TPSTA3_CHEN_Zeyu_GROUP_2.1

Zeyu

2018/4/2

1.Ajuster une loi Bernoulli

<–1.a–>

```
   rb <- rbinom(10,1,0.7)
```

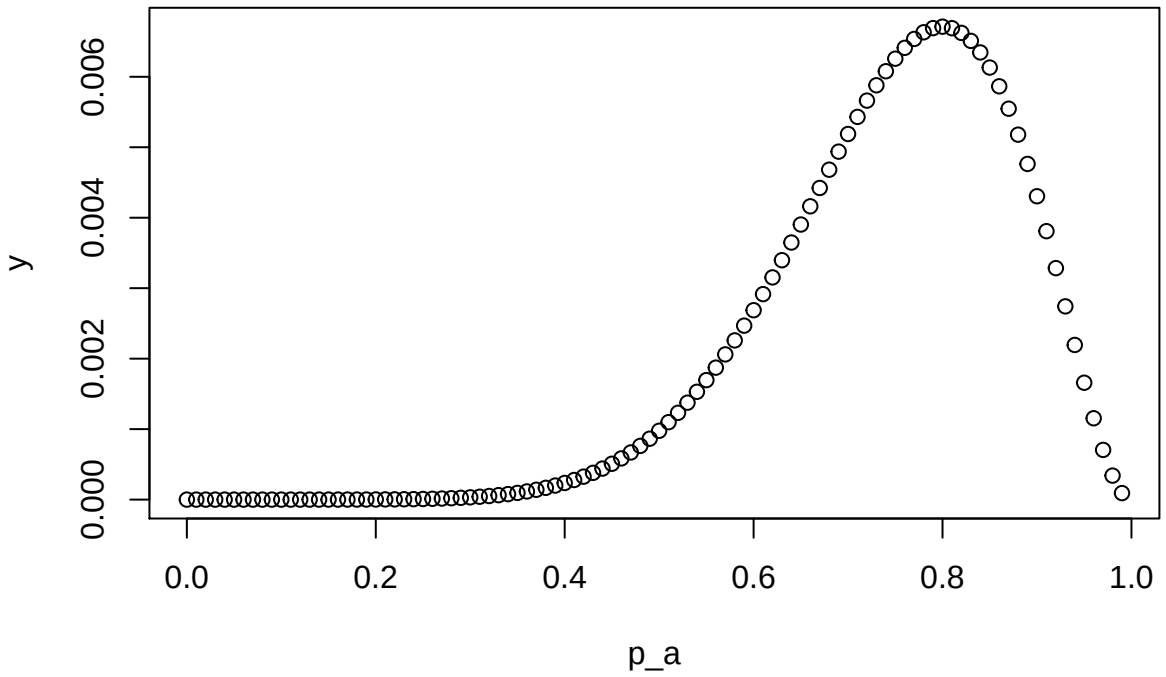
Une façon simple pour estimer p c'est faire la somme de rb et divisé par size.

<–1.b–>

```
#x_n est un échantillon
L_bern <- function(p,x_n)
{
  L <- (p**(sum(x_n)))*((1-p)**(sum(1-x_n)))
}
```

<–1.c–>

```
#x_n est un échantillon
p_a = seq(0, by = 0.01, length = 100)
rb <- rbinom(10,1,0.7)
y = seq(0,length=100)
for(i in 1:100)
{
  y[i] <- L_bern(p_a[i],x_n = rb)
}
plot(p_a,y)
```



###

On remarque que la vraisemblance est une fonction quadratique, admettant un maximum

<–1.d–>

```
#x_n est un échantillon
optimise(L_bern,interval = c(0,1),maximum = TRUE,x_n = rb)
```

```
## $maximum
## [1] 0.799982
##
## $objective
## [1] 0.006710886
```

