Rapport utilisant le logiciel 

Modèles logistiques

D. Chessel & A.B. Dufour

5 février 2008

La fiche propose un exercice pour apprendre à ajuster une courbe logistique à des observations. La classe des modèle `selfStart`.

Table des matières

1 Introduction

La question est posée par D. Nandris (Laboratoire de phyto-pathologie, ORSTOM, Nouméa). Pendant plusieurs années et pour plusieurs stations, à intervalles réguliers pendant la période de végétation, trois notes de l'état sanitaire des parcelles de caféier sont établies. Une des maladies étudiées est la rouille, maladie la plus grave de l'arabica, pouvant conduire à la défoliation sévère. La note intègre les feuilles vues infectées à la date d'observation mais aussi toutes celles qui ont disparu depuis la première visite. Le but de l'étude porte sur les déterminismes du développement de la maladie et une modélisation du déroulement annuel de cette maladie est souhaitable. Pour illustrer la modélisation par un modèle logistique de cette situation, on extrait les mesures faites dans 3 stations (BAN, EMA et MOU2) pendant trois années (92, 94 et 96). On utilise les en-têtes des variables sta pour le nom de la station, an pour l'année du relevé, dat pour le numéro du jour dans l'année du relevé, rou pour la note sanitaire de la rouille (la valeur minimum étant 0).

Récupérer le fichier de données :

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

2 Représentation des données

```
dn[1:4, ]
```

```
  sta  an dat  rou
1 EMA2 A92  36 0.00
2 EMA2 A92  73 0.04
3 EMA2 A92 106 0.12
4 EMA2 A92 142 0.40
```

```
summary(dn)
```

```
  sta       an          dat             rou        
BAN1:26   A92:26   Min.   : 23.0   Min.   :0.0000  
EMA2:27   A94:27   1st Qu.:101.0   1st Qu.:0.2100  
MOU2:27   A96:27   Median :173.0   Median :0.7800  
                   Mean   :175.4   Mean   :0.8959  
                   3rd Qu.:245.2   3rd Qu.:1.3550  
                   Max.   :351.0   Max.   :2.3800  
```

`sta` est une variable à 3 modalités, `an` est une variable qualitative à 3 modalités, `dat` et `rou` sont des variables quantitatives. Pour obtenir la figure 1 :

```
coplot(dn$rou ~ dn$dat | dn$an * dn$sta)
```

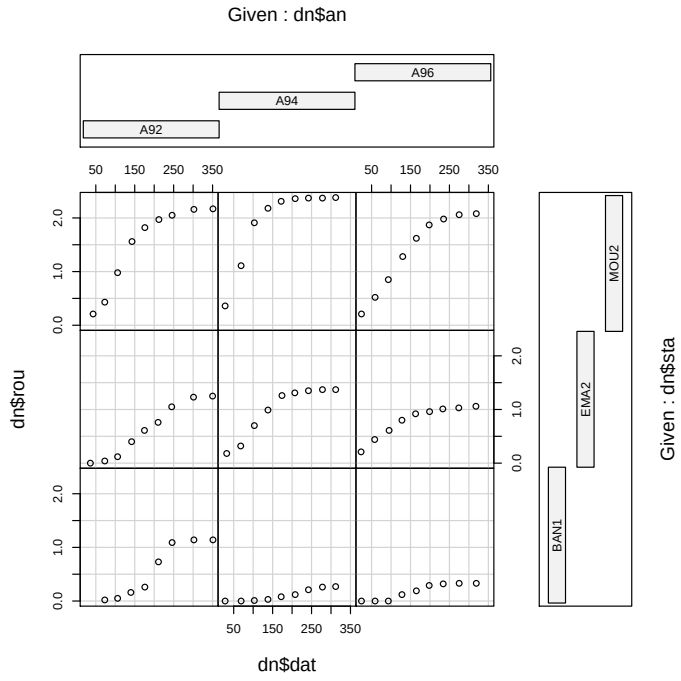


FIG. 1 – Représentation des données brutes par la fonction `coplot` de R.

