

TP1 : Méthode de Monte Carlo pour la simulation probabiliste

GAUDIN Emma

DIDIER Thomas

1) La simulation et les méthodes de Monte Carlo

Introduction

Dans ce TP nous verrons comment simuler différentes lois grâce aux méthodes de Monte-Carlo. Dans un premier temps, on se familiarisera avec la simulation d'une loi uniforme sur $[0,1]$. Puis nous étudierons 2 méthodes de simulation : la méthode de l'inverse, et la méthode de l'acceptation-rejet. Pour ces 2 méthodes, on commencera par une étude théorique, puis nous coderons ces méthodes sur R.

2) Simuler un échantillon de nombres choisis au hasard entre 0 et 1

Code R2

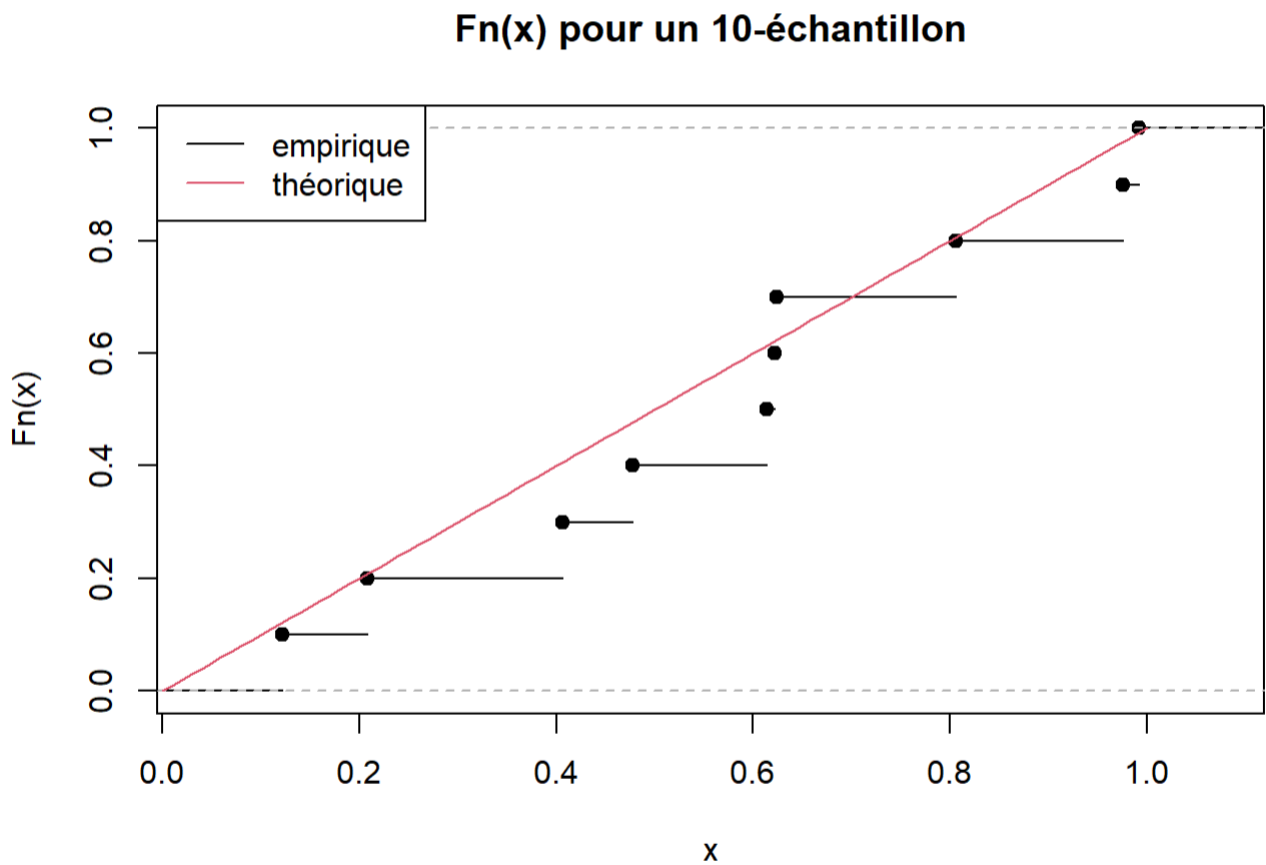
Dans cette section on a comparé les fonctions de répartition théorique et empirique pour la loi uniforme sur $[0,1]$. Pour un n -échantillon, on trace sa fonction de répartition empirique grâce à la fonction `ecdf` :