<–1.e–>

```
p_a = seq(0, by = 0.01, length = 100)
#N est la taille d'échantillon
max_p <- function (N)
{
  rb <- rbinom(N,1,0.7)
  max <- p_a[0]
  for(i in 1:99)
  {
        if( (L_bern(p_a[i+1],x_n=rb)) > (L_bern(p_a[i],x_n=rb)) )
        max <- p_a[i+1]
  }
  return(max)
}

#N est la taille d'échantillon
compare <- function (N)
```

```
{
   rb <- rbinom(N,1,0.7)
   a <-max_p (N)
   b <-as.double(optimise(L_bern,interval = c(0,1),maximum = TRUE,x_n=rb))

   list("N"=N,"max theorique"=b[1],"max obtenu"=a,"l'ecart"=b[1]-a)
}
compare(10)
```

```
## $N
## [1] 10
##
## $`max theorique`
## [1] 0.799982
##
## $`max obtenu`
## [1] 0.8
##
## $`l'ecart`
## [1] -1.803363e-05
```

```
compare(50)
```

```
## $N
## [1] 50
##
## $`max theorique`
## [1] 0.7600075
##
## $`max obtenu`
## [1] 0.76
##
## $`l'ecart`
## [1] 7.471001e-06
```

```
compare(100)
```

```
## $N
## [1] 100
##
## $`max theorique`
## [1] 0.6500013
##
## $`max obtenu`
## [1] 0.65
##
## $`l'ecart`
## [1] 1.283531e-06
```

```
compare(200)
```

```
## $N
## [1] 200
##
## $`max theorique`
## [1] 0.6550009
```

```
##
## $`max obtenu`
## [1] 0.68
##
## $`l'ecart`
## [1] -0.0249991
```

```
compare(500)
```

```
## $N
## [1] 500
##
## $`max theorique`
## [1] 0.6820099
##
## $`max obtenu`
## [1] 0.7
##
## $`l'ecart`
## [1] -0.01799009
```

```
compare(1000)
```

```
## $N
## [1] 1000
##
## $`max theorique`
## [1] 0.6750112
##
## $`max obtenu`
## [1] 0.68
##
## $`l'ecart`
## [1] -0.004988777
```

```
compare(2000)
```

```
## $N
## [1] 2000
##
## $`max theorique`
## [1] 0.9999339
##
## $`max obtenu`
## numeric(0)
##
## $`l'ecart`
## numeric(0)
```

on remarque que quand N=2000, la valeur obtenu est numeric(0)

On peut combattre l'instabilité des calculs en passons au log-vraissemblance.

Le produit devient alors une somme, et il n'y a plus d'instabilités dû au produit de probabilités.

<–1.f–>

```
x <- read.csv("distribution_inconue_2_100_realisations.csv",header = TRUE)
Log_bern <- function(p,x_n)
{
  L <- sum(x_n)*log(p)+sum(1-x_n)*log(1-p)
}
optimise(Log_bern,interval = c(0,1),maximum = TRUE,x_n=x[2])
```

```
## $maximum
## [1] 0.5800008
##
## $objective
## [1] -68.0292
```

Parce que l'enchantillon ne contient que 2 types de valeurs distinctes: 0 ou 1.

2.Ajuster d'une loi normale d'écart type connu

<–2.a–>

```
r_norm <- rnorm(n=300,mean = 2,sd=1)
L_norm <- function(mu,x_n) {
  return (prod(dnorm(x_n, mu, 1)))
}
L_norm( 1,r_norm)
```

```
## [1] 1.596199e-236
```

```
L_norm( 2,r_norm)
```

