TP7 Correction

Clement LAROCHE

8 avril 2019

```
library(ggplot2)
library(latex2exp)
library(ggExtra)
```

Dans ce TP, je vous montre la syntaxe d'un package graphique couramment utilisé en `R` qui s'appelle `ggplot2`. **Il n'est absolument pas obligatoire de savoir s'en servir**, je vous le montre pour la curiosité (il a servi à faire les graphes des TP5 et 6). Je vous ai également mis les codes pour faire les graphes de manière "normale" en commentaire dans les codes.

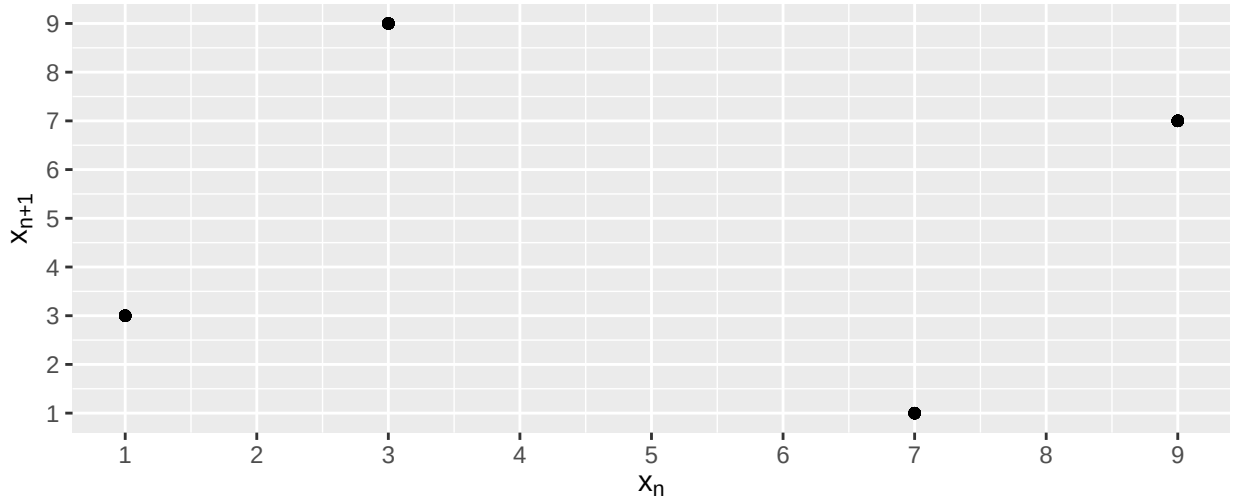
Génération de nombres pseudo-aléatoires

On suit l'énoncé, on code la fonction demandée. Faites attention cependant, il faut mettre le `3*x` entre parenthèses si vous voulez avoir des nombres compris entre `0` et `9`. De plus, il sera plus intéressant d'initialiser le x_0 à 1 plutôt qu'à 0.

```
# la fonction de l'énoncé
pseudo1 <- function(x)
{
  (3*x)%%10
}
# x_0
res <- 1
n <- 100
# création des termes x_n avec une boucle
# j'ajuste pour que mon vecteur res soit de
# taille 100 une fois ma boucle terminée
for(i in 1:(n-1))
{
  res <- c(res,pseudo1(res[length(res)]))
}
# pour regarder les points (x_n,x_{n+1}), on pourra
# utiliser la commande
# plot(res[-length(res)],res[-1])
# pour regarder les points (x_n,x_{n+2}), on pourra
# utiliser la commande
# plot(res[-c(length(res)-1,length(res))],res[c(-1,-2)])

# en ggplot pour les points (x_n,x_{n+1}).
ggplot()+geom_point(aes(x = res[-n],y = res[-1]))+
  scale_x_continuous(breaks = seq(0,9,1))+
  scale_y_continuous(breaks = seq(0,9,1))+
  xlab(TeX("$x_n$"))+
```

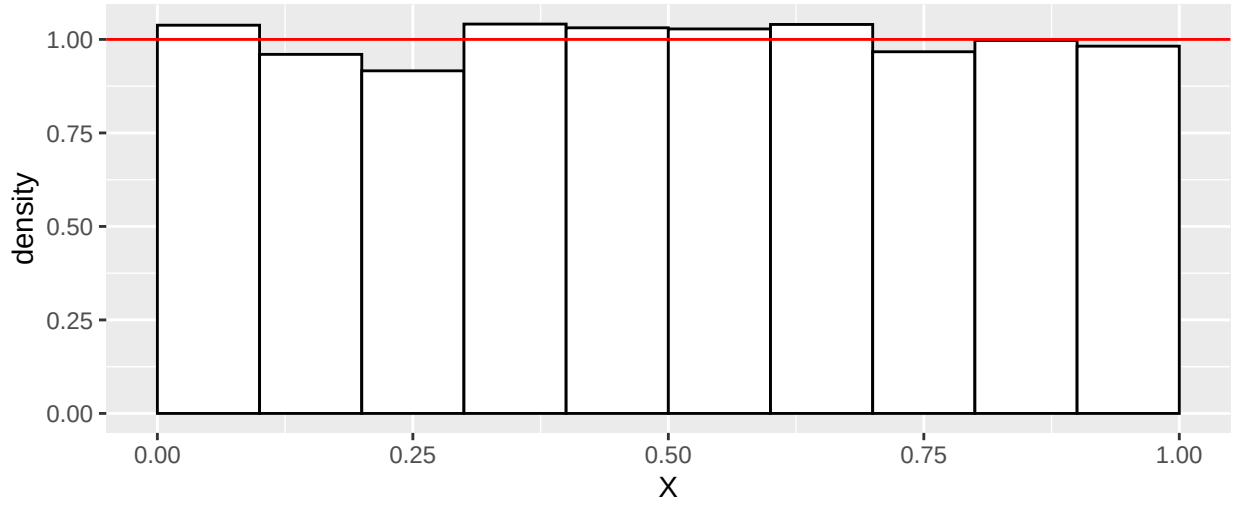
```
  ylab(TeX("$x_{n+1}$"))
```



On voit que nous obtenons un graphe avec 4 points alors que notre suite comporte 100 termes. Cela signifie que les points (x_n, x_{n+1}) sont superposés sur les 4 positions du graphes. Si vous affichez le vecteur `res` dans votre console, vous vous rendrez compte que c'est tout à fait normale car `res` est cyclique : il est composé de répétitions du motif `1 3 9 7`. On ne simule donc pas des nombres aélatoires, si je connais la valeur de x_n, je peux deviner (en regardant le motif) la valeur de x_{n+1}.

