sqd

1) Ajuster une loi Bernoulli

a) Quelle est une façon simple d'estimer p?

```
N <- 10
p <- 0.7
Ber_echan <- rbinom(N, 1,prob=p)
mean(Ber_echan)
```

```
## [1] 0.8
```

b) Générez une fonction de vraisemblance, nommée L_bern, qui donne la vraisemblance d'un échantillon de Bernoulli pour une valeur donnée de p. p**(sum xi)*(1-p)**(n- sum xi)

```
L_bern<-function(echan, p){
res <- 1
for (i in echan){
res <- res * (p**i * (1-p)**(1-i))
}
return(res)
}
```

c) Pour votre échantillon, estimez la vraisemblance de l'échantillon pour 100 lois Bernoulli de paramètresp allant de 0 à 1. Tracez la courbe des valeurs calculées. Que remarquez-vous?

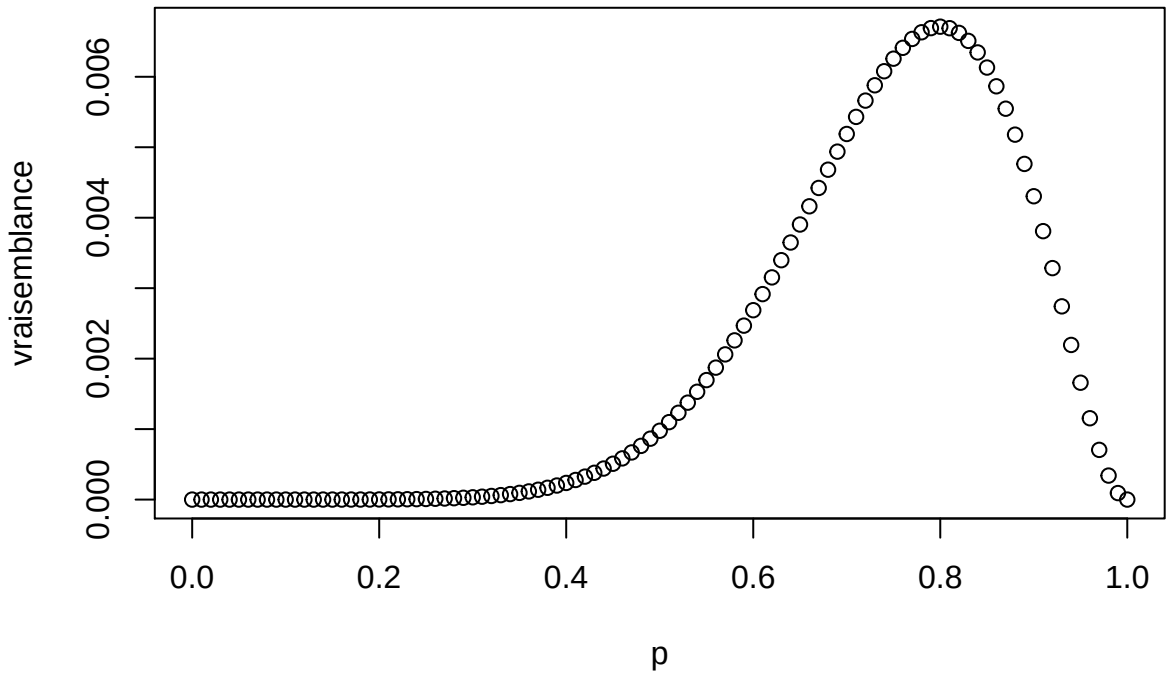
On remarque que la vraisemblance est une fonction continue, admettant un maximum.

```
Ber_courbe<-function(N){
  p <- seq(0, 1, 1/N)
  L <- vector(length = length(p))
  for(i in 1:length(p)){
    L[i]=L_bern( Ber_echan ,p[i] )
  }

  plot(L,x=p,ylab="vraisemblance")

}

Ber_courbe(100)
```