```
#Code R2 ----

x10 = runif(10)
plot(ecdf(x10), main = "Fn(x) pour un 10-échantillon")

curve(punif, 0, 1, add = TRUE, col = 2)

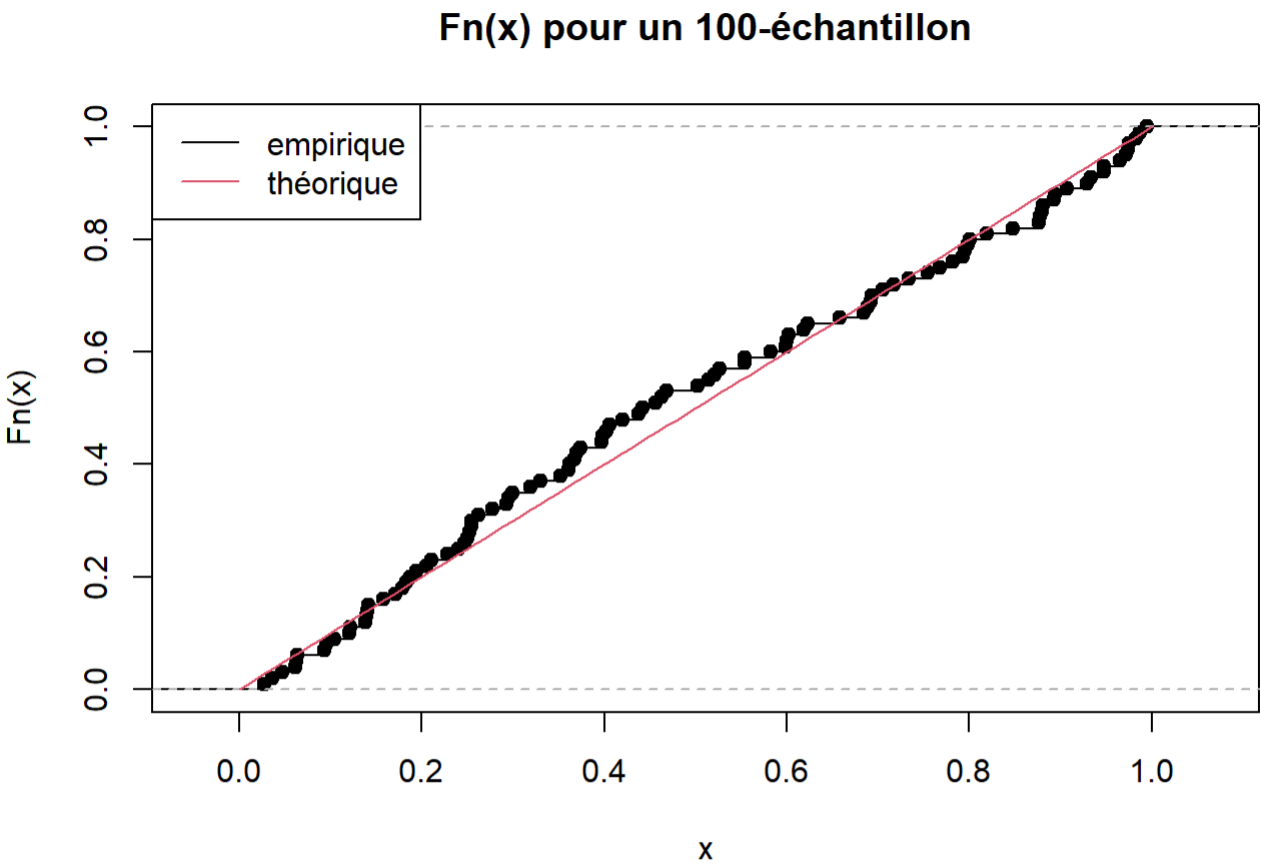
legend("topleft", legend = c("empirique", "théorique"), col = c(1, 2), lty = c(1, 1));
```



```
x100 = runif(100)
plot(ecdf(x100), main = "Fn(x) pour un 100-échantillon")

curve(punif, 0, 1, add = TRUE, col = 2)

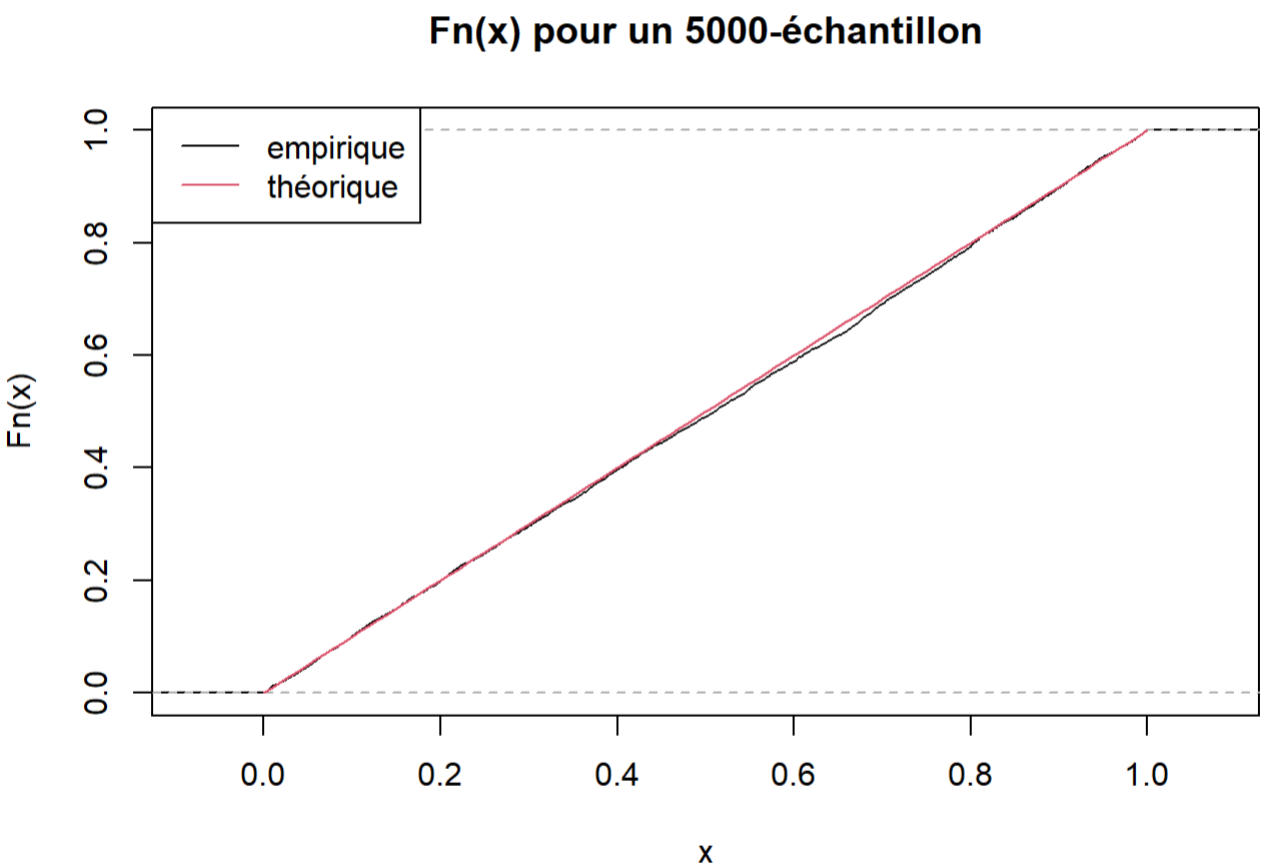
legend("topleft", legend = c("empirique", "théorique"), col = c(1, 2), lty = c(1, 1));
```



```
x5000 = runif(5000)
plot(ecdf(x5000), main = "Fn(x) pour un 5000-échantillon")

curve(punif, 0, 1, add = TRUE, col = 2)

legend("topleft", legend = c("empirique", "théorique"), col = c(1, 2), lty = c(1, 1));
```



