Fiche TD avec le logiciel R : tdr46

Régression non linéaire

P^r Jean R. Lobry

Contents

1 Une petite simulation **2**

2 Estimation d'un paramètre **3**

3 nlm (non-linear minimization) **6**

4 Le piège des minimums locaux **7**

5 Le piège des mesas **14**

6 Un cas pratique : le modèle CTMI **18**

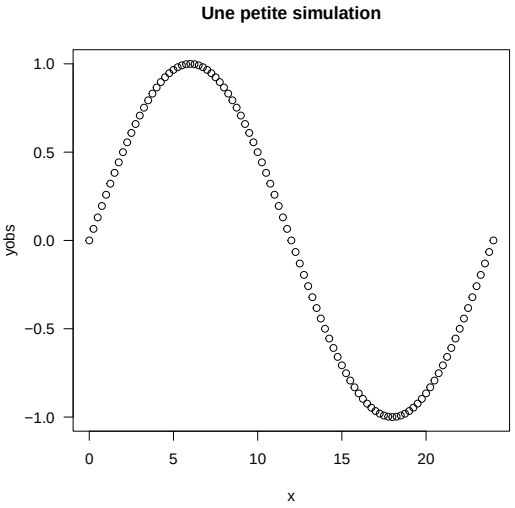
7 Régions de confiance **20**

Références **22**

1 Une petite simulation

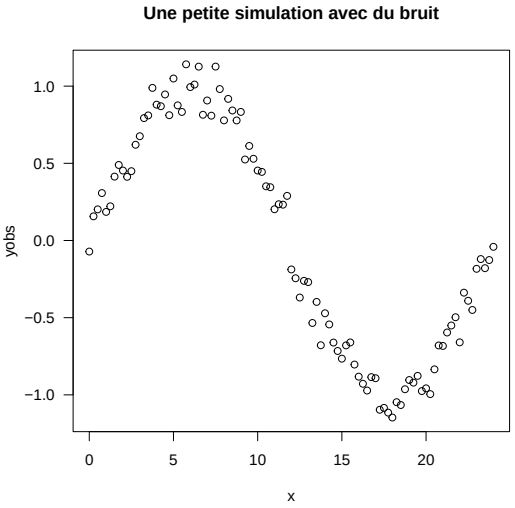
Simulons un rythme circadien où `x` représente le temps exprimé en heures et `yobs` une variable quantitative variant cycliquement avec une période de 24 heures, et mesurée toutes les 15 minutes :

```
x <- seq(from = 0, to = 24, by = 0.25)
yobs <- sin(2*pi*x/24)
plot(x = x, y = yobs, main = "Une petite simulation", las = 1)
```



Ajoutons un peu de bruit gaussien pour être plus réaliste :

```
yobs <- yobs + rnorm( n = length(yobs), mean = 0, sd = 0.1)
plot(x = x, y = yobs, main = "Une petite simulation avec du bruit", las = 1)
```



Pour la reproductibilité des résultats, il serait utile de sauvegarder une simulation pour sa réutilisation ultérieure. Cette opération a déjà été effectuée pour vous avec les instructions suivantes (pour information seulement) :

```
circa <- as.data.frame(cbind(x,yobs))
write.table(circa, file = "circa.txt")
```

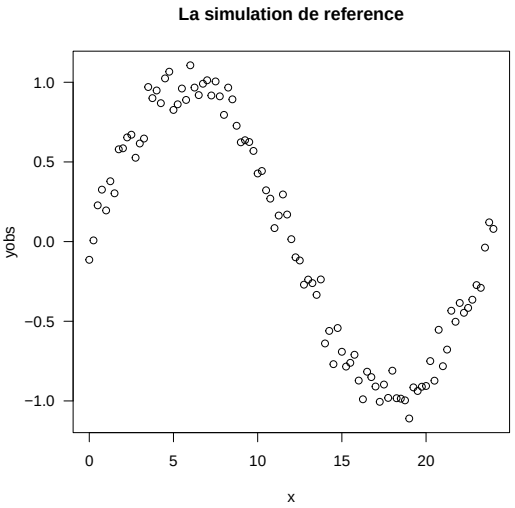
2 Estimation d'un paramètre

À partir de maintenant nous travaillerons avec une simulation de référence :

```
circa <- read.table("http://pbil.univ-lyon1.fr/R/donnees/circa.txt")
x <- circa$x
yobs <- circa$yobs
summary(circa)
```

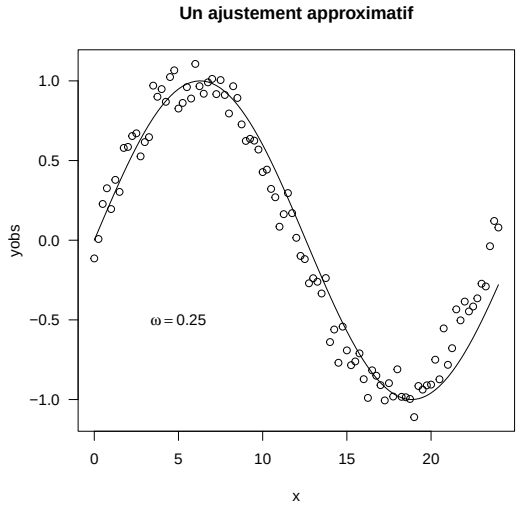
```
       x           yobs          
 Min.   : 0   Min.   :-1.11034  
 1st Qu.: 6   1st Qu.:-0.69152  
 Median :12   Median : 0.01517  
 Mean   :12   Mean   : 0.01333  
 3rd Qu.:18   3rd Qu.: 0.64676  
 Max.   :24   Max.   : 1.10648  
```

```
plot(circa, main = "La simulation de reference", las = 1)
```



Supposons que ce soient de véritables observations et que nous cherchions à ajuster un modèle à un seul paramètre $y = sin(\omega x)$. Pour simplifier, l'amplitude est connue et égale à 1. Reste à trouver ω. Essayons approximativement.

```
omega <- 0.25
ytheo <- sin(omega*x)
plot(circa, main = "Un ajustement approximatif", las = 1)
text(x = 5, y = -0.5, labels = bquote(omega == .(omega)))
lines(x, ytheo)
```