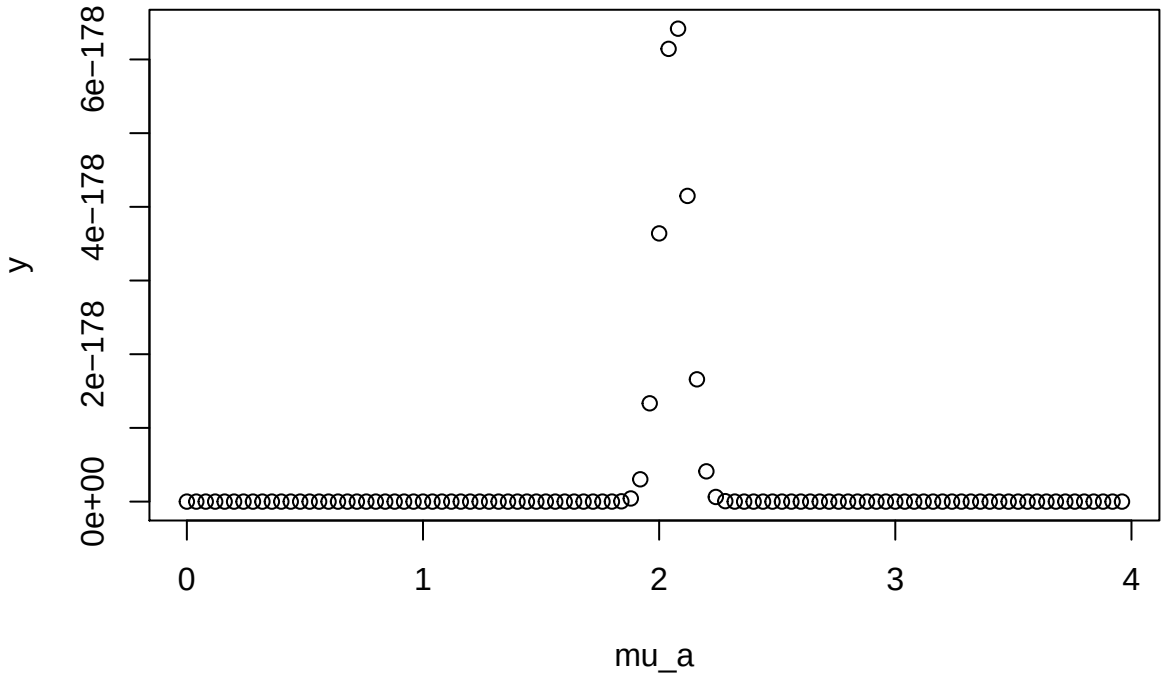
```
## [1] 7.012073e-176
```

```
L_norm( 3,r_norm)
```

```
## [1] 1.585847e-245
```

<–2.b–>

```
r_norm <- rnorm(n=300,mean = 2,sd=1)
mu_a = seq(0, by = 0.04, length = 100)
y = seq(0,length=100)
for (i in 1:100)
{
  y[i] = L_norm(mu_a[i],r_norm)
}
plot(mu_a,y)
```



###

on remarque que la vraisemble atteint maximum quand mu est proche de 2

<–2.c–>

```
optimise(L_norm,interval = c(0,4),maximum = TRUE,x_n=r_norm)
```

```
## $maximum
## [1] 2.063634
##
## $objective
## [1] 6.680719e-178
```