Grâce à ces différentes courbes, on remarque que plus le nombre d'échantillon augmente, plus la précision de la fonction de répartition empirique augmente; elle se rapproche de la courbe théorique. C'est une illustration de la loi forte des grands nombres.

3) Deux méthodes classiques de générations d'échantillon d'une loi

3.1) Méthode de l'inverse

Lemma 3.2 Soit U une v.a. de loi uniforme sur $[0, 1]$. Alors pour toute v.a. X , la v.a. $Q_X(U)$ admet la même loi que X .

Preuve:

Soit $x \in \mathbb{R}$, en posant $Y = Q_X(U)$, on a :

$$\begin{aligned} F_Y(x) &= \mathbb{P}(Y \leq x) \\ &= \mathbb{P}(Q_X(U) \leq x) \\ &= \mathbb{P}(U \leq F_X(x)) \\ &= \begin{cases} 0, & \text{si } F_X(x) \leq 0, \\ 1, & \text{si } F_X(x) \geq 1, \\ F_X(x), & \text{sinon.} \end{cases} \end{aligned}$$

Ainsi, $Q_X(U)$ admet la même loi que X , puisque $F_X(\mathbb{R}) \subset [0, 1]$.

Code R5: Loi de Laplace

Une variable aléatoire Y suit une loi de Laplace de paramètre $a > 0$ si elle admet pour densité $g_a(x) = \frac{a}{2} \exp(-a|x|)$ pour tout $x \in \mathbb{R}$, associée à une loi dite doublement exponentielle ou de Laplace.

On montre que l'on peut générer une valeur d'une v.a. de loi de Laplace par la méthode de l'inverse:

Soit Y une variable aléatoire de densité :

$$g_a(x) = \frac{a}{2} \exp(-a|x|), \quad x \in \mathbb{R}.$$

Sa fonction de répartition est :

$F(x) =$

$$\begin{cases} \frac{a}{2} e^{ax}, & x \leq 0, \\ 1 - \frac{a}{2} e^{-ax}, & x \geq 0. \end{cases}$$

On considère $U \sim \text{Unif}([0, 1])$ et on cherche $Y = G(U)$ tel que Y suive la loi de Laplace.

La méthode de l'inverse consiste à poser :

$$U = F(Y) \iff Y = G(U).$$

Ainsi, la fonction inverse est donnée par :

$G(U) =$

$$\begin{cases} \frac{1}{a} \ln(2U), & U \in [0, 1/2[, \\ -\frac{1}{a} \ln(2(1 - U)), & U \in [1/2, 1[. \end{cases}$$

En conclusion, $Y = G(U)$ suit bien la loi de Laplace : la méthode de l'inverse permet de générer une valeur de cette loi à partir d'une variable uniforme $U \sim \text{Unif}([0, 1])$.

On construit maintenant une fonction `laplace(n,a)`, où n est un entier, qui simule la réalisation d'un n -échantillon (Y_1, \dots, Y_n) de la loi de Laplace par la méthode de l'inverse

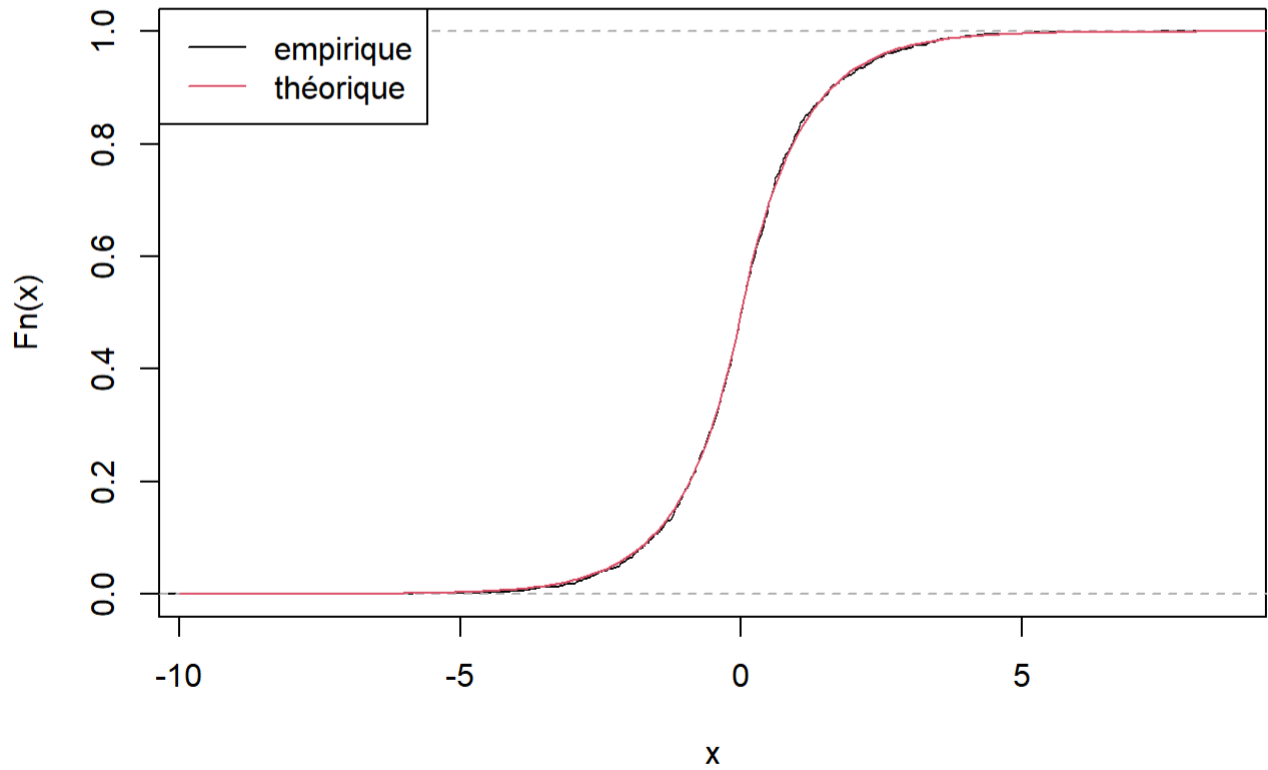
```
#Code R5 -----
rlaplace=function(n,a){

  u=runif(n);
  x=u
  x[u<=1/2]=1/a*log(2*x[u<=1/2]);
  x[u>1/2]=-1/a*log(2*(1-x[u>1/2]));

  return(x);
}

plot(ecdf(rlaplace(1000,1)),main="Simulation de la loi de Laplace par la méthode de l'inverse");
curve(1-1/2*exp(-1*x), from=0, to=10,add=TRUE, col=2)
curve(1/2*exp(1*x), from=-10, to=0,add=TRUE, col=2)
legend("topleft", legend = c("empirique", "théorique"), col = c(1, 2), lty = c(1, 1));
```

Simulation de la loi de Laplace par la méthode de l'inverse



On remarque grace à cette courbe qu’avec un nombre d’échantillons assez grand la méthode de l'inverse marche très bien et on retrouve la loi de laplace théorique.

3.2) Méthode de l’acceptation-rejet (AR) ou “Hit and Miss”

Lemma 3.3 Si Y est une v.a. de densité g et si $U \sim Unif([0, 1])$ indépendante de Y alors : 1. la probabilité d'accepter à l'étape 3 la valeur produite à l'étape 1 de l'algorithme est :

$$\mathbb{P}\{Ucg(Y) \leq f_X(Y)\} = \frac{1}{c}$$

Preuve:

$$\begin{aligned} \mathbb{P}(Ucg(Y) \leq f_X(Y)) &= \iint_{(u,y) \in [0,1] \times \mathbb{R},\, ucg(y) \leq f_X(y)} f_{U,Y}(u,y) \, du \, dy \\ &= \iint_{(u,y) \in [0,1] \times \mathbb{R},\, ucg(y) \leq f_X(y)} f_U(u) f_Y(y) \, du \, dy \end{aligned}$$

car U et Y sont indépendantes

On sait que U suit une loi uniforme : $U \sim Unif([0, 1])$ et la densité de Y est g donc :

$$\begin{aligned} \mathbb{P}(Ucg(Y) \leq f_X(Y)) &= \iint_{(u,y) \in [0,1] \times \mathbb{R},\, ucg(y) \leq f_X(y)} 1_{[0,1]}(u) g(y) \, du \, dy \\ \text{or } \mathbb{P}(ucg(y) \leq f_X(y)) &= \mathbb{P}\left(u \leq \frac{f_X(y)}{cg(y)}, g(y) > 0\right) + \cancel{\mathbb{P}(g(y) = 0)} \\ \text{Donc } \mathbb{P}(Ucg(Y) \leq f_X(Y)) &= \int_{\mathbb{R}} \int_{-\infty}^{\frac{f_X(y)}{cg(y)}} 1_{[0,1]}(u) g(y) \, du \, dy \end{aligned}$$

