Fiche TD avec le logiciel R : tdr221

Estimer un mélange de lois normales

D. Chessel, A.B. Dufour & J.R. Lobry

Maximum de vraisemblance, solution numérique, simulations et estimations.

Table des matières

1 Maximum de vraisemblance

Implanter la fonction de vraisemblance d'un échantillon de la loi de Poisson :

```
llpois <- function(lambda, obs) {
    -sum(dpois(x = obs, lambda = lambda, log = TRUE))
}
```

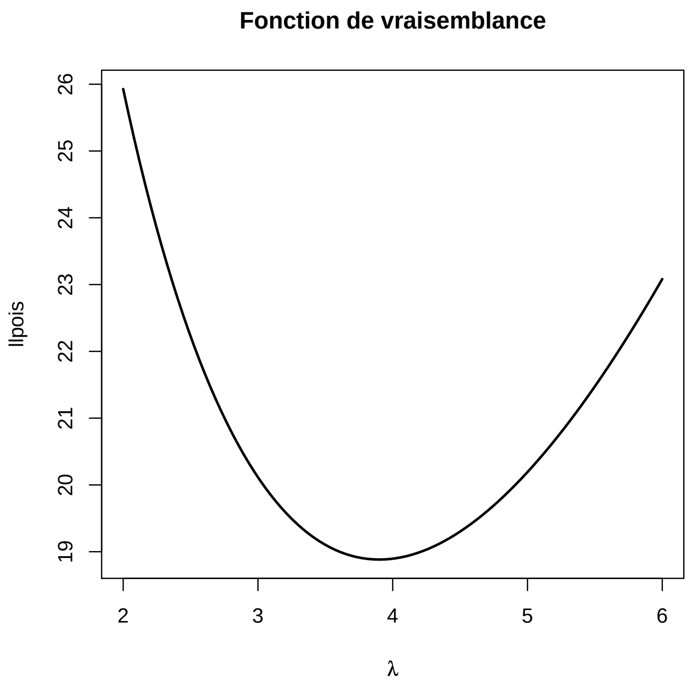
Construire un échantillon :

```
obs <- rpois(n = 10, lambda = 4)
mean(obs)
```

```
[1] 3.9
```

Tracer la fonction de vraisemblance :

```
x0 <- seq(from = 2, to = 6, length = 100)
plot(x = x0, y = sapply(x0, function(x) llpois(x, obs)), xlab = expression(lambda),
    ylab = "llpois", main = "Fonction de vraisemblance", type = "l",
    lwd = 2)
```



Commenter. Vérifier numériquement le résultat du cours.

2 Solution numérique

La fonction `nlm()` minimise une fonction. On l'utilise pour minimiser la log-vraisemblance négative, soit :

$$L(\mathbf{x}) = \prod_{i=1}^{n} f(x_i, \mathbf{p}) \Rightarrow LL(\mathbf{x}) = \sum_{i=1}^{n} \log(f(x_i, \mathbf{p})) \Rightarrow -LL(\mathbf{x}) = -\sum_{i=1}^{n} \log(f(x_i, \mathbf{p}))$$

où $\mathbf{p}$ est le vecteur des paramètres, $\mathbf{x}$ est le vecteur des observations et f est la densité de probabilité recherchée.

```
nlm(f = llpois, p = 3, obs = obs)
```

```
$minimum
[1] 18.88271

$estimate
[1] 3.899998

$gradient
[1] 8.380765e-08

$code
[1] 1

$iterations
[1] 5
```

