Consultation statistique avec le logiciel R

Problème de convergence pour estimer « una mezcla de dos distribuciones normales » avec *Triatoma dimidiata*

Pr Jean R. Lobry

1 Question posée

Jorge Rabinovich est un collègue argentin, vous pouvez vous faire une idée de ses travaux de recherche avec la fiche `tdr335`[1]. Il m'a demandé par courriel la chose suivante :

> City Bell, Monday, May 4, 2009, 16:02
> Hello Jean:
> I wonder if you remember me, visiting Lyon several times to work with Carlos Bernstein and Fred Menu.
> I will bother you with a minor problem, but as it refers to a program resource you developed in R, I thought that you were the best person to contact.
> I need to checked for the possibility of separating an apparent mix of two normal distributions in the development time of a species of triatomine (« pounaise »). I have applied your procedure (obtained from your « Fiche TD avec le logiciel R. `tdr222` ») to several species, without any problem, but I cannot make it work with the species *Triatoma dimidiata*. I get a message of `NAN` in the `logvraineg()` function. I think that maybe the problem is that this function in very sensitive to the first estimates of the parameters of the first and second normals, and also that maybe there are not two normals, because I only have a increase in the right tail of the distribution. However, with very similar data in other species it worked well.
> I here enclose the R script (`ScriptNormalBiModal.txt`) I am using and also de data file (`Tdimidiata (revisado 05-mayo-09).txt`) with the original development times ($N = 139$) of the problematic species. I also attach an image file with the results of the separation of normals to another species (`BiModal_DT_Tbrasiliensis.jpg`) so you can see the similarity between both species.
> If you could have a little bit of time to take a look at this problem and let me know where I have made a mistake, I would appreciate it very much.

[1] Densité de population et ingestion de nourriture chez un insecte vecteur de la maladie de Chagas : `http://pbil.univ-lyon1.fr/R/pdf/tdr335.pdf`

A bientôt,
Jorge

BIEN sûr que je me souviens de Jorge ! Comment pourrais je l'oublier ? Il a publié en 2006 un article [2] citant mon tout premier article scientifique [1] publié en 1991 dans une obscure revue de rang Z. Que voulez-vous : qu'un programme écrit en bon vieux FORTRAN 77 puisse encore rendre service 15 ans après fait tout chaud au cœur.

1.1 Les commandes R de Jorge

LE FICHIER de commandes R est le suivant, notez comme les statistiques ont tout de suite un air plus sympathique en espagnol :

```
# Análisis de mezcla de dos distribuciones normales
# Fiche TD avec le logiciel R. tdr222 - J.R. Lobry

# Se utiliza la función logvraineg() que devuelve el valor del logaritmo de la
# función de máxima verosimilitud en el caso de una mezcla de dos
# distribuciones normales (en realidad, lo opuesto a este valor, ya que lo que
# se busca es maximizar la verosimilitud y la función de optimización aquí utilizada,
# nlm(), busca la minimización del valor de la función pasada como argumento).


logvraineg <-function(param, obs) {
p <-param[1]
m1 <-param[2]
sd1 <-param[3]
m2 <-param[4]
sd2 <-param[5]
-sum(log(p * dnorm(obs, m1, sd1) + (1 -p) * dnorm(obs, m2, sd2)))
}

# Se usa la función simulmixnor() para simular una mezcla de dos distribuciones normales
# En esta función los parámetros representan la frecuencia relativa de la primera
# población en la mezcla de las dos poblaciopnes, m1 y m2 son las medias de la primera
# y de la segunda población, respectivamente, sd1 y sd2 son las desviaciones estándar
# de la primera y de la segunda población, respectivamente.

simulmixnor <-function(n, p, m1, sd1, m2, sd2) {
n1 <-rbinom(1, n, p)
x1 <-rnorm(n1, m1, sd1)
x2 <-rnorm(n -n1, m2, sd2)
c(x1, x2)
}

# Se establece el directorio de trabajo

setwd("D:\\ECOS2\\2009\\ANR Emergent Diseases\\Visita Fred 2009\\Bomidalidad Normal en R")

# se lee el archivo de datos

data <-read.table(file = "Tdimidiata (revisado 05-mayo-09).txt",h = T, sep = "\t", quote = "\"")

# se verifican los encabezados

names(data)

# Se especifica cuál de las columnas se va a analizar

x <-data$DevTime

# Se produce el histograma de todo el conjunto de datos

hist(x, col = grey(0.8), proba = TRUE, main = paste("Distribution of development times ",
"(N= ", length(x),")\n", "T dimidiata Female & Male 5th instar nymphs"),
xlab = "Development time (days)", ylab = "Density")

xseq <-seq(from = min(x), to = max(x), length = 100)
```

```
lines(xseq, dnorm(xseq, mean(x), sd(x)), lwd = 2)

# Se verifica si la distribución de los tiempos de desarrollo tiene una distribución normal,
# para lo que se visualiza bien mediante un gráfico de cuantiles-cuantiles:

qqnorm(x)
qqline(x)

# Suponiendo quelos datos son una mezcla heterogénea, se ensaya la estimación
# de una mezcla de distribuciones normales

#  La optimización de la función logvraineg() produce:

resnlm <-nlm(f = logvraineg, p = c(1, 290, 5, 380, 5), obs = x)
resnlm$estimate

xseq <-seq(min(x), max(x), le = 300)
est <-resnlm$estimate
y1 <-est[1] * dnorm(xseq, est[2], est[3])
y2 <-(1 -est[1]) * dnorm(xseq, est[4], est[5])

hist(x, proba = T, ylim = c(0, max(y1 + y2)), col = grey(0.8),
main = paste("Distribution of development times ","(N= ", length(x),")\n",
"T dimidiata Female & Male 5th instar nymphs"),
xlab = "Development time (days)", ylab = "Normal probability density")

lines(xseq, y1 + y2)
lines(xseq, y1, col = "red", lwd = 2)
lines(xseq, y2, col = "blue", lwd = 2)
```

