Fiche TD avec le logiciel R : tdr4SS

Galerie des modèles non-linéaires de la classe `selfStart`

Pr Jean R. Lobry

Il y a 10 modèles non-linéaires de la classe `selfStart` disponibles en standard grâce aux contributions de José Pinheiro and Douglas Bates. Ils ont tous une fonction associée `getInitial()` qui utilise une estimation initiale automatique de la valeur des paramètres du modèle, ce qui facilite grandement l'estimation d'iceux par régression non-linéaire. L'objectif de cette fiche de TD est d'explorer graphiquement la signification des paramètres de ces modèles pour avoir une vue d'ensemble des courbes de réponse disponibles.

1 Introduction

1.1 Mise en œuvre

On veut estimer ici les paramètres du modèle de cinétique enzymatique de Michaelis et Menten [11] à partir des données publiées par Michaelis et Menten en 1913[1]. On utilise à cet effet la fonction `SSmicmen()` que l'on détaillera dans la section 10 page 17.

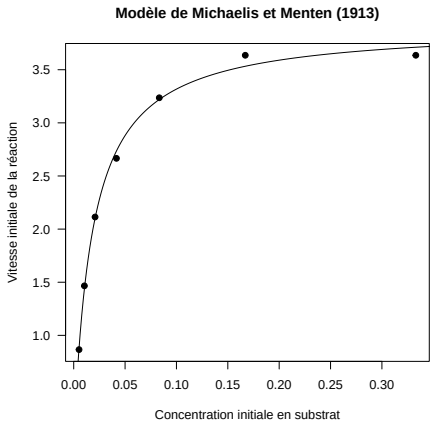
```
mm1913 <- structure(list(s = c(0.333, 0.167, 0.0833, 0.0416, 0.0208, 0.0104,
0.0052), v = c(3.636, 3.636, 3.236, 2.666, 2.114, 1.466, 0.866
)), class = "data.frame", row.names = c(NA, -7L))
mm1913
```

```
      s     v
1 0.3330 3.636
2 0.1670 3.636
3 0.0833 3.236
4 0.0416 2.666
5 0.0208 2.114
6 0.0104 1.466
7 0.0052 0.866
```

Le tableau de données `mm1913` ainsi défini comporte sept lignes et deux colonnes : `s` pour la concentration initiale en substrat et `v` pour la vitesse initiale de la réaction.

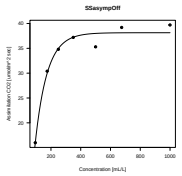
[1] Pour plus de détails voir la fiche de TD « [a]justement par régression non-linéaire du modèle de Michaelis-Menten aux données de Michaelis et Menten » à `http://pbil.univ-lyon1.fr/R/pdf/tdr47.pdf`

```
plot(v ~ s, data = mm1913, xlab = "Concentration initiale en substrat",
  ylab = "Vitesse initiale de la réaction", las = 1, pch = 19,
  main = "Modèle de Michaelis et Menten (1913)")
resnls <- nls(v ~ SSmicmen(s, Vm, K), data = mm1913)
s <- seq(0, 0.4, length.out = 256)
lines(s, predict(resnls, list(s = s)))
```

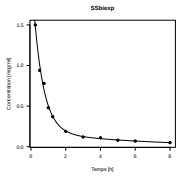


La fonction `plot()` produit le graphe des points expérimentaux. L'ajustement du modèle se fait avec la fonction `nls()`. La syntaxe est simple puisqu'elle reprend les notations utilisées avec les modèles linéaires, par exemple `y ~ x` pour obtenir la droite de régression linéaire qui prédit `y` en fonction de `x`. De façon analogue, `v ~ SSmicmen(s, Vm, K)` se lit la vitesse `v` prédite pour la concentration `s` par le modèle de Michaelis et Menten de paramètre `Vm` et `K`. Le nom des variables `v` et `s` doit correspondre au nom des colonnes dans le tableau de données indiqué par le paramètre `data`, soit `mm1913` dans notre cas. Pour tracer la courbe de réponse on a besoin de suffisamment de points pour donner l'illusion d'une courbe continue, avec 256 points c'est largement assez, d'où la construction du vecteur de concentration `s` en abscisse. Les valeurs correspondantes en ordonnée prédites par le modèle sont données par la fonction `predict()`, il ne reste plus qu'à relier les points par des lignes avec la fonction `lines()` pour ajouter la courbe du modèle.