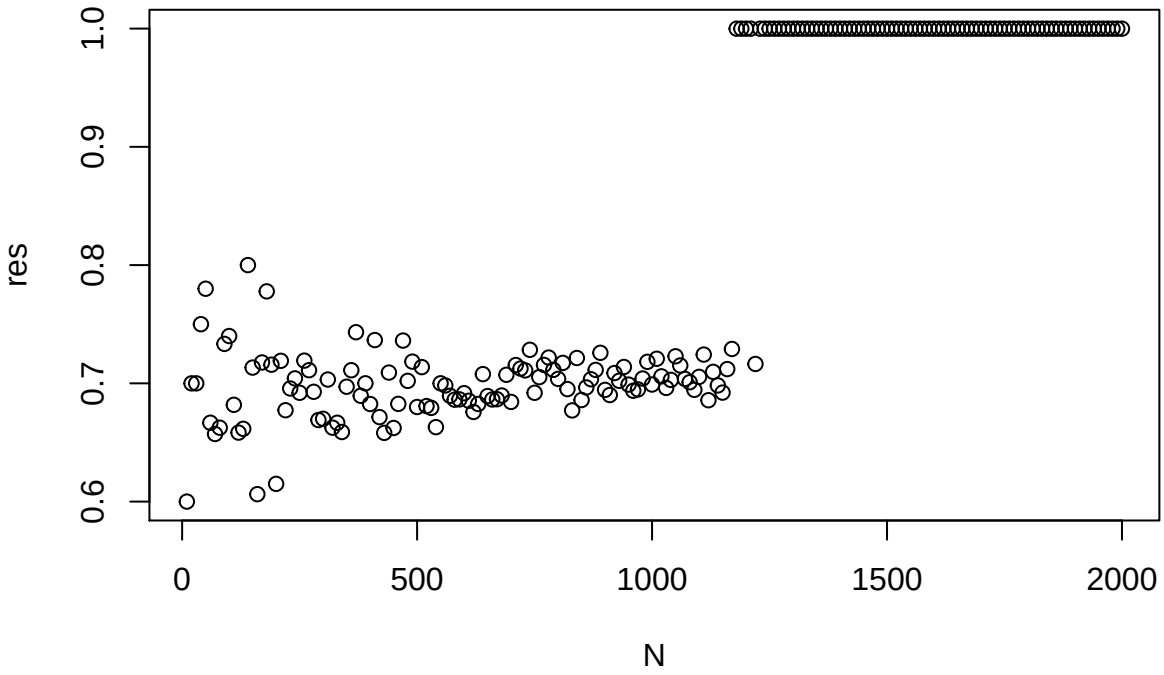
On marque que la courbe des vraisemblances sont tous à peu près égal lorsque p est différent de 0 ou 1. Elles sont toutes situés dans le bas du graphe.(sauf p = 0 et p = 1)

d) En utilisant la fonction optimize de R, trouvez la valeur de p la plus probable d'avoir généré cetéchantillon. (Attention à utiliser l'option maximum = TRUE et de borner les valeurs de p en utilisantl'option interval).

```
x<-optimize(L_bern, echan=Ber_echan, interval=c(0,1), maximum=TRUE)
```

on remarque que la valeur est proche de notre discretisation obtenue en (c) #e) Testez avec des échantillons de taille allant de N = 10 à N = 2000 et comparez l'écart entre lavaleur théorique attendue et la valeur obtenue. Que remarquez-vous? Comment combattre l'instabilité numérique due aux multiplications de probabilités?

```
N <-seq(10,2000,10)
res=vector(length = length(N))
for(k in 1:length(N) ){
  xx=rbinom(N[k], 1 ,0.7)
pk=optimize(L_bern, echan=xx, interval=c(0,1), maximum=TRUE)
res[k]=pk$maximum
}
plot(y= res,x=N)
```




```
mean(res[1:120])
```

```
## [1] 0.7060906
```

On remarque 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.

f)Chargez le fichier "distribution_inconue_2_100_realisations.csv" et trouvez l'estimateur de maximum de vraisemblance si on considére que c'est une loi Bernoulli. Pourquoi cette hypothèse est possible?

```
N<-read.csv("distribution_inconue_2_100_realisations.csv",header = TRUE)["x"]
x=as.vector(as.matrix(N))
optimize(L_bern, echan=x, interval=c(0,1), maximum=TRUE)
```

```
## $maximum
## [1] 0.580002
##
## $objective
## [1] 2.852948e-30
```