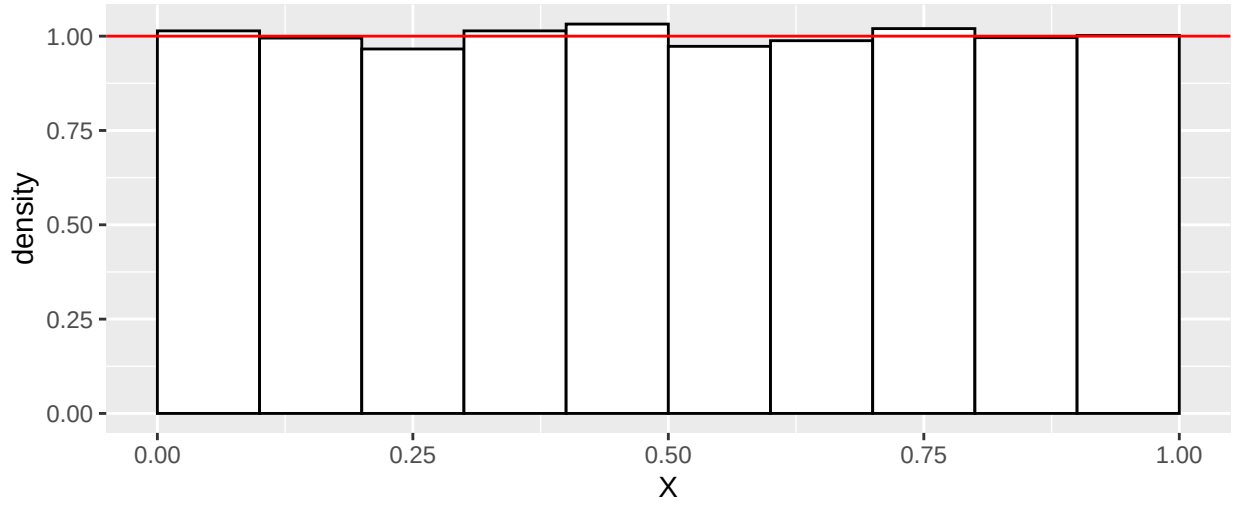
Recopions maintenant la deuxième fonction de l'énoncé.

```
# code de la fonction
pseudo2 <- function(N,x0)
{
  a=7^7
  b=2^31-1
  x=1:N
  X=1:N
  x[1] <- x0
  X[1] <- (x0%%b)/b
  for (j in c(1:(N-1)))
  {
    x[j+1] <- (a*x[j]+1)%%b
    X[j+1] <- x[j+1]/b
  }
  return(X)
}
# on stocke le résultat dans un vecteur
X <- pseudo2(10000,123456)
#hist(X) qui en ggplot s'écrit
ggplot()+geom_histogram(aes(x = X,y = ..density..),boundary=0,binwidth = 0.1,color="black", fill="white
  geom_hline(aes(yintercept = 1),col="red")
```



On voit ici que nous avons simulé des nombres compris entre 0 et 1 qui sont uniformément répartis sur tout les valeurs de $[0, 1]$. La courbe rouge représente la densité d'une loi uniforme sur $[0, 1]$ donc ça colle.
Le problème réside dans le fait que, si je ne change pas la valeur du paramètre `x0` de ma fonction `Pseudo2`.... le résultat ne change pas non plus. **En revanche**, si je change la valeur de `x0`, mon vecteur `X` (et donc l'allure de mon histogramme) change tout en gardant une pseudo distribution uniforme sur $[0, 1]$.
Donc, jusque là nous n'avons rien fait d'aléatoire même si on est heureux de l'allure de l'histogramme qui nous donne des nombres uniformément répartis (à peu près) sur $[0, 1]$. On obtient une allure différente pour chaque valeur de `x0`.
Maintenant, je vous propose le code de la fonction `Pseudo3` qui va nous permettre d'accéder au monde du pseudo-aléatoire. Je veux m'affranchir de ce paramètre `x0`. Ce que je voudrais c'est que cette valeur change toute seule, sans que je le fasse moi même en exécutant des lignes de codes. De cette manière en appelant ma fonction `Pseudo3`, j'obtiendrais un histogramme différent à chaque execution. Je pourrais ainsi simuler des nombres pseudo-aléatoires compris entre $[0, 1]$. On procède la manière suivante.

```
# code de la fonction
pseudo3 <- function(N)
{
  a=7^7
  b=2^31-1
  x=1:N
  X=1:N
  x0 <- as.numeric(Sys.time())
  x[1] <- x0
  X[1] <- (x0%%b)/b
  for (j in c(1:(N-1)))
  {
    x[j+1] <- (a*x[j]+1)%%b
    X[j+1] <- x[j+1]/b
  }
  return(X)
}
# on stocke le résultat dans un vecteur
X <- pseudo3(10000)
#hist(X) qui en ggplot s'écrit
ggplot()+geom_histogram(aes(x = X,y = ..density..),boundary=0,binwidth = 0.1,color="black", fill="white
  geom_hline(aes(yintercept = 1),col="red")
```



Si je définis mon paramètre `x0` comme étant l'heure qu'il est sur ma machine, vous serez d'accord qu'il ne prendra jamais deux fois la même valeur. `=)`
Je vous laisse le soin de tracer plusieurs histogrammes de vecteurs créés à partir de `pseudo3`, vous vous rendrez compte qu'ils sont tous différents.

On connait maintenant le processus historique qui a mené à la simulation de nombres pseudo-aléatoires compris entre 0 et 1. Dans le reste du TP, on utilisera cependant la commande `runif` pour simuler des nombres aléatoires suivant une loi uniforme.

Simulation d'autres lois que la loi uniforme

Dans cette section, on part toujours de la loi uniforme sur $[0,1]$ que je vais noter X dans la suite. On a que $X \sim \mathcal{U}[0,1]$. On va voir que l'on peut simuler beaucoup d'autres lois à partir de X.
Je rappelle que la fonction de répartition F_X de X vaut :

$$F_X(t) = \mathbb{P}(X \leq t) = t \quad \forall t \in [0,1]$$

Loi uniforme sur [a,b]

Pour simuler une loi uniforme sur $[a,b]$ à partir de X, rien de plus simple. On procède par transformations mathématiques. On a que :

$$\begin{aligned} X \sim \mathcal{U}[0,1] &\iff (b-a)X \sim \mathcal{U}[0,(b-a)] \\ &\iff (b-a)X + a \sim \mathcal{U}[a,b] \end{aligned}$$

Pour illustrer brièvement avec du code :

```
# simulation de 5 réalisations
# d'uniformes sur [3,7]
# je pars de mes uniformes sur [0,1]
X <- runif(5)
a <- 3
b <- 7
# j'applique la formule qui me donne le résultat
(b-a)*X+a
```

```
## [1] 4.438269 3.163353 3.822699 4.321864 3.348284
```

```r
# la commande runif(n = 5,min = 3,max = 7)
# permet d'obtenir directement le résultat
```

Lois discrètes

On va montrer que les Z et W définis dans l'énoncé sont de même loi. On nous demande de passer par la fonction de répartition. Nous allons donc prouver l'égalité entre la fonction de répartition F_Z de Z et F_W de W. On a que :

$$
\begin{aligned}
F_Z(t) &= \mathbb{P}(Z \leq t) \\
&= \sum_i \mathbb{P}(Z = z_i, z_i \leq t) \\
&= \sum_{z_i \leq t} p_i \\
F_W(t) &= \mathbb{P}(W \leq t) \\
&= \mathbb{P}(\sum_i^n z_i \mathbb{1}_{p_0+\ldots+p_{i-1} \leq X \leq p_0+\ldots+p_i} \leq t) \\
&= \sum_{z_i \leq t} \mathbb{P}(W = z_i) \\
&= \sum_{z_i \leq t} \mathbb{P}(p_0 + \ldots + p_{i-1} \leq X \leq p_0 + \ldots + p_i) = \sum_{z_i \leq t} \Big(\mathbb{P}(X \leq p_0 + \ldots + p_i) - \mathbb{P}(X \leq p_0 + \ldots + p_{i-1}) \Big) \\
&= \sum_{z_i \leq t} (p_0 + \ldots + p_i) - (p_0 + \ldots + p_{i-1}) \\
&= \sum_{z_i \leq t} p_i
\end{aligned}
$$

Passons au code :

```r
# pour 100 réalisations de Bernoulli de paramètre 0.5
# nombre de réalisations
n <- 100
# valeur du paramètre de la Bernouilli
p <- 0.5
# nos réalisations d'uniformes
X <- runif(n)
# on initialise notre vecteur dans lequel
# on stockera les réalisations de Bernouilli
Bern <- rep(0,n)
# critère de décision
Bern[X < p] <- 0
Bern[X > p] <- 1
Bern
```

```
##   [1] 1 0 1 1 1 0 1 0 1 0 0 1 0 1 1 1 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 1 0 0 1 1 0
##  [38] 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 0 0 1 0 1 1 1 0 0 1 1 1 0 1 0 1 1 0
##  [75] 1 0 0 0 0 0 1 1 1 0 0 1 1 1 0 1 0 1 1 0 0 0 0 0 0 1
```

```r
sum(Bern)
```

```
## [1] 55
```

On voit bien qu'on a environ 50 succès sur les 100 expériences de Bernouilli, le résultat est cohérent.

```
# simulation de 1000 lancers d'un dé équilibré
# nombre de réalisations
n <- 1000
# on part de nos uniformes comme toujours
x <- runif(n)
# on initialise un vecteur dans lequel on
# stockera les résultats des lancers
de <- rep(0,n)
de[x < 1/6] <- 1
de[x > 1/6 & x < 1/3] <- 2
de[x > 1/3 & x < 1/2] <- 3
de[x > 1/2 & x < 2/3] <- 4
de[x > 2/3 & x < 5/6] <- 5
de[x > 5/6 & x < 1] <- 6
c(sum(de==1),sum(de==2),sum(de==3),
  sum(de==4),sum(de==5),sum(de==6))
```

```
## [1] 162 176 172 166 155 169
```

On voit qu'on a bien obtenu les faces du dé avec des proportions à peu près égales.