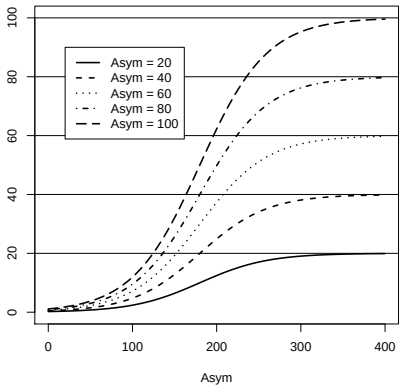
3 Le modèle logistique

Toutes les courbes sont croissantes et ont une allure de modèle logistique, simple et fort utilisé en écologie. On utilise la paramétrisation de R :

$$y(t) = \frac{Asym}{1 + \exp\left(\frac{t_{mid} - t}{Scal}\right)}$$

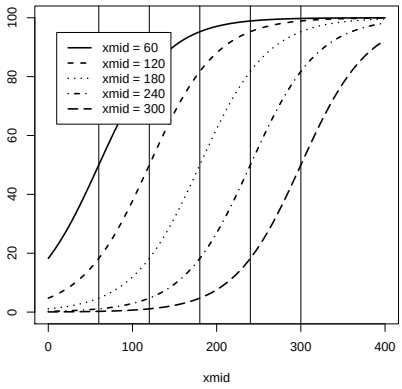
y est la note sanitaire et t le temps en jour (0 - 365). Les paramètres ont une signification claire. *Asym* définit l'intensité limite de la maladie en fin de saison et on peut l'appeler l'**optimum** : des valeurs fortes de *Asym* correspondent à une infection sévère. *Scal* caractérise l'inverse de la vitesse de développement de la maladie et on peut l'appeler l'**étalement** : des valeurs fortes de *Scal* correspondent à une infection se développant progressivement sur une longue période. t_{mid}est simplement la date à laquelle la courbe atteint la moitié du maximum ($Asym/2$) et on peut l'appeler la **date moyenne** des valeurs fortes de t_{mid}correspondent à un développement tardif. On va utiliser la fonction `nls` (*nonlinear least squares*)et les fonction SS (*self-starting*). Examinons d'abord l'allure des courbes logistiques. Elles dépendent de trois paramètres. Le premier est `Asym`, l'ordonnée de l'asymptote :

```
w0 <- seq(0, 400, le = 100)
plot(0, 0, xlim = c(0, 400), ylim = c(0, 100), type = "n", xlab = "Asym",
    ylab = "")
for (i in 1:5) lines(w0, SSlogis(w0, i * 20, 180, 40), lty = i,
    lwd = 2)
abline(h = 20 * (1:5))
legend(20, 90, paste("Asym =", 20 * 1:5), lty = 1:5, lwd = 2, bg = "white")
```



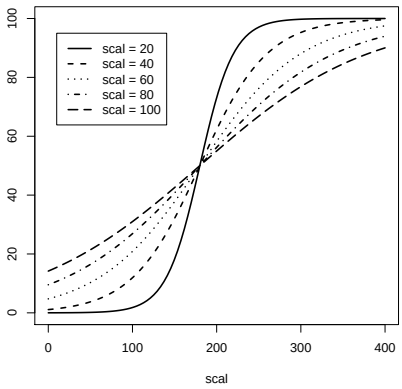
Le second est `xmid`, l'abscisse du point d'inflexion :

```
plot(0, 0, xlim = c(0, 400), ylim = c(0, 100), type = "n", xlab = "xmid",
    ylab = "")
for (i in 1:5) lines(w0, SSlogis(w0, 100, i * 60, 40), lty = i,
    lwd = 2)
abline(v = 60 * (1:5))
legend(10, 95, paste("xmid =", 60 * (1:5)), lty = 1:5, lwd = 2,
    bg = "white")
```



Le troisième est `scal` qui caractérise la pente au point d'inflexion :

```
plot(0, 0, xlim = c(0, 400), ylim = c(0, 100), type = "n", xlab = "scal",
    ylab = "")
for (i in 1:5) lines(w0, SSlogis(w0, 100, 180, i * 20), lty = i,
    lwd = 2)
legend(10, 95, paste("scal =", 20 * (1:5)), lty = 1:5, lwd = 2,
    bg = "white")
```



Les indications contenues dans l'ouvrage de A. Pavé (1994)(1) sur le modèle et son estimation sont très utiles pour l'emploi de la fonction `nls`. On se heurte alors à la principale difficulté (p. 513).

> Le problème, que nous n'avons pas abordé jusqu'à présent, est délicat car un bon comportement d'un algorithme de minimisation dépend fortement de la qualité des estimations initiales. Pour le modèle exponentiel on peut proposer de prendre les estimations obtenues par régression linéaire sur les logarithmes, par contre pour des modèles plus compliqués on est souvent réduit à des bricolages plus ou moins avouables.