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

Soit une loi Normale (rnorm) avec = 2 et = 1. Simuler un échantillon i.i.d de taille N = 30

```
N <- 30
mu= 2
delta = 1
N_echan <- rnorm(N, mu ,delta)
print(N_echan)
```

```
##  [1] 1.9833037 3.5202776 2.7992716 1.6021226 4.4643501 2.2518596 1.3947981
##  [8] 1.8392871 2.6465709 2.0176355 2.6871378 2.4153771 1.2862734 2.0030151
## [15] 2.0311437 4.2390800 3.8274786 0.8401202 0.9428379 1.2330702 1.4444947
## [22] 1.7756962 0.9676564 1.4226996 0.5756146 2.9378151 1.1736246 3.5881029
## [29] 2.2212204 1.3006273
```

a) Générez une fonction de vraisemblance, nommée L_norm, qui donne la vraisemblance d'un échantillon pour une valeur donnée de . (Voir la fonctiondnorm')

```
L_norm<-function(echan,mu){
  som<-0
for (i in echan) {
  som<-som+(i-mu)**2
}
  res<-(2*pi)**(-length(echan)/2)*exp(-0.5*som)
return(res)
}
L_norm(N_echan,2)
```

```
## [1] 1.951993e-19
```

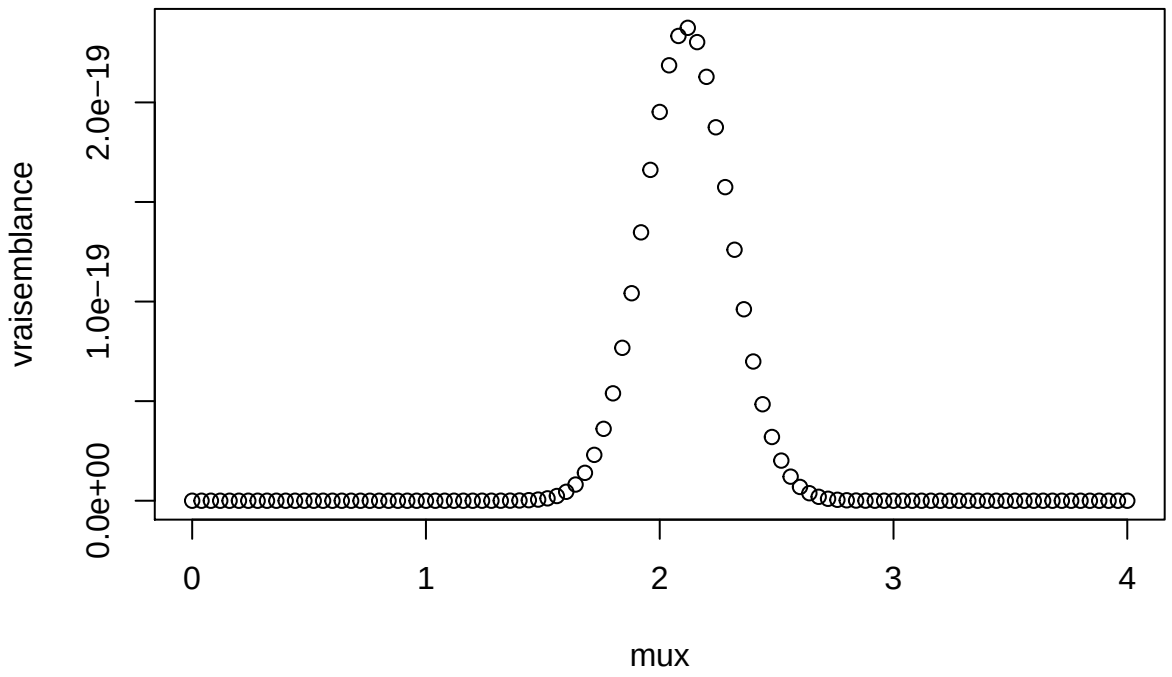
b) Pour votre échantillon, estimez la vraisemblance de l'échantillon pour 100 lois normales de paramètres allant de 0 à 4. Tracez la courbe des valeurs calculées. Que remarquez-vous?

On remarque la vraissemblance est d'autant plus élévé que le paramètre testé se rapproche du vrai paramètre

```
N_courbe<-function(n){
  mux=seq(0,4,0.04)
  Ne <- vector(length =length(mux))
  for(i in 1:100){
    Ne[i]=L_norm( N_echan , mux[i] )
  }

  plot(y=Ne,x=mux,ylab="vraisemblance")
```

```
}
N_courbe(100)
```



c) En utilisant la fonction optimize de R, trouvez la valeur de p la plus probable d'avoir généré cet échantillon. (Attention à utiliser l'option maximum = TRUE et de borner les valeurs de p en utilisant l'option interval).