Lois continues

Dans notre exemple, c'est-à-dire avec $I = \mathbb{R}$, la fonction F définit une bijection entre I et $]0, 1[$. donc on peut définir F^{-1}. On va maintenant s'intéresser à la manière de simuler des variables aléatoires à partir de l'inverse de leur fonction de répartition. L'idée vient du constat suivant, soit $X \sim U_{[0,1]}$, on a pour $t \in [0, 1]$:

$$\mathbb{P}(X \leq t) = t$$

Très bien, partons maintenant de la variable de l'énoncé $F_Z^{-1}(X)$. Pour connaitre la loi de cette variable, on se souvient que si on arrive à trouver et identifier une formule explicite de la fonction de répartition de cette variable, cela nous donnera sa loi. On a que :

$$\begin{aligned} \mathbb{P}(F_Z^{-1}(X) \leq t) &= \mathbb{P}(X \leq F_Z(t)) \quad \text{car F est une bijection croissante dans notre cas de figure} \\ &= F_Z(t) \quad \text{car X est une variable suivant une loi uniforme} \end{aligned}$$

On obtient donc que la fonction de répartition de la variable $F_Z^{-1}(X)$ est la même que celle de Z. Elles ont donc la même loi.

Calculons la fonction inverse de la fonction de répartition d'une loi exponentielle. On a tout d'abord que pour $Y \sim Exp(\lambda)$:

$$F_Y(t) = \int_0^t \lambda \exp(-\lambda y) \mathrm{d}y = [-\exp(-\lambda y)]_0^t = 1 - \exp(-\lambda t)$$

On obtient donc que :

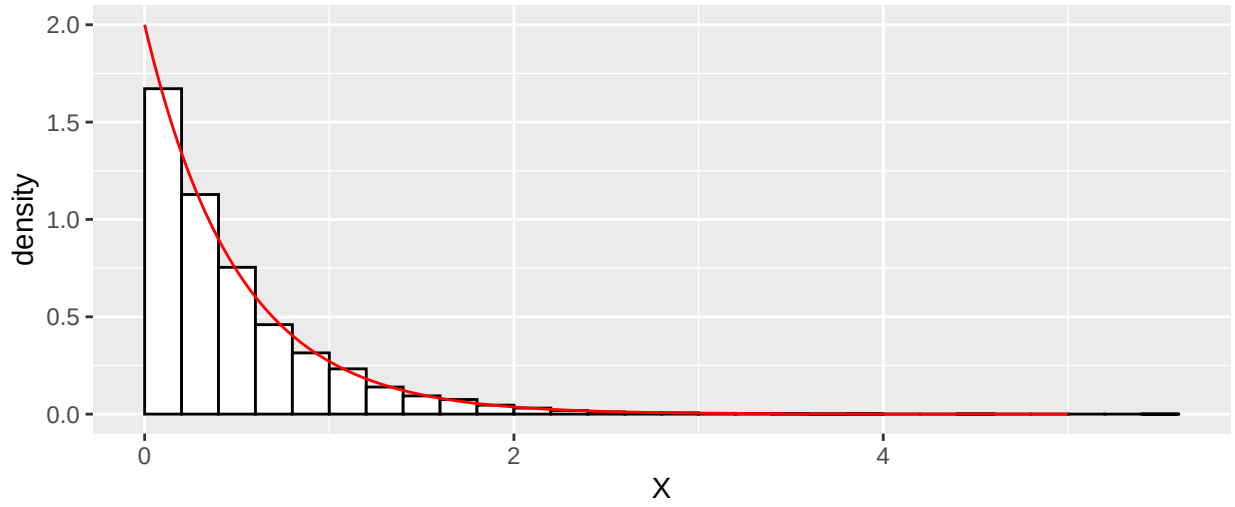
$$x = 1 - \exp(-\lambda t) \iff t = -\frac{1}{\lambda} \log(1 - x)$$

Donc $F_Y^{-1}(x) = -\frac{1}{\lambda} \log(1 - x)$.

On passe au code :

```
n <- 10000
X <- runif(n)
X <- -1/2*log(1-X)
# code pour illustrer graphiquement sans ggplot
```

```r
# {
#   hist(X,breaks = 50,freq = FALSE)
#   lines(seq(from = -100,to = 100,by = 0.01),dexp(seq(from = -100,to = 100,by = 0.01),2))
# }
# code avec ggplot
ggplot()+geom_histogram(aes(x = X,y = ..density..),
                        boundary = 0,binwidth = 0.2,col = "black",fill = "white")+
  geom_line(aes(x = seq(0,5,length.out = 1000),dexp(x = seq(0,5,length.out = 1000),rate = 2)),col="red")
```



On voit qu'on est proche de la densité théorique d'une loi exponentielle de paramètre 2 (droite en rouge).

Ensuite, on calcule la fonction de répartition d'une variable aléatoire suivant une loi de Cauchy :

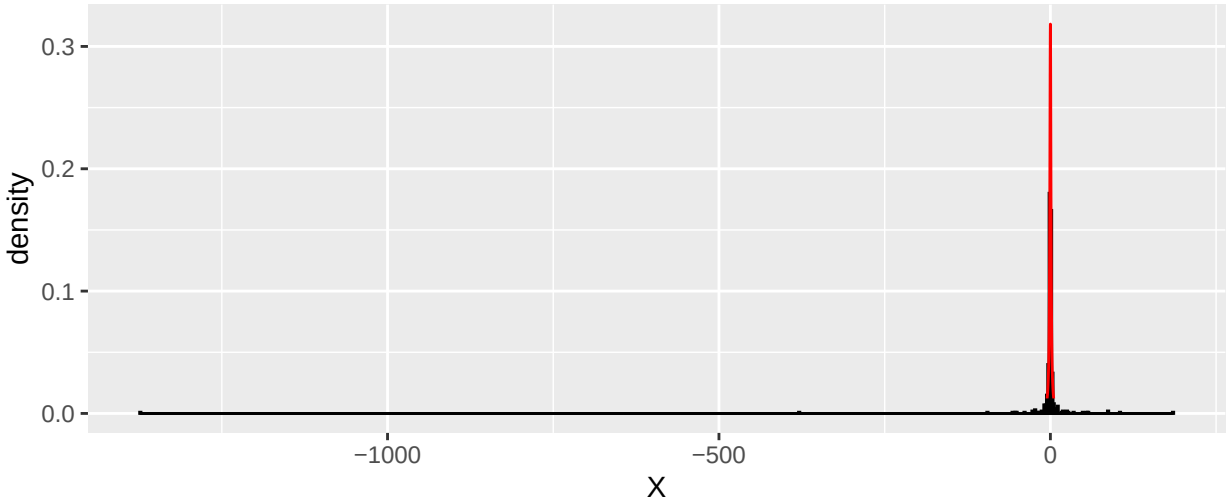
$$F_z(t) = \int_{-\infty}^{t} \frac{1}{\pi(1+x^2)} \mathrm{d}x = \frac{1}{\pi} \arctan(t) + \frac{1}{2}$$