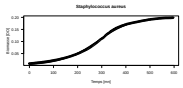
1.2 Typologie rapide



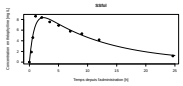
Les courbes qui s'écrasent au plafond : `SSasymp()` section 2 page 4, `SSasympOff()` section 3 page 5, `SSasympOrig()` section 4 page 7, `SSmicmen()` section 10 page 17 et `SSweibull()` section 11 page 18.



Celle qui s'écrase au plancher : `SSbiexp()` section 5 page 8, plus souple qu'une simple exponentielle décroissante.



Les courbes qui partent du plancher *puis* s'écrasent au plafond : `SSlogis()` section 9 page 13, `SSfpl()` section 7 page 10, `SSgompertz()` section 8 page 12



Celle qui n'est pas pas monotone, monte rapidement puis descend lentement : `SSfol()` section 6 page 9

1.3 La fonction `getInitial()`

Une petite remarque en passant. Tous les modèles de la classe `selfStart` ont une fonction associée `getInitial()` qui fait appel à l'attribut `initial`. Par exemple, dans le cas de la fonction `SSmicmen()` on a :

```
 attr(SSmicmen, "initial")
function (mCall, data, LHS, ...)
{
    xy <- sortedXyData(mCall[["input"]], LHS, data)
    if (nrow(xy) < 3) {
        stop("too few distinct input values to fit a Michaelis-Menten model")
    }
    pars <- as.vector(coef(lm(1/y ~ I(1/x), data = xy)))
    pars <- coef(nls(y ~ x/(K + x), data = xy, start = list(K = abs(pars[2L]/pars[1L])),
        algorithm = "plinear", ...))
    setNames(pars[c(".lin", "K")], mCall[c("Vm", "K")])
}
<environment: namespace:stats>
```

On reconnait ici la transformation en double inverse de Lineweaver et Burk [9] qui est utilisée ici pour avoir une première estimation initiale de la valeur des paramètres. La fonction `getInitial()` ne porte pas très bien son nom puisqu'elle enchaîne ensuite avec une régression non linéaire et renvoie finalement la valeur des paramètres après convergence. Il ne faudra pas s'étonner si `nls()` rapporte un nombre d'itérations nul, c'est simplement qu'elles ont été faites en amont.

```
 summary(resnls)
Formula: v ~ SSmicmen(s, Vm, K)
Parameters:
   Estimate Std. Error t value Pr(>|t|)
Vm 3.9109352  0.0557700   70.13 1.12e-08 ***
K  0.0178867  0.0009928   18.02 9.68e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

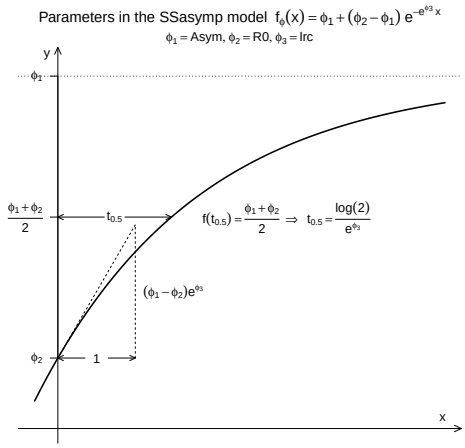
```
Residual standard error: 0.06719 on 5 degrees of freedom

