Soufyan Bahera
Lorrain Barrabé
Corentin Chautard
Enzo Francois
Théo Paesa
Justine Ribas

Probabilités et Simulation
ING1 GI

TP - Simulation variables aléatoires réelles

Sommaire :

Exercice 1 : Simulation de variables aléatoires réelles

Loi de bernoulli de paramètre p

Code source :

```
#Q1
library(methods)
bernoulli <- setRefClass("Loi de Bernoulli", fields = list(p =
"numeric"), methods = list(
  esp = function() #Donne l'esperance
  {
    return(p)
  },
  var = function() #Donne la variance
  {
    return(p*(1-p))

  },
  simulation = function(n) #Génère une liste de taille n de 0 et 1 en
fonction de la probabilité p donné
  {
    x=sample(c(0,1),n,replace=T,prob=c(1-p,p))
    return(x)
  },
  simu_proba = function(n) #Retrouve une approximation de P a partir
d'un vecteur généré avec la fonction simulation
  {
    print(table(simulation(n))/n)
  }
))
ber = bernoulli(p=0.5)
ber$simulation(100)
```

Résultat :

On obtient aléatoirement des valeurs comprises entre 0 et 1, dans la console

```
> ber$simulation(100)
  [1] 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 1 0 1 0 0 0 1 1 0 1 1 1 0 0
 [68] 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 1 1 1 0 0 1 1 0
```

Loi binomiale de paramètre (n, p)

Code source :

```
#Q2
library(methods)
binomiale <- setRefClass("Loi binomiale", fields = list(n="numeric", p =
"numeric"), methods = list(
  esp = function() #Donne l'esperance
  {
    return(n*p)
  },
  var = function() #Donne la variance
  {
    return(n*p*(1-p))

  },
  simulation = function(r) #Génère un vecteur de r entier entre 0 et n
en fonction des probabilités
  {
    probabilites = c(0:n) #on fait une liste pour les probabilités de
chaque valeur
    for(k in 0:n)
    {
      probabilites[k+1] =
(factorial(n)/(factorial(k)*factorial(n-k)))*(p^k*((1-p)^(n-k))) #on
calcule ces probabilités
    }
    x=sample(seq(from =0, to=n, by=1),r,replace=T,prob=probabilites)#on
crée notre vecteur
    return(x)
  }
))

binom = binomiale(n=10,p=0.45)
binom$simulation(100)
```

Résultat :

On obtient aléatoirement des valeurs comprises entre 0 et n (n=10 dans notre cas), dans la console

```
> binom$simulation(100)
  [1] 3 3 4 5 6 6 4 5 5 2 6 5 6 7 5 5 2 4 3 6 2 4 5 4 4 7 6 5 5 4 5 6 6 2 7 6 3 5 7 4 3 5 4 6 2 3 3 6 5 4 4 2 4 5 6 5 6 3 6 6 5 6 5 4 4 5 4
 [68] 4 7 4 3 4 4 4 4 4 5 5 1 4 7 5 5 7 6 6 5 3 6 7 4 5 4 6 3 5 3 6 5 6
```

Loi géométrique de paramètre p

Code source :

```
#Q3
library(methods)
geometrique <- setRefClass("Loi geometrique", fields = list(p =
"numeric"), methods = list(
  esp = function() #Donne l'espérance
  {
    return(1/p)
  },
  var = function() #Donne la variance
  {
    return((1-p)/(p*p))

  },
  simulation = function(r) #on simule une loi géométrique avec une loi
de bernoulli
  {
    j=1
    c=1
    result <- integer(r)
    bernoulli_approach <- bernoulli(p=p)
    bernoulli_list = bernoulli_approach$simulation(r)
    for(i in 0:r)
    {
      c=0
      while(bernoulli_list[j]==0&&j<r)#On compte le nombre d'échec avant
un succès
      {
        c=c+1
        j=j+1
      }
      result[i] = c
      c=1
      j=1
      bernoulli_list = bernoulli_approach$simulation(r)
    }
    return(result)
  },
  simu_proba = function(r)
  {
    print(table(simulation(r))/r)
  }
```

```
))
geom = geometrique(p=0.3)
geom$simulation(100)
```

Résultat :

On obtient les valeurs suivantes dans la console.

```
> geom$simulation(100)
  [1]  0  0  1  0  0  0 11  9  0  0  2  0  1  2  0  0 12  0  0  5  2  1  0  0  3  6  2  0  3  9  5  2  0  0  0  0  2  3  1  1  4  7  0  1  3
 [46]  0  1  3  4  5  3  0  1  0  3  2  2  0  4  1  1  0  0  0  3  0  0  0  2  0  3 11  6  4  4  0  0  0  1  8  0  0  4  9  0  1  3  2  0  0
 [91]  0  0  0  6  7  6  0  1  4  0
```

Loi uniforme sur [1;k]

Code source :

```
#Q4
library(methods)
uniformeSegment <- setRefClass("Loi uniforme sur [1,k]", fields = list(k
= "numeric"), methods = list(
  esp = function() #Donne l'espérance
  {
    if(k<2)
    {
      return("Erreur: il faut k >= 2")
    }
    return((1+k)/2)
  },
  var = function() #Donne la variance
  {
    if(k<2)
    {
      return("Erreur: il faut k >= 2")
    }
    return((k*k)/12)

  },
  simulation = function(n)
  {
    if(k<2)
    {
      return("Erreur: il faut k >= 2")
    }
    x=round(runif(50,1,50))
    return(x)
  }
))
uniSegment = uniformeSegment(k=5)
uniSegment$simulation(1000)
```

Résultat :

