Minimiser une fonction de 2 variables

- $(x, y) \mapsto f(x, y) = 2y^2 - x(x-1)^2$
  > f = function(x) 2*x[2]^2-x[1]*(x[1]-1)^2

Minimiser une fonction de 2 variables

- $(x, y) \mapsto f(x, y) = 2y^2 - x(x-1)^2$
  > f = function(x) 2*x[2]^2-x[1]*(x[1]-1)^2
- $\partial f/\partial x = \partial f/\partial y = 0 \Longrightarrow$ points critiques: $(1/3, 0)$, $(1, 0)$
  $(1/3, 0)$: minimum local $\quad f(1/3, 0) = -4/27 = -0.148148$

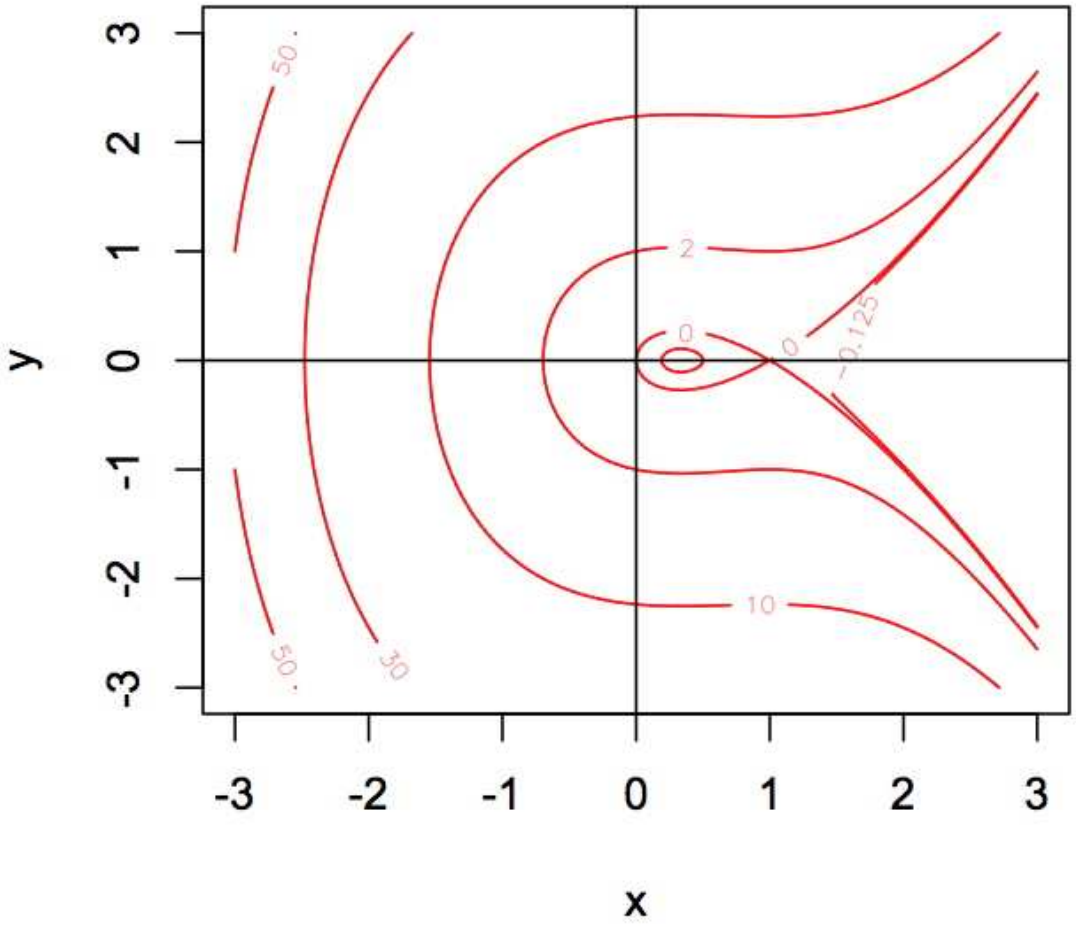
Minimiser une fonction de 2 variables

- $(x, y) \mapsto f(x, y) = 2y^2 - x(x-1)^2$
  > f = function(x) 2*x[2]^2-x[1]*(x[1]-1)^2
- $\partial f/\partial x = \partial f/\partial y = 0 \Longrightarrow$ points critiques: $(1/3, 0)$, $(1, 0)$
  $(1/3, 0)$: minimum local $f(1/3, 0) = -4/27 = -0.148148$
- Contours dans le carré $[-3, 3] \times [-3, 3]$
  > a = seq(-3,3,length=200); b = a #grille
  > v = matrix(numeric(200^2),nrow=200)
  > for (i in 1:200) {for (j in 1:200) v[i,j] = f(c(a[i],b[j]))}
  > contour(a,b,v,levels=c(50,30,10,2,0,-.125),xlab='x',ylab='y',col='red')

Minimiser une fonction de 2 variables

- $(x, y) \mapsto f(x, y) = 2y^2 - x(x-1)^2$
  > f = function(x) 2*x[2]^2-x[1]*(x[1]-1)^2
- $\partial f/\partial x = \partial f/\partial y = 0 \Longrightarrow$ points critiques: $(1/3, 0)$, $(1, 0)$
  $(1/3, 0)$: minimum local $f(1/3, 0) = -4/27 = -0.148148$
- Contours dans le carré $[-3, 3] \times [-3, 3]$
  > a = seq(-3,3,length=200); b = a #grille
  > v = matrix(numeric(200^2),nrow=200)
  > for (i in 1:200) {for (j in 1:200) v[i,j] = f(c(a[i],b[j]))}
  > contour(a,b,v,levels=c(50,30,10,2,0,-.125),xlab='x',ylab='y',col='red')

Contours d'une fonction de 2 var.



Fonction optim

- Par défaut, calcule un point minimum selon la méthode de Nelder-Mead (simplexe)

Fonction optim

- Par défaut, calcule un point minimum selon la méthode de Nelder-Mead (simplexe)
- optim(par, fn, gr = NULL, ..., method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN"), lower = -Inf, upper = Inf, control = list(), hessian = FALSE)