On inverse cette fonction de répartition, on obtient :

$$F_Z^{-1}(x) = \tan(\pi(x - \frac{1}{2}))$$

Reste plus que le code.

```r
n <- 500
X <- runif(n)
X <- tan(pi*(X-0.5))
# code pour illustrer graphiquement sans ggplot
# {
#   hist(X,breaks = 100,freq = FALSE)
#   lines(seq(from = -100,to = 100,by = 0.01),dcauchy(seq(from = -100,to = 100,by = 0.01)))
# }
# code avec ggplot
ggplot()+geom_histogram(aes(x = X,y = ..density..),
                        boundary = 0,binwidth = 2,col = "black",fill = "white")+
  geom_line(aes(x = seq(-5,5,length.out = n),dcauchy(x = seq(-5,5,length.out = n))),col="red")
```



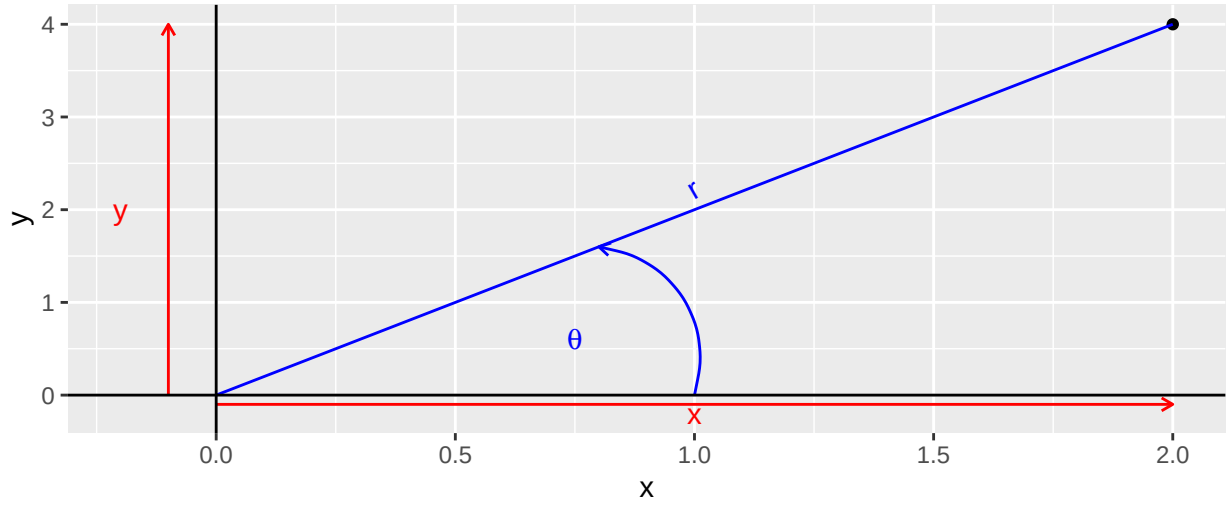
On voit qu'on est plutôt proche de la densité théorique d'une loi de Cauchy (droite en rouge).

Box-Muller

Quand on rentre dans le cas de la simulation de variables gaussiennes, on se rend compte que la méthode d'inversion est un peu problématique. En effet, je vous mets au défi de trouver une formule explicite de l'inverse de la fonction de répartition d'une loi normale. D'autres méthodes existent pour atteindre cet objectif. Je vous précise ici la démarche de la méthode de **Box-Muller**.

Intuition graphique de la méthode

Pour comprendre la méthode de Box-Muller, rappelons qu'il existe deux systèmes de coordonnées pour représenter un point de $\mathbb{R}^2$.
On peut représenter un point en utilisant les coordonnées cartésiennes (x, y) qui consistent à caractériser un point par son abscisse et son ordonnée (en rouge sur le graphe).
On peut également utiliser les coordonnées polaires (r, θ) qui consistent à caractériser un point par son rayon et son angle par rapport à l'origine du graphe *i.e.* le point de coordonnées $(0, 0)$ (en bleu sur le graphe).

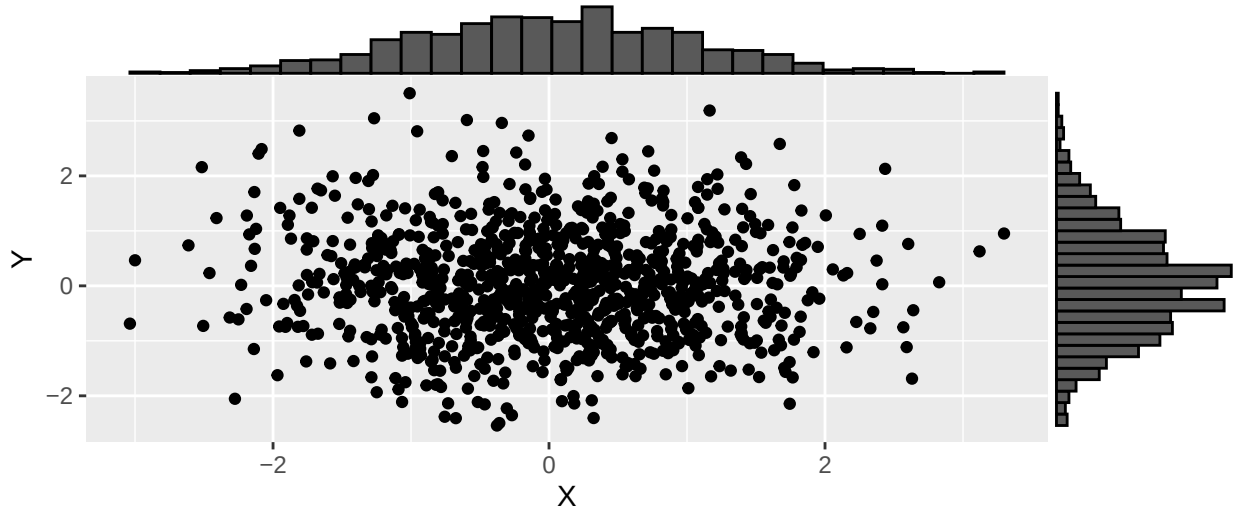


On peut relier les deux systèmes de coordonnées au moyen des équations suivantes :

$$x = r\cos(\theta)$$
$$y = r\sin(\theta)$$
$$r = \sqrt{x^2 + y^2}$$

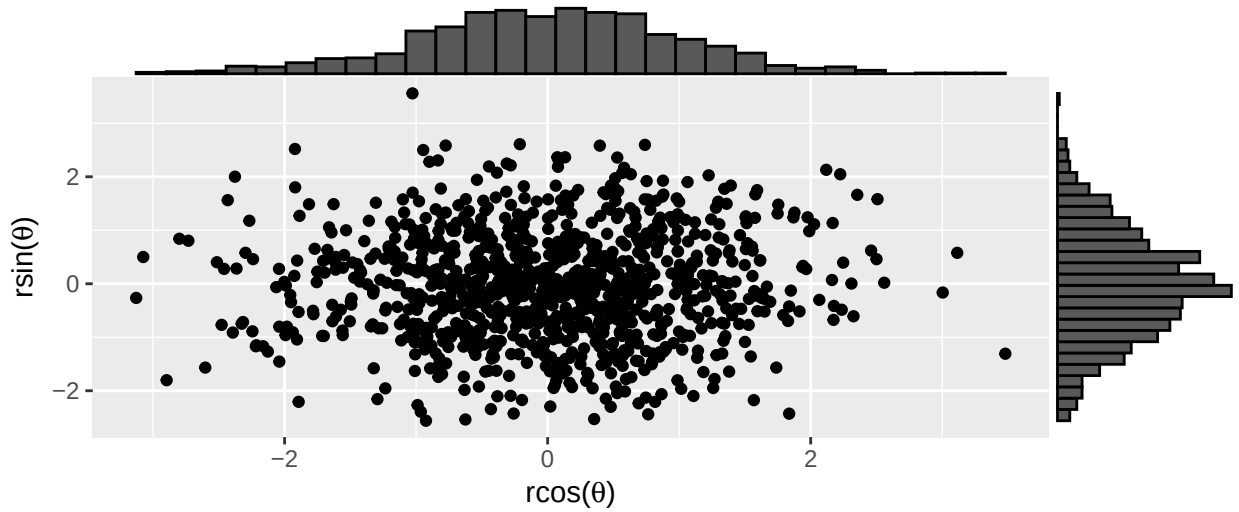
Un nuage de points gaussiens est un ensemble de points (ici de dimension 2) dont chaque coordonnée (l'abscisse et l'ordonnée) est une réalisation d'une loi normale (ici centrée réduite). Ce sont les points (x_i, y_j) simulés selon le couple (X_1, X_2) avec $X_1 \sim X_2 \sim \mathcal{N}(0, 1)$. La méthode de Box-Muller permet de montrer que je peux contruire un nuage de points gaussiens de deux manières différentes.
La première manière, la plus évidente, est de simuler les points **en passant par les coordonnées cartésiennes**. Cela donne :

```
# abscisses
X <- rnorm(1000)
# ordonnées
Y <- rnorm(1000)
p <- ggplot()+geom_point(aes(x = X,y = Y))
ggMarginal(p,type="histogram")
```