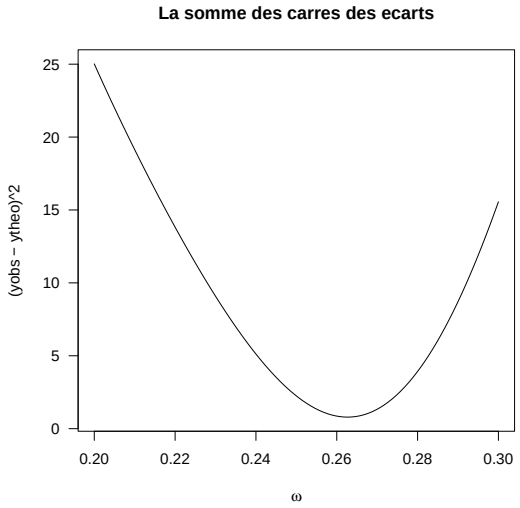
Ce n'est pas si mal, mais pour faire mieux il va falloir se donner un critère objectif d'ajustement entre les données et le modèle. Prenons par exemple le critère des moindres carrés, et cherchons la valeur de ω qui le minimise. Avec notre estimation grossière $\omega = 0.25$, nous avons une somme des carrés des écarts égale à :

```
sum( (yobs - ytheo)^2 )
```

```
[1] 2.245973
```

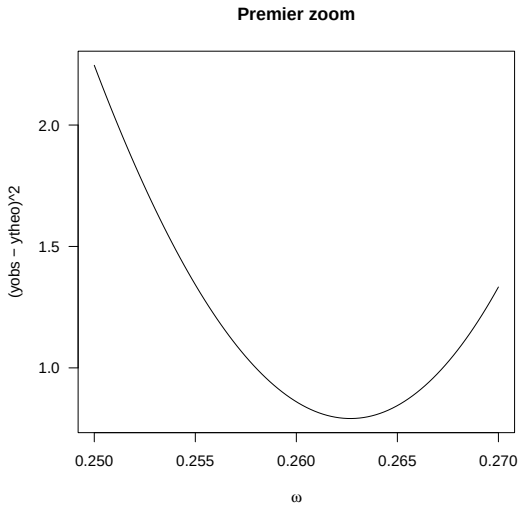
Il faut donc essayer de faire mieux que 2.246, c'est-à-dire trouver une valeur de ω telle que la somme des carrés des résidus soit plus faible. Définissons une fonction qui nous calcule la somme des carrés des écarts pour une valeur de ω donnée, et représentons la au voisinage de 0.25.

```
sce <- function(param, x, yobs)
{
  ytheo <- sin(param[1]*x)
  return( sum( (yobs-ytheo)^2) )
}
omegax <- seq(from = 0.2, to = 0.3, length = 100)
plot( x = omegax, y = sapply(omegax, function(omega) { sce(omega, x, yobs) } ),
  type = "l", xlab = expression(omega), ylab = "(yobs - ytheo)^2",
  main = "La somme des carres des ecarts" , las = 1)
```

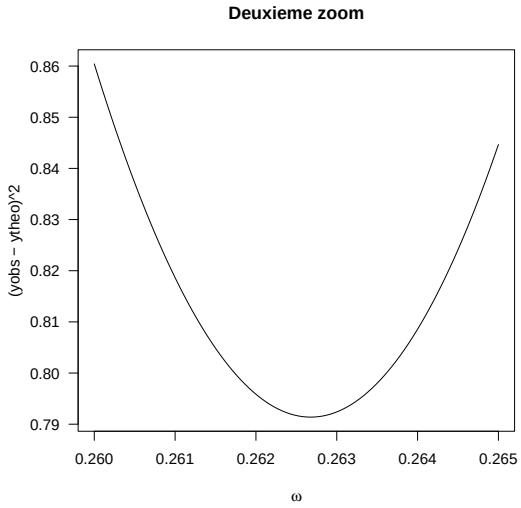


La valeur du paramètre qui nous intéresse se trouve donc au niveau du minimum. On pourrait envisager de préciser sa valeur en zoomant progressivement :

```
omegax <- seq(from = 0.25, to = 0.27, length = 100)
plot( x = omegax, y = sapply(omegax, function(omega) { sce(omega, x, yobs) } ),
  type = "l", xlab = expression(omega), ylab = "(yobs - ytheo)^2",
  main = "Premier zoom" , las = 1)
```




```
omegax <- seq(from = 0.26, to = 0.265, length = 100)
plot( x = omegax, y = sapply(omegax, function(omega) { sce(omega, x, yobs) } ),
  type = "l", xlab = expression(omega), ylab = "(yobs - ytheo)^2",
  main = "Deuxieme zoom" , las = 1)
```



Mais ceci devient très vite fastidieux. Il existe des outils pour rechercher le minimum automatiquement. C'est précisément l'objet de cette fiche de TD sous R.

3 nlm (non-linear minimization)

La fonction `nlm()` permet de minimiser une fonction. Le premier argument obligatoire est le nom de la fonction à minimiser, ici la fonction `sce()` qui calcule la somme des carrés des résidus. Le deuxième argument obligatoire est un vecteur contenant une première estimation grossière des paramètres à estimer. Les arguments suivants sont optionnels, ce sont des arguments que `nlm()` va transmettre fidèlement à la fonction à minimiser. Ici, `sce()` a besoin de connaître les valeurs en abscisse, `x`, et les valeurs observées, `yobs`.

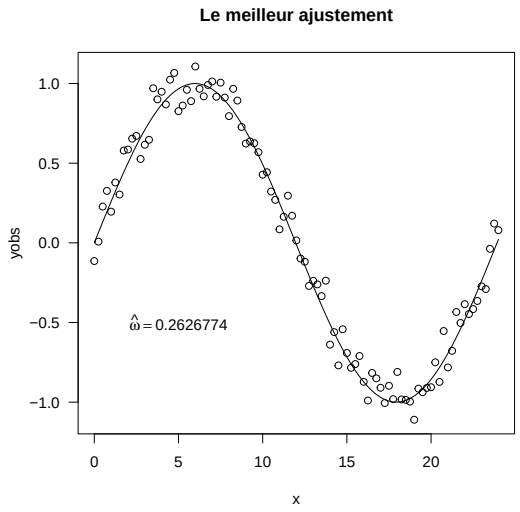
```
 nlm( f = sce, p = 0.25, x = x, yobs = yobs)
$minimum
[1] 0.7913835
$estimate
[1] 0.2626774

$gradient
[1] 2.964295e-08

$code
[1] 1

$iterations
[1] 5
```

Le premier résultat (`minimum`) est la valeur minimale de la fonction trouvée par `nlm()`. On constate que `nlm()` a amélioré notre score précédent (`0.791 < 2.246`). Le deuxième résultat (`estimate`) est celui qui nous intéresse le plus, c'est la valeur du paramètre ω telle que la somme des carrés des résidus soit minimale. En partant de notre estimation grossière $\omega = 0.25$, `nlm()` a trouvé que $\omega =$ `0.263` était meilleure au sens des moindres carrés.