<–2.d–>

```
mu_a = seq(0, by = 0.04, length = 100)
#N est la taille d'échantillon
# max_mu :: calcule le maximum empirique de vraisemblance
max_mu <- function (N)
{
  r<- rnorm(N,2,1)
  max <-mu_a[0]
  for(i in 0:99)
  {
        if( (L_norm(mu_a[i+1],x_n=r)) > (L_norm(mu_a[i],x_n=r)) )
        max <- mu_a[i+1]
  }
  return(max)
}

#N est la taille d'échantillon
# compare: comparel'écart entre le maximum empirique de vraisemblance et celui théorique
```

```r
compare_sans_log <- function (N)
{
   r <- rnorm(N,2,1)
   a <-max_mu (N)
   b <-as.double(optimise(L_norm,interval = c(0,4),maximum = TRUE,x_n=r))
   list("N"=N,"max theorique"=b[1],"max obtenu"=a,"l'ecart"=b[1]-a)
}
compare_sans_log(10)
```

```
## $N
## [1] 10
##
## $`max theorique`
## [1] 1.992371
##
## $`max obtenu`
## [1] 2.24
##
## $`l'ecart`
## [1] -0.2476291
```

```r
compare_sans_log(50)
```

```
## $N
## [1] 50
##
## $`max theorique`
## [1] 2.044987
##
## $`max obtenu`
## [1] 2
##
## $`l'ecart`
## [1] 0.04498749
```

```r
compare_sans_log(100)
```

```
## $N
## [1] 100
##
## $`max theorique`
## [1] 1.859027
##
## $`max obtenu`
## [1] 2.04
##
## $`l'ecart`
## [1] -0.180973
```

```r
compare_sans_log(200)
```

```
## $N
## [1] 200
##
## $`max theorique`
## [1] 1.960129
```

```
##
## $`max obtenu`
## [1] 1.84
##
## $`l'ecart`
## [1] 0.1201291
```

```
compare_sans_log(500)
```

```
## $N
## [1] 500
##
## $`max theorique`
## [1] 1.967545
##
## $`max obtenu`
## [1] 1.96
##
## $`l'ecart`
## [1] 0.007545057
```

```
compare_sans_log(1000)
```

```
## $N
## [1] 1000
##
## $`max theorique`
## [1] 3.99994
##
## $`max obtenu`
## numeric(0)
##
## $`l'ecart`
## numeric(0)
```

```
compare_sans_log(2000)
```

```
## $N
## [1] 2000
##
## $`max theorique`
## [1] 3.99994
##
## $`max obtenu`
## numeric(0)
##
## $`l'ecart`
## numeric(0)
```

on remarque que max obtenu devient NA quand N est grand. Donc, on utilise log-vraissemblance

```
#x_n est un échantillon de lois Gaussien
Log_norm <- function(mu,x_n)
{
```

```
  N <- length(x_n)
  L <- sum(log(dnorm(x = x_n,mu,1)))
  return(L)
}

#N est la taille  d'échantillon
# max_mu_lg : calcule le maximum empirique de log_vraisemblance
max_mu_lg <- function (N)
{
  r<- rnorm(N,2,1)
  max <-mu_a[0]
  for(i in 1:99)
  {
        if( (Log_norm(mu_a[i+1],x_n=r)) > (Log_norm(mu_a[i],x_n=r)) )
        max <- mu_a[i+1]
  }
  return(max)
}

#N est la taille  d'échantillon
# compare_Lg : comparel'écart entre le maximum empirique de log_vraisemblance et celui théorique
compare_Lg <- function(N)
{
   r <- rnorm(N,2,1)
   a <-max_mu_lg(N)
   b <-as.double(optimise(Log_norm,interval = c(0,4),maximum = TRUE,x_n=r))

   list("N"=N,"max theorique"=b[1],"max obtenu"=a,"l'ecart"=b[1]-a)
}
compare_Lg(10)
```

```
## $N
## [1] 10
##
## $`max theorique`
## [1] 2.491415
##
## $`max obtenu`
## [1] 1.52
##
## $`l'ecart`
## [1] 0.9714155
```

```
compare_Lg(50)
```

```
## $N
## [1] 50
##
## $`max theorique`
## [1] 1.988054
##
## $`max obtenu`
## [1] 2.28
##
```

```
## $`l'ecart`
## [1] -0.2919456
```

```
compare_Lg(100)
```

```
## $N
## [1] 100
##
## $`max theorique`
## [1] 1.914153
##
## $`max obtenu`
## [1] 1.96
##
## $`l'ecart`
## [1] -0.04584721
```

```
compare_Lg(200)
```

```
## $N
## [1] 200
##
## $`max theorique`
## [1] 2.019838
##
## $`max obtenu`
## [1] 1.92
##
## $`l'ecart`
## [1] 0.0998383
```

```
compare_Lg(500)
```

```
## $N
## [1] 500
##
## $`max theorique`
## [1] 1.989238
##
## $`max obtenu`
## [1] 2.04
##
## $`l'ecart`
## [1] -0.0507622
```

```
compare_Lg(1000)
```

```
## $N
## [1] 1000
##
## $`max theorique`
## [1] 2.015446
##
## $`max obtenu`
## [1] 2
##
## $`l'ecart`
```

```
## [1] 0.01544626
```

```
compare_Lg(2000)
```

```
## $N
## [1] 2000
##
## $`max theorique`
## [1] 1.990477
##
## $`max obtenu`
## [1] 2
##
## $`l'ecart`
## [1] -0.0095227
```