Fonction optim

- Par défaut, calcule un point minimum selon la méthode de Nelder-Mead (simplexe)
- optim(par, fn, gr = NULL, ..., method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN"), lower = -Inf, upper = Inf, control = list(), hessian = FALSE)
- `par` valeur initiale `fn` fonction (argument x vectoriel)

Fonction optim

- Par défaut, calcule un point minimum selon la méthode de Nelder-Mead (simplexe)
- optim(par, fn, gr = NULL, ..., method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN"), lower = -Inf, upper = Inf, control = list(), hessian = FALSE)
- `par` valeur initiale `fn` fonction (argument x vectoriel)
- `BFGS` : méthode de type quasi-Newton vue en classe
  `hessian = T`: calcule la matrice hessienne en $\widehat{\theta}$
  $\Longrightarrow l''(\widehat{\theta})$ si `fn` $= l$ une log-vraisemblance.
  Utile pour obtenir une estimation de la variabilité des estimateurs de vraisemblance maximale: matrice de covariance asymptotique : $\Longrightarrow (I(\widehat{\theta}))^{-1} \approx (-l''(\widehat{\theta}))^{-1}$.

Fonction optim

- Par défaut, calcule un point minimum selon la méthode de Nelder-Mead (simplexe)
- optim(par, fn, gr = NULL, ..., method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN"), lower = -Inf, upper = Inf, control = list(), hessian = FALSE)
- `par` valeur initiale `fn` fonction (argument x vectoriel)
- `BFGS` : méthode de type quasi-Newton vue en classe
  `hessian = T`: calcule la matrice hessienne en $\widehat{\theta}$ $\Longrightarrow l''(\widehat{\theta})$ si `fn` $= l$ une log-vraisemblance.
  Utile pour obtenir une estimation de la variabilité des estimateurs de vraisemblance maximale: matrice de covariance asymptotique : $\Longrightarrow (I(\widehat{\theta}))^{-1} \approx (-l''(\widehat{\theta}))^{-1}$.
- `control` : permet de choisir les facteurs α, β, γ de Nelder-Mead, le n. d'itérations, la tolérance, etc.