Notons que c'est une valeur proche de la valeur que nous avons utilisée lors de la simulation, $\omega = 2\pi/24 =$ `0.262`, nous le savons, mais `nlm()` ne le savait pas.

Le troisième résultat (`gradient`) donne une idée de la pente au niveau du minimum, elle devrait être faible si tout s'est bien passé. C'est une indication rassurante mais non suffisante. Le quatrième résultat (`code`) est un code donnant une indication sur le bon déroulement de la procédure de minimisation, il doit être aussi petit que possible. C'est une indication rassurante mais non suffisante. Le dernier résultat (`iterations`) est le nombre d'itérations avant arrêt : `nlm()` procède pas à pas, par améliorations successives. Il est rassurant de constater que `nlm()` a convergé rapidement. Mais à chercher des solutions parfaites dans un monde parfait, on s'expose forcément à quelques déconvenues... quand il nous faut des solutions imparfaites dans un monde imparfait. Soit, les mises en garde vont être longuement développées par la suite, ne boudons pas notre plaisir de l'instant et visualisons le résultat clef en main de `nlm()` :

```
omegaestime <- nlm( f = sce, p = 0.25, x = x, yobs = yobs)$estimate
ytheo <- sin(omegaestime*x)
plot(x = x, y = yobs, las = 1, main = "Le meilleur ajustement")
lines(x, ytheo)
text(x = 5, y = -0.5, labels = bquote(hat(omega) == .(omegaestime)))
```



4 Le piège des minimums locaux

Il est facile d'avoir une bonne estimation intiale des paramètres lorsque que l'on travaille sur des données simulées. En pratique, ce n'est pas le cas. Jouons les idiots en fournissant à `nlm()` une estimation initiale un peu plus naïve, à peine plus du double de notre estimation précédente.

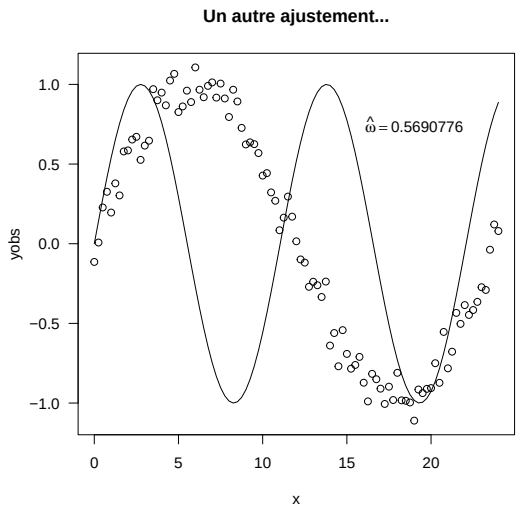
```
nlm1 <- nlm(f = sce, p = 0.57, x = x, yobs = yobs)
nlm1
```

```
$minimum
[1] 86.39535
$estimate
[1] 0.5690776

$gradient
[1] 2.032152e-06

$code
```

```
[1] 1

$iterations
[1] 3
```

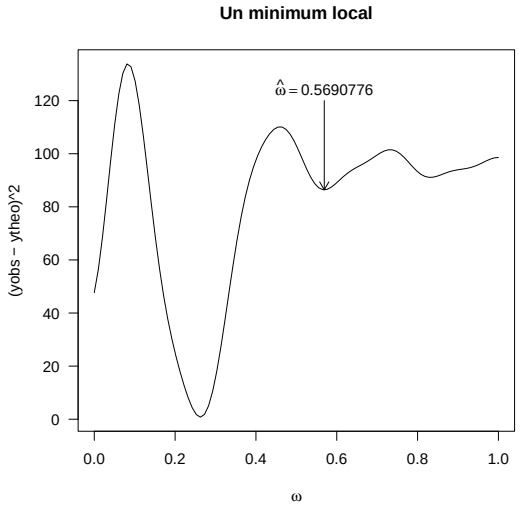
Que se passe-t-il ? Toutes les conditions rassurantes sont là, mais la somme des carrés des résidus semble franchement élevée. Voyons le modèle.

```
ytheo <-  sin(nlm1$estimate*x)
plot(x = x, y = yobs, las = 1, main = "Un autre ajustement...")
lines(x, ytheo)
text(19, 0.75, bquote(hat(omega) == .(nlm1$estimate)))
```



Un être humain normalement constitué ne pourrait pas se satisfaire d'un tel modèle. Pourquoi `nlm()` s'est fourvoyé ? Prenons un peu de recul et examinons le comportement de la somme des carrés des résidus en nous écartant un peu du voisinage de la "vraie" valeur $\omega = 0.263$.

```
omegax <- seq(from = 0, to = 1, length = 100)
plot( x = omegax, y = sapply(omegax, function(omega) { sce(omega, x, yobs) } ),
  type="l", las = 1, xlab = expression(omega), main = "Un minimum local",
  ylab = "(yobs - ytheo)^2" )
arrows(x0 = nlm1$estimate, y0 = 120, x1 = nlm1$estimate, y1 = nlm1$min, length = 0.1)
text(nlm1$est, 125, bquote(hat(omega) == .(nlm1$estimate)))
```



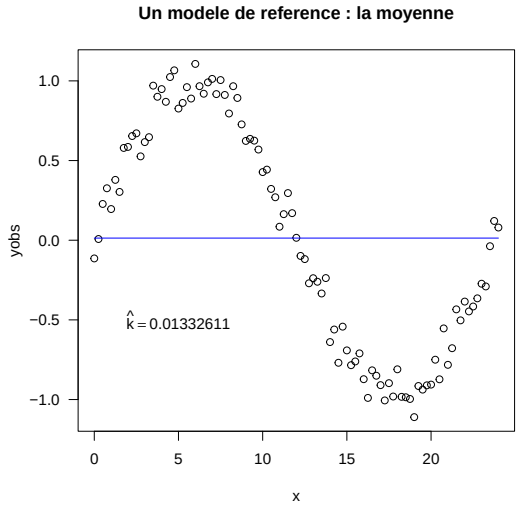
Il y a un minimum local de la fonction objective pour $\omega = 0.569$, et en fournissant une estimation initiale trop proche de ce minimum local, `nlm()` est restée piégée dedans. `nlm()` utilise un algorithme de type Newton, c'est-à-dire basé sur une estimation locale du gradient, si le gradient local mène à un minimum local, `nlm()` restera piégé dans ce minimum local, sans qu'aucune procédure de contrôle interne ne puisse indiquer qu'il y a un problème.