Number of iterations to convergence: 0
Achieved convergence tolerance: 2.224e-06
```

2 Modèle asymptotique `SSasymp()`

C'est un modèle à trois paramètres (`Asym`, `R0` et `lrc`) dont la valeur vaut `Asym + (R0 - Asym)*exp(-exp(lrc)*input)`. C'est un cas particulier de modèle de croissance de Weibul (*cf.* `SSweibull()` page 18). En utilisant les notations données en annexe page 20 :

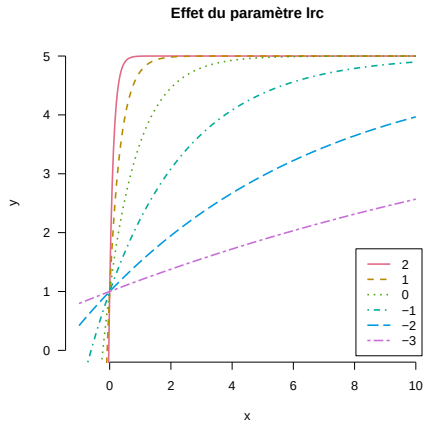
$$y = f(x) = y_\infty + (y_0 - y_\infty)e^{-e^{\ln \lambda} x} = y_\infty + (y_0 - y_\infty)e^{-\lambda x} \tag{1}$$

On a donc une branche d'exponentielle qui part de $f(0) = y_0$ et va s'écraser sur la droite d'équation $y = y_\infty$. L'exécution des exemples de la fonction `SSasymp()` produit un graphique qui met en évidence que l'on sera à mi-parcours pour $x_{0.5} = \frac{\ln 2}{\lambda}$:



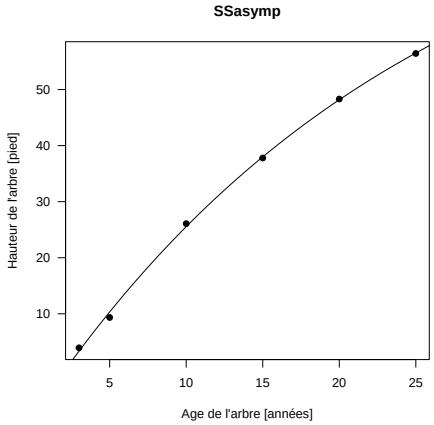
La significations des paramètres est donc la suivante. `Asym` représente la valeur de l'asymptote horizontale à droite, obtenue pour les très grandes valeurs de la variable de la fonction. `R0` contrôle la valeur initiale de la fonction. `lrc` est le logarithme népérien du taux de croissance, plus il est élevé, plus on s'écrase rapidement sur l'asymptote :

```
x <- seq(-1, 10, length = 255)
par(lwd = 2, lend = "butt") ; plot.new() ; plot.window(xlim = c(-1, 10), ylim = c(0, 5))
axis(1) ; axis(2, las = 1) ; title(main = "Effet du paramètre lrc", xlab = "x", ylab = "y")
lrcseq <- 2:-3 ; n <- length(lrcseq) ; mycol <- hcl.colors(n, "Dark 3")
for(i in seq_len(n)) points(x, SSasymp(x, Asym = 5, R0 = 1, lrc = lrcseq[i]), type = "l",
                            col = mycol[i], lty = i)
legend("bottomright", inset = 0.02, legend = lrcseq, lty = 1:n, col = mycol, box.lwd = 1)
```



La documentation de la fonction `SSasymp()` utilise le jeu de données standard `Loblolly` [7, 12] pour illustrer un exemple d'utilisation. Ce sont des données de croissance de pins loblolly (*Pinus taeda*) dont on ne conserve que celles issues de la graine 329.