3.Ajuster d'une loi à plusieurs paramètres

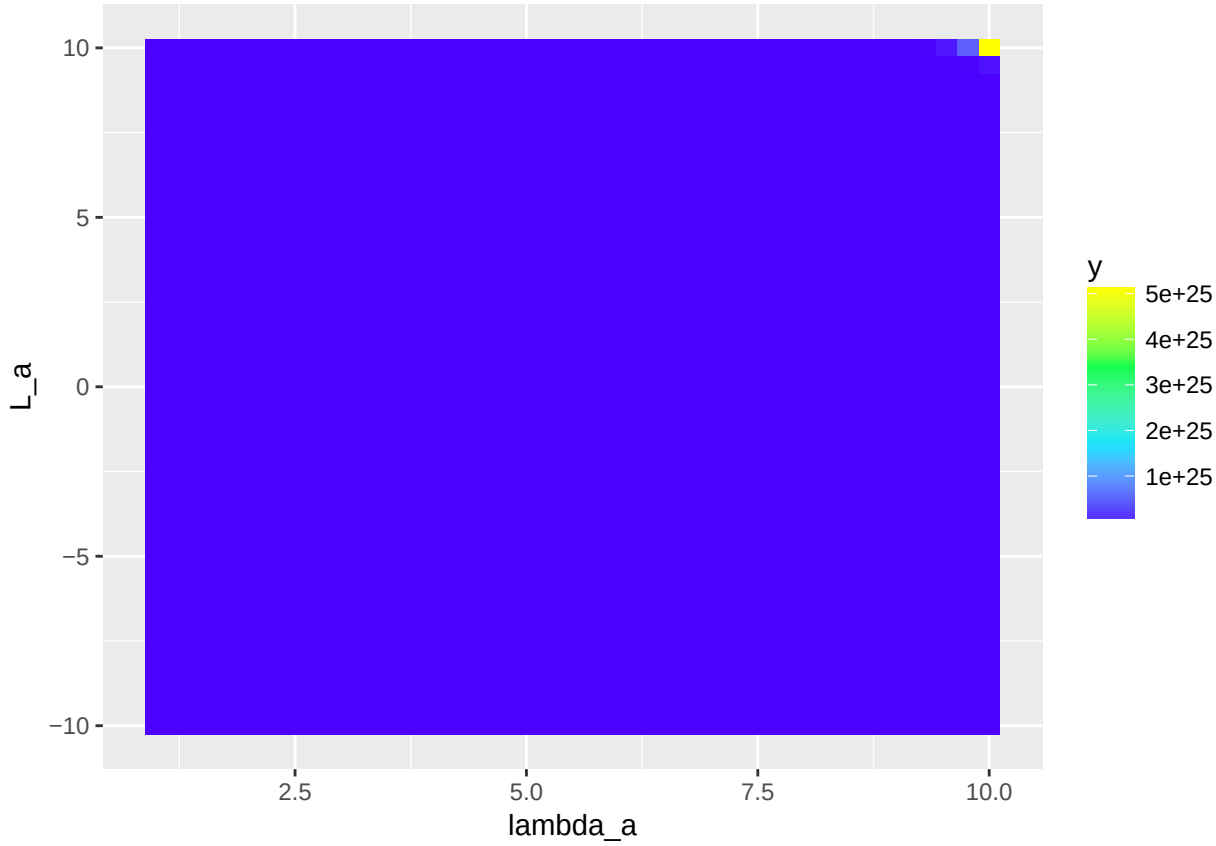
<–3.a–>

```
library(stats4)
library(ggplot2)

lambda <- 2
L <- 4
N = 100
r_exp <- rexp(n=N,lambda)*(exp(lambda*L))

x0 = -2
alpha = 0.4
r_cau <- rcauchy(n=N,x0,alpha)

# log_vraissemblance pour une lois exponentielle
# x_n est un échantillon
Log_vrs_exp <- function(param,x_n)
{
  lambda = param[1]
  L = param[2]
  N <- length(x_n)
  -sum(dexp(x_n,lambda,log=TRUE)*exp(lambda*L))
  #return(sum(log(lambda) - lambda * (x_n-L)))
}
n <- 40
# generation du dataframe
lambda_a <- rep(seq(1,10,9/(n - 1)), each=n)
length(lambda_a)
```

```
## [1] 1600
```