1.2 Les données de Jorge

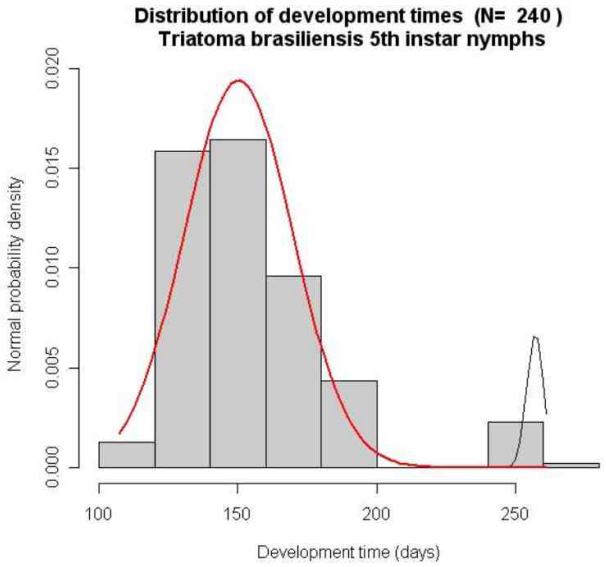
Le fichier comportant les données est le suivant :

```
 dput(read.table("http://pbil.univ-lyon1.fr/R/donnees/Tdimidiata.txt", header = TRUE))
```

```
structure(list(DevTime = c(202L, 202L, 202L, 202L, 202L, 216L,
216L, 216L, 216L, 216L, 226L, 226L, 226L, 226L, 226L, 226L, 226L,
226L, 226L, 235L, 235L, 235L, 235L, 235L, 235L, 235L, 235L, 235L,
246L, 246L, 246L, 246L, 246L, 246L, 246L, 256L, 256L, 256L, 256L,
256L, 256L, 256L, 256L, 256L, 256L, 256L, 256L, 256L, 256L, 256L,
256L, 266L, 266L, 266L, 266L, 266L, 266L, 266L, 266L, 266L, 266L,
266L, 266L, 266L, 266L, 266L, 266L, 276L, 276L, 276L, 276L, 276L,
276L, 276L, 276L, 276L, 276L, 286L, 286L, 286L, 286L, 286L, 286L,
286L, 286L, 286L, 286L, 286L, 286L, 286L, 286L, 286L, 286L, 296L,
296L, 296L, 296L, 296L, 296L, 296L, 296L, 296L, 296L, 296L, 296L,
296L, 308L, 308L, 308L, 308L, 308L, 308L, 308L, 308L, 308L, 319L,
319L, 319L, 319L, 319L, 319L, 319L, 339L, 339L, 339L, 339L, 339L,
339L, 339L, 339L, 372L, 372L, 372L, 372L, 372L, 372L, 372L,
372L)), class = "data.frame", row.names = c(NA, -139L))
```