La deuxième méthode (et tout l'intérêt de Box-Muller) permet de simuler un nuage de points de même loi **en utilisant les coordonnées polaires**. Pour cela, il faut que les rayons de mes points suivent une loi exponentielle de paramètre $\frac{1}{2}$ et les angles suivent une loi uniforme sur $[0, 2\pi]$. Graphiquement, cela donne :

```
# abscisses
r <- rexp(n = 1000,rate = 0.5)
# ordonnées
theta <- runif(n = 1000,min = 0,max = 2*pi)
p <- ggplot()+geom_point(aes(x = sqrt(r)*cos(theta),y = sqrt(r)*sin(theta)))+xlab(TeX("$r\\cos(\\theta)$
ylab(TeX("$r\\sin(\\theta)$"))
ggMarginal(p,type="histogram")
```

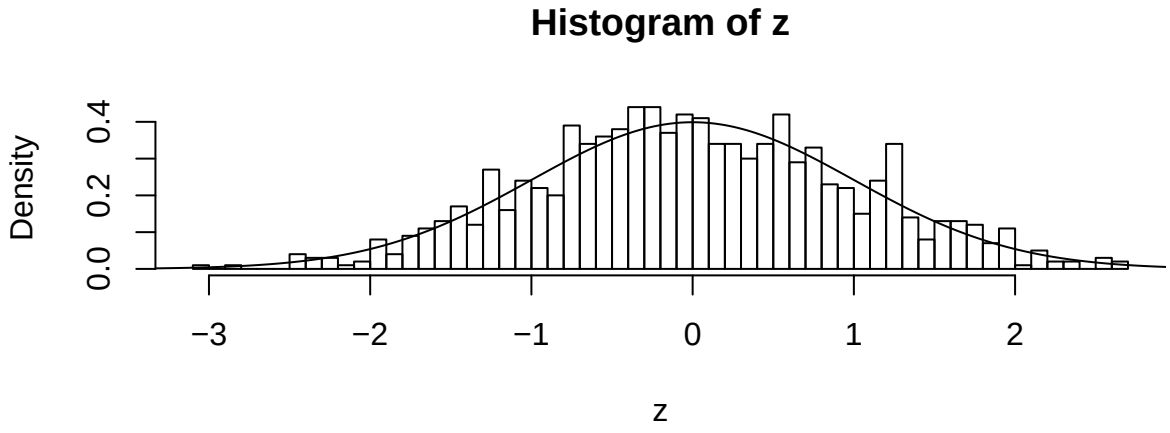


On peut se rendre compte que les deux nuages de points que j'ai tracé sont de loi identique (regardez les histogrammes associés aux axes des abscisses et des ordonnées qui sont semblables dans les deux cas).

Je vous ai fait en TD une démonstration "avec les mains" de la méthode de Box-Muller. Pour une démonstration rigoureuse, je vous invite à consulter le lien suivant : https://www.youtube.com/watch?v=vjPh1EBkxCI
Je rappelle que cette démonstration n'est pas au programme, ce lien est mis à disposition pour votre curiosité.

On sait donc maintenant simuler des gaussiennes (centrées réduites) grâce à la méthode de Box-Muller !!!
Passons donc à la simulation.

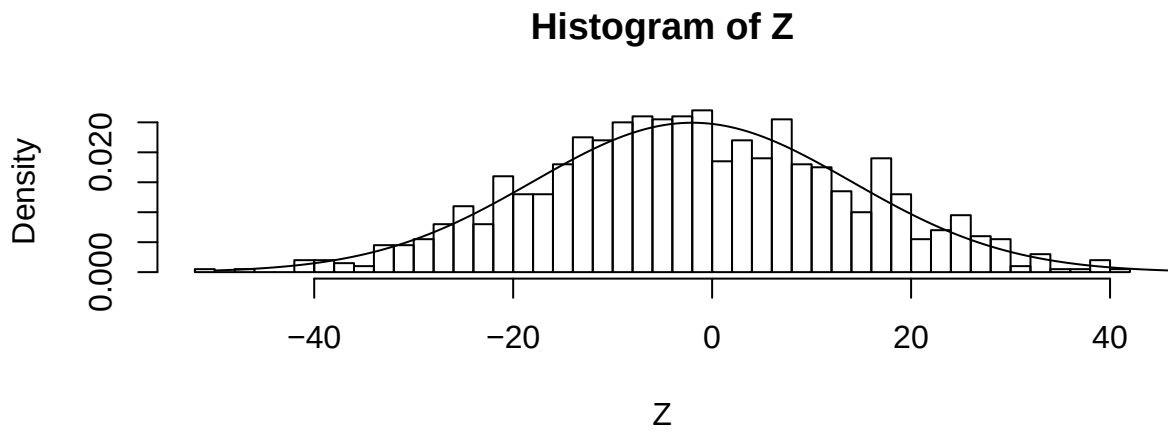
```
u1 <- runif(1000) # simulation de 1000 réalisations d'une loi uniforme sur [0,1]
u2 <- runif(1000) # de même
z <- sqrt(-2*log(u1))*cos(2*pi*u2) # on applique la formule de Box Muller
# on vérifie qu'on obtient déjà une loi normale centrée réduite
X <- seq(from = -4,to = 4,by = 0.001)
{
  hist(z,breaks = 50,freq = FALSE)
  lines(X,dnorm(x = X))
}
```




```
# youpi !!!
```

Maintenant que l'on peut simuler une loi normale centrée réduite, pour simuler la loi $\mathcal{N}(-2, 4^2)$, la procédure est assez directe. Si $X \sim \mathcal{N}(0,1)$, alors $\sigma X + \mu \sim \mathcal{N}(\mu, \sigma^2)$. Donc, passons au code :

```
Z <- 16*z-2
X <- seq(from = -50,to = 50,by = 0.01)
{
  hist(x = Z,breaks = 50,freq = FALSE)
  lines(x = X,y = dnorm(X,mean = -2,sd = 16))
}
```



Loi des grands nombres et théorème de la limite centrale

Nous allons illustrer la loi des grands nombres et le théorème de la limite centrale pour toutes les lois étudiées dans ce TP, à savoir :

- loi uniforme
- loi de Bernoulli
- loi exponentielle
- loi de Cauchy
- loi normale

La loi uniforme

Calcul de l'espérance :