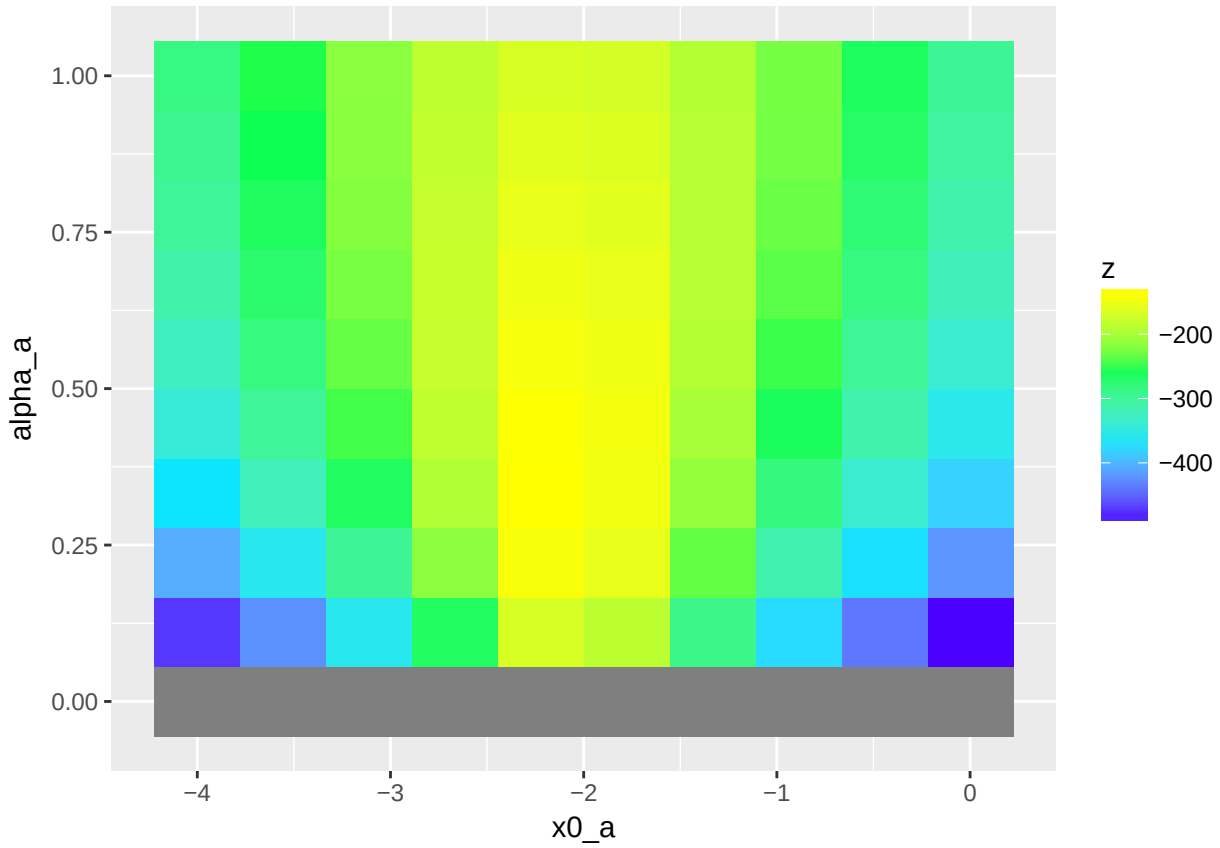
```
L_a   <- rep(seq(-10,10,20/(n - 1)), times=n)
y <- seq(0,length=1600)
for(i in 1:1600)
{
  y[i] = Log_vrs_exp(c(lambda_a[i],L_a[i]),r_exp)/exp(55)
```

```
}

df <- data.frame(lambda_a, L_a, y)
ggplot(df,aes(lambda_a,L_a))+geom_raster(aes(fill = y))+scale_fill_gradientn(colours = topo.colors(4))
```