1.3 La figure de Jorge

La figure pour *Triatoma brasiliensis* est la suivante :



2 Reproduction et analyse des résultats de Jorge

Nous essayons donc de reproduire ici les problèmes de Jorge. Nous avons le code R et les données, donc tout ce dont nous avons besoin *a priori*. On commence par définir les mêmes fonctions que dans la fiche `tdr222`[2] sur les mélanges de lois normales.

2.1 Défintion de `logvaineg()` et `simulmixnor()`

```
logvraineg <-function(param, obs) {
  p <-param[1]
  m1 <-param[2]
  sd1 <-param[3]
  m2 <-param[4]
  sd2 <-param[5]
  -sum(log(p * dnorm(obs, m1, sd1) + (1 -p) * dnorm(obs, m2, sd2)))
}
simulmixnor <-function(n, p, m1, sd1, m2, sd2) {
  n1 <-rbinom(1, n, p)
  x1 <-rnorm(n1, m1, sd1)
  x2 <-rnorm(n -n1, m2, sd2)
  c(x1, x2)
}
```

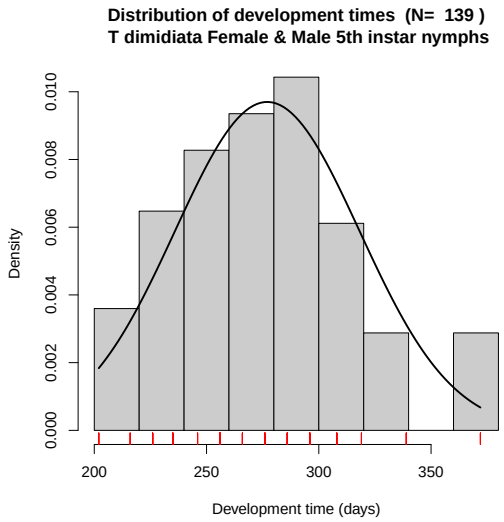
On importe le jeu de données :

```
data <- read.table("http://pbil.univ-lyon1.fr/R/donnees/Tdimidiata.txt", header = TRUE)
names(data)
[1] "DevTime"
dim(data)
[1] 139   1
```

On retrouve ici $n = 139$ comme Jorge nous l'avait annoncé, pas de problème à ce niveau. On les représente graphiquement :

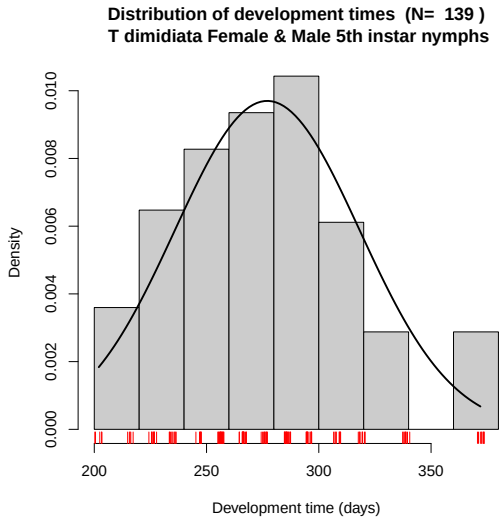
[2] `http://pbil.univ-lyon1.fr/R/pdf/tdr222.pdf`

```
x <-data$DevTime
hist(x, col = grey(0.8), proba = TRUE, main = paste("Distribution of development times ",
"(N= ", length(x),")\n", "T dimidiata Female & Male 5th instar nymphs"),
xlab = "Development time (days)", ylab = "Density")
xseq <-seq(from = min(x), to = max(x), length = 100)
lines(xseq, dnorm(xseq, mean(x), sd(x)), lwd = 2)
rug(x, col = "red")
```