3 Application à un mélange de lois normales

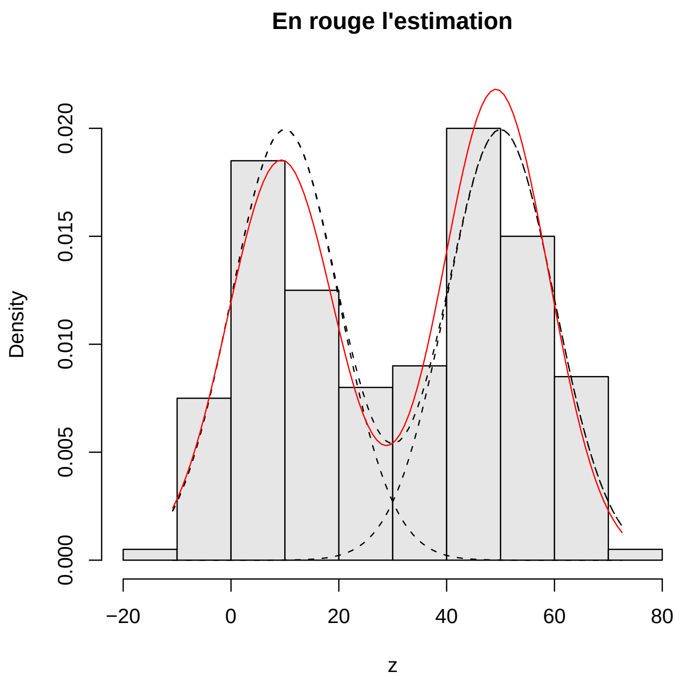
```
logvraineg <- function(param, obs) {
    p <- param[1]
    m1 <- param[2]
    sd1 <- param[3]
    m2 <- param[4]
    sd2 <- param[5]
    -sum(log(p * dnorm(x = obs, mean = m1, sd = sd1) + (1 - p) *
        dnorm(x = obs, mean = m2, sd = sd2)))
}
stp <- read.table("http://pbil.univ-lyon1.fr/R/donnees/t3var.txt",
    header = TRUE)
x <- stp[, "tai"]
nlm(f = logvraineg, p = c(0.3, 165, 6, 180, 6), obs = x)
```

```
$minimum
[1] 239.9957

$estimate
[1]   0.1570021 161.5495030   4.9179701 176.3906346   7.8894623

$gradient
[1] -9.123369e-06  5.682600e-08  1.271414e-07  2.196193e-07 -3.242241e-07

$code
[1] 1

$iterations
[1] 49
```

La convergence dépend étroitement des valeurs de départ. Pour tester cette procédure on écrit une fonction qui simule un échantillon d'un mélange de lois normales, trace l'histogramme de l'échantillon, la densité de probabilité exacte, estime les paramètres et trace la densité estimée.

```
simulmixnor <- function(n, p, m1, sd1, m2, sd2) {
    n1 <- rbinom(n = 1, size = n, prob = p)
    x1 <- rnorm(n = n1, mean = m1, sd = sd1)
    x2 <- rnorm(n = n - n1, mean = m2, sd = sd2)
    c(x1, x2)
}
mixnor <- function(eff = 200, freq = 0.3, moy1 = 1, et1 = 1, moy2 = 5,
    et2 = 2) {
    z <- simulmixnor(n = eff, p = freq, m1 = moy1, sd1 = et1, m2 = moy2,
        sd2 = et2)
    paramvrai <- c(freq, moy1, et1, moy2, et2)
    paramdeb <- paramvrai + runif(n = 5, min = -0.1 * paramvrai,
        max = 0.1 * paramvrai)
    q <- nlm(f = logvraineg, p = paramdeb, obs = z)$estimate
    print(q)
    w0 <- seq(from = min(z), to = max(z), length = 100)
    pop1 <- freq * dnorm(x = w0, mean = moy1, sd = et1)
    pop2 <- (1 - freq) * dnorm(x = w0, mean = moy2, sd = et2)
    pop <- pop1 + pop2
    est1 <- q[1] * dnorm(x = w0, mean = q[2], sd = q[3])
    est2 <- (1 - q[1]) * dnorm(x = w0, mean = q[4], sd = q[5])
    est <- est1 + est2
    maxhist <- max(hist(z, proba = TRUE, plot = FALSE)$density)
    hist(z, proba = TRUE, col = grey(0.9), ylim = c(0, max(pop,
        est, maxhist)), main = "En rouge l'estimation")
    lines(w0, pop1, lty = 2)
    lines(w0, pop2, lty = 2)
    lines(w0, pop, lty = 2)
    lines(w0, est, col = "red")
}
mixnor(200, 0.5, 10, 10, 50, 10)
```

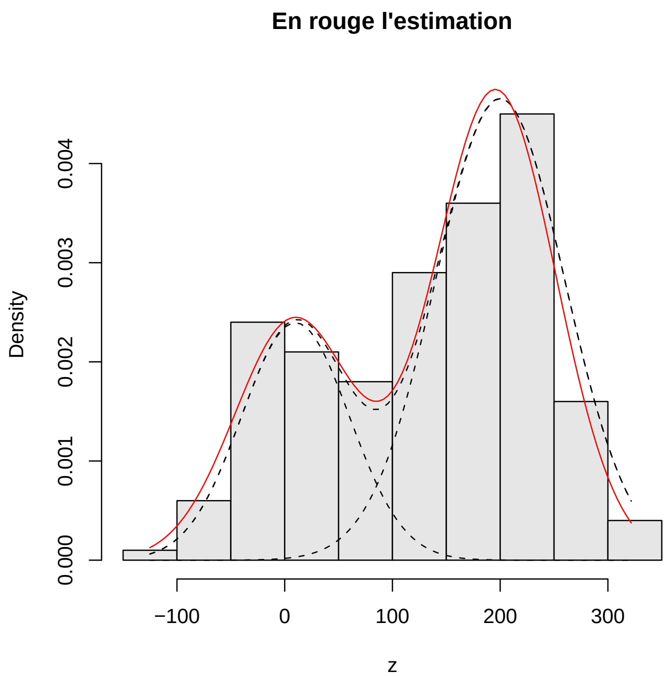
```
[1]  0.4679546  7.9598837  9.8973180 48.2547861 11.3447281
```