Pour détecter ce genre de situation il est bon d'avoir un modèle de référence dont on connaît la solution exacte. Par exemple, le modèle $y = k$, où k est une constante. Quelle est la valeur de k qui minimise la somme des carrés des résidus ?

```
scek <- function(param, x, yobs)
{
  ytheo <- sapply( x, function(x) { param[1] } )
  return( sum( (yobs-ytheo)^2) )
}
nlmk <- nlm( f = scek, p = 0.0, x = x, yobs = yobs)
nlmk
```

```
$minimum
[1] 47.66545
$estimate
[1] 0.01332611

$gradient
[1] -7.105427e-09

$code
[1] 1

$iterations
[1] 2
```

```
ytheo <- sapply(x, function(x) { nlmk$estimate } )
plot(x = x, y = yobs, las = 1, main = "Un modele de reference : la moyenne")
lines(x, ytheo, col = "blue")
text(x = 5, y = -0.5, labels = bquote(hat(k) == .(nlmk$estimate)))
```

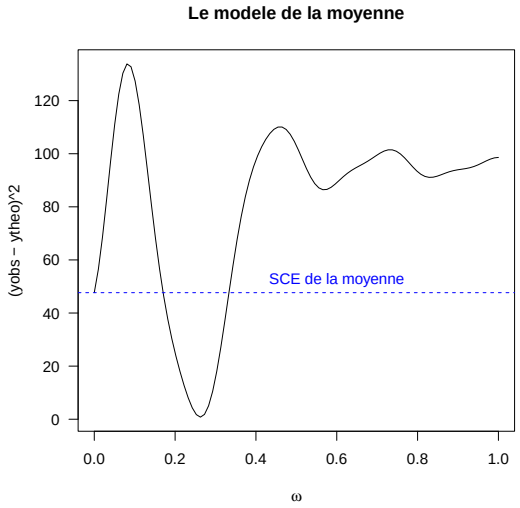


Mais nous connaissions déjà la solution, puisque la valeur qui minimise la somme des carrés des écarts n'est rien d'autre que la moyenne des valeurs observées, et la valeur de la fonction objective donne la variance, à $1/(n-1)$ près.

```
c(mean(yobs), nlmk$estimate)
```
```
[1] 0.01332661 0.01332611
```
```
c(var(yobs), nlmk$minimum/(length(yobs)-1))
```
```
[1] 0.4965151 0.4965151
```

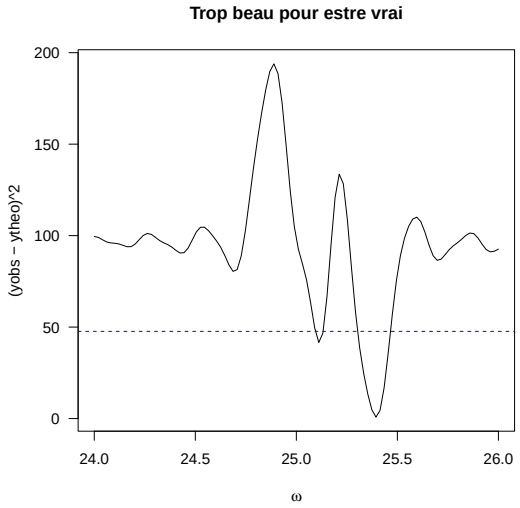
Il n'est donc pas inutile de garder `(length(yobs) - 1)*var(yobs) = 47.665` comme valeur de référence pour la somme des carrés des résidus, tout modèle au-dessus de cette valeur est moins bon que le modèle résumant les données par leur simple moyenne. Le modèle précédent avec $\omega =$ `0.569` est donc franchement mauvais :

```
sceminref <- (length(yobs)-1)*var(yobs)
omegax <- seq(from = 0, to = 1,length = 100)
plot( x = omegax, y = sapply(omegax, function(omega) { sce(omega, x, yobs) } ),
 type="l", las = 1, xlab = expression(omega), ylab = "(yobs - ytheo)^2",
 main = "Le modele de la moyenne" )
abline(h = sceminref, col = "blue", lty = 2)
 text(0.6, sceminref, "SCE de la moyenne", pos = 3, col = "blue")
```



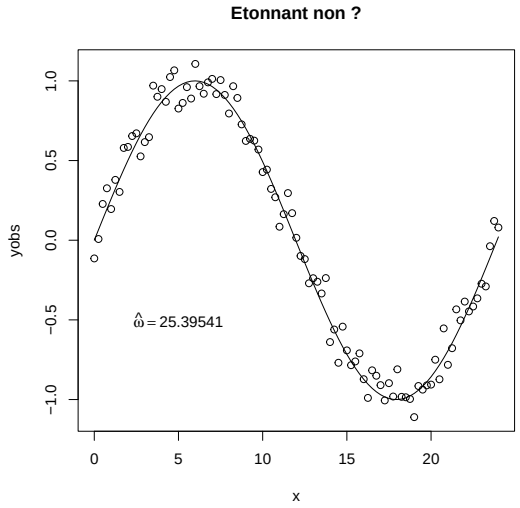
Le minimum local à `0.569` est au-dessus de la somme des carrés des résidus de référence représentée par la ligne horizontale. Nous ne tomberons donc pas dans le piège de ce minimum local. Mais sommes-nous armés contre tous les minimums locaux ? Non, ce serait trop beau, en effet :

```
omegax <- seq(from = 24, to = 26, length =100)
plot( x = omegax, y = sapply(omegax, function(omega) { sce(omega, x, yobs) } ),
 type="l", las = 1, xlab = expression(omega), ylab = "(yobs - ytheo)^2",
 main = "Trop beau pour estre vrai" )
 abline(h = sceminref, col = "blue", lty = 2)
```



Nous avons donc au voisinage de $\omega = 25.4$ un minimum local qui semble être très bon. Quel est donc ce nouveau piège ? Que se passe-t-il ici ?

```
badomega <- nlm( f = sce, p = 25.4, x = x, yobs = yobs)$estimate
ytheo <- sin(badomega*x)
plot(x = x, y = yobs, main = "Etonnant non ?")
lines(x, ytheo)
text(5, -0.5, bquote(hat(omega) == .(badomega)))
```