```
data("Loblolly") ; Lob.329 <- subset(Loblolly, Seed == "329")
plot(height ~ age, data = Lob.329, pch = 19,
     xlab = "Age de l'arbre [années]", las = 1,
     ylab = "Hauteur de l'arbre [pied]", main = "SSasymp")
resnls <- nls(height ~ SSasymp(age, Asym, R0, lrc), data = Lob.329)
age <- seq(0, 30, length.out = 128)
lines(age, predict(resnls, list(age = age)))
```

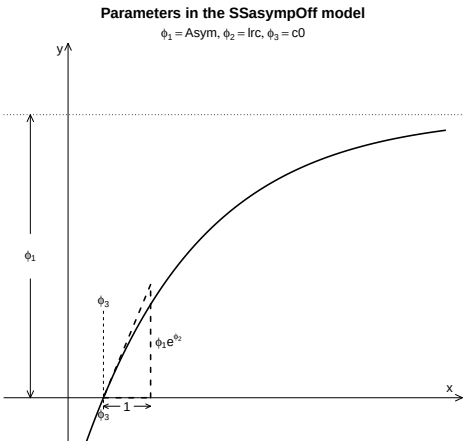


3 Modèle asymptotique avec décalage `SSasympOff()`

C'est un modèle à trois paramètres équivalent à `SSasymp()` page 4) mais avec une paramétrisation différente (`Asym`, `lrc` et `c0`) dont la valeur vaut `Asym*(1 - exp(-exp(lrc)*(input - c0)))`. En utilisant les notations données en annexe page 20 :

$$y = f(x) = y_\infty(1 - e^{-e^{\ln\lambda}(x-x_0)}) = y_\infty(1 - e^{-\lambda(x-x_0)}) \tag{2}$$

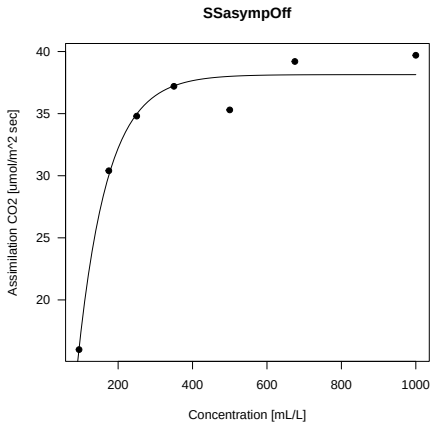
L'EXÉCUTION des exemples de la fonction `SSasympOff()` produit un graphique qui met en évidence que l'on a toujours une branche d'exponentielle qui part de $f(x_0) = 0$ et va s'écraser sur la droite d'équation $y = y_\infty$:



LA SIGNIFICATIONS des paramètres est donc la suivante. `Asym` représente la valeur de l'asymptote horizontale à droite, obtenue pour les très grandes valeurs de la variable de la fonction. `lrc` est le logarithme népérien du taux de croissance, plus il est élevé, plus on s'écrase rapidement sur l'asymptote. `c0` est la valeur pour laquelle la fonction est nulle.

LA documentation de la fonction `SSasympOff()` utilise le jeu de données standard `CO2` [13, 12] pour illustrer un exemple d'utilisation. Ce sont des données d'assimilation du gaz carbonique par *Echinochloa crus-galli* (le panic pied-de-coq) dont on ne conserve que celles issues de la plante `Qn1`.

```
data("CO2") ; CO2.Qn1 <- subset(CO2, Plant == "Qn1")
plot(uptake ~ conc, data = CO2.Qn1, pch = 19,
     xlab = "Concentration [mL/L]", las = 1,
     ylab = "Assimilation CO2 [umol/m^2 sec]", main = "SSasympOff")
resnls <- nls(uptake ~ SSasympOff(conc, Asym, lrc, c0), data = CO2.Qn1)
conc <- seq(0, 1000, length.out = 101)
lines(conc, predict(resnls, list(conc = conc)))
```

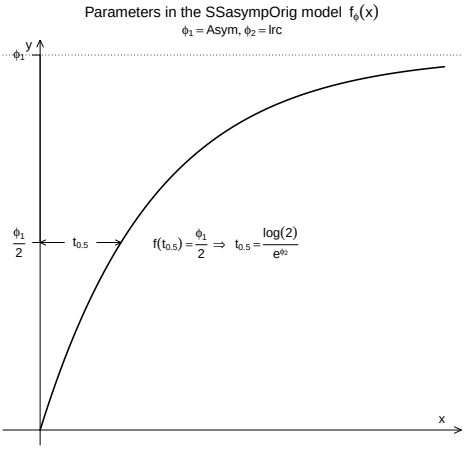


4 Modèle asymptotique via l'origine `SSasympOrig()`

C'est un modèle à deux paramètres (`Asym` et `lrc`) qui est un cas particulier de `SSasympOff()` page 5) avec le paramètre `c0` nul. Il vaut `Asym*(1 - exp(-exp(lrc)*input))` soit en utilisant les notations données en annexe page 20 :

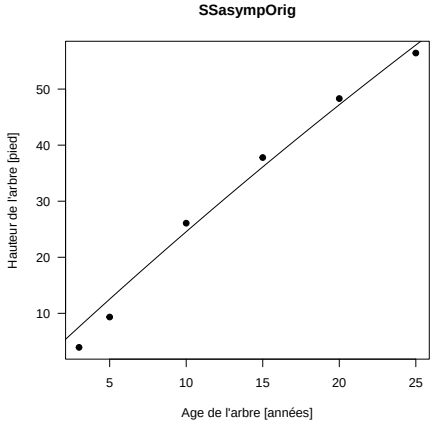
$$y = f(x) = y_\infty(1 - e^{-e^{\ln\lambda}x}) = y_\infty(1 - e^{-\lambda x}) \tag{3}$$

L'exécution des exemples de la fonction `SSasympOrig()` produit un graphique qui met en évidence que l'on a une branche d'exponentielle passant par l'origine et que l'on est à mi-hauteur pour $x_{0.5} = \frac{\ln 2}{\lambda}$:



La documentation de la fonction `SSasympOrig()` utilise le jeu de données standard `Loblolly` [7, 12] pour illustrer un exemple d'utilisation. Ce sont des données de croissance de pins loblolly (*Pinus taeda*) dont on ne conserve que celles issues de la graine 329.