On peut essayer des situations variées. L'estimation des variances est toujours moins bonne que celle des moyennes et l'estimation de **p** est d'autant meilleure que les deux groupes sont disjoints :

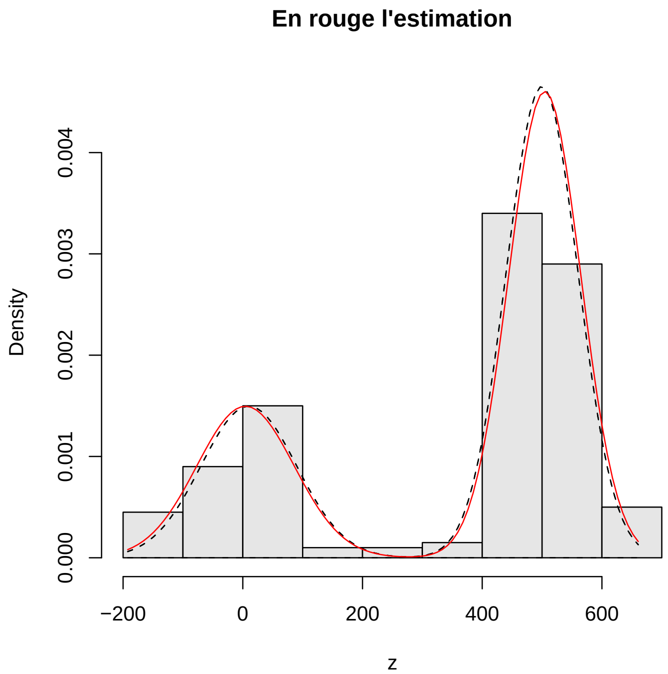
```
mixnor(200, 0.3, 10, 50, 200, 60)
```

```
[1]   0.2548936   0.9314191  42.4795686 202.0753919  55.9540267
```



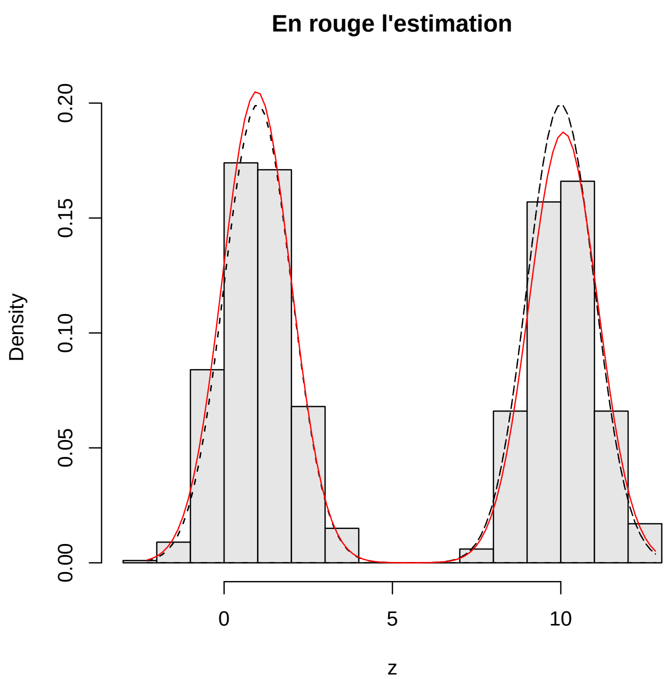

```
mixnor(200, 0.3, 10, 80, 500, 60)
```

```
[1]   0.2801980  30.5625968  98.5699888 501.1469643  60.6062066
```



```
mixnor(1000, 0.5, 1, 1, 10, 1)
```