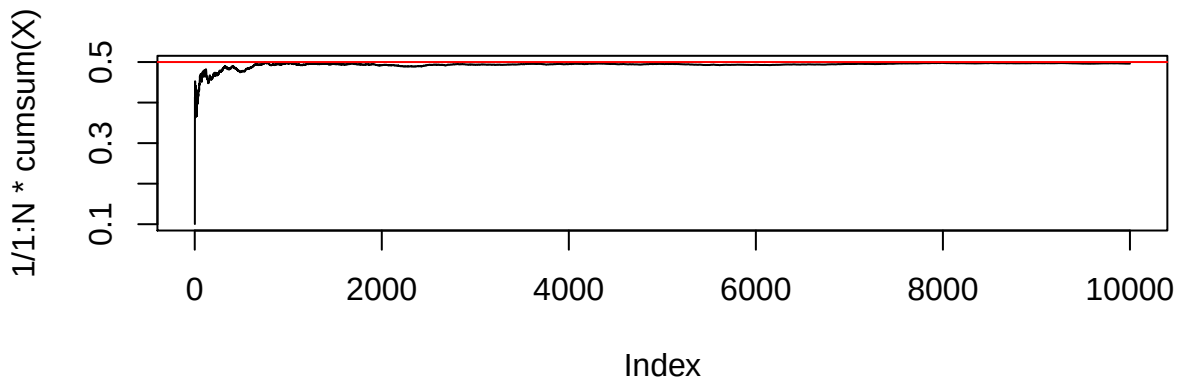
$$\begin{aligned}\mathbb{E}(X) &= \int_{\mathbb{R}} x \mathbb{1}_{[0,1]}(x) \mathrm{d}x \\ &= \int_0^1 x \mathrm{d}x = \frac{1}{2}\end{aligned}$$

Calcul de la variance :

$$\begin{aligned}\mathbb{E}(X^2) &= \int_{\mathbb{R}} x^2 \mathbb{1}_{[0,1]}(x)\mathrm{d}x \\ &= \int_0^1 x^2 \mathrm{d}x = \frac{1}{3} \\ \mathbb{V}(X) &= \mathbb{E}(X^2) - \mathbb{E}(X)^2 \\ &= \frac{1}{3} - \frac{1}{4} = \frac{1}{12}\end{aligned}$$

On illustre la loi des grands nombres.

```
# on simule N uniformes
N <- 10^4
X <- runif(N)
# esperance de X
E <- 0.5
{
  plot(1/1:N*cumsum(X),type = "l") # tracé de mes
# moyennes empiriques (les (n,X_n) de l'énoncé)
  abline(h = E,col = "red") # tracé de l'esperance
}
```

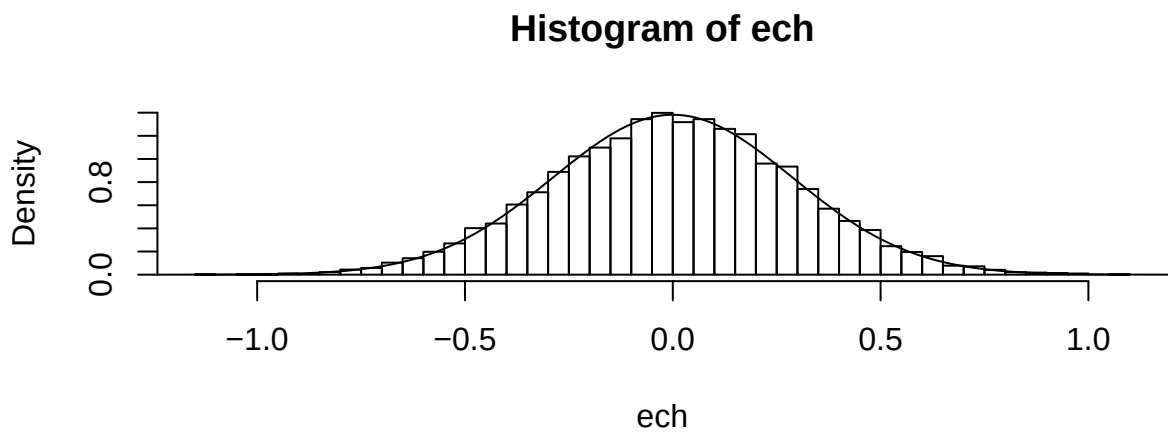



```
# convergence
```

On illustre le théorème de la limite centrale.

```
# simule M echantillons
# de N réalisations de loi
# uniforme
N <- 10^4
M <- 10^4
# variance de X
V <- 1/12
###
ech <- c()
for(i in 1:M)
{
  X <- runif(N)
  ech <- c(ech,sqrt(N)*(mean(X)-E))
```

```
}
abs <- seq(from = -5,to = 5,by = 0.01)
{
  hist(x = ech,breaks = 50,freq = FALSE)
  lines(x = abs,y = dnorm(x = abs,sd = sqrt(V))) # juste
# pour bien se convaincre que ça a une allure de loi
# normale
}
```



La loi de Bernouilli

Calcul de l'espérance :

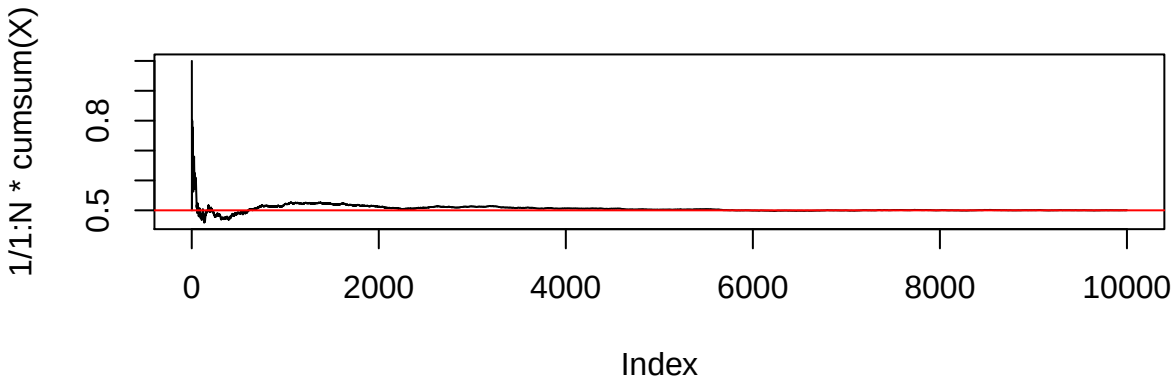
$$\begin{aligned}\mathbb{E}(X) &= \sum_i x_i \mathbb{P}(X = x_i) \\ &= 0 \times 0.5 + 1 \times 0.5 = 0.5\end{aligned}$$

Calcul de la variance :

$$\begin{aligned}\mathbb{E}(X^2) &= \sum_i x_i^2 \mathbb{P}(X = x_i) \\ &= 0 \times 0.5 + 1 \times 0.5 \\ \mathbb{V}(X) &= \mathbb{E}(X^2) - \mathbb{E}(X)^2 \\ &= \frac{1}{2} - \frac{1}{4} = \frac{1}{4}\end{aligned}$$

On illustre la loi des grands nombres.

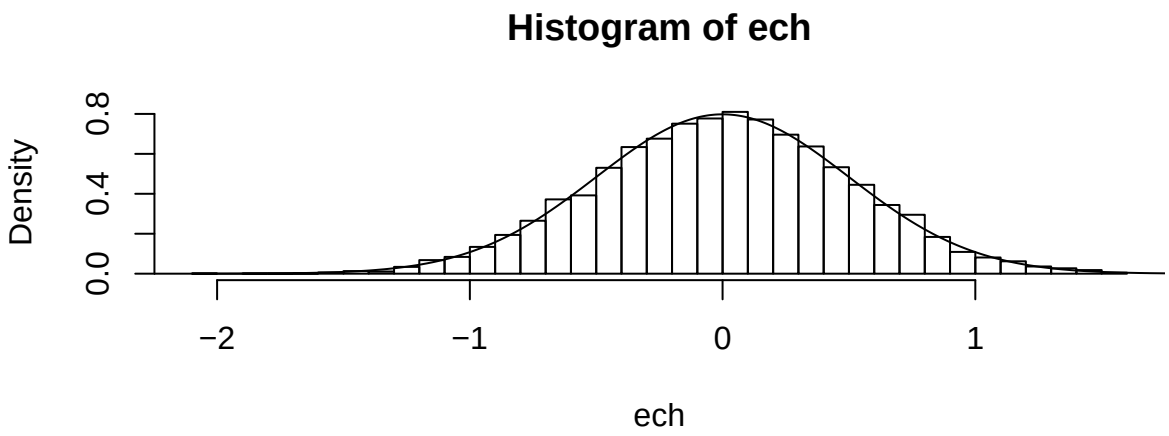
```
# on simule N réalisations de loi de Bernoulli
X <- rbinom(n = N,size = 1,prob = 0.5)
# esperance
E <- 0.5
{
  plot(1/1:N*cumsum(X),type = "l") # tracé de mes
# moyennes empiriques (les (n,X_n) de l'énoncé)
  abline(h = E,col = "red") # tracé de la moyenne
# théorique
}
```




```r
# convergence
```