On obtient les valeurs suivantes dans la console.

```
> uniSegment$simulation(1000)
 [1] 48 44 10 24 19  2 30 13 49 37 22  2  5 22 43 35  8 36  3 27 49 13 47  8  7 43 24 45  9 21 38 24  6 44 42 25 12 24 37 24 37 17 19 17 21
[46] 41 10 22 47 25
```

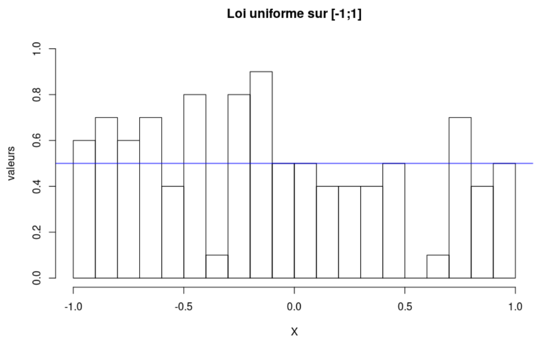
Loi uniforme sur [-1;1]

Code source :

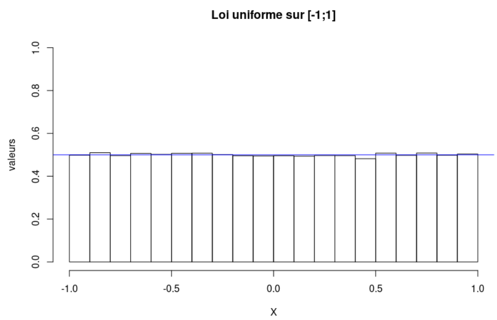
```
#Q5
library(methods)
uniforme_11 <- setRefClass("Loi uniforme sur [-1,1]", methods = list(
  esp = function() #Donne l'espérance
  {
    return((-1+1)/2)
  },
  var = function() #Donne la variance
  {
    return(1/12)

  },
  graph = function(n) #on affiche sur un même graph la densité et
l'histogramme de la loi
  {
    x = runif(n,-1,1)
    hist(x, main="Loi uniforme sur [-1;1]", breaks=20, xlab = "X", ylab
= "valeurs", probability = TRUE, xlim = c(-1,1), ylim = c(0,1))
    abline(h = 1/2, col="blue")
  }
))
uni = uniforme_11()
uni$graph(1000)
```

Résultat :

Pour n = 100



Pour n =100000



Observation :

Plus n est grand et plus on se rapproche de la densité

Exercice 2 : Simulation de variables aléatoires réelles par inversion

Loi de Bernoulli de paramètres p

Code source :

```
#Question 1
# Loi de Bernoulli X~B(p)
bernoulliInversion <- function(p){
  u = runif(1) # loi uniforme entre 0 et 1
  # fonction inverse de la fonction de répartition de Bernoulli
  if(u<1-p){
    binom = 0
  } else {
    binom = 1
  }
  binom
}

p = 0.5
bernoulliInversion(p)
```

Résultat :

On obtient la valeur de 1 ou 0 aléatoirement dans la console.

```
> p = 0.5
> bernoulliInversion(p)
[1] 0
```

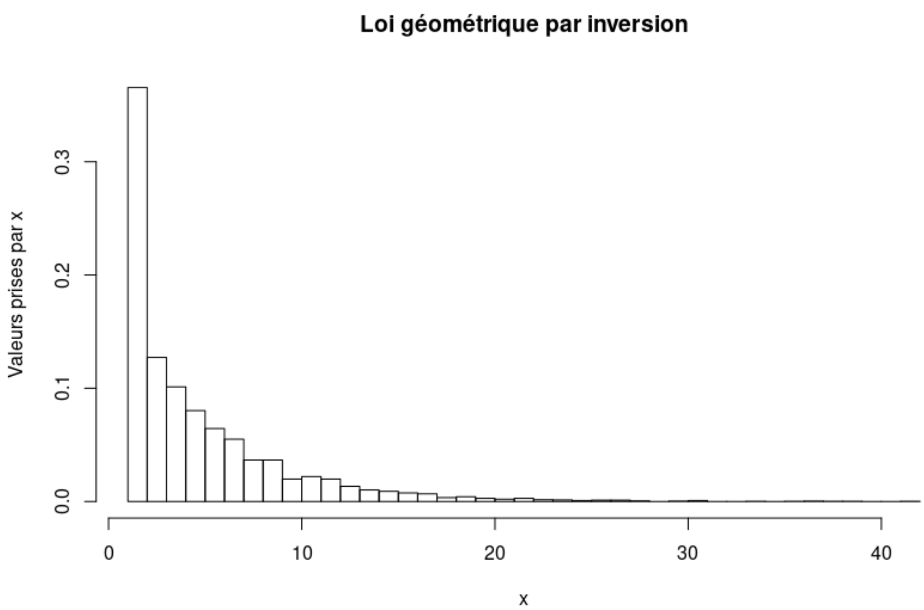
```
> p = 0.5
> bernoulliInversion(p)
[1] 1
```

Loi Géométrique de paramètre p

Code source :

```
#Question 2
# Loi Géométrique X~G(lambda)
geometriqueInversion <- function(n, lambda){
  i = 0
  geom = array()
  while (i < n) {
    u = runif(1)
    # Fonction inverse de la fonction de répartition d'une loi
géométrique
    geom[i] = floor(log(1-u)/log(1-lambda))+1
    i = i+1
  }
  hist(geom, breaks=50, probability = TRUE, main="Loi géométrique par
inversion", xlab="x", ylab="Valeurs prises par x", freq=F)
}

n = 5000
lambda = 0.2
geometriqueInversion(n, lambda)
```

Résultat :



Loi de poisson de paramètre λ

Code source :

```
# Question 3
# Loi de Poisson X~P(lambda)
poissonInversion <- function(n, lambda){
  i = 0
  poiss = array()
  while (i < n) {
    # Fonction inverse de la fonction de répartition de la loi de
Poisson
    p = 1
    x = 0
    while(p>=exp(-lambda)){
      u = runif(1)
      p = p*u
      x = x+1
    }
    poiss[i] = x
    i = i+1
  }
  poiss
}

n = 200
lambda = 0.5
poissonInversion(n, lambda)
```

Résultat :

On obtient les valeurs suivantes dans la console.