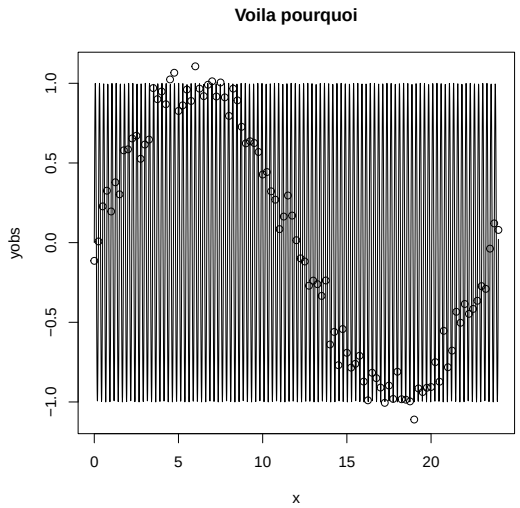
ÉTONNANT non ? Voila un modèle qui semble fort satisfaisant à première vue. Quel est le gag ? Calculons la période de ce modèle :

```
2*pi/badomega
```

```
[1] 0.2474142
```

C'EST très proche de 0.25, la fréquence d'échantillonnage de notre simulation. Mais pour le voir, il nous faut un pas plus petit sur l'axe temporel pour représenter le modèle.

```
xfin <- seq(from = 0, to = 24, by = 0.01)
ytheo <- sin(badomega*xfin)
plot(x = x, y = yobs, main = "Voila pourquoi")
lines(xfin, ytheo)
```



EN oscillant avec une période proche de celle de l'échantillonnage, le modèle arrive à passer au voisinage des points observés. Au sens des moindres car-

rés, le modèle est bon. Mais d'un point de vue biologique, un rythme circadien de 15 minutes n'a pas de sens.

L'EXEMPLE de minimum local présenté ici n'est évidement qu'à visée pédagogique, il ne doit pas masquer le fait que les minimums locaux constituent un problème très concret dont les conséquences peuvent être dramatiques. Un exemple sorti de votre propre réfrigérateur : *Listeria monocytogenes* est une bactérie pathogène capable de se développer même à très basses température. Le graphique ci-après extrait de [3] montre qu'il y a un problème de minimum local pour les paramètres d'un modèle de la croissance de *L. monocytogenes* à basse température :

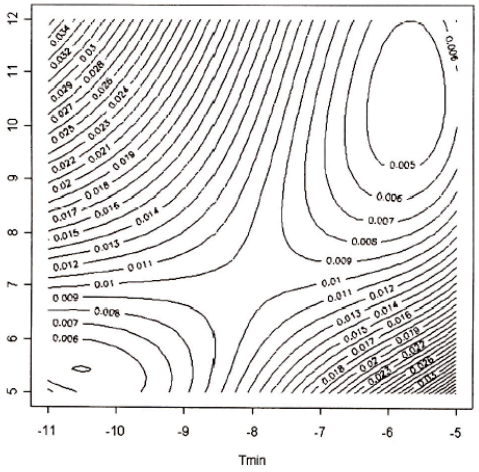


FIGURE 4

Contour iso-somme des carrés résiduels (souche CIP)

AIMERIEZ-VOUS que les dates de péremption de vos yaourts soient calibrées à partir d'une estimation des paramètres du modèle de croissance de *L. monocytogenes* tombée dans le piège d'un minimum local ? Le premier paragraphe de la conclusion de [3] mérite d'être reprise *in extenso* :

> De cette étude, nous pensons que la recherche des paramètres d'intérêt en "microbiologie prévisionnelle" n'est pas une tâche qui peut être entièrement automatisée. La mise en oeuvre d'un modèle dont les paramètres sont biologiquement interprétables est importante; mais son utilisation mérite un regard très critique. En particulier, la validation du modèle est une phase essentielle qui nous semble trop souvent négligée. La présence de deux minima, même si l'un donne des estimations peu vraisemblables de T_{min}, indique que le choix des conditions initiales pour la procédure de régression non linéaire n'est absolument pas anodin. Les conclusions hâtives (en utilisant de façon un peu trop automatique les logiciels standard) peuvent entraîner ainsi des conséquences importantes pour la santé publique.

À méditer sérieusement ! On trouvera un autre exemple de minimum local avec des données réelles dans la fiche de TD[1] « [a]justement du modèle de DE RÉAUMUR (1735) aux données de DE RÉAUMUR (1735-1740). »

[1] tdr4R : `https://pbil.univ-lyon1.fr/R/pdf/tdr4R.pdf`

5 Le piège des mesas

Pour illustrer ce problème, nous allons utiliser une simulation du modèle de croissance logistique,

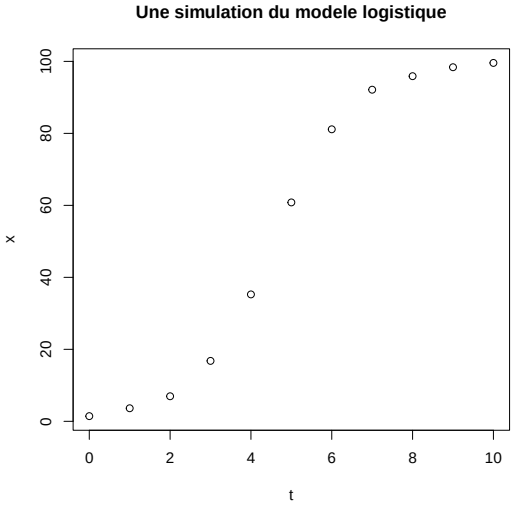
$$x(t) = x_0 e^{\mu t} \frac{x_m}{x_m - x_0 + x_0 e^{\mu t}}$$

extrait de l'annexe 6.4 de [4].

```
simlog <- read.table(file="http://pbil.univ-lyon1.fr/R/donnees/simlog.txt", header = TRUE)
summary(simlog)
```

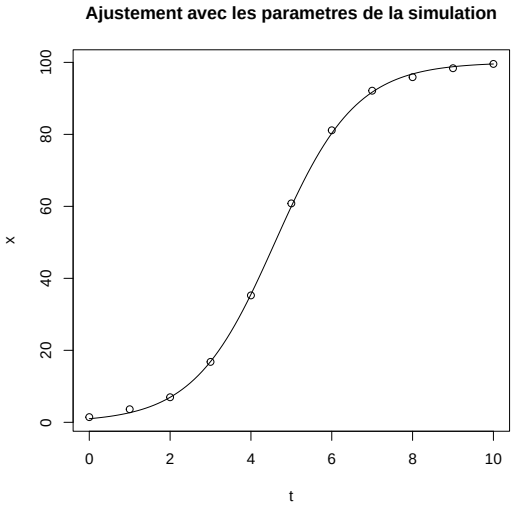
```
       t                x          
 Min.   : 0.0   Min.   : 1.44  
 1st Qu.: 2.5   1st Qu.:11.88  
 Median : 5.0   Median :60.82  
 Mean   : 5.0   Mean   :53.82  
 3rd Qu.: 7.5   3rd Qu.:94.01  
 Max.   :10.0   Max.   :99.57  
```

```
plot(simlog, main = "Une simulation du modele logistique")
```