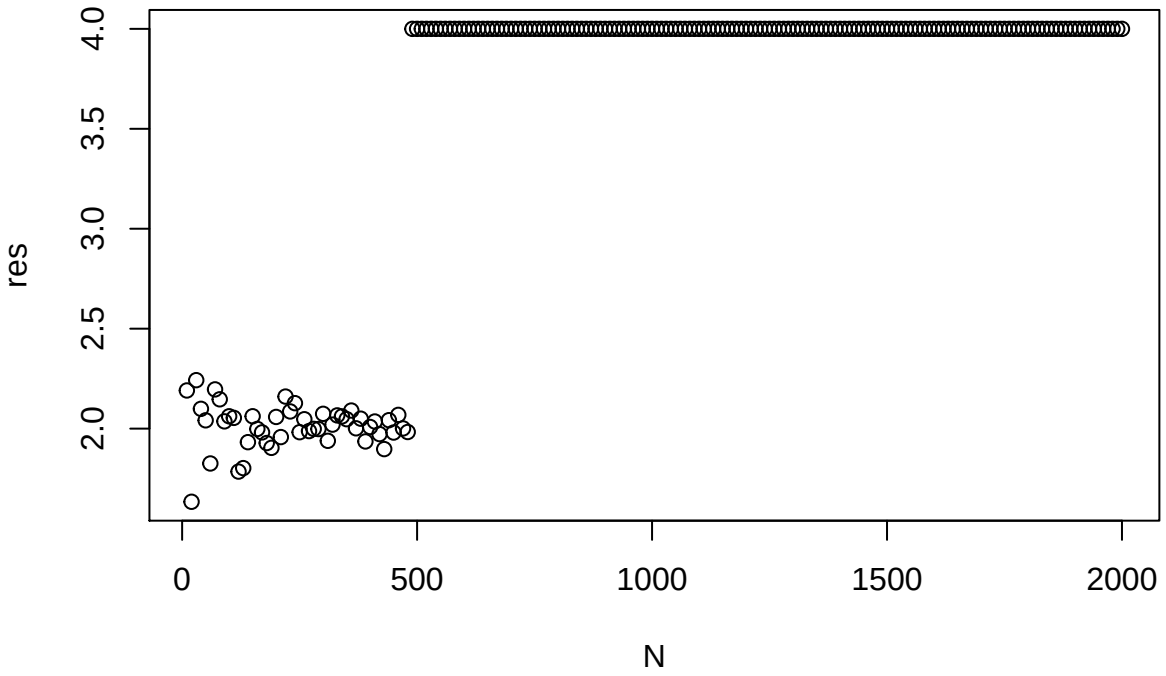
```
x<-optimize(L_norm, echan=N_echan, interval=c(0,4), maximum=TRUE)
x
```

```
## $maximum
## [1] 2.114418
##
## $objective
## [1] 2.375542e-19
```

d) Testez avec des échantillons de tailles allant de N = 10 à N = 2000 et comparez l'écart entre la valeur théorique attendue et la valeur obtenue. Utilisez une technique qui permet d'éviter ces instabilités en caclulant des sommes.

```
N <-seq(10,2000,10)
res=vector(length = length(N))
for(k in 1:length(N) ){
```

```
  xx=rnorm(N[k], mu ,delta)
pk=optimize(L_norm, echan=xx, interval=c(0,4), maximum=TRUE)
res[k]=pk$maximum
}
plot(y= res,x=N)
```