Par le théorème de Fubini–Tonelli,

$$\mathbb{P}(Ucg(Y) \leq f_X(Y)) = \int_{\mathbb{R}} g(y) \left(\int_0^{\min(1, \frac{f_X(y)}{cg(y)})} du \right) dy$$

Or $\frac{f_X(y)}{cg(y)} \leq 1$, donc :

$$\begin{aligned} \mathbb{P}(Ucg(Y) \leq f_X(Y)) &= \int_{\mathbb{R}} g(y) \frac{f_X(y)}{cg(y)} dy \\ &= \frac{1}{c} \int_{\mathbb{R}} f_X(y) dy \\ &= \frac{1}{c} \text{ car } f_X \text{ est une densité} \end{aligned}$$

On obtient bien:

$$\mathbb{P}(Ucg(Y) \leq f_X(Y)) = \frac{1}{c}$$

2. La loi de Y sachant que l'événement $\{Ucg(Y) \leq f_X(Y)\}$ est réalisé admet pour densité f_X .

Preuve:

Soit $A := \{Ucg(Y) \leq f_X(Y)\}$ avec $\mathbb{P}(A) = \frac{1}{c}$

$$\begin{aligned} F_{Y|A}(t) &= \mathbb{P}(Y \leq t \mid A) \\ &= \frac{\mathbb{P}((Y \leq t) \cap A)}{\mathbb{P}(A)} \\ &= \frac{\mathbb{P}((Y \leq t) \cap (Ucg(Y) \leq f_X(Y)))}{\mathbb{P}(A)} \\ &= c \int_{-\infty}^t \left(\int_0^{\frac{f_X(y)}{cg(y)}} du \right) g(y) dy \\ &= \int_{-\infty}^t f_X(y) dy \\ &= F_X(t). \end{aligned}$$

Commentaire 3.2

1. Il n'est pas nécessaire que f_X soit une densité. En effet, si l'on garde l'inégalité : $0 \leq f(x) \leq cg(x)$ avec $\int_{\mathbb{R}} f(x)dx \neq 1$, alors on peut montrer :

a. $\mathbb{P}(Ucg(Y) \leq f(Y)) = \int_{\mathbb{R}} f(y)dy/c.$

Preuve :

$$\begin{aligned} \mathbb{P}(Ucg(Y) \leq f(Y)) &= \mathbb{P}(U \leq \frac{f(Y)}{c\,g(Y)},\, g(Y) \neq 0) + \mathbb{P}(0 \leq f(Y),\, g(Y) = 0) \\ &= \mathbb{P}(U \leq \frac{f(Y)}{c\,g(Y)},\, g(Y) \neq 0) + \cancel{\mathbb{P}(g(Y) = 0)} \\ &= \mathbb{P}(U \leq h(Y)) \text{ où } h(Y) = \begin{cases} \frac{f(Y)}{cg(Y)} & \text{si } g(Y) \neq 0 \\ 0 & \text{si } g(Y) = 0 \end{cases} \\ &= \int_{\mathbb{R}} \mathbb{P}(U \leq h(y))\, g(y)\, dy, \, 0 \leq h(y) \leq 1 \\ &= \int_{\mathbb{R}} h(y)\, g(y)\, dy \\ &= \int_{\{g(y)>0\}} \frac{f(y)}{c}\, dy + \int_{\{g(y)=0\}} 0\, dy \\ &= \int_{\{g(y)>0\}} \frac{f(y)}{c}\, dy + \int_{\{g(y)=0\}} \frac{f(y)}{c}\, dy \\ &= \int_{\mathbb{R}} \frac{f(y)}{c}\, dy \end{aligned}$$

b. la loi de Y sachant que l'événement $\{Ucg(Y) \leq f(Y)\}$ est réalisé admet pour densité $f_X(\cdot) = f(\cdot)/\int_{\mathbb{R}} f(y)dy$

Preuve :

Soit $B := \{Ucg(Y) \leq f(Y)\}$ avec $\mathbb{P}(B) = \frac{\int_{\mathbb{R}} f(x)dx}{c}$

$$\begin{aligned} F_{Y|B}(t) &= \mathbb{P}(Y \leq t \mid B) \\ &= \frac{\mathbb{P}((Y \leq t) \cap B)}{\mathbb{P}(B)} \\ &= \frac{\mathbb{P}((Y \leq t) \cap (Ucg(Y) \leq f(Y)))}{\mathbb{P}(B)} \\ &= \frac{c}{\int_{\mathbb{R}} f(x)dx} \int_{-\infty}^t \left(\int_0^{h(y)} du \right) g(y)\, dy \\ &= \frac{c}{\int_{\mathbb{R}} f(x)dx} \int_{-\infty}^t h(y)\, g(y)\, dy \\ &= \int_{-\infty}^t \frac{c}{\int_{\mathbb{R}} f(x)dx} \frac{f(y)}{c}\, dy \\ &= \int_{-\infty}^t \frac{f(y)}{\int_{\mathbb{R}} f(x)dx}\, dy \\ &= F(t) \end{aligned}$$

Ainsi la méthode de l'acceptation-rejet peut être utilisée dans un cas plus large lorsque f_X n'est plus une densité ou quand elle n'est connue qu'à une constante multiplicative près.