Définissons le modèle de croissance logistique et représentons le pour la valeur des paramètres de la simulation : la condition initiale $x_0 = 1$, la biomasse finale $x_m = 100$ et le taux de croissance $\mu = 1$.

```
logistique <- function(t, param)
{
  x0 <- param[1]
  xm <- param[2]
  mu <- param[3]
  tmp <- x0*exp(mu*t)
  return( tmp*xm/(xm - x0 + tmp) )
}
plot(simlog, main = "Ajustement avec les parametres de la simulation")
xaxix <- seq(from = 0, to = 10, by = 0.1)
lines(xaxix, sapply(xaxix, function(x) { logistique(x, c(1,100,1)) } ) )
```



Cherchons la valeur des paramètres qui minimise la somme des carrés des résidus.

```
scelog <- function(param, data)
{
  tobs <- data[ ,1]
  xobs <- data[ ,2]
  xtheo <- sapply(tobs, logistique, param)
  return( sum((xobs-xtheo)^2) )
}
nlm( f = scelog, p = c(1.1, 110, 1.1), data = simlog )
```

```
$minimum
[1] 2.666712
$estimate
[1]  0.9434911 99.5689749  1.0183770

$gradient
[1]  1.304830e-04 -1.407955e-05 -1.063770e-04

$code
[1] 3

$iterations
[1] 38
```

```
sceminref <- (length(simlog$x)-1)*var(simlog$x)
sceminref
```

```
[1] 17285.35
```

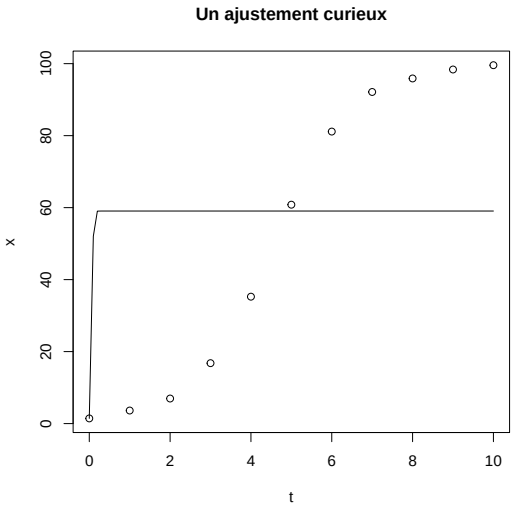
Nous retrouvons des valeurs pour les paramètres très proches de celles utilisées lors de la simulation, et la somme des carrés des résidus est bien meilleure que pour le modèle moyenne. Mais c'est toujours facile quand on connaît la solution à l'avance. Utilisons une valeur initiale pour les paramètres un peu moins bonne.

```
nlm( f = scelog, p = c(1, 100, 10), data = simlog )
```

```
$minimum
[1] 14267.56
$estimate
[1]  1.439807 59.055530 56.964309

$gradient
[1] -5.053427e-06 -9.178799e-06  3.193209e-08

$code
[1] 1

$iterations
[1] 22
```

Le minimum est un peu meilleur que pour le modèle moyenne, mais nous sommes très loin du minimum global. Que se passe-t-il ?

```
est <- nlm( f = scelog, p = c(1,100,10), data = simlog )$estimate
plot(simlog, main = "Un ajustement curieux")
lines( xaxix, sapply( xaxix, function(x) { logistique(x, est) } ) )
```



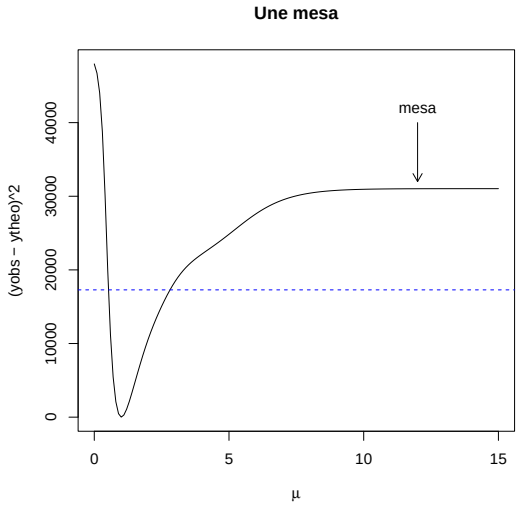
Le modèle est un peu meilleur que le modèle moyenne parce que le premier point est bien prédit, par contre pour les suivants le modèle avec $x_m =$ `59.05553` prédit la moyenne :

```
mean(simlog$x[-1])
```

```
[1] 59.05556
```

Encore un minimum local ? Pas exactement. Pour comprendre ce qu'il se passe, observons l'influence du paramètre μ sur la somme des carrés des résidus, les deux autres paramètres étant fixés à la valeur utilisée pendant la simulation.

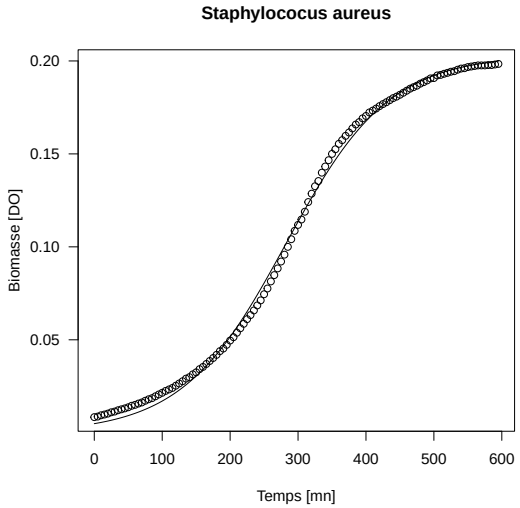
```
muaxis <- seq(from = 0, to = 15, by = 0.1)
plot( muaxis, sapply( muaxis, function(x) { scelog( c(1,100,x), simlog) } ), t = "l",
  ylab = "(yobs - ytheo)^2", xlab = expression(mu), main = "Une mesa")
abline(h = sceminref, lty = 2, col = "blue")
arrows(12, 40000, 12, 32000, le = 0.1)
text(12, 40000, "mesa", pos = 3)
```



POUR les fortes valeurs de μ, nous avons une mesa : entre 10 et 15 la fonction objective est complètement plate. Dans un monde parfait avec une précision de calcul infinie, ce ne serait pas exactement vrai, mais dans le monde réel avec une précision de calcul limitée, la fonction objective devient rigoureusement plate. Quand le taux de croissance μ est trop fort tous les points théoriques -sauf le premier- tendent vers la valeur de la biomasse maximale x_m, la fonction objective devient alors insensible à une variation du paramètre μ. En fournissant une estimation initiale trop grande pour μ, `nlm()` ne pourra pas deviner comment optimiser μ parce que `nlm()` n'a qu'une connaissance locale de la pente. Quand c'est plat, `nlm()` ne sait pas où aller.