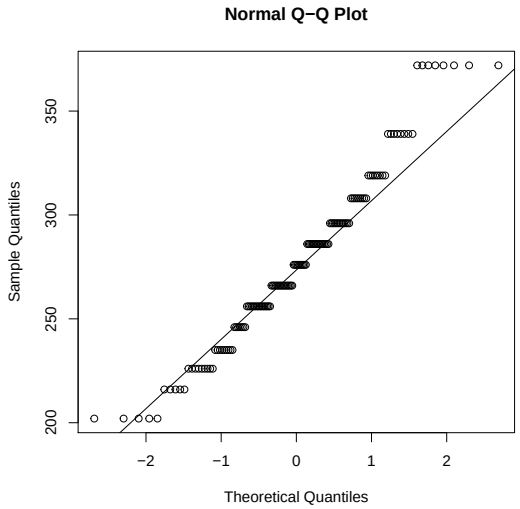
On voit tout de suite ici qu'il va être difficile d'estimer les paramètres d'un mélange de lois normales. Il n'y a pas deux modes bien évidents et les données sont de nature discrète. Un petit coup de `jitter()` pour y voir plus clair :

```
x <-data$DevTime
hist(x, col = grey(0.8), proba = TRUE, main = paste("Distribution of development times ",
"(N= ", length(x),")\n", "T dimidiata Female & Male 5th instar nymphs"),
xlab = "Development time (days)", ylab = "Density")
xseq <-seq(from = min(x), to = max(x), length = 100)
lines(xseq, dnorm(xseq, mean(x), sd(x)), lwd = 2)
rug(jitter(x), col = "red")
```



OK, on continue :

```
qqnorm(x)
qqline(x)
```



Il y a bien une queue de distribution anormalement épaisse à droite par rapport à ce qui serait attendu sous une distribution normale. Comme le nombre de données n'est pas très important on fait un petit test de normalité pour se donner bonne conscience :

```
shapiro.test(x)
```

```
        Shapiro-Wilk normality test
data:  x
W = 0.96551, p-value = 0.001392
```

Pas de problème, on rejette l'hypothèse d'une distribution normale des données avec un risque de première espèce $\alpha = 0.05$. Allons y pour la partie qui fâche quand on essaye d'estimer un mélange de lois normales :

```
resnlm <-try(nlm(f = logvraineg, p = c(1, 290, 5, 380, 5), obs = x) )
resnlm
```

```
[1] "Error in nlm(f = logvraineg, p = c(1, 290, 5, 380, 5), obs = x) : \n  valeur non finie fournie par 'nlm'\n
attr(,"class")
[1] "try-error"
attr(,"condition")
<simpleError in nlm(f = logvraineg, p = c(1, 290, 5, 380, 5), obs = x): valeur non finie fournie par 'nlm'>
```

Donc `nlm()` est partie en vrille, essayons de comprendre pourquoi en augmentant le niveau du détail d'affichage :

```
resnlm <-try(nlm(f = logvraineg, p = c(1, 290, 5, 380, 5), obs = x, print.level = 1 ))
```

```
iteration = 0
Step:
[1] 0 0 0 0 0
Parameter:
[1]   1 290   5 380   5
Function Value
[1] 5487.064
Gradient:
[1]         Inf    72.04081 -2026.44493     0.00000     0.00000
```

Nous avons donc un gradient infini sur le premier paramètre, difficile de converger dans ces conditions. Le premier paramètre représente la proportion des individus appartenant à la première population. Comme condition initiale on lui dit de considérer que tous les individus appartiennent à la première population. Ce n'est pas une très bonne condition initiale pour estimer un mélange que de considérer au départ qu'il n'y a pas de mélange. Essayons de trouver une meilleure condition initiale pour ce paramètre.

```
 table(x)
```

```
x
202 216 226 235 246 256 266 276 286 296 308 319 339 372
  5   5   9   9   7  16  16  10  16  13   9   8   8   8
```