Code R6

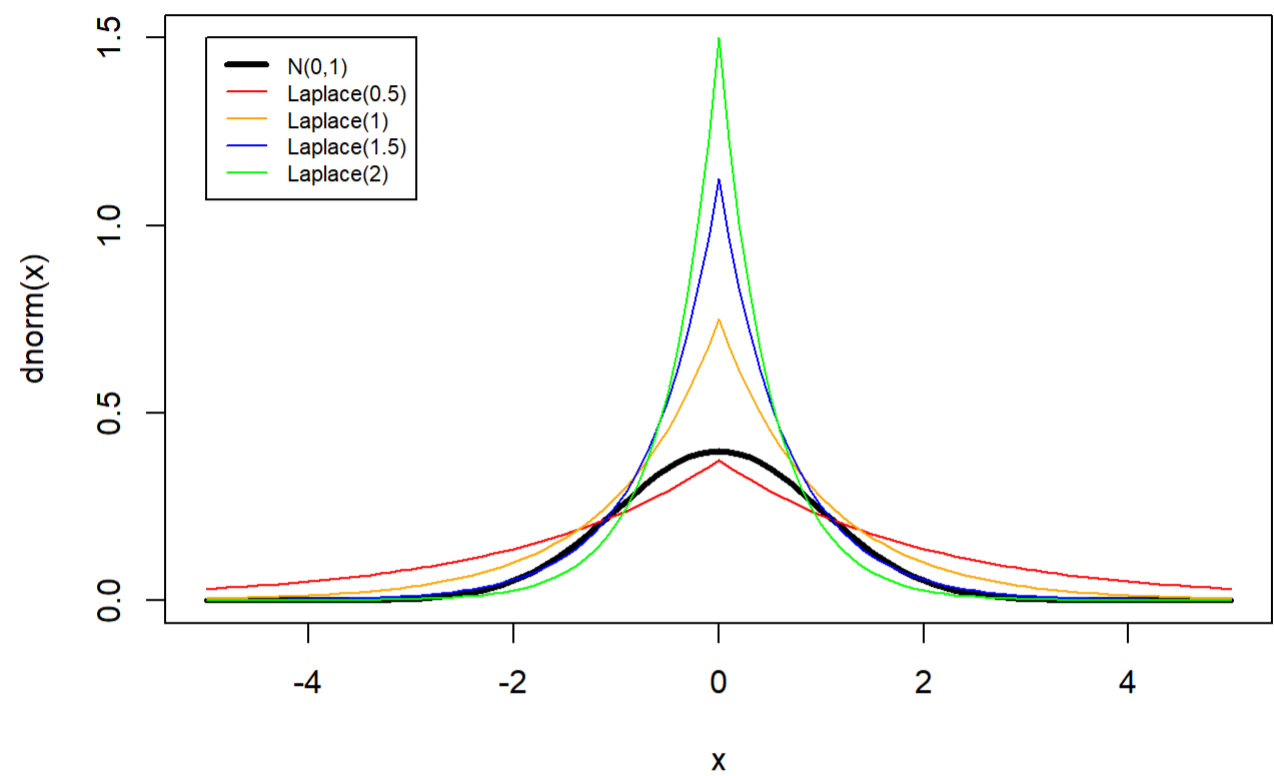
On souhaite maintenant simuler une réalisation d'une v.a. $X \sim N(0, 1)$. On considère la densité instrumentale suivante : pour $a > 0$ $g_a(x) = \frac{a}{2}\exp(-a|x|)$ pour tout $x \in \mathbb{R}$, associée à une loi dite doublement exponentielle ou de Laplace.

D'abord grâce à la fonction curve nous superposons les densités dnorm et g_a pour quelques valeurs de a.

```
#fonction densité de La loi de Laplace de paramètre a
dlaplace=function(a,x){
  return(a/2*exp(-a*abs(x)));
}

#On trace la densité de La loi N(0,1) et la loi de Laplace pour plusieurs
#valeurs du paramètre a
curve(dnorm,from=-5,to=5,col="black",ylim=c(0,1.5),xname = "x",lwd=3);
curve(1.5*dlaplace(0.5,x),from = -5, to=5,add=TRUE,col="red"); #a=0.5
curve(1.5*dlaplace(1,x),from = -5, to=5,add=TRUE,col="orange"); #a=1
curve(1.5*dlaplace(1.5,x),from = -5, to=5,add=TRUE,col="blue"); #a=1.5
curve(1.5*dlaplace(2,x),from = -5, to=5,add=TRUE,col="green",ylim=2); #a=2
title("Densité de la loi normale et des loi de Laplace")
legend(x=-5,y=1.5,
       legend=c("N(0,1)","Laplace(0.5)","Laplace(1)","Laplace(1.5)","Laplace(2)"),
       col=c("black","red","orange","blue","green"),
       lwd=c(3,1,1,1,1),cex=0.7)
```

Densité de la loi normale et des loi de Laplace



On remarque grâce à ce graphique qu’il est nécessaire de multiplier les densités g_a par une constante c_a afin de garantir l’inégalité $f_X(x) \leq c_a g_a(x)$ pour tout $x \in \mathbb{R}$. Mais alors, que vaut cette constante c_a ?

On a $\forall x \in \mathbb{R} : f_X(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$, et $g_a(x) = \frac{a}{2} e^{-a|x|}$ donc $\frac{f_X(x)}{g_a(x)} = \sqrt{\frac{2}{\pi}} \frac{1}{a} e^{-\frac{x^2}{2} + a|x|}$ Cette fonction est paire. On peut donc étudier l’intervalle où $x \geq 0$, le raisonnement étant similaire pour $x \leq 0$.

Ainsi, sur $x \geq 0$

$$\frac{-x^2}{2} + a|x| = \frac{-x^2}{2} + ax$$

est un polynôme du 2nd degrés admettant un maximum pour $x = a$. Par parité, si $x \leq 0$, le maximum est atteint pour $x = -a$, mais la valeur du maximum demeure inchangée.

Finalement, $\forall x \in \mathbb{R} : \frac{-x^2}{2} + a|x| \leq \frac{a^2}{2}$, et donc, comme la fonction e^x est croissante sur \mathbb{R} , on obtient l’inégalité suivante :

$$\frac{f_X(x)}{g_a(x)} \leq c_a := \sqrt{\frac{2}{\pi}} \frac{1}{a} e^{\frac{-a^2}{2}}$$

On cherche à présent la valeur minimale de c_a sur $]0, +\infty[$. En effet plus c_a est petit, plus la probabilité d’accepter une valeur de y est importante (d’après Lemme 3.3). Pour se faire on dérive c_a par rapport à a :

$$\frac{dc_a}{da} = \sqrt{\frac{2}{\pi}} \left(\frac{-1}{a^2} e^{\frac{-a^2}{2}} + \frac{1}{a} a e^{\frac{-a^2}{2}} \right) = \sqrt{\frac{2}{\pi}} e^{\frac{-a^2}{2}} \left(1 - \frac{1}{a^2} \right)$$

On voit que c_a admet un extremum pour $a = 1$. De plus si $a < 1$ la fonction est décroissante, et si $a > 1$, la fonction est croissante. On peut ainsi en conclure que c_a est minimale pour $a = 1$.

On prends donc $c = \sqrt{\frac{2e}{\pi}}$.

D’après le lemme 3.3 : $\mathbb{P}(U c g(Y) \leq f_X(Y)) = \frac{1}{c} = \sqrt{\frac{\pi}{2e}}$. La loi du nombre de répétitions de l’algorithme AR pour accepter une valeur candidate est géométrique de paramètre $\sqrt{\frac{\pi}{2e}}$. Donc l’espérance vaut $\sqrt{\frac{2e}{\pi}}$, qui correspond au nombre de tirage qu’il faut en moyenne pour otenir une valeur de X .