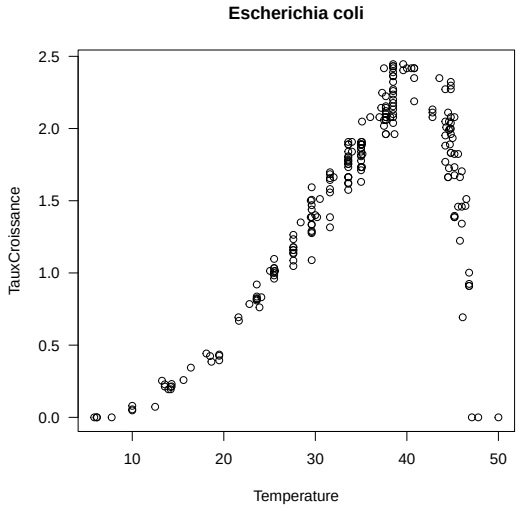
QUELQUES données réelles tant que nous avons le modèle logistique sous la main. Il s'agit de la courbe de croissance de *Staphylococus aureus* (extrait de l'annexe 6.7 de [4]).

```
saureus <- read.table(file="http://pbil.univ-lyon1.fr/R/donnees/saureus.txt", header = TRUE)
plot(saureus, main = "Staphylococus aureus",
  xlab = "Temps [mn]", las = 1, ylab = "Biomasse [DO]")
est <- nlm( f = scelog, p = c(0.01,0.2,0.02), data = saureus )$estimate
xtheo <- sapply( saureus$t, function(x) { logistique(x, est) } )
lines( saureus$t , xtheo)
```



6 Un cas pratique : le modèle CTMI

```
barber <- read.table(file="http://pbil.univ-lyon1.fr/R/donnees/barber.txt", header = TRUE)
plot(barber, main = "Escherichia coli", las = 1)
```



Les données sont extraites de [1]. Elles consistent en 217 mesures du taux de croissance (h^{-1}) de la bactérie *Escherichia coli* à différentes températures (°C). L'influence de la température T, sur le taux de croissance en phase exponentielle des micro-organismes, μ, est donné en première approximation, par :

$$\mu(T) = \begin{cases} 0 & \text{si } T \notin [T_{min}, T_{max}] \\ \frac{\mu_{opt}(T-T_{max})(T-T_{min})^2}{(T_{opt}-T_{min})[(T_{opt}-T_{min})(T-T_{opt})-(T_{opt}-T_{max})(T_{opt}+T_{min}-2T)]} & \text{si } T \in [T_{min}, T_{max}] \end{cases}$$

où T_{min} représente la température en deça de laquelle il n'y a plus de croissance, T_{max} la température au delà de laquelle il n'y a plus de croissance, T_{opt} la température pour laquelle le taux de croissance atteint son maximum μ_{opt}. C'est le modèle dit des températures cardinales [5].

POUR estimer la valeur des paramètres de ce modèle, nous commençons par définir une fonction donnant la valeur prédite du taux de croissance pour une température et un jeu de valeurs des paramètres donnés :

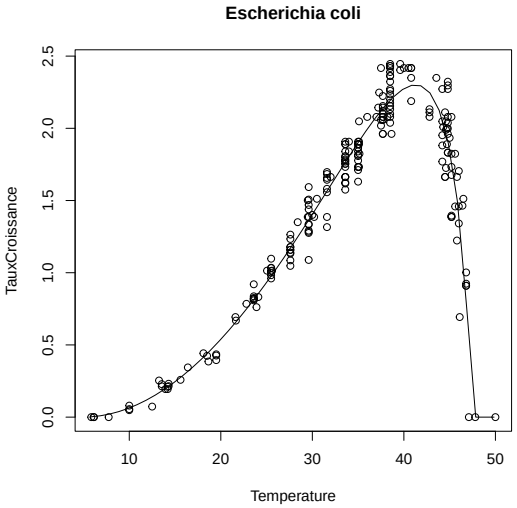
```
CTMI <- function(T, param)
{
  Tmin <- param[1]
  Topt <- param[2]
  Tmax <- param[3]
  Muopt <- param[4]
  if( T <= Tmin || T >= Tmax )
  {
    return(0)
  }
  else
  {
    Num <- (T-Tmax)*(T-Tmin)^2
    Den <- (Topt-Tmin)*((Topt-Tmin)*(T-Topt)-(Topt-Tmax)*(Topt+Tmin-2*T))
    return(Muopt*Num/Den)
  }
}
```

NOUS définissons ensuite notre fonction objective, ici la somme des carrés des écarts entre les valeurs observées et prédites :

```
sceCTMI <- function(param, data)
{
  xobs <- data[ ,1]
  yobs <- data[ ,2]
  ytheo <- sapply(xobs, CTMI, param)
  return( sum((yobs-ytheo)^2) )
}
```

NOUS allons maintenant utiliser `nlm()` pour estimer la valeur des paramètres. Pour cela, nous avons besoin d'une première estimation grossière de la valeur des paramètres. D'après la représentation graphique des données, on voit que l'on a $T_{min} \approx 10$, $T_{opt} \approx 40$, $T_{max} \approx 50$ et $\mu_{opt} \approx 2.5$.

```
guess <- c( 10, 40, 50, 2.5 )
nlm2 <- nlm( f = sceCTMI, p = guess, data = barber )
xaxis <- seq(from = min(barber$Temp), to = max(barber$Temp), by = 1)
ytheo <- sapply( xaxis, function(x) { CTMI(x, nlm2$estimate) } )
plot(barber, main = "Escherichia coli")
lines(xaxis, ytheo)
```



L'AJUSTEMENT est raisonnablement bon, nous pouvons résumer les 217 données en disant que dans le cas de *Escherichia coli* nous avons $T_{min} \approx 4.9$ °C, $T_{opt} \approx 41.3$ °C, $T_{max} \approx 47.5$ °C et $\mu_{opt} \approx 2.3\ h^{-1}$.