On illustre le théorème de la limite centrale.

```r
# simule M echantillons
# de N réalisations de loi
# uniforme
N <- 10^4
M <- 10^4
# variance de X
V <- 1/4
###
ech <- c()
for(i in 1:M)
{
  X <- rbinom(n = N,size = 1,prob = 0.5)
  ech <- c(ech,sqrt(N)*(mean(X)-E))
}
abs <- seq(from = -5,to = 5,by = 0.01)
{
  hist(x = ech,breaks = 50,freq = FALSE)
  lines(x = abs,y = dnorm(x = abs,sd = sqrt(V))) # juste
# pour bien se convaincre que ça a une allure de loi
# normale
}
```



Loi exponentielle

Calcul de l'espérance :

$$\begin{aligned}
\mathbb{E}(X) &= \int_0^{+\infty} x\lambda \exp(-\lambda x)\mathrm{d}x \\
&= \Big[-x \exp(-\lambda x) \Big]_0^{+\infty} + \int_0^{+\infty} \exp(-\lambda x)\mathrm{d}x \\
&= \int_0^{+\infty} \exp(-\lambda x)\mathrm{d}x \\
&= \Big[-\frac{1}{\lambda} \exp(-\lambda x) \Big]_0^{+\infty} = \frac{1}{\lambda}
\end{aligned}$$

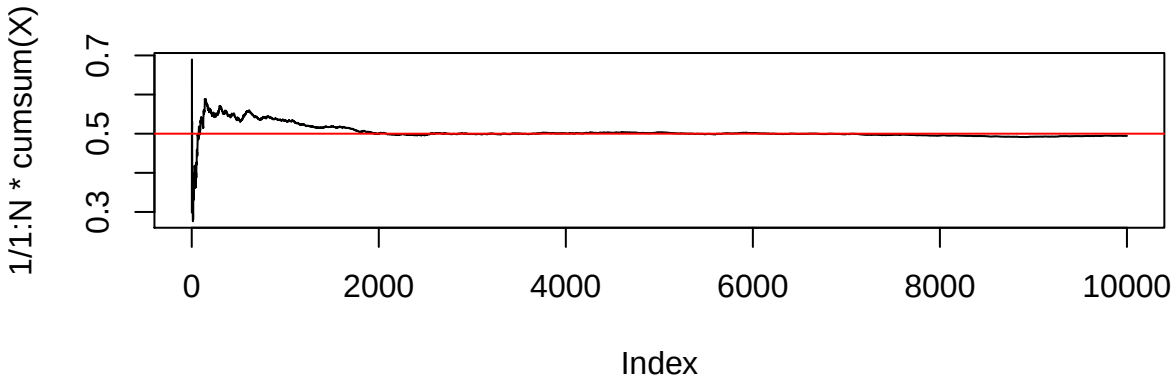
Calcul de la variance :

$$\begin{aligned}
\mathbb{E}(X^2) &= \int_0^{+\infty} x^2\lambda \exp(-\lambda x)\mathrm{d}x \\
&= \Big[-x^2 \exp(-\lambda x) \Big]_0^{+\infty} + \int_0^{+\infty} 2x \exp(-\lambda x)\mathrm{d}x \\
&= \int_0^{+\infty} 2x \exp(-\lambda x)\mathrm{d}x = \frac{2}{\lambda} \int_0^{+\infty} \lambda x \exp(-\lambda x)\mathrm{d}x \\
&= \frac{2}{\lambda^2} \\
\mathbb{V}(X) &= \mathbb{E}(X^2) - \mathbb{E}(X)^2 \\
&= \frac{2}{\lambda^2} - \frac{1}{\lambda^2} = \frac{1}{\lambda^2}
\end{aligned}$$

On illustre la loi des grands nombres.

```
# on simule N réalisations
# de loi exponentielle et
# de paramètre lambda
X <- rexp(N,2)
# esperance de X
E <- 1/2
{
```

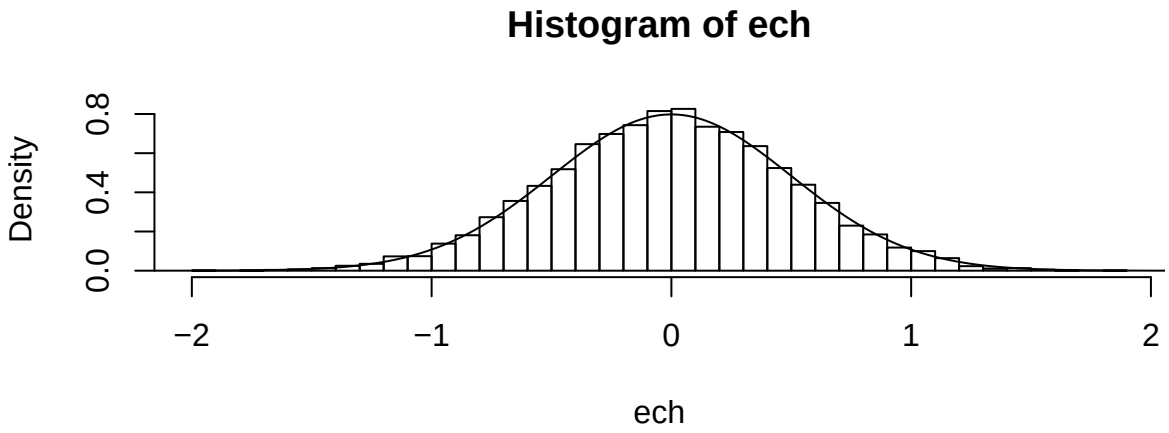
```
  plot(1/1:N*cumsum(X),type = "l") # tracé de mes moyennes empiriques (les (n,X_n) de l'énoncé)
  abline(h = E,col = "red") # tracé de la moyenne théorique
}
```




```
# convergence
```

On illustre le théorème de la limite centrale.

```
# simule M echantillons
# de N réalisations de loi
# uniforme
N <- 10^4
M <- 10^4
# variance de X
V <- 1/4
###
ech <- c()
for(i in 1:M)
{
  X <- rbinom(n = N,size = 1,prob = 0.5)
  ech <- c(ech,sqrt(N)*(mean(X)-E))
}
abs <- seq(from = -5,to = 5,by = 0.01)
{
  hist(x = ech,breaks = 50,freq = FALSE)
  lines(x = abs,y = dnorm(x = abs,sd = sqrt(V))) # juste
# pour bien se convaincre que ça a une allure de loi
# normale
}
```



Loi normale

Calcul de l'espérance :

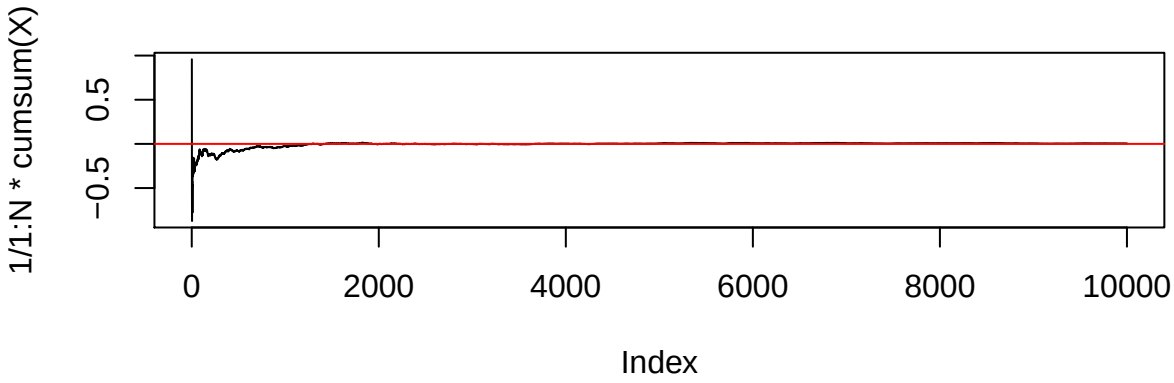
$$\mathbb{E}(X) = \mu$$

Calcul de la variance :

$$\mathbb{V}(X) = \sigma^2$$

On illustre la loi des grands nombres.