Question : je n'arrive pas à obtenir le valeur max au travers de "ggplot"

```
# log_vraissemblance pour une lois cauchy
# x_n est un échantillon
Log_vrs_cau <- function (param,x_n)
{
  x0 = param[1]
  alpha = param[2]
  N <- length(x_n)
  L <- -N*log(pi)+N*log(alpha)-sum(log((x_n-x0)**2+alpha**2))
  return(L)
}
n <- 10
x0_a <- rep(seq(-4,0,4/(n - 1)), times=n)
alpha_a   <- rep(seq(0,1,1/(n - 1)), each=n)
z <- seq(0,length=100)
for(i in 1:100)
{
  z[i] = Log_vrs_cau(c(x0_a[i],alpha_a[i]),r_cau)
}
df <- data.frame(x0_a, alpha_a, z)
ggplot(df,aes(x0_a,alpha_a))+geom_raster(aes(fill = z))+scale_fill_gradientn(colours = topo.colors(4))
```



on peut ramarquer que la valeur dépend de lambda et x0

```
max_vrs_exp<- function (N)
{
  r_exp <- rexp(n=N,2)*exp(2*4)
  max <- c(lamda_a[0],L_a[0])
  for(i in 1:99)
  {
      if( (Log_vrs_exp(c(lamda_a[i+1],L_a[i+1]),x_n=r_exp)) >   Log_vrs_exp(c(lamda_a[i],L_a[i]),x_n=r_
       max =  c(lamda_a[i+1],L_a[i+1])
  }
  return(max)
}

max_vrs_cauchy<- function (N)
{
  r_cau <- rcauchy(N,-2,0.4)
  max <- c(x0_a[0],alpha_a[0])
  for(i in 1:99)
  {
      if( (Log_vrs_cau(c(x0_a[i+1],alpha_a[i+1]),x_n=r_cau)) >  Log_vrs_exp(c(x0_a[i],alpha_a[i]),x_n=r
       max =  c(x0_a[i+1],alpha_a[i+1])
  }
  return(max)
}
N<-100

r_exp <- rexp(n=N,2)*(exp(2*4))
```

```r
Loi_exp <- function (lambda,L)
{
   -sum(dexp(r_exp,lambda,log=TRUE)*exp(lambda*L))
}

r_cau <- rcauchy(n=N,-2,0.4)

Loi_cauchy <- function (x0,alpha)
{
    -sum(dcauchy(r_cau,x0,alpha,log=TRUE))
}
#la  log-vraisemblance théorique de loi exponentielle
#lambda commence à 0.01 car lambda > 0
mle(Loi_exp,start = list(lambda= 2,L = 4),method= "L-BFGS-B",lower = c(0.01,-Inf),upper=c(10,Inf))
```

```
##
## Call:
## mle(minuslogl = Loi_exp, start = list(lambda = 2, L = 4), method = "L-BFGS-B",
##     lower = c(0.01, -Inf), upper = c(10, Inf))
##
## Coefficients:
##    lambda         L
##  8.193337 -4.172730
```

```r
#la  log-vraisemblance théorique de loi cauchy
#alpha commence à 0.1 car alpha > 0
mle(Loi_cauchy,start = list(x0 = -2, alpha = 0.4),method= "L-BFGS-B",lower = c(-5,0.1),upper=c(5,10))
```

```
##
## Call:
## mle(minuslogl = Loi_cauchy, start = list(x0 = -2, alpha = 0.4),
##     method = "L-BFGS-B", lower = c(-5, 0.1), upper = c(5, 10))
##
## Coefficients:
##         x0      alpha
## -2.0342577  0.3673263
```