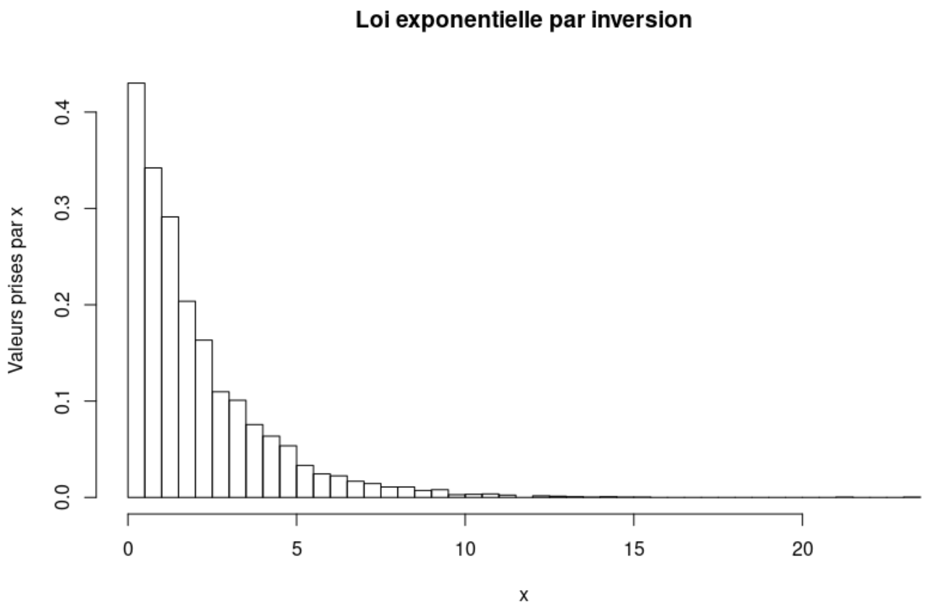
```
> n = 100
> lambda = 0.5
> poissonInversion(n, lambda)
 [1] 1 1 1 1 1 1 1 1 1 1 2 1 1 2 2 1 1 1 1 2 2 3 1 3 1 1 1 1 1 1 1 1 2 2 2 2 1 1 2 1 3 1 1 2 1 2 1 1 1 1 1 3 1 1 1 2 1 1 1 3 1 1 2 2 1 1 1 1
[69] 1 1 1 2 1 1 2 3 1 1 1 3 1 1 1 1 1 1 1 2 2 1 2 1 1 1 1 1 1 1 2
```

Loi Exponentielle de paramètre λ

Code source :

```
#Question 4
# Loi Exponentielle X~E(lambda)

exponentielleInversion <- function(n, lambda){
  i = 0
  expo = array()
  while (i < n) {
    u = runif(1)
    # Fonction inverse de la fonction de répartition de la loi
exponantielle
    expo[i] = -log(1-u)/lambda
    i = i+1
  }
  hist(expo, breaks=50, probability = TRUE, main="Loi exponentielle par
inversion", xlab="x", ylab="Valeurs prises par x", freq=F)
}

n = 5000
lambda = 0.5
exponentielleInversion(n, lambda)
```

Résultat :



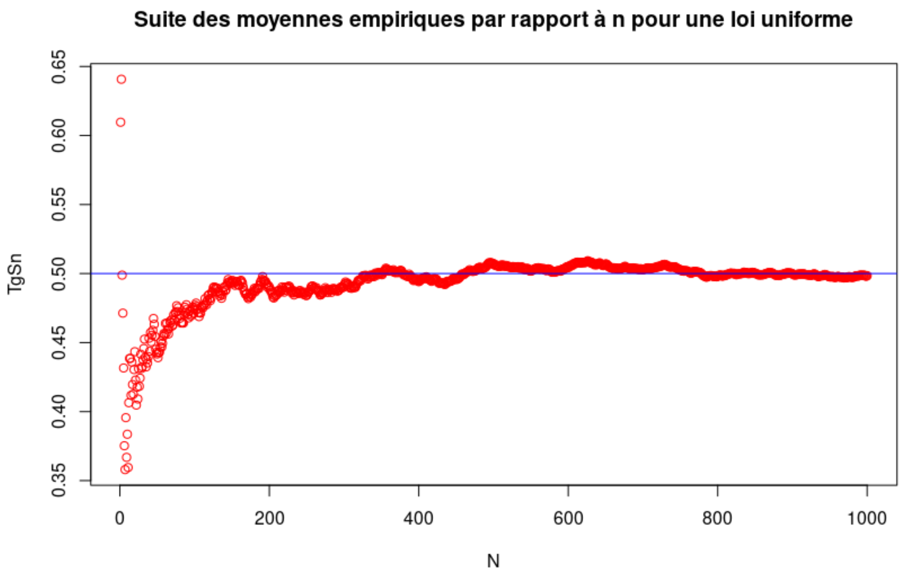
Exercice 3 : Lois des grands nombres

Loi uniforme sur [0;1]