Fonction nlm

- Calcule un point minimum selon une méthode de type Newton.

Fonction nlm

- Calcule un point minimum selon une méthode de type Newton.
- nlm(f, p, hessian = FALSE, typsize=rep(1, length(p)), fscale=1, print.level = 0, ndigit=12, gradtol = 1e-6, steptol = 1e-6, iterlim = 100, ...)

Fonction nlm

- Calcule un point minimum selon une méthode de type Newton.
- nlm(f, p, hessian = FALSE, typsize=rep(1, length(p)), fscale=1, print.level = 0, ndigit=12, gradtol = 1e-6, steptol = 1e-6, iterlim = 100, ...)
- Par défaut utilise des dérivées numériques.

Fonction nlm

- Calcule un point minimum selon une méthode de type Newton.
- nlm(f, p, hessian = FALSE, typsize=rep(1, length(p)), fscale=1, print.level = 0, ndigit=12, gradtol = 1e-6, steptol = 1e-6, iterlim = 100, ...)
- Par défaut utilise des dérivées numériques.
- Produit une liste contenant la valeur minimale de la fonction, le point minimum, le gradient au point minimum ainsi qu'une évaluation de la qualité de l'itération (de 1 à 5). Produit aussi sur demande la matrice hessienne au point minimum: `hessian = T`.

Fonction nlm

- Calcule un point minimum selon une méthode de type Newton.
- nlm(f, p, hessian = FALSE, typsize=rep(1, length(p)), fscale=1, print.level = 0, ndigit=12, gradtol = 1e-6, steptol = 1e-6, iterlim = 100, ...)
- Par défaut utilise des dérivées numériques.
- Produit une liste contenant la valeur minimale de la fonction, le point minimum, le gradient au point minimum ainsi qu'une évaluation de la qualité de l'itération (de 1 à 5). Produit aussi sur demande la matrice hessienne au point minimum: `hessian = T`.
- Fonction moins polyvalente qu'`optim`.

`optim` et `nlm` appliquées

- > optim(c(0,0),f)# par défaut Nelder-Mead, minimisation
  $par # fonction f de la p. 1
  [1] 3.333333e-01 4.934999e-09 $\approx (1/3, 0)$
  $value
  [1] -0.1481481 $\approx f(1/3, 0) = -4/27$

`optim` et `nlm` appliquées

- > optim(c(0,0),f)# par défaut Nelder-Mead, minimisation
  $par # fonction f de la p. 1
  [1] 3.333333e-01 4.934999e-09 $\approx (1/3, 0)$
  $value
  [1] -0.1481481 $\approx f(1/3, 0) = -4/27$
- > nlm(f,c(0,0))#par défaut type Newton, minimisation
  $minimum
  [1] -0.1481481
  $estimate
  [1] 3.333324e-01 -4.558233e-07

`optim` et `nlm` appliquées

- > optim(c(0,0),f)# par défaut Nelder-Mead, minimisation
  $par # fonction f de la p. 1
  [1] 3.333333e-01 4.934999e-09 $\approx (1/3, 0)$
  $value
  [1] -0.1481481 $\approx f(1/3, 0) = -4/27$
- > nlm(f,c(0,0))#par défaut type Newton, minimisation
  $minimum
  [1] -0.1481481
  $estimate
  [1] 3.333324e-01 -4.558233e-07
- > optim(c(2,0),f) #autre valeur initiale
  $par
  [1] 3.776441e+55 -5.926356e+54 # $\Rightarrow$ divergence!
  $value [1] -5.385772e+166

Vraisemblance à 2 paramètres

- Modèle de Weibull à paramètres $\theta, \alpha > 0$

$$f(y|\theta,\alpha) = \frac{\alpha}{\theta}\left(\frac{y}{\theta}\right)^{\alpha-1} \exp\left[-\left(\frac{y}{\theta}\right)^{\alpha}\right], \quad y > 0$$

Vraisemblance à 2 paramètres

- Modèle de Weibull à paramètres $\theta, \alpha > 0$

$$f(y|\theta,\alpha) = \frac{\alpha}{\theta}\left(\frac{y}{\theta}\right)^{\alpha-1}\exp\left[-\left(\frac{y}{\theta}\right)^{\alpha}\right], \quad y > 0$$