On code sur R, une fonction (nommée rnormAR) qui simule la réalisation d’un n-échantillon d’une loi N(0, 1) en utilisant la méthode d’acceptation-rejet.

```
#simulation d'un n echantillon d'une loi normale en utilisant la méthode
#de l'acceptation-rejet
rnormAR=function(n){

  y=rlaplace(round(sqrt(2*exp(1)/pi)*n),1);
  cgy=sqrt(2*exp(1)/pi)*dlaplace(1,y);
  u=runif(round(sqrt(2*exp(1)/pi)*n));
  v=u*cgy

  return(y[v<=dnorm(y)]);
}
```

On réalise plusieurs tests sur la fonction rnormAR.

Tout d’abord on test le nombre d’éléments en sortie, afin de voir si il est cohérent au paramètre d’entrée n.

```
#On vérifie que l'on obtient bien environ n tirages
length(rnormAR(10000))
```

```
## [1] 9985
```

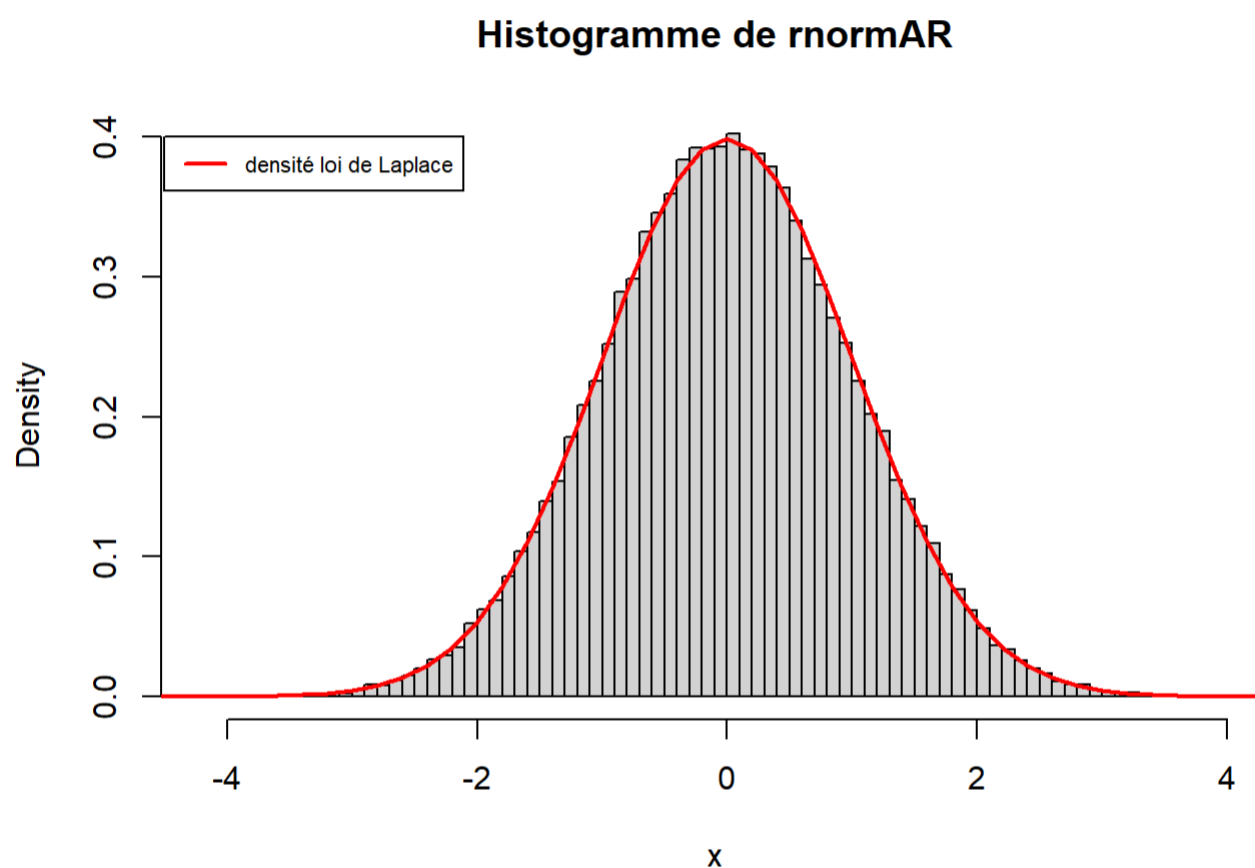
On remarque que le nombre de tirages réalisés par la fonction `rnormAR` est proche de n . Cela confirme le calcul d'espérance réalisé précédemment.

Ensuite, pour tester si les valeurs obtenues suivent bien une loi normale, on trace l'histogramme d'un n -échantillon, et on y superpose la fonction densité de la loi $N(0, 1)$ que l'on cherche à simuler.

```
#On trace l'histogramme d'un échantillon tirés selon notre fonction rnormAR
hist(rnormAR(100000),freq=F,main="Histogramme de rnormAR",nclass=100,xlab="x")

#On compare avec la fonction densité de la loi N(0,1)
curve(dnorm(x),from=-10,to=10,col="red",add=T,lwd=2)

legend(y=0.4,x=-4.5,legend="densité loi de Laplace",col="red",cex=0.7,lwd=2)
```



Le graphique est convainquant, l'histogramme correspondant relativement bien à la densité de la loi normale centrée réduite.

Pour confirmer que notre fonction `rnormAR` suit bien la loi normale, on réalise le test de Kolmogorov-Smirnov. Il permet de comparer les écarts entre un n -échantillon et une loi entrée en paramètre.

```
#Test de Kolmogorov-Smirnov
S=0
for (i in 1:1000){
  x=rnormAR(1000)
  test = ks.test(x,"pnorm")
  S=S+test$p.value
}
S/1000
```

```
## [1] 0.4961543
```

Dans le test de Kolmogorov-Smirnov, plus p -value est grande, plus les valeurs de l'échantillon sont des valeurs "raisonnables" de la loi normale. On réalise donc 1000 tests, qui nous donnent une valeur moyenne de p -value bien supérieure à 0,01. On en conclut que notre fonction `rnormAR` simule bien la loi $N(0, 1)$.

On s'intéresse maintenant à la rapidité d'exécution de la fonction `rnormAR`, comparée à `rnorm` implémentée par défaut dans R.

```
#Comparaison des vitesses d'exécutions
N = 10^(1:7)
temps_classique = rep(0,7)
temps_AR = rep(0,7)

for (k in 1:7){
  temps_classique[k]=round(system.time(rnorm(N[k]))[1],6)
  temps_AR[k]=round(system.time(rnormAR(N[k]))[1],6)
}
matrice = rbind(temps_classique,temps_AR)
colnames(matrice) = N
matrice
```

```
##           10 100 1000 10000 1e+05 1e+06 1e+07
## temps_classique  0   0   0     0  0.00  0.03  0.36
## temps_AR        0   0   0     0  0.03  0.31  2.86
```

On remarque ici que la fonction rnormAR est moins rapide que rnorm, d'autant plus lorsque le nombre de valeurs devient important.