Code source :

```
##1

n = 1000 #Nombre d'essaie
esp = 1/2 #Espérance des Xk
Xk= runif(n,min=0,max=1)
Sn= cumsum(Xk) #Calcule de Sn
N = seq(1,n, by=1)
TgSn = Sn/N

plot(N,TgSn, col="red", main = "Suite des moyennes empiriques par
rapport à n pour une loi uniforme")
abline(h=esp, col = "blue") #Tracer l'espérance sur le graphe
```

Résultat :



Observation :

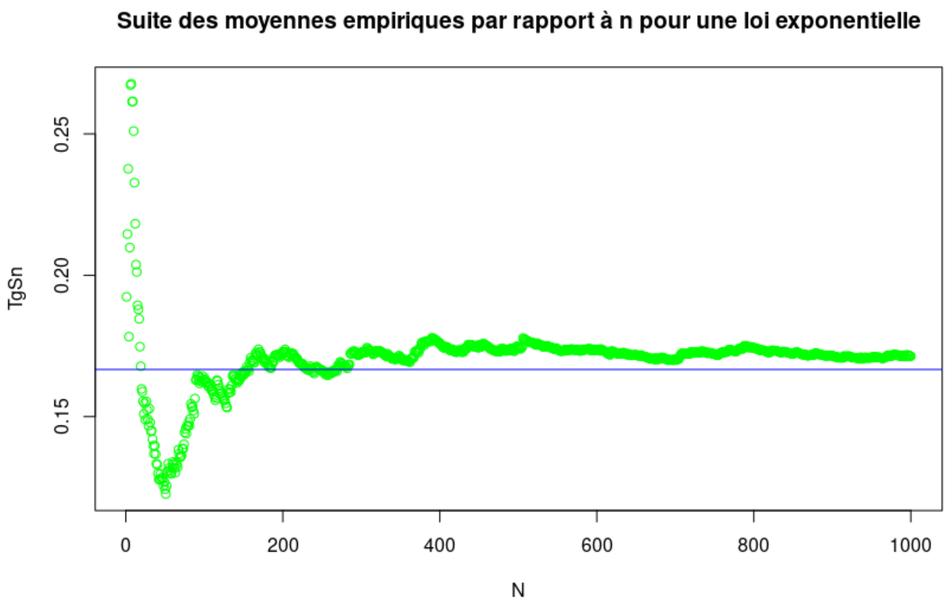
La suite des moyennes empiriques semble converger vers l'espérance des X_k

Loi exponentielle

Code source :

```
##2

n = 1000 #Nombre d'essaie
lambda = 6 #Parametre pour la loi exponentielle
esp = 1/lambda #Espérance des Xk
Xk = rexp(n,lambda)
Sn = cumsum(Xk) #Calcule de Sn
N = seq(1,n, by=1)
TgSn=Sn/N

plot(N,TgSn, col="green", main = "Suite des moyennes empiriques par
rapport à n pour une loi exponentielle")
abline(h=esp, col="blue") #Tracer l'espérance sur le graphe
```

Résultat :



Observation :

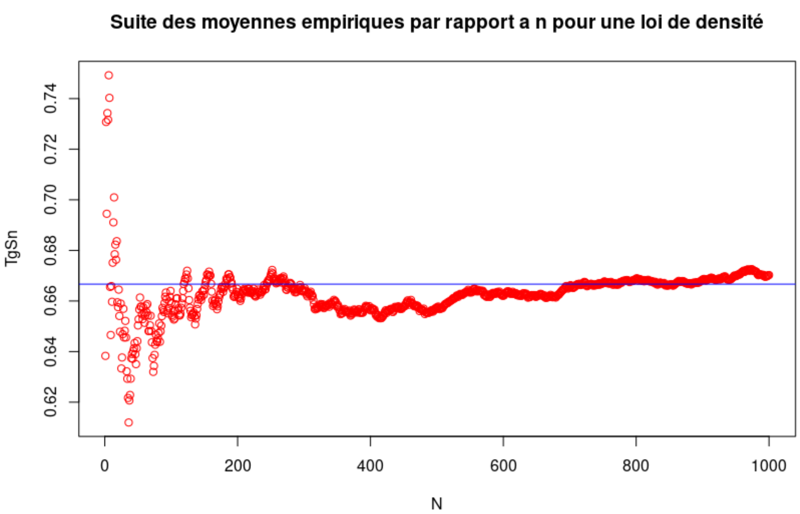
La suite des moyennes empiriques semble converger vers l'espérance des X_k

Variables aléatoires à densité f

Code source :

```
##3

n = 1000
f = function(x){ #fonction pour calculer l'espérance
  2*x*x
}
Xk = c()
i = 0

for (i in 1:n) {
  Xk[i] = sqrt(runif(1))
}

esp = integrate(f, lower = 0, upper = 1)
Sn = cumsum(Xk) #Calcule de Sn
N = seq(1,n, by=1)
TgSn = Sn/N
plot(N, TgSn, col="red", main= "Suite des moyennes empiriques par
rapport a n pour une loi de densité")
abline(h = esp$value, col="blue") #Tracer de l'espérance sur le graphe
```

Résultat :



Observation :