- Log-vraisemblance p. r. aux observations $y_1, \ldots, y_n$

$$l(\theta,\alpha) = -n\log\theta + n\log\alpha + (\alpha-1)\sum_1^n \log\left(\frac{y_i}{\theta}\right) - \sum_1^n \left(\frac{y_i}{\theta}\right)^{\alpha}$$

Vraisemblance à 2 paramètres

- Modèle de Weibull à paramètres $\theta, \alpha > 0$

$$f(y|\theta,\alpha) = \frac{\alpha}{\theta}\left(\frac{y}{\theta}\right)^{\alpha-1}\exp\left[-\left(\frac{y}{\theta}\right)^{\alpha}\right], \quad y > 0$$

- Log-vraisemblance p. r. aux observations $y_1, \ldots, y_n$

$$l(\theta,\alpha) = -n\log\theta + n\log\alpha + (\alpha-1)\sum_1^n \log\left(\frac{y_i}{\theta}\right) - \sum_1^n \left(\frac{y_i}{\theta}\right)^{\alpha}$$

- Fonction score

$$l'(\theta,\alpha)^T = \begin{pmatrix} -n\alpha/\theta + \alpha\theta^{-1}\sum(y_i/\theta)^{\alpha} \\ n/\alpha + \sum\log(y_i/\theta) - \sum(y_i/\theta)^{\alpha}\log(y_i/\theta) \end{pmatrix}$$

Vraisemblance à 2 paramètres (suite)

- Matrice hessienne $= l''(\theta, \alpha) = -$ information observée

$$l_{\theta\theta} = \alpha(\alpha+1)/\theta^2 \sum (y_i/\theta)^\alpha - n\alpha\theta^{-2}$$

$$l_{\theta\alpha} = \theta^{-1} \sum [1 - (y_i/\theta)^\alpha \{1 + \alpha \log(y_i/\theta)\}]$$

$$l_{\alpha\alpha} = n\alpha^2 + \sum (y_i/\theta)^\alpha \{\log(y_i/\theta)\}^2$$

Vraisemblance à 2 paramètres (suite)

- Matrice hessienne $= l''(\theta, \alpha) =$ − information observée

$$l_{\theta\theta} = \alpha(\alpha+1)/\theta^2 \sum (y_i/\theta)^\alpha - n\alpha\theta^{-2}$$

$$l_{\theta\alpha} = \theta^{-1} \sum [1 - (y_i/\theta)^\alpha \{1 + \alpha \log(y_i/\theta)\}]$$

$$l_{\alpha\alpha} = n\alpha^2 + \sum (y_i/\theta)^\alpha \{\log(y_i/\theta)\}^2$$

- $-l(\theta, \alpha)$ #on change le signe pour minimiser!

```
> lneg = function(p,x){ e =
log(p[2])-log(p[1])+(p[2]-1)*log(x/p[1])-(x/p[1])^p[2]
-sum(e)}   #p vecteur: (p[1],p[2]) = (θ, α)
```

Vraisemblance à 2 paramètres (suite)

- Matrice hessienne $= l''(\theta, \alpha) = -$ information observée

$$l_{\theta\theta} = \alpha(\alpha+1)/\theta^2 \sum (y_i/\theta)^\alpha - n\alpha\theta^{-2}$$

$$l_{\theta\alpha} = \theta^{-1} \sum [1 - (y_i/\theta)^\alpha \{1 + \alpha \log(y_i/\theta)\}]$$

$$l_{\alpha\alpha} = n\alpha^2 + \sum (y_i/\theta)^\alpha \{\log(y_i/\theta)\}^2$$

- $-l(\theta, \alpha)$ #on change le signe pour minimiser!
  > lneg = function(p,x){ e = log(p[2])-log(p[1])+(p[2]-1)*log(x/p[1])-(x/p[1])^p[2] -sum(e)} #p vecteur: (p[1],p[2]) = (θ, α)
- Premier objectif: identifier la région du point maximum avec le graphique des contours.