Dans R, ce ne sera pas utile. La fonction logistique est appelée `SSlogis` parce que c'est un *selfStart model*. Elle autorise l'emploi de la fonction `getInitial` qui

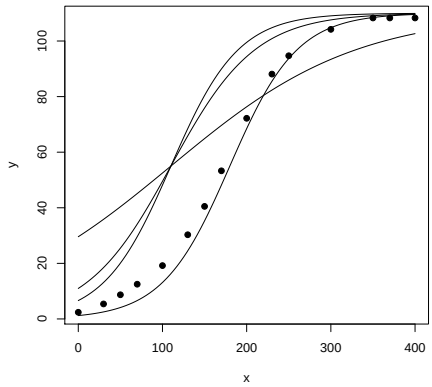
donne des estimations initiales des paramètres. Pour s'en convaincre, on reprendra l'exemple de l'ouvrage cité p. 159 (croissance d'une population bactérienne) :

```
x <- c(0, 30, 50, 70, 100, 130, 150, 170, 200, 230, 250, 300, 350,
    370, 400)
y <- c(2.4, 5.4, 8.7, 12.5, 19.2, 30.3, 40.5, 53.3, 72.2, 88.1,
    94.7, 104.2, 108.3, 108.3, 108.3)
```

4 Estimation des paramètres

Lire la fiche de documentation de `selfStart`, la classe des modèles non linéaires qui ont des procédures automatiques d'estimation initiale. Vérifier par `class(SSlogis)` que `SSlogis` en fait partie. `SSlogis` est une fonction ordinaire, en ce sens qu'elle calcule le modèle si on lui donne les paramètres. Contrôler votre habileté en cherchant les paramètres à la main.

```
plot(x, y, pch = 20, cex = 1.5)
lines(w0, SSlogis(w0, 110, 110, 110))
lines(w0, SSlogis(w0, 110, 110, 50))
lines(w0, SSlogis(w0, 110, 110, 40))
lines(w0, SSlogis(w0, 110, 180, 40))
```



On n'est déjà pas si loin de la solution. Mais c'est un jeu qui demande une procédure !

Quand on possède une valeur de départ, on utilise la fonction nls pour obtenir les valeurs des paramètres qui minimise la somme des carrés des écarts :

```
nls(y ~ Asym/(1 + exp((xmid - x)/scal)), start = list(Asym = 110,
    xmid = 180, scal = 40))
```

```
Nonlinear regression model
  model:  y ~ Asym/(1 + exp((xmid - x)/scal))
   data:  parent.frame()
  Asym   xmid   scal
110.07 171.41  44.91
 residual sum-of-squares: 17.86

Number of iterations to convergence: 5
Achieved convergence tolerance: 1.159e-06
```

Une étude approfondie est proposée dans la fiche :

http://pbil.univ-lyon1.fr/R/fichestd/tdr46.pdf

Quand on n'en a pas et qu'on utilise un modèle `selfStart`, la fonction qui calcule le modèle a un attribut qui calcule des valeurs initiale des paramètres. Vérifier par :

```
attributes(SSlogis)
```

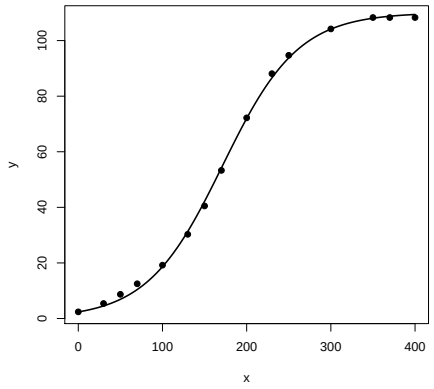
Cette fonction sera utilisée par `getInitial` :

```
getInitial(y ~ SSlogis(x, Asym, xmid, scal), data = cbind.data.frame(x,
    y))
```

```
    Asym     xmid     scal
110.0666 171.4110  44.9059
```

Ces valeurs initiales sont ici les valeurs optimales. Pour ajuster un modèle logistique, il suffira donc de faire :

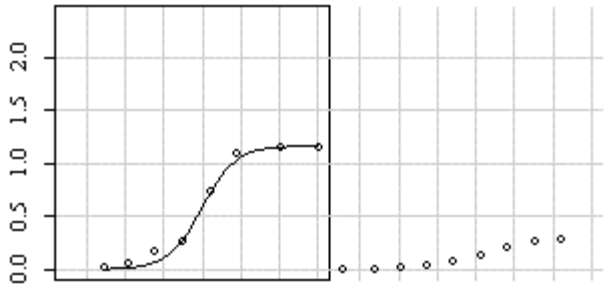
```
para <- getInitial(y ~ SSlogis(x, Asym, xmid, scal), data = cbind.data.frame(x,
    y))
plot(x, y, pch = 20, cex = 1.5)
lines(w0, SSlogis(w0, para[1], para[2], para[3]), lwd = 2)
```