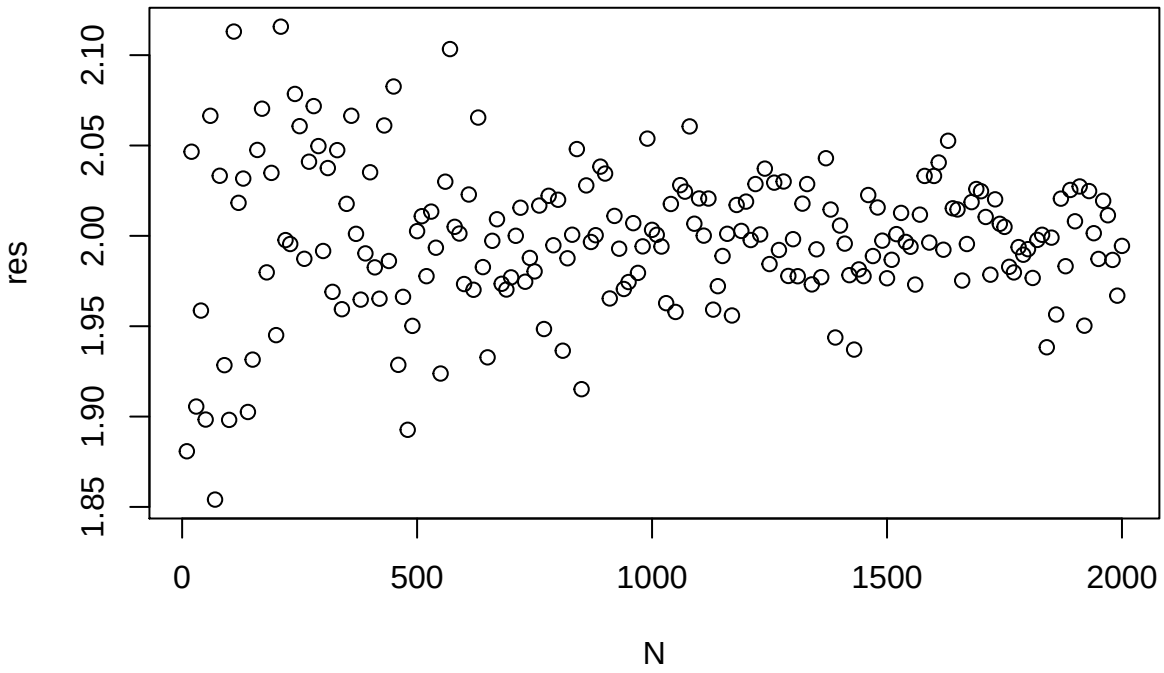

```
mean(res[100:200])
```

```
## [1] 3.99994
```

Utilisez une technique qui permet d'éviter ces instabilités en caclulant des sommes.

```
log_L_norm <- function(X_N, mu) {
  #return (prod(dnorm(X_N, mu, 1)))
  return (sum(log(dnorm(X_N, mu, 1))))
}

for (i in 1:length(N)) {
  X_Ni  <- rnorm(N[i], 2, 1)
  mu    <- optimize(function(mu) { log_L_norm(X_Ni, mu) }, interval=c(0,4), maximum=TRUE)
  res[i] <- mu$maximum
}
plot(N, res)
```




```
mean(res)
```

```
## [1] 1.998172
```

3) Ajuster d'une loi à plusieurs paramètres

Pour des lois à plusieurs paramètres, résoudre un probléme numérique nécessite de trouver un maximum sur une surface. Ceci pourrait être résolu en utilisant des solveurs plus compliqués que "optimize". Un solveur typique est le solveur mle (maximum likelihood estimator) du package stats4 (intall.packages(stats4), puis library(stats4)).

```
library(stats4)
```

Générez des échantillons issus:

1. de la loi exponentielle translatée avec paramètres = 2 et L = 4. (Pdf: f(x) = ∗ e− (x−L) avec x L)

```
lambda<-2
L<-4
E_echan<-rexp(100,rate=lambda) * exp(lambda * L)
L_exp <- function(X_N, lambda, L) {
```

```
  return (sum(log(lambda) - lambda * (X_N - L)))
}
```

2. de la loi de Cauchy de paramètres x0 = −2 and = 0.4. (Pdf: f(x;x0,) = 1/ * / ((x − x0)2 + 2), avec() > 0)

```
C_echan<-rcauchy(100,location=-2,scale=0.4)
L_cauchy <- function(X_N, x0, alpha) {
  pi <- 3.1415926
  return (sum(log(alpha / pi) - log((X_N - x0)**2 + alpha**2)))
}
```

Pour ces deux lois, calculez la log-vraisemblance des échantillons que vous avez générés, en faissant varier un paramètre à la fois. Présentez graphiquement la surface de log-vraisemblance que vous calculez, en utilisant la librairie ggplot2. Commentez.

```
library(ggplot2)
```

Comparez la valeur obtenue de cette façon à la valeur que vous obtenez en appliquant le solveur mle de la librairie stats4. Commentez.Effectuez une étude de sensibilité des vos résultats en fonction de la taille de votre échantillon.