Contours de la vraisemblance

- Temps de panne
  # t0 = c(225,171,198,189,189,135,162,135,117,162)
  > x = seq(1,15,length=200) #grille des α
  > y = seq(100,250,length=200) #grille des θ

Contours de la vraisemblance

- Temps de panne
  # t0 = c(225,171,198,189,189,135,162,135,117,162)
  > x = seq(1,15,length=200) #grille des α
  > y = seq(100,250,length=200) #grille des θ
- Valeurs de la log-vraisemblance $l(\theta, \alpha)$ sur la grille
  > logv = matrix(numeric(200^2),nrow=200)
  > for (i in 1:200) {for (j in 1:200) logv[i,j] =
  -lneg(c(y[i],x[j]),t0)} # t0: temps de panne

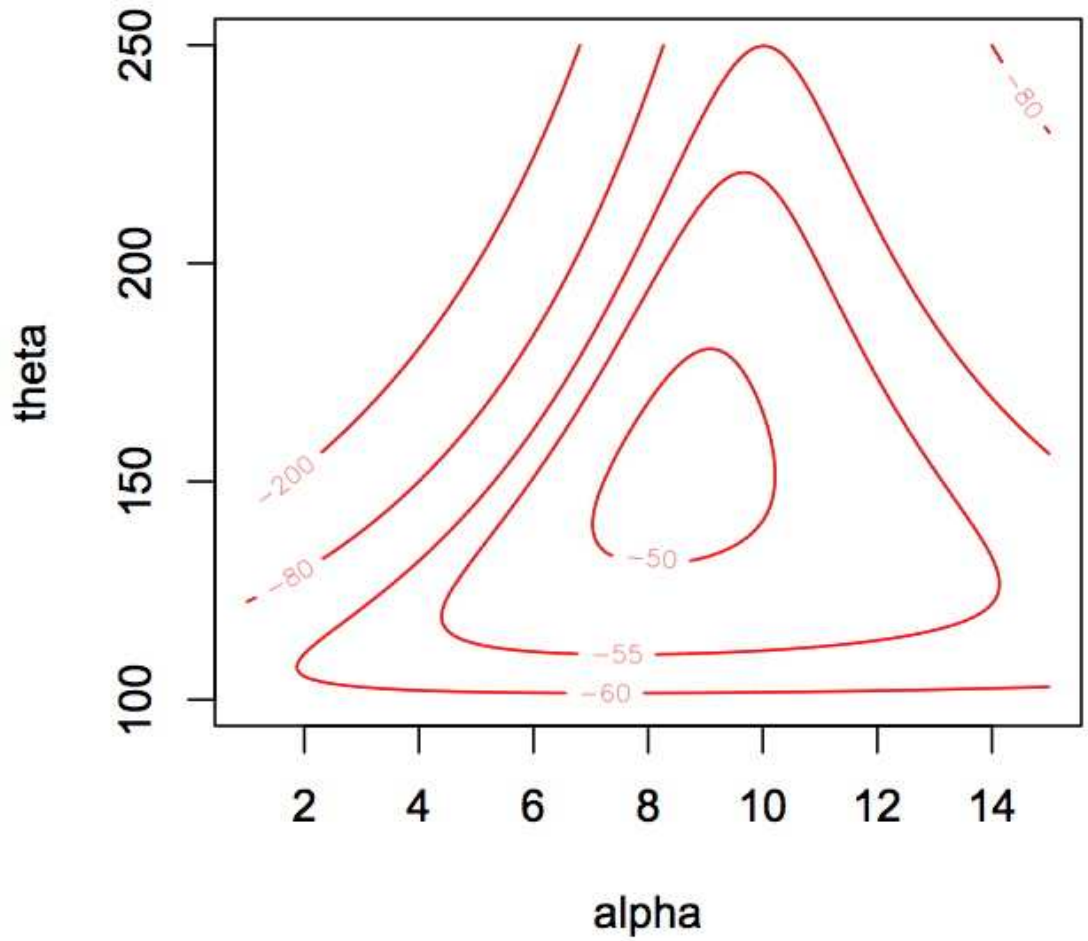
Contours de la vraisemblance

- Temps de panne
  # t0 = c(225,171,198,189,189,135,162,135,117,162)
  > x = seq(1,15,length=200) #grille des α
  > y = seq(100,250,length=200) #grille des θ
- Valeurs de la log-vraisemblance $l(\theta, \alpha)$ sur la grille
  > logv = matrix(numeric(200^2),nrow=200)
  > for (i in 1:200) {for (j in 1:200) logv[i,j] =
  -lneg(c(y[i],x[j]),t0)} # t0: temps de panne
- Minimum et maximum des valeurs de $l(\theta, \alpha)$ sur la grille
  > range(logv) #intervalle des valeurs
  [1] -254028.28687 -48.75647

Contours de la vraisemblance

- Temps de panne
  # t0 = c(225,171,198,189,189,135,162,135,117,162)
  > x = seq(1,15,length=200) #grille des α
  > y = seq(100,250,length=200) #grille des θ
- Valeurs de la log-vraisemblance $l(\theta, \alpha)$ sur la grille
  > logv = matrix(numeric(200^2),nrow=200)
  > for (i in 1:200) {for (j in 1:200) logv[i,j] =
  -lneg(c(y[i],x[j]),t0)} # t0: temps de panne
- Minimum et maximum des valeurs de $l(\theta, \alpha)$ sur la grille
  > range(logv) #intervalle des valeurs
  [1] -254028.28687 -48.75647
- Contours
  > contour(x,y,logv,levels=c(-50,-55,-60,-80,-
  200),xlab='alpha',ylab='theta',col='red')

Contours de la log-vraisemblance



Estimation

- Fonction `optim`: (Nelder-Mead)
  > p0 = c(165,1)
  #valeur initiale (modèle exponentiel $\alpha = 1$)
  > optim(p0,lneg,x=t0)
  $par