On peut donc écrire une petite fonction pour compléter le `coplot` :

```
"flogis" <- function(x, y, col = 1, pch = 1) {
    a <- cbind.data.frame(x, y)
    points(x, y, pch = 20, cex = 1.5)
    para <- getInitial(y ~ SSlogis(x, Asym, xmid, scal), data = a)
    w0 <- seq(min(x), max(x), le = 50)
    if (sum(is.na(para)) > 0)
        return()
    lines(w0, SSlogis(w0, para[1], para[2], para[3]), lwd = 2)
}
coplot(dn$rou ~ dn$dat | dn$an * dn$sta, panel = flogis)
```

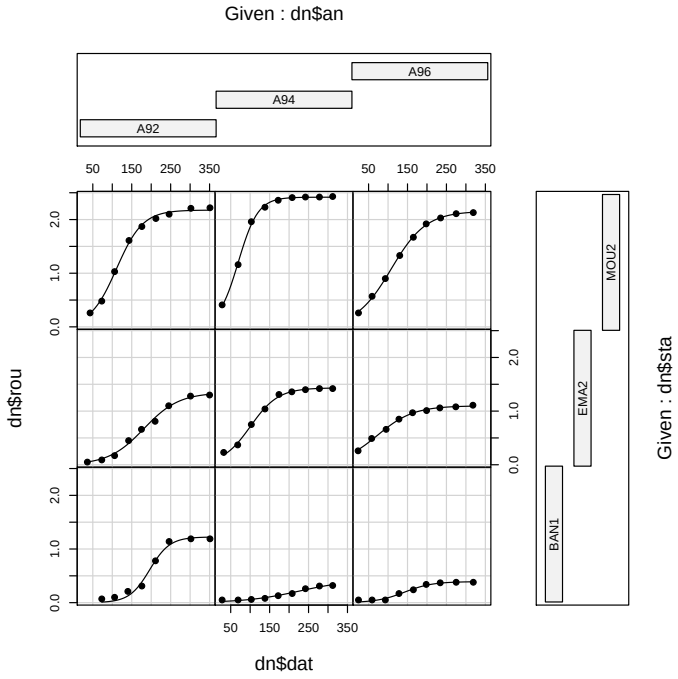
Observer que la procédure est prise en défaut :



On ne peut estimer directement ce modèle. En fait la difficulté tient aux valeurs nulles :

```
dn$rou <- dn$rou + 0.05
```

Refaire la figure complète.



La figure 4 étend la procédure aux données complètes (non disponible). La modélisation par une courbe logistique est satisfaisante dans la plupart des cas et une étude de la forme de la courbe est possible en fonction du temps et de l'espace.

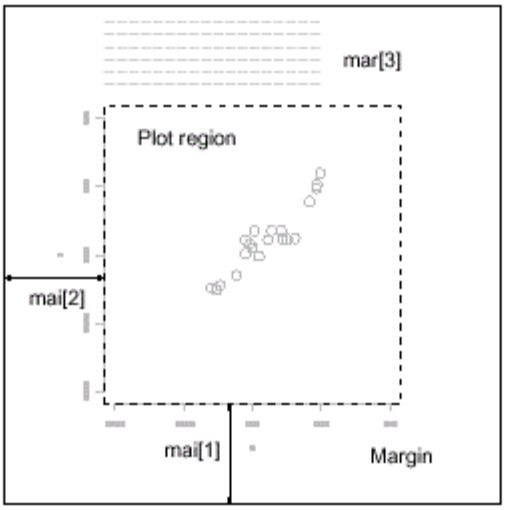
Pour continuer cet exercice, utiliser :

`http://pbil.univ-lyon1.fr/R/pps/pps021.pdf`

Pour améliorer les figures, utiliser au menu `Aide`, l'option `Manuels (en pdf)`, l'onglet `An introduction to R` et le chapitre `Chapter 12 : Graphical pro-`

cedures. En particulier, on pourra ainsi maîtriser les marges avec les paramètres mar et mai :

Chapter 12: Graphical procedures 74



Graphics parameters controlling figure layout include:

`mai=c(1, 0.5, 0.5, 0)`
Widths of the bottom, left, top and right margins, respectively, measured in inches.