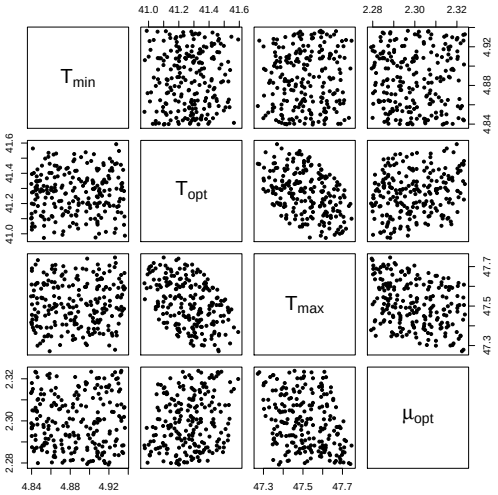
7 Régions de confiance

UNE région de confiance pour la valeur des paramètres avec un risque de première espèce α est donnée [2] par l'ensemble des valeurs des paramètres telles que la somme des carrés des résidus n'excède pas un seuil donné,

$$\theta / S(\theta) \leq S(\hat{\theta})(1 + \frac{p}{n-p} F^{\alpha}_{p;n-p})$$

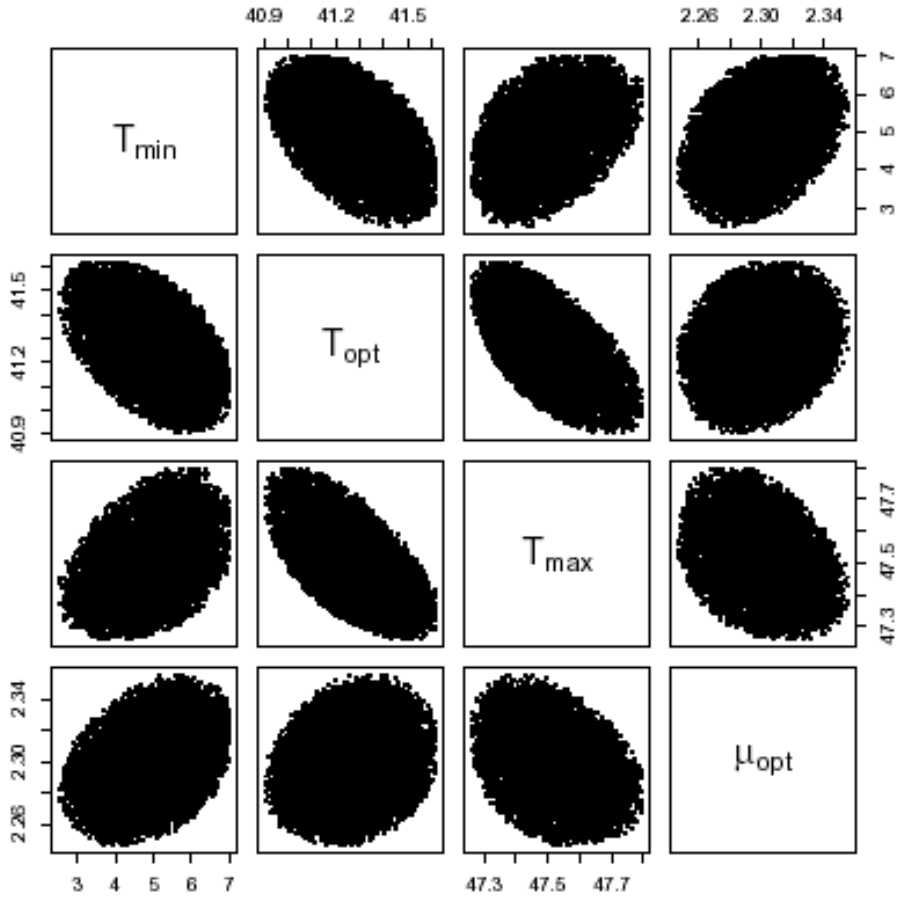
où p est le nombre de paramètres du modèle, n le nombre de points disponibles dans le jeu de données, et $\hat{\theta}$ le vecteur des valeurs des paramètres tel que le critère soit minimal. R permet très facilement de visualiser ces régions de confiance.

```
scemin <- nlm2$minimum
n <- nrow(barber)
p <- 4
alpha = 0.05
seuil <- scemin*( 1 + (4*qf(p = 1 - alpha, df1 = p, df2 = n - p))/(n - p) )
estmin <- 0.99*nlm2$estimate
estmax <- 1.01*nlm2$estimate
pt1000 <- sapply(1:4, function(x) { runif(n=1000, min=estmin[x], max=estmax[x])})
inside <- pt1000[apply(pt1000,1,function(x) sceCTMI(param=x,data=barber)<=seuil),]
pairs(inside,
  label = c(expression(T[min]), expression(T[opt]), expression(T[max]), expression(mu[opt])),
  pch = 20)
```



On n'est pas trop mauvais pour T_{opt} et T_{max}, mais pas assez large pour T_{min} et μ_{opt}. On affine et on recommence.

```
estmin <- c(2.5, 40.9, 47.25, 2.245)
estmax <- c(7.0, 41.62, 47.8, 2.357)
pt50000 <- sapply(1:4, function(x) { runif(n=50000, min=estmin[x], max=estmax[x])})
inside2 <- pt50000[ apply(pt50000, 1, function(x) sceCTMI(param=x, data = barber) <= seuil), ]
pairs(inside2,
  label = c(expression(T[min]), expression(T[opt]), expression(T[max]), expression(mu[opt])),
  pch=".")
```



On retrouve les résultats de la figure 4 de [5] avec $T_{min} \in [2.5, 7]$, $T_{opt} \in [40.9, 41.62]$, $T_{max} \in [47.25, 47.8]$ et $\mu_{opt} \in [2.245, 2.357]$.

References

[1] M.A. Barber. The rate of multiplication of *Bacillus coli* at different temperatures. *Journal of Infectious diseases*, 5:379–400, 1908.

[2] E.M.L. Beale. Confidence regions in non-linear estimation. *Journal of the Royal Statistical Society*, 22B:41–88, 1960.

[3] S. Charles-Bajard, J.-P. Flandrois, and R. Thomassone. Quelques problèmes statistiques rencontrés dans l'estimation de la température minimale de croissance de *Listeria monocytogenes*. *Revue de Statistique Appliquée*, LI:59–71, 2003.

[4] J.R. Lobry. *Ré-évaluation de modèle de croissance de Monod. Effet des antibiotiques sur l'énergie de maintenance*. PhD thesis, University Claude Bernard, Lyon I, 1991.

[5] L. Rosso, J.R. Lobry, and J.-P. Flandrois. An unexpected correlation between cardinal temperatures of microbial growth highlighted by a new model. *Journal of Theoretical Biology*, 162(4):447–463, 1993.