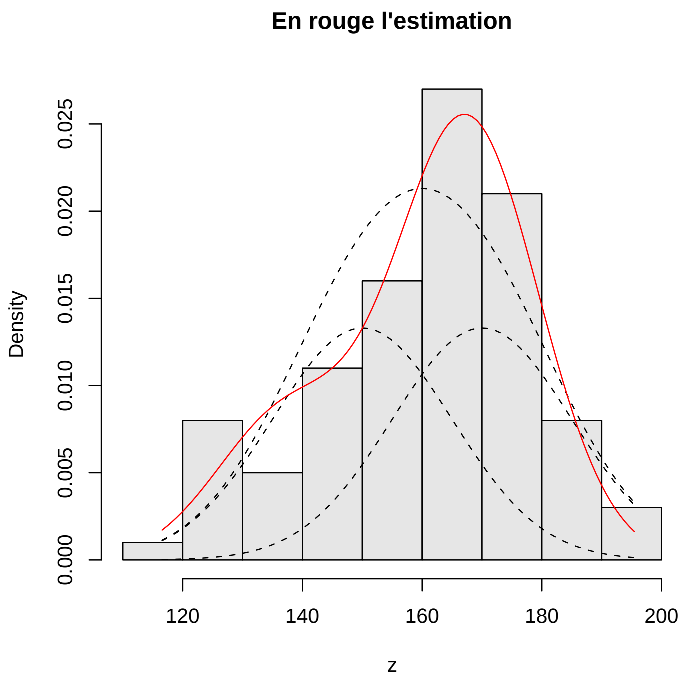
```
[1]  0.4799733  0.9871386  1.0645887 10.0518969  0.9829263
```



Il arrive que le modèle ne converge pas (on suppose le problème du point de départ résolu en introduisant simplement une erreur de 10 % sur les vraies valeurs). On peut avoir plus ou moins de chance :

```
mixnor(100, 0.5, 150, 15, 170, 15)
```

```
[1]   0.1810174 146.8580766   5.1258874 158.9338963  21.7410596
```

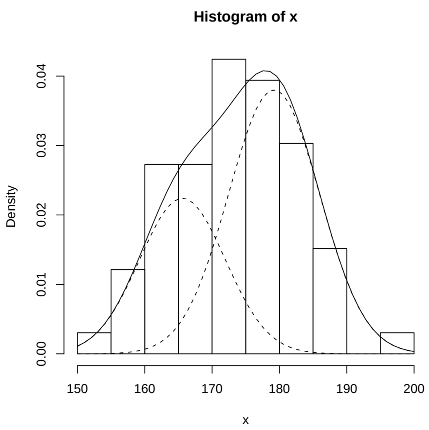


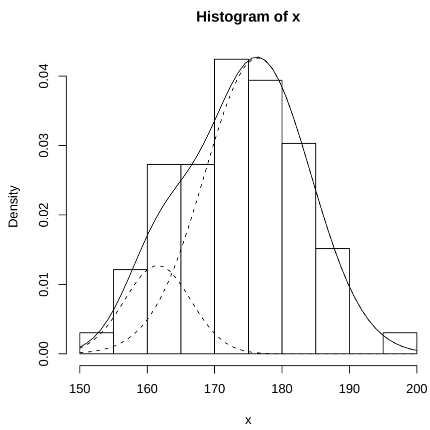
On était parti de la taille de 41 garçons de moyenne 179.2 et d'écart-type 6.74 et de la taille de 25 filles de moyenne 165.6 et d'écart-type 6.35. Si on estime les paramètres en ne connaissant que le mélange :

```
par(mfrow = c(1, 2))
hist(x, proba = T, nclass = 10)
provi <- seq(150, 200, le = 50)
x1 <- dnorm(provi, 165.6, sqrt(40.32))
x2 <- dnorm(provi, 179.2, sqrt(45.41))
x3 <- 0.357 * x1 + 0.643 * x2
lines(provi, 0.357 * x1, lty = 2)
lines(provi, 0.643 * x2, lty = 2)
lines(provi, x3)
nlm(logvraineg, c(0.5, 160, 10, 180, 10), obs = x)$estimate
```

```
[1]   0.1570022 161.5495047   4.9179722 176.3906365   7.8894609
```

```
hist(x, proba = T, nclass = 10)
provi <- seq(150, 200, le = 50)
x1 <- dnorm(provi, 161.55, 4.92)
x2 <- dnorm(provi, 176.4, 7.89)
x3 <- 0.157 * x1 + 0.843 * x2
lines(provi, 0.157 * x1, lty = 2)
lines(provi, 0.843 * x2, lty = 2)
lines(provi, x3)
```





A gauche, ce qu'on sait quand on connaît les groupes, à droite ce qu'on infère quand on ne les connaît pas. On est loin du compte parce que 5 paramètres estimés avec 66 mesures c'est trop. L'estimation des mélanges de lois normales est un problème délicat pour lequel on a des outils à manier avec précaution.