Code R7

Le modèle mathématique d'un tirage au hasard d'un point dans le carré $([0, 1]^2)$ est donné par la probabilité sur le plan définie par :

$\forall S \subset \mathbb{R}^2, \quad \mathbb{P}\{(X, Y) \in S\} = \text{Aire}(S \cap [0, 1]^2) \cdot 1$

a)On a :

$$c \cdot g(x, y) = \frac{1}{\text{Aire}(S)} \cdot 1_{[0,1]^2}(x, y) \geq \frac{1}{\text{Aire}(S)} \cdot 1_S(x, y) \quad \text{puisque } S \subset [0, 1]^2$$

, d'où $c \cdot g(x, y) \geq f(x, y)$.

b. Nous acceptons un couple (x, y) tiré au hasard dans le carré $[0, 1]^2$, si : $Ucg(x, y) \leq f(x, y)$

C'est-à-dire :

$$U \cdot \frac{1}{\text{Aire}(S)} \cdot 1_{[0, 1]^2}(x, y) \leq \frac{1}{\text{Aire}(S)} \cdot 1 * S(x, y),$$
$$U * \cdot 1_{[0, 1]^2}(x, y) \leq 1_S(x, y).$$

Si $(x, y) \in S$, alors $(x, y) \in [0, 1]^2$ et l'inégalité devient $U \leq 1$, ce qui est toujours vrai puisque U suit une loi uniforme. Ainsi, un couple (x, y) est accepté si et seulement si $(x, y) \in S$.

2. On génère maintenant un tirage au hasard dans une surface

$$S := (x, y) \in \mathbb{R}^2 : s(x, y) \leq 0$$

(avec $s(\cdot, \cdot)$ donnée) incluse dans le carré $[0, 1]^2$. On choisit donc de simuler un cercle de centre $(0.5, 0.5)$ et de rayon 0.5.

```

# Définition du cercle
cercle <- function(x, y) {
  # Retourne TRUE si (x, y) est dans le disque de centre (1/2, 1/2) et rayon 1/2
  (x - 0.5)^2 + (y - 0.5)^2 <= 0.25
}

# Aire du disque
aireS <- pi / 4

#Fonction indicatrice
indicatriceS <- function(x, y) {
  as.numeric(cercle(x, y)) # 1 si (x, y) dans le disque S, 0 sinon
}

# Méthode d'acceptation-rejet
acceptation_rejet <- function(n) {
  c <- 1 / aireS
  res <- matrix(NA, n, 2)
  i <- 1

  plot(0, 0, xlim=c(0,1), ylim=c(0,1), main="Méthode d'Acceptation-Rejet", asp=1,ylab="y",xlab="x")
  symbols(0.5, 0.5, circles=0.5, inches=FALSE, add=TRUE, lwd=2)
  abline(v=c(0,1), h=c(0,1))

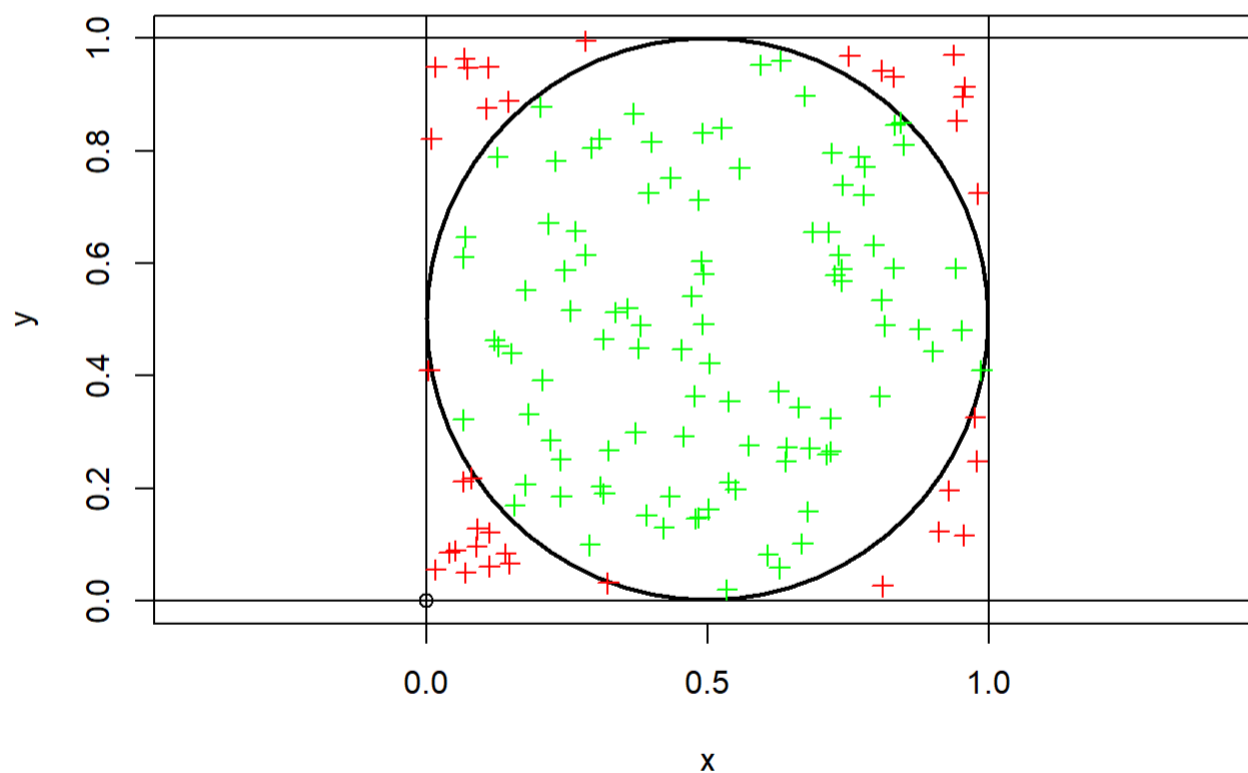
  while (i <= n) {
    x <- runif(1)
    y <- runif(1)
    z <- runif(1)

    if (z * c <= indicatriceS(x, y) / aireS) {
      res[i, ] <- c(x, y)
      points(x, y, col="green", pch=3)
      i <- i + 1
    } else {
      points(x, y, col="red", pch=3)
    }
  }
}

acceptation_rejet(100)

```

Méthode d'Acceptation-Rejet



Après modélisation, nous remarquons que la plupart des points se situent à l'intérieur de la surface du disque ce qui est cohérent car la surface du cercle représente une très grande partie de notre surface d'étude. Sur notre graphique, les points verts sont les points acceptés et les points rouges les non-acceptés. On remarque que la répartition des points est assez homogène donc bien aléatoire.