Supposons que les 16 individus à 339 et 372 soient dans la deuxième population, et on essaye avec cette nouvelle condition initiale :

```
 (p10 <- (length(x)-16)/length(x))
```

```
[1] 0.8848921
```

```
 resnlm <-try(nlm(f = logvraineg, p = c(p10, 290, 5, 380, 5), obs = x, print.level = 1 ))
```

```
iteration = 0
Step:
[1] 0 0 0 0 0
Parameter:
[1]   0.8848921 290.0000000   5.0000000 380.0000000   5.0000000
Function Value
[1] 4355.896
Gradient:
[1]  3.465175e-04  1.139606e+02 -1.445646e+03  1.568007e+01 -1.084794e+02
iteration = 100
Parameter:
[1] -7369.0979   -64.9410  4493.7059   294.6543   292.7827
Function Value
[1] -309.5856
Gradient:
[1]  0.0188598231  0.0001654832 -0.0021630570  0.0305902996  0.4963424767

Limite d'itérations dépassé. L'algorithme a échoué.
```

C'est mieux, on ne plante pas tout de suite, mais le nombre d'itérations maximum autorisé est dépassé. Essayons de préciser les conditions initiales des autres paramètres, toujours sous l'hypothèse que les 16 individus à 339 et 372 soient dans la deuxième population.

```
 x1 <- x[x < 339]
 (p20 <- mean(x1))
```

```
[1] 266.8374
```

```
 (p30 <- sd(x1))
```

```
[1] 31.07246
```

```
 x2 <- x[x >= 339]
 (p40 <- mean(x2))
```

```
[1] 355.5
```

```
 (p50 <- sd(x2))
```

```
[1] 17.04113
```

```
 resnlm <-try(nlm(f = logvraineg, p = c(p10, p20, p30, p40, p50), obs = x, print.level = 1 ))
```

```
iteration = 0
Step:
[1] 0 0 0 0 0
Parameter:
[1]   0.8848921 266.8373984  31.0724626 355.5000000  17.0411267
Function Value
[1] 709.8682
Gradient:
[1] -18.07267847  -0.16211853  -0.33470976  -0.02299345  -0.15091918
iteration = 100
Parameter:
```

```
[1] 9.256639e-01 2.634061e+02 3.440259e+01 3.720005e+02 1.987333e-04
Function Value
[1] 647.4915
Gradient:
[1] -3.390106e+01 -8.675596e-01 -2.434263e-01  1.358614e+05 -1.980949e+05

Limite d'itérations dépassé. L'algorithme a échoué.
```

```
 resnlm
```

```
$minimum
[1] 647.4915
$estimate
[1] 9.256639e-01 2.634061e+02 3.440259e+01 3.720005e+02 1.987333e-04

$gradient
[1] -3.390106e+01 -8.675596e-01 -2.434263e-01  1.358614e+05 -1.980949e+05

$code
[1] 4

$iterations
[1] 100
```

RIEN À FAIRE, `nlm()` part dans les choux, même avec les meilleures conditions initialles possibles. On décide de ruser en bruitant les données pour éviter de passer par des écart-types nuls dans le deuxième groupe.

```
 xx <- jitter(x)
 xx1 <- xx[xx < 339]
 (p20 <- mean(xx1))
```

```
[1] 269.7109
```

```
 (p30 <- sd(xx1))
```

```
[1] 33.46842
```

```
 xx2 <- xx[xx >= 339]
 (p40 <- mean(xx2))
```

```
[1] 363.022
```

```
 (p50 <- sd(xx2))
```

```
[1] 14.76728
```

```
 resnlm <-try(nlm(f = logvraineg, p = c(p10, p20, p30, p40, p50), obs = xx, print.level = 1 ))
```