  [1] 181.409455 5.976163q $(\widehat{\theta}, \widehat{\alpha})$
  $value #valeur minimale de `lneg`
  [1] 48.75639 $-l(\widehat{\theta}, \widehat{\alpha})$

Estimation

- Fonction `optim`: (Nelder-Mead)
  > p0 = c(165,1)
  #valeur initiale (modèle exponentiel $\alpha = 1$)
  > optim(p0,lneg,x=t0)
  $par
  [1] 181.409455 5.976163q $(\widehat{\theta}, \widehat{\alpha})$
  $value #valeur minimale de `lneg`
  [1] 48.75639 $-l(\widehat{\theta}, \widehat{\alpha})$
- Méthode BFGS (quasi-Newton)
  > optim(p0,lneg,x=t0,method='BFGS')
  $par
  [1] 181.44731 5.97875
  $value
  [1] 48.75639

Estimation (suite)

- Calcul de l'information observée $= -l''(\widehat{\theta}, \widehat{\alpha})$ (= estimation de $I(\widehat{\theta}, \widehat{\alpha})$).

```
> optim(p0,lneg,x=t0,method='BFGS',hessian=TRUE)
$hessian
                [,1]        [,2]
[1,] 0.01084155 -0.02432539
[2,] -0.02432539 0.52368771
```

Estimation (suite)

- Calcul de l'information observée $= -l''(\widehat{\theta}, \widehat{\alpha})$ (= estimation de $I(\widehat{\theta}, \widehat{\alpha})$).
  > optim(p0,lneg,x=t0,method='BFGS',hessian=TRUE)
  $hessian
  [,1] [,2]
  [1,] 0.01084155 -0.02432539
  [2,] -0.02432539 0.52368771

- $\widehat{I(\widehat{\theta}, \widehat{\alpha})^{-1}} = (-l''(\widehat{\theta}, \widehat{\alpha}))^{-1}$: estimation de la covariance asymptotique de $(\widehat{\theta}, \widehat{\alpha})$
  > solve(optim(p0,lneg,x=t0,method='BFGS',
  hessian=TRUE)$hessian)
  [,1] [,2]
  [1,] 102.969336 4.782945 $\widehat{\text{Var}(\widehat{\theta})} = 102.97$, se $= 10.15$
  = [2,] 4.782945 2.131704 $\widehat{\text{Var}(\widehat{\alpha})} = 2.13$, se $= 1.46$

Fonction `fitdistr` du package MASS

- fitdistr(x, densfun, start, ...) #ajustement par EMV
  x : vecteur de données univariées
  `densfun` : densité ou fonction de masse = ”beta”, ”cauchy”, ”chi-squared”, etc. (principales lois continues ou discrètes). Peut aussi être définie par l’usager.
  `start` : vecteur des valeurs initiales sous forme d’une liste. Pas toujours nécessaire. Voir les exemples.

