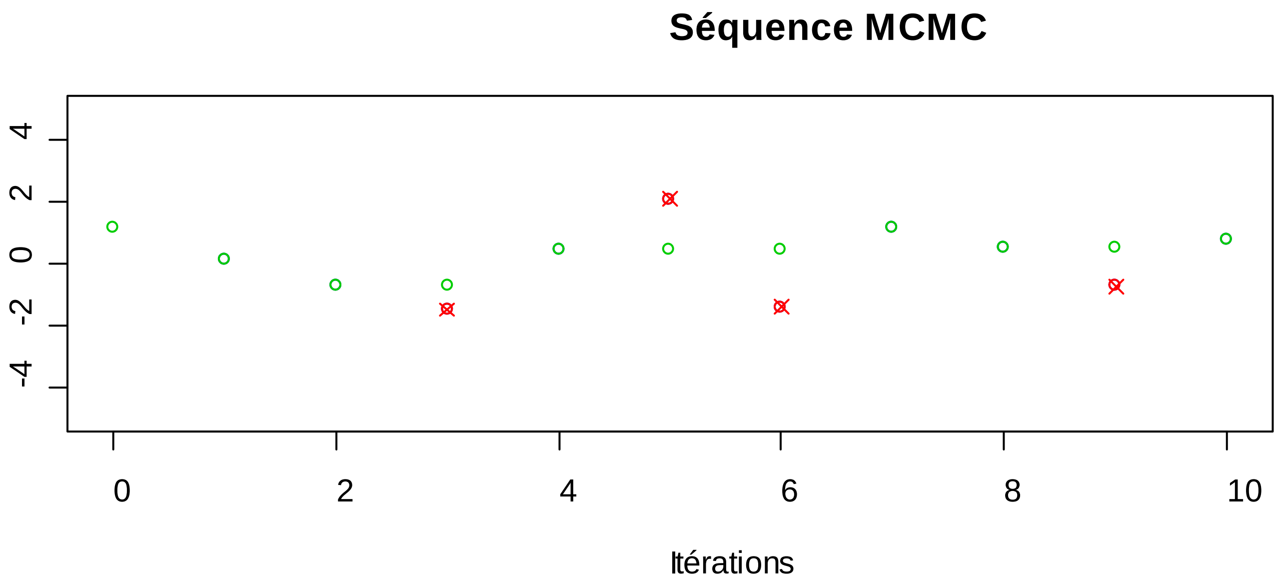
- indépendante : $g(y \mid x)$ ne dépend pas de x

Algorithme de Metropolis-Hastings

Exemple

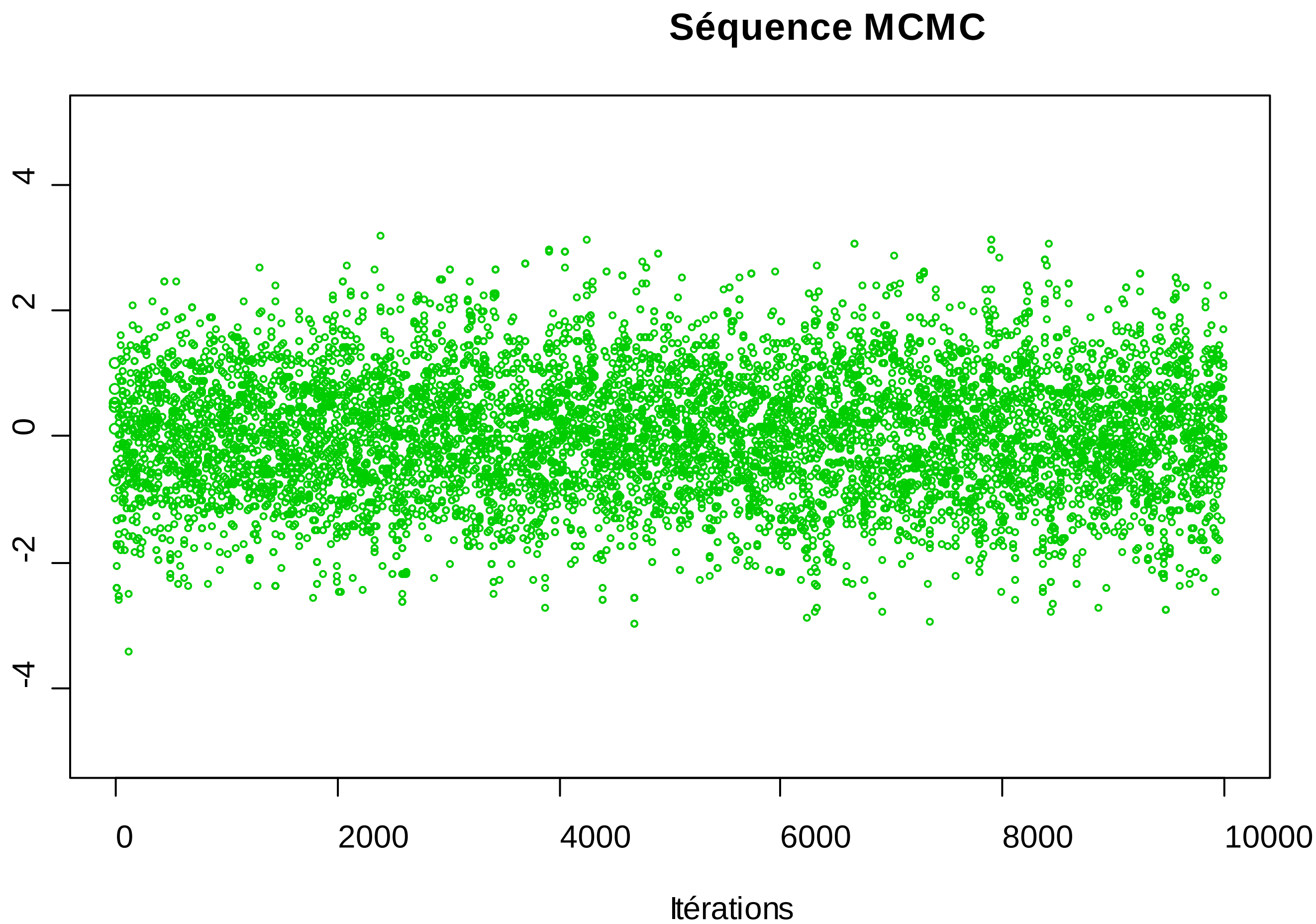
Densité cible : $f \sim N(0,1)$

Loi de proposition : $g(y\,|\,x)$ Uniforme sur $[x\text{-}\delta, x\text{+}\delta]$



Algorithme de Metropolis-Hastings

Exemple



Algorithme de Metropolis-Hastings

Exemple