La suite des moyennes empiriques semble converger vers l'espérance

Exercice 4 : Théorème Central Limite

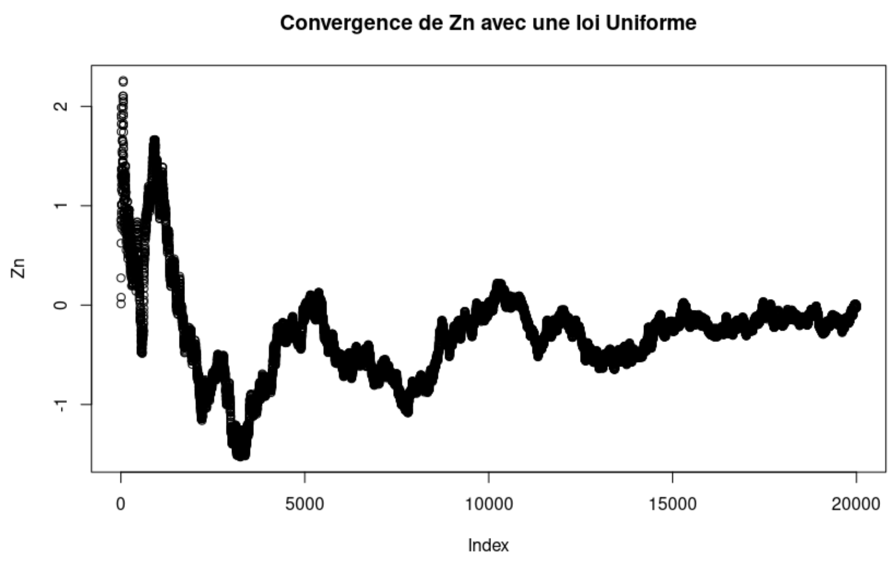
Convergence de Zn avec Xn qui suit une loi uniforme sur [0;1]

Convergence

Code source :

```
#Question 1.a

x1 = c()
n = 20000

##Calcul de la loi
x = runif(n, min = 0, max = 1)
## esperance de la série
esp = mean(x)
## variance de la série
sigmacarre = mean(x^2) - esp^2
##Def de Sn
Sn = cumsum(x)
N = seq(1, n, by = 1)
##Def de Zn
Zn = (sqrt(N) / sqrt(sigmacarre)) * ((Sn / N) - esp)

plot(Zn,main = "Convergence de Zn avec une loi Uniforme")
```

Résultat :

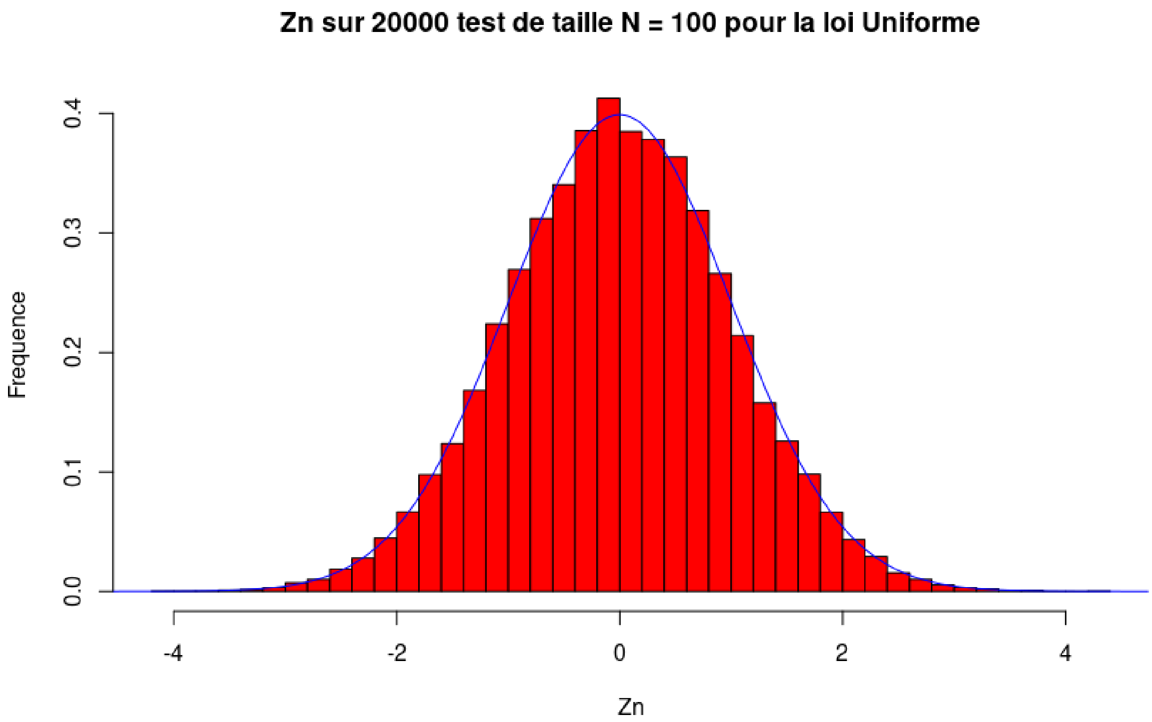


Convergence en loi

Code source :

```
#Question 1.b

## f(x) correspond a une loi normale de paramètres N(µ,ϕ²)
x1 = c()

##esperance
esp = 1 / 2
##variance
sigmacarre = 1 / 12
n = 20000
tries = 100
#calcul loi uniforme
fsimu1 = function(echantillon) {
  for (i in 1:echantillon) {
    w = runif(1)
    x1[i] = qunif(w)
  }
  x1
}
# on a effectué la simulation de la v.a.r. de densité f(x) sur n
SumZn = c()
for (j in 1:n) {
  Sn = fsimu1(tries)
  tmpZn = (sqrt(tries) / sqrt(sigmacarre)) * ((sum(Sn) / tries) - esp)
  SumZn[j] = tmpZn
}

hist(SumZn,ylab = "Frequence", xlab = "Zn",main = "Zn sur 20000 test de
taille N = 100 pour la loi Uniforme", breaks = 30,col =
"red",probability = TRUE)

loiNormale <- seq(-5, 5, by = 0.01) # tracé de la loi normale
y <- dnorm(loiNormale)
lines(loiNormale, y, col = "blue")
```

Résultat :



Observation :

On remarque que Zn converge en loi vers variable qui suit une loi Normale de paramètre (0;1). Ce qui correspond bien au théorème de la limite centrale.