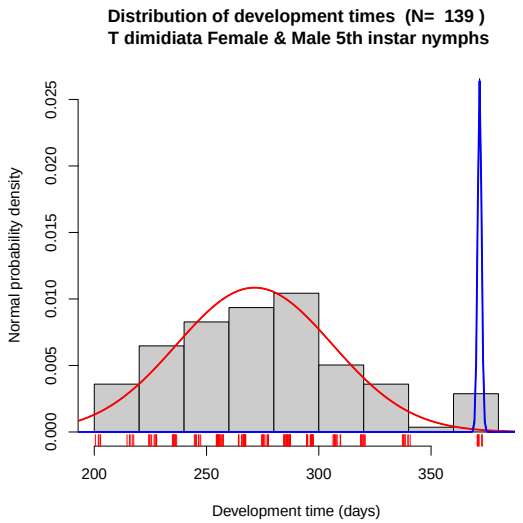
```
iteration = 0
Step:
[1] 0 0 0 0 0
Parameter:
[1]   0.8848921 269.7109413  33.4684225 363.0219644  14.7672825
Function Value
[1] 709.5889
Gradient:
[1] -49.86143074  -0.01859968  -0.04218489   0.06188414  -0.06638605
iteration = 72
Parameter:
[1]   0.9430604 271.3873864  34.6766402 371.6306917   0.8607639
Function Value
[1] 690.7799
Gradient:
[1] -6.195933e-05  1.998200e-06 -8.065073e-07 -5.362663e-07  5.911716e-06

Gradient relatif proche de zéro.
L'itération courante est probablement la solution.
```

```
 resnlm
```

```
$minimum
[1] 690.7799
$estimate
[1]   0.9430604 271.3873864  34.6766402 371.6306917   0.8607639

$gradient
[1] -6.195933e-05  1.998200e-06 -8.065073e-07 -5.362663e-07  5.911716e-06

$code
[1] 1

$iterations
[1] 72
```

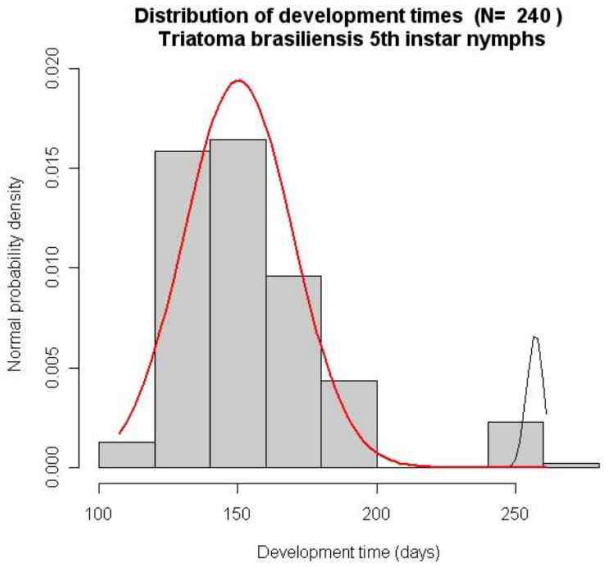
Voyons le résultat :

```
xseq <-seq(0.9*min(xx), 1.1*max(xx), le = 300)
est <-resnlm$estimate
y1 <-est[1] * dnorm(xseq, est[2], est[3])
y2 <-(1 -est[1]) * dnorm(xseq, est[4], est[5])
hist(xx, proba = T, ylim = c(0, max(y1 + y2)), col = grey(0.8),
main = paste("Distribution of development times ","(N= ", length(x),")\n",
"T dimidiata Female & Male 5th instar nymphs"),
xlab = "Development time (days)", ylab = "Normal probability density")
lines(xseq, y1 + y2)
lines(xseq, y1, col = "red", lwd = 2)
lines(xseq, y2, col = "blue", lwd = 2)
rug(xx,col="red")
```



Voici donc l'explication du problème de convergence. Avec un mélange de deux lois normales la meilleure solution consiste à regrouper dans la deuxième population tous les individus ayant une valeur de 372. Dans les données originelles il n'y a plus aucune variabilité intra-population, l'écart type vaut 0 et donc la fonction de densité de probabilité tend vers $+\infty$, d'où les `NaN` produits. En bruitant les données on a résolu le problème de convergence mais l'écart-type estimé de la deuxième population n'a plus aucun sens.

Par rapport à *Triatoma brasiliensis* il y a une grosse différence, en effet dans cette dernière espèce la variabilité intra-populationnelle est bien documentée :



HTH Jorge, as we say in french.

References

[1] J.R. Lobry, L. Rosso, and J.-P. Flandrois. A FORTRAN subroutine for the determination of parameter confidence limits in non-linear models. *Binary*, 3:86–93, 1991.

[2] J. Rabinovich, S. Pietrokovsky, and C. Wisnivesky-Colli. Temperature and development rate of *Triatoma guasayana* (Hemiptera: Reduviidae) eggs under laboratory conditions: physiological and adaptive aspects. *Physiological Entomology*, 31:361–370, 2006.