3. On prend maintenant un cercle de centre (0,0) et de rayon 1.

```
# Définition du disque unité
cercle <- function(x, y) {
  # Retourne TRUE si (x, y) est dans le disque de centre (0,0) et de rayon 1
  (x)^2 + (y)^2 <= 1
}

# Aire du disque
aireS <- pi

# Fonction g1 : densité uniforme sur le carré [-1,1]^2
g1 <- function(x, y) {
  as.numeric(x >= -1 & x <= 1 & y >= -1 & y <= 1)
}

# Fonction indicatrice du disque
indicatriceS <- function(x, y) {
  as.numeric(cercle(x, y)) # 1 si (x, y) dans S, 0 sinon
}

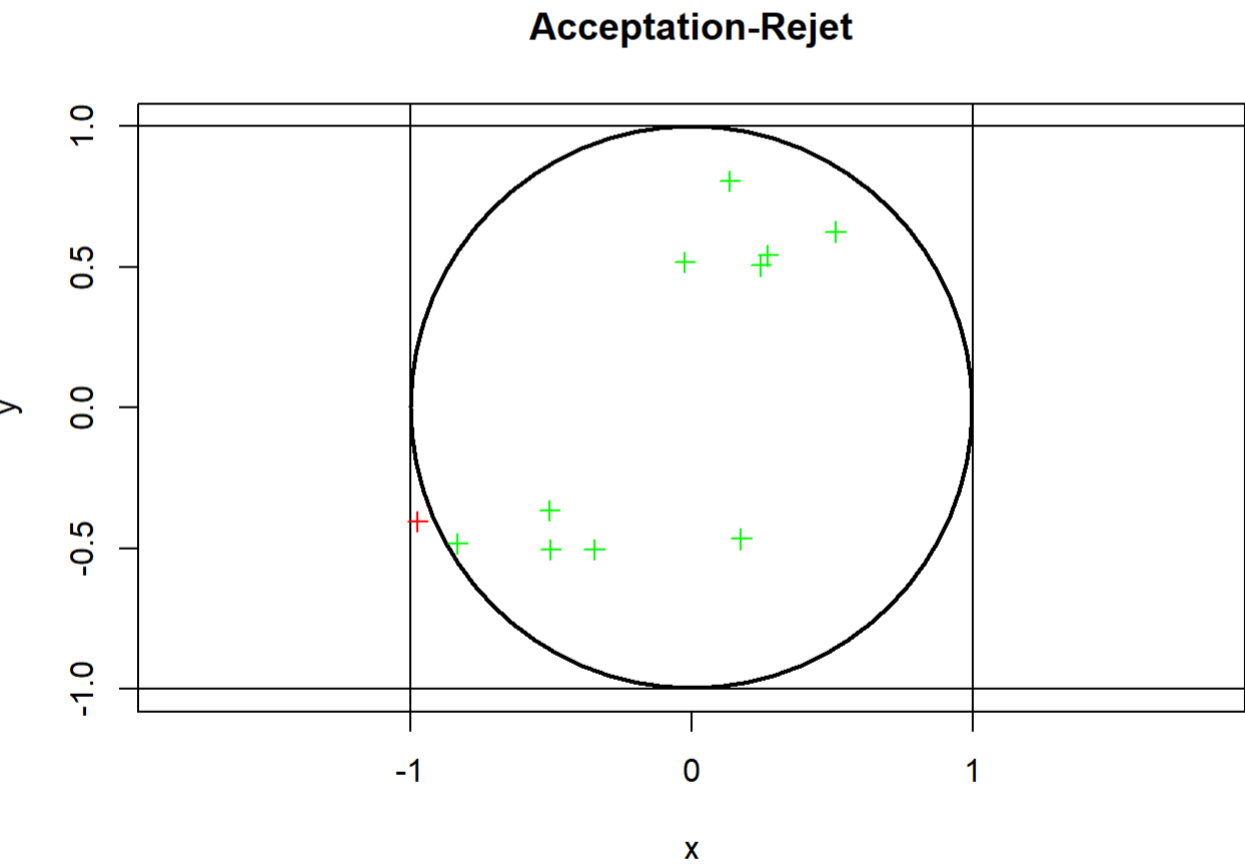
# Méthode d'acceptation-rejet
acceptation_rejet <- function(n) {
  c <- 1 / aireS
  res <- matrix(NA, n, 2)
  i <- 1

  plot(0, 0, xlim=c(-1,1), ylim=c(-1,1), pch=NA, main="Acceptation-Rejet", asp=1, ylab="y", xlab="x")
  abline(v=c(-1,1), h=c(-1,1))
  symbols(0, 0, circles=1, inches=FALSE, add=TRUE, lwd=2)

  while (i <= n) {
    x <- runif(1, -1, 1)
    y <- runif(1, -1, 1)
    z <- runif(1)

    if (z * c * g1(x, y) <= indicatriceS(x, y) / aireS) {
      res[i, ] <- c(x, y)
      points(x, y, col="green", pch=3)
      i <- i + 1
    } else {
      points(x, y, col="red", pch=3)
    }
  }
}

acceptation_rejet(10)
```



Ici ,nous avons seulement changé le domaine d’étude et nous en venons à la même conclusion que précédemment; les points sont répartis homogénéiquement(ils sont bien aléatoires).

Conclusion

Durant ce TP, nous avons pu nous initier à la simulation de lois de probabilité et à l'utilisation des méthodes de Monte Carlo. Le TP a débuté par des définitions et des remarques préliminaires sur la simulation de la loi uniforme sur $[0,1]$ et sur la fonction quantile, ce qui nous a permis de simuler une loi de Laplace à l'aide de la **méthode de l'inverse** (code R5). Cette méthode repose sur le lemme 3.2 et établit une bijection entre une variable uniforme U et la variable aléatoire X : elle permet ainsi de générer non seulement des réalisations de la loi de Laplace, mais également de la loi exponentielle à partir d'une loi uniforme.

Nous avons ensuite étudié la méthode de l'**acceptation-rejet** (AR) à travers le lemme 3.3 et le commentaire 3.2. Cette méthode consiste à majorer la densité cible à une constante près, puis à appliquer un algorithme pour accepter ou rejeter des candidats. Elle nous a permis de simuler la loi normale centrée réduite à partir d'une loi de Laplace (code R6). Des tests ont été réalisés afin de vérifier que les échantillons simulés suivaient bien la loi souhaitée.

Enfin, nous avons appliqué la méthode AR pour générer un couple de variables aléatoires uniformes dans un pavé de \mathbb{R}^2 et simuler des points dans un domaine donné (code R7).

Ce TP nous a permis de comprendre la puissance des méthodes de simulation probabiliste : elles permettent de générer des échantillons suivant des lois complexes, d'estimer des probabilités ou des quantiles et de modéliser des phénomènes aléatoires lorsque les calculs analytiques sont difficiles.