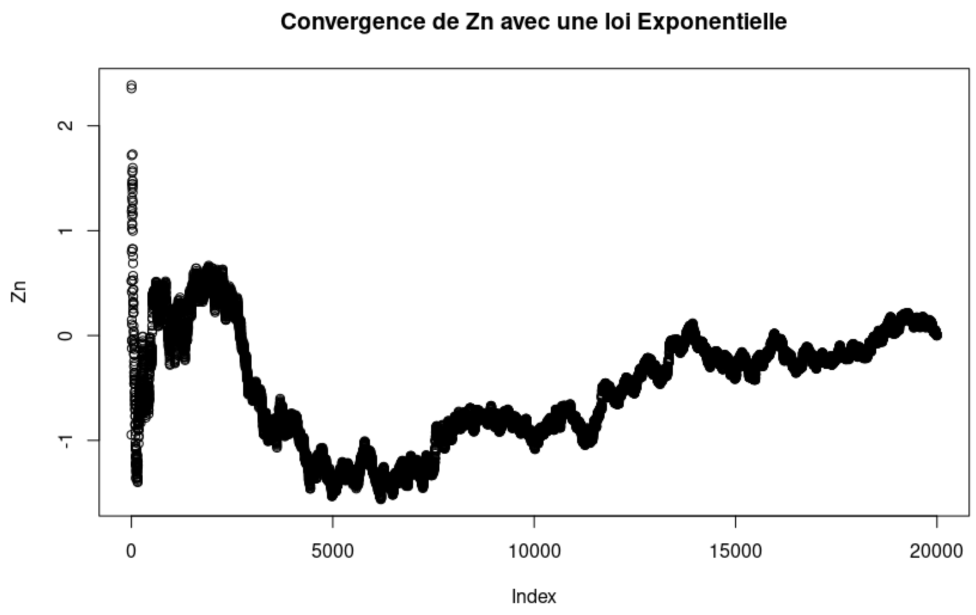
Convergence de Zn avec Xn qui suit une loi exponentielle

Convergence

Code source :

```
# Question 2.a

x1 = c()
n = 20000

##Calcul de la loi
x = rexp(n)
##Esperance de la série
esp = mean(x)
##Variance
sigmacarre = (1 / (1/esp)^2)
Sn = cumsum(x)
N = seq(1, n, by = 1)
Zn = (sqrt(N) / sqrt(sigmacarre)) * ((Sn / N) - esp)


plot(Zn,main = "Convergence de Zn avec une loi Exponentielle")
```

Résultat :

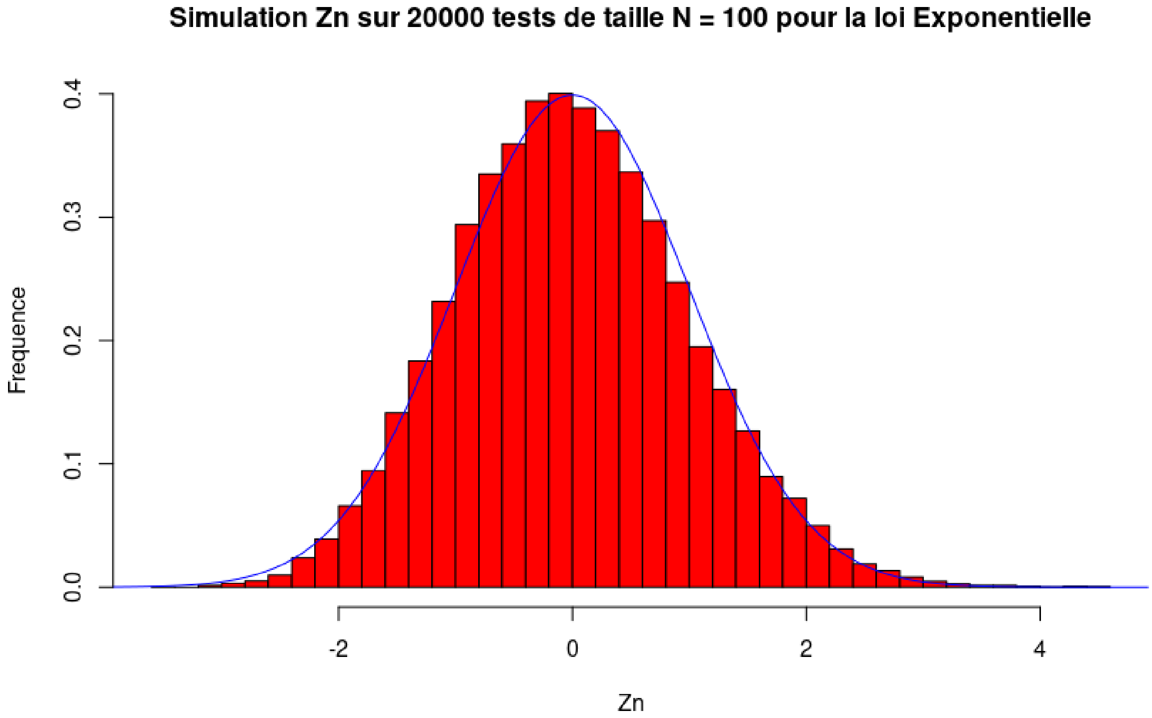


Convergence en loi

Code source :

```
# Question 2.b

x2 = c()

lambda = 0.5
##Variance
sigmacarre = 1 / (lambda ^ 2)
##Esperance
esp = 1 / lambda

tries = 100
#calcul loi expo
fsimu2 = function(echantillon) {
  for (i in 1:echantillon) {
    w = runif(1)
    x2[i] = qexp(w,lambda)
  }
  x2
}
SumZn = c()
for (j in 1:n) {
  Sn = fsimu2(tries)
  tmpZn = (sqrt(tries) / sqrt(sigmacarre)) * ((sum(Sn) / tries) - esp)
  SumZn[j] = tmpZn
}

hist(SumZn,ylab = "Frequence", xlab = "Zn",main = "Simulation Zn sur
20000 tests de taille N = 100 pour la loi Exponentielle", breaks =
30,col = "red",probability = TRUE)

# tracé de la loi normale
loiNormale <- seq(-5, 5, by = 0.01)
y <- dnorm(loiNormale)
lines(loiNormale, y, col = "blue")
```