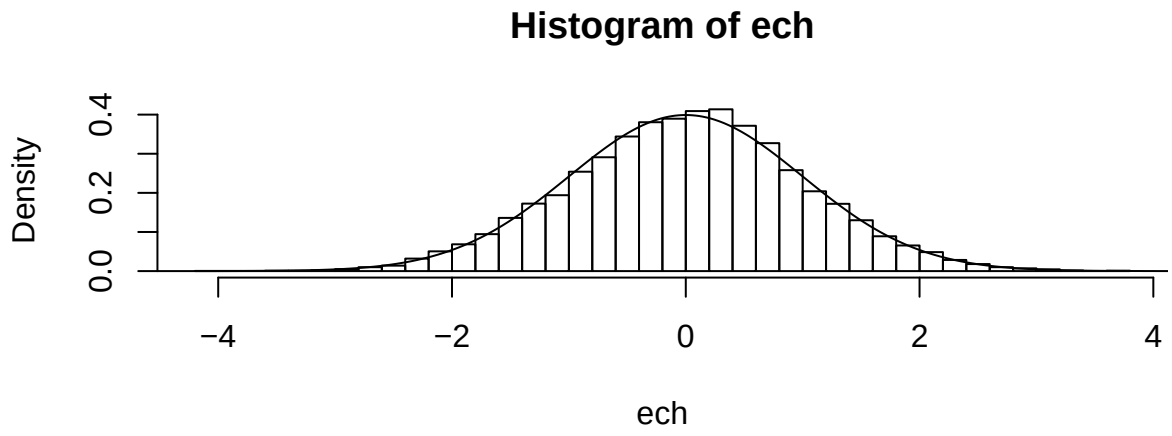
```
# on simule N réalisations
# de loi normale et
# centrée réduite
X <- rnorm(N)
# esperance de X
E <- 0
{
  plot(1/1:N*cumsum(X),type = "l") # tracé de mes moyennes empiriques (les (n,X_n) de l'énoncé)
  abline(h = E,col = "red") # tracé de la moyenne théorique
}
```



```r
# convergence
```

On illustre le théorème de la limite centrale.

```r
# simule M echantillons
# de N réalisations de loi
# uniforme
N <- 10^4
M <- 10^4
# variance de X
V <- 1
###
ech <- c()
for(i in 1:M)
{
  X <- rnorm(n = N)
  ech <- c(ech,sqrt(N)*(mean(X)-E))
}
abs <- seq(from = -5,to = 5,by = 0.01)
{
  hist(x = ech,breaks = 50,freq = FALSE)
  lines(x = abs,y = dnorm(x = abs,sd = sqrt(V))) # juste
# pour bien se convaincre que ça a une allure de loi
# normale
}
```



Loi de Cauchy

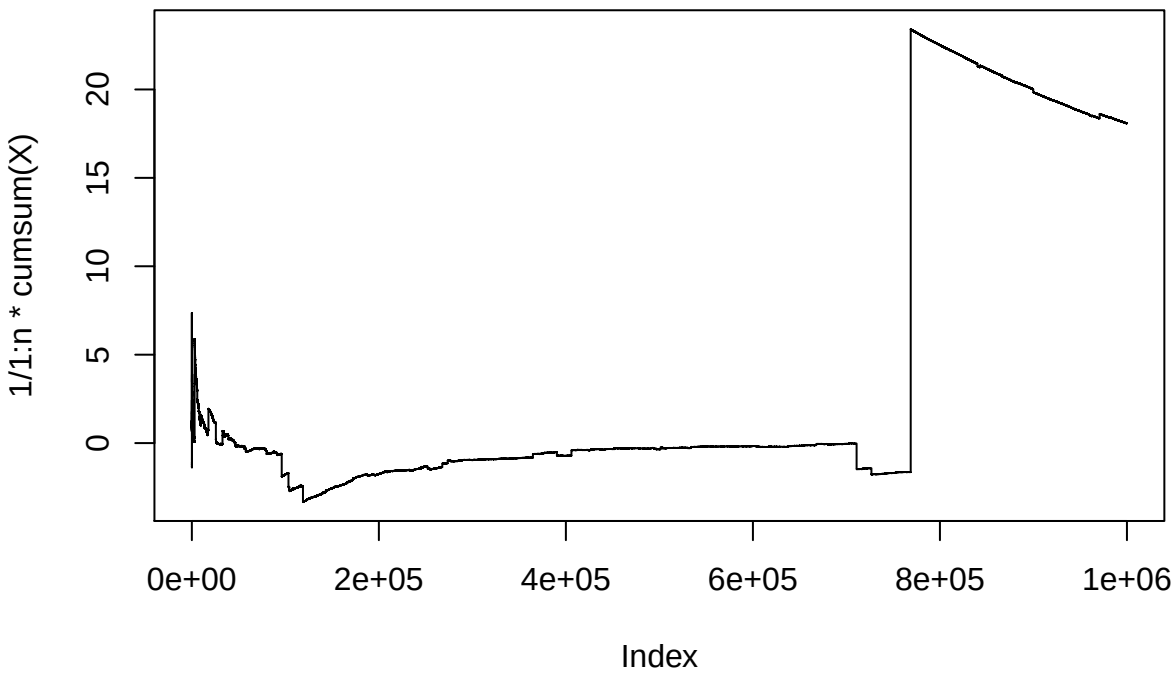
Illustration de la loi des grands nombres.

```r
# loi de cauchy
n <- 10^6
X <- rcauchy(n)
mean(X)
```

```
## [1] 18.07752
```

```r
{
  plot(1/1:n*cumsum(X),type = "l") # tracé de mes moyennes empiriques (les (n,X_n) de l'énoncé)
```

```
}
```




```
# ne converge pas
```

On voit ici que toutes les lois convergent sauf la loi de Cauchy !!! Pourquoi la loi de Cauchy (au demeurant fort sympatique) ne converge pas ? Eh bien lançons nous dans le calcul de son espérance théorique. On a, pour $X \sim Cauchy(0,1)$:

$$\begin{aligned}\mathbb{E}(X) &= \int_{-\infty}^{+\infty} x \frac{1}{\pi} \frac{1}{1+x^2} \mathrm{d}x \\ &= [x \arctan(x)]_{-\infty}^{+\infty} - \int_{-\infty}^{+\infty} \arctan(x) \mathrm{d}x \quad \text{par intégration par parties}\end{aligned}$$

Or le terme de gauche diverge !!! La loi de Cauchy n'admet donc pas d'espérance théorique. Le paramètre de gauche rentré dans la loi de Cauchy n'est qu'un paramètre de position.

On recommence pour l'illustration du théorème de la limite centrale.

```
# ceci marche pour toutes les lois sauf celle de Cauchy, observons
ech <- c()
for(i in 1:M)
{
  X <- rcauchy(n = N)
  ech <- c(ech,sqrt(N)*(mean(X)-0.5))
}
abs <- seq(from = -5,to = 5,by = 0.01)
{
  hist(x = ech,breaks = 50,freq = FALSE)
```

```
  lines(x = abs,y = dnorm(x = abs,sd = sd(ech))) # juste pour bien se convaincre que ça N'A PAS une all
}
```