Fonction `fitdistr` du package MASS

- fitdistr(x, densfun, start, ...) #ajustement par EMV
  x : vecteur de données univariées
  `densfun` : densité ou fonction de masse = ”beta”, ”cauchy”, ”chi-squared”, etc. (principales lois continues ou discrètes). Peut aussi être définie par l’usager.
  `start` : vecteur des valeurs initiales sous forme d’une liste. Pas toujours nécessaire. Voir les exemples.
- Utilise `optim` avec les méthodes de Nelder-Mead en univarié et BFGS en multivarié.

Fonction `fitdistr` du package MASS

- fitdistr(x, densfun, start, ...) #ajustement par EMV
  x : vecteur de données univariées
  `densfun` : densité ou fonction de masse = ”beta”, ”cauchy”, ”chi-squared”, etc. (principales lois continues ou discrètes). Peut aussi être définie par l’usager.
  `start` : vecteur des valeurs initiales sous forme d’une liste. Pas toujours nécessaire. Voir les exemples.
- Utilise `optim` avec les méthodes de Nelder-Mead en univarié et BFGS en multivarié.
- Produit une liste de longueur 3
  $estimate: estimateurs des paramètres
  $sd: écarts types estimés des estimateurs (calculés à partir de l’information observée)
  $loglik: valeur maximale de la log-vraisemblance

Applications de `fitdistr`

- > library(MASS)

Applications de `fitdistr`

- > library(MASS)
- > t0 #temps de panne
  [1] 225 171 198 189 189 135 162 135 117 162
  > fitdistr(t0,”weibull”)
  shape scale
  5.979282 181.452512
  (1.460124) (10.146980)

Applications de `fitdistr`

- > library(MASS)
- > t0 #temps de panne
  [1] 225 171 198 189 189 135 162 135 117 162
  > fitdistr(t0,”weibull”)
  shape scale
  5.979282 181.452512
  (1.460124) (10.146980)
- > set.seed(123)
  > x = rgamma(100, shape = 5, rate = 0.1)
  > fitdistr(x, "gamma")
  shape rate
  6.45947303 0.13593172
  (0.89052010) (0.01948648) # écarts types estimés

Applications de `fitdistr`

- > set.seed(123)
  > x = rgamma(10000, shape = 5, rate = 0.1)
  > fitdistr(x, dgamma, list(shape = 1, rate = 0.1))
  shape rate
  5.061290962 0.101984691
  (0.069304895) (0.001467997)

Applications de `fitdistr`

- > set.seed(123)
  > x = rgamma(10000, shape = 5, rate = 0.1)
  > fitdistr(x, dgamma, list(shape = 1, rate = 0.1))
  shape rate
  5.061290962 0.101984691
  (0.069304895) (0.001467997)
- > set.seed(123)
  > x4 = rnegbin(500, mu = 5, theta = 4)
  > fitdistr(x4, "Negative Binomial")
  size mu
  4.2159071 4.9447685
  (0.5043658) (0.1466082)

Applications de `fitdistr`

- > set.seed(123)
  > x = rgamma(10000, shape = 5, rate = 0.1)
  > fitdistr(x, dgamma, list(shape = 1, rate = 0.1))
  shape rate
  5.061290962 0.101984691
  (0.069304895) (0.001467997)
- > set.seed(123)
  > x4 = rnegbin(500, mu = 5, theta = 4)
  > fitdistr(x4, "Negative Binomial")
  size mu
  4.2159071 4.9447685
  (0.5043658) (0.1466082)
- Selon les auteurs de MASS, `fitdistr` à utiliser avec prudence s'il y a des observations aberrantes.