Résultat :



Observation :

On remarque que Zn converge en loi vers variable qui suit une loi Normale de paramètre (0;1). Ce qui correspond bien au théorème de la limite centrale.

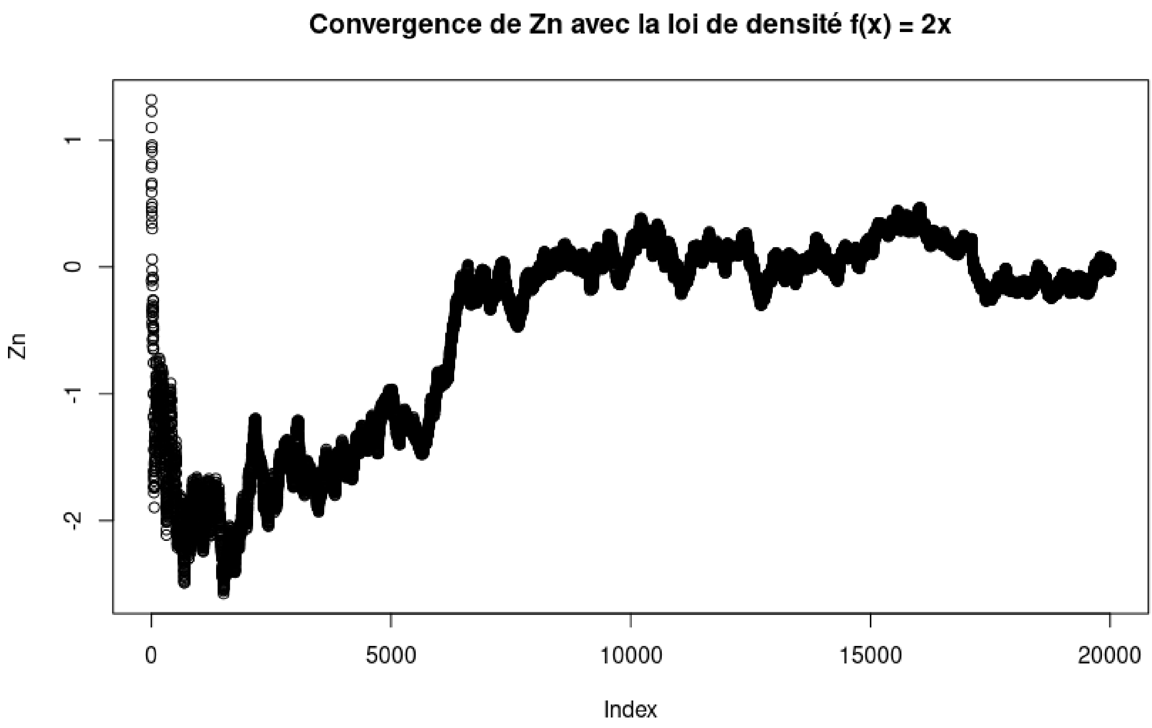
Convergence de Zn avec Xn qui a f comme fonction de densité

Convergence

Code source :

```
#Question 3.a

n = 20000
x = c()
for (i in 1:n) { # calcul de la loi
  w = runif(1)
  x[i] = sqrt(w)
}
esp = mean(x) # espérance de la série
sigmacarre = mean(x^2) - (esp^2) # variance de la série
Sn = cumsum(x)
N = seq(1, n, by = 1)
Zn = (sqrt(N) / sqrt(sigmacarre)) * ((Sn / N) - esp)

plot(Zn,main = "Convergence de Zn avec la loi de densité f(x) = 2x")
```

Résultat :