`mar=c(4, 2, 2, 1)`
Similar to `mai`, except the measurement unit is text lines.

Références

[1] Pavé A. (1994) Modélisation en biologie et écologie. Aléas, Lyon, 559 p.

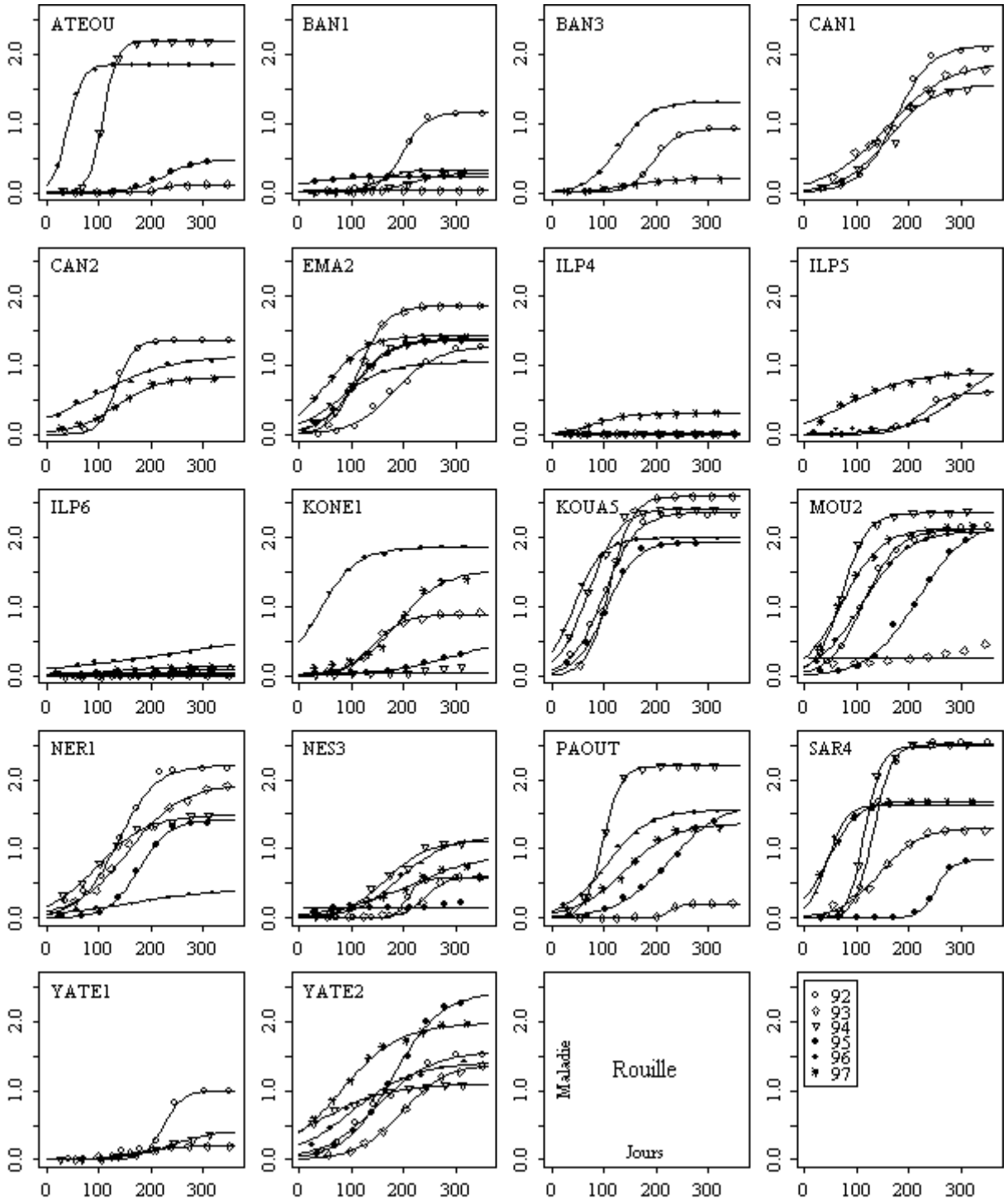


FIG. 2 – Composantes stationnelles du développement annuel de la rouille dans des parcelles de caféier en Nouvelle Calédonie. Données non publiées de D. Nandris et F. Pellerin (IRD).