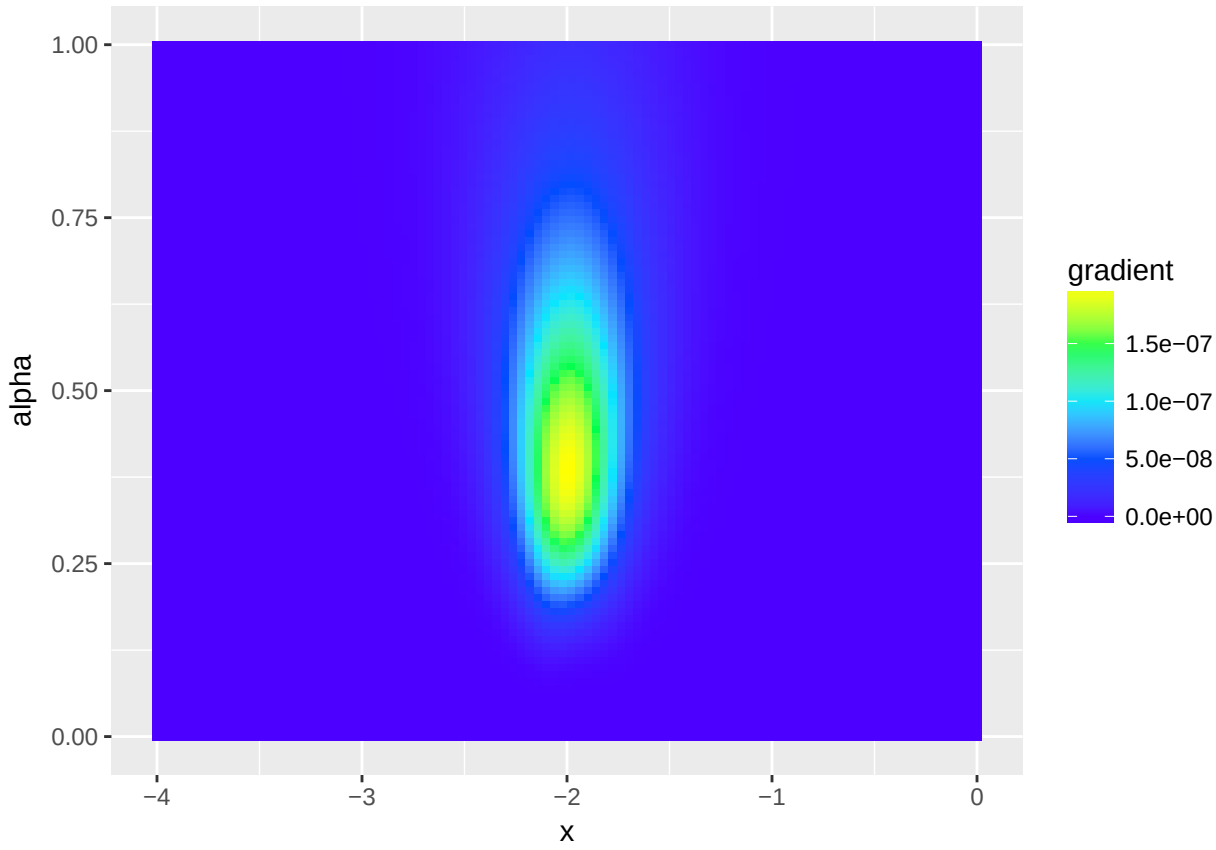
```
n <- 100
m <- 1
x            <- rep( seq(-4.0, 0.0, (0.0 - -4.0) / (n - 1)), times=n  )
alpha         <- rep( seq(0.0, 1.0, (1.0 - 0.0) / (n - 1)), each=n  )
vraissemblance <- c()
gradient       <- c()

for (i in 1:(n*n)) {
  vraissemblance[i] <- L_cauchy(C_echan, x[i], alpha[i])

  gradient[i]       <- exp( vraissemblance[i] * 0.1 )
  m <- if (vraissemblance[i] > vraissemblance[m]) i else m
}

df <- data.frame(x, alpha, vraissemblance, gradient)

ggplot(df, aes(x, alpha)) +
    geom_raster(aes(fill = gradient)) +
    scale_fill_gradientn(colours = topo.colors(5))
```




```
c("x0" = -2, "alpha" = 0.4)
```

```
##    x0 alpha
##  -2.0   0.4
```

```
c("x0_max" = x[m], "alpha_max" = alpha[m])
```

```
##     x0_max  alpha_max
## -1.9797980  0.3838384
```