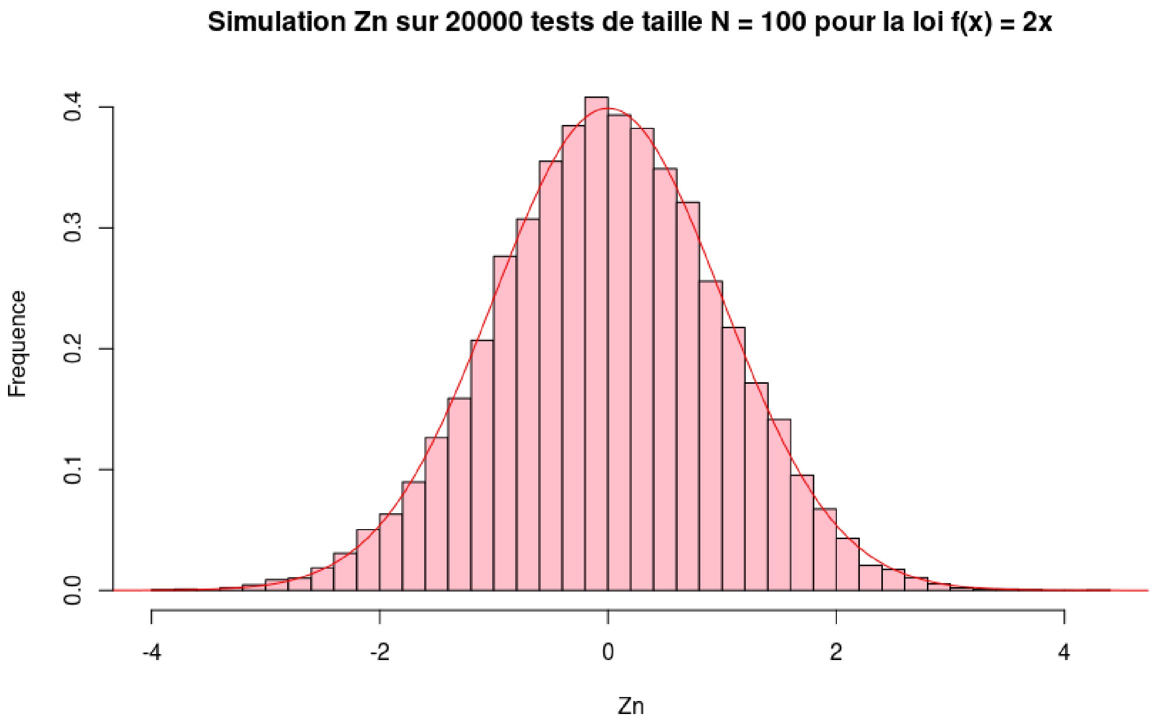


Convergence en loi

Code source :

```
#Question 3.b

x3 = c()

tries = 100
sigmacarre = 1 / 18 # variance
esp = 2 / 3 # espérance
fsimu3 = function(echantillon) {
  # on a f(x) = 2x donc F(x) = x² et F⁻¹(x) = sqrt(x)
  for (i in 1:echantillon) {
    w = runif(1)
    x3[i] = sqrt(w)
  }
  x3
}
# on a donc effectué la simulation de la v.a.r. de densité f(x) sur n
SumZn = c()
for (j in 1:n) {
  Sn = fsimu3(tries)
  tmpZn = (sqrt(tries) / sqrt(sigmacarre)) * ((sum(Sn) / tries) - esp)
  SumZn[j] = tmpZn
}

hist(SumZn,ylab = "Frequence", xlab = "Zn",main = "Simulation Zn sur
20000 tests de taille N = 100 pour la loi f(x) = 2x", breaks = 30, col =
"pink", probability = TRUE)

loiNormale <- seq(-5, 5, by = 0.01) # tracé de la fonction gaussienne
standard
y <- dnorm(loiNormale)
lines(loiNormale, y, col = "red")
```

Résultat :



Observation :

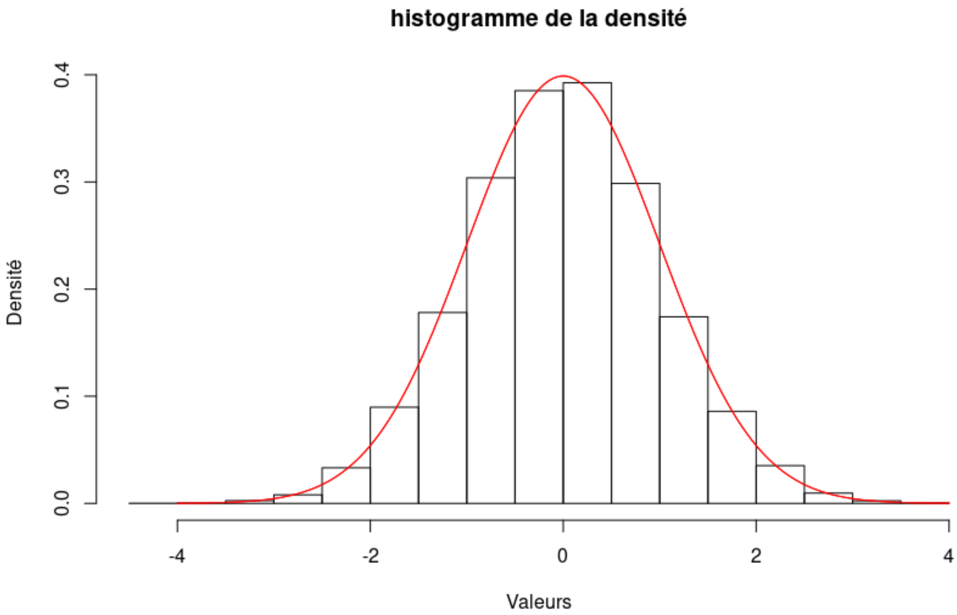
On remarque que Zn converge en loi vers variable qui suit une loi Normale de paramètre (0;1). Ce qui correspond bien au théorème de la limite centrale.

Exercice 5 : Méthode de Box-Muller

Code source :

```
n=5000
Xn= c() # Suite de VAR Xn
Yn= c() #Suite de VAR Yn

for (i in 1:n) {
  U=runif(1) #Variable aléatoire U
  R= sqrt(-2*log(U)) #valeur de R
  V=runif(1) #Variable aléatoire V
  teta=2*pi*V
  X= R*cos(teta)
  Y=R*sin(teta)

  Xn[i]=X
  Yn[i]=Y
}

#loi normale
normale=rnorm(n)
sortie=c(Xn, Yn, normale)
traceLoiNormale <- seq(-4,4,by=0.05)
y<-dnorm(traceLoiNormale, mean=0, sd=1)

#representation histogramme
hist(sortie, main="histogramme de la densité", ylab= "Densité", xlab=
"Valeurs", probability =  TRUE)
#representation fonction gaussienne
lines(traceLoiNormale, y, col="red")
```

Résultat :



Observation :

On remarque également qu'il y a convergence vers la loi Normale de paramètres (0;1)