```
data("Loblolly") ; Lob.329 <- subset(Loblolly, Seed == "329")
plot(height ~ age, data = Lob.329, pch = 19,
     xlab = "Age de l'arbre [années]", las = 1,
     ylab = "Hauteur de l'arbre [pied]", main = "SSasympOrig")
resnls <- nls(height ~ SSasympOrig(age, Asym, lrc), data = Lob.329)
age <- seq(0, 30, length.out = 101)
lines(age, predict(resnls, list(age = age)))
```

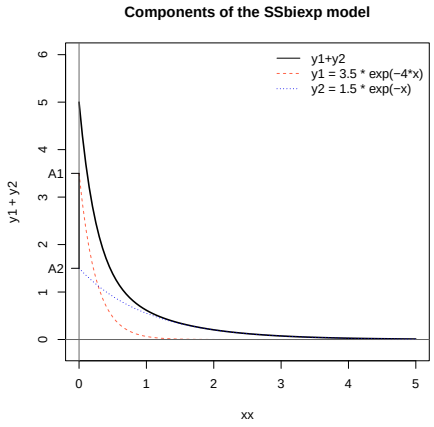


5 Le modèle bi-exponentiel `SSbiexp()`

C'est un modèle à 4 paramètres (`A1`, `lrc1`, `A2` et `A2`) dont la valeur est `A1*exp(-exp(lrc1)*input)+A2*exp(-exp(lrc2)*input)`. En utilisant les notations données en annexe page 20 :

$$y = f(x) = \alpha_1 e^{-e^{\ln \lambda_1} x} + \alpha_2 e^{-e^{\ln \lambda_2} x} = \alpha_1 e^{-\lambda_1 x} + \alpha_2 e^{-\lambda_2 x} \tag{4}$$

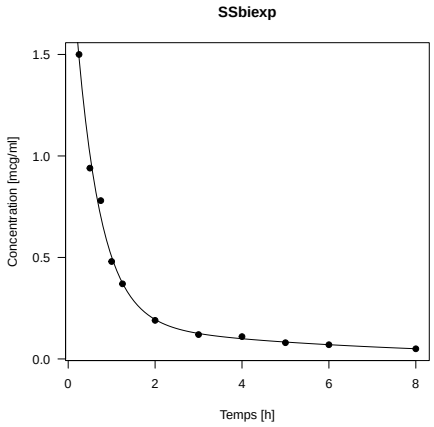
L'exécution des exemples de la fonction `SSbiexp()` produit un graphique qui met en évidence que l'on a comme courbe de réponse la somme de deux exponentielles décroissantes :



La documentation de la fonction `SSbiexp()` utilise le jeu de données standard `Indometh` [8, 4, 12] pour illustrer un exemple d'utilisation. Ce sont des données de pharmacocinétique de l'indomethacine dont on ne conserve que celles du premier patient.

```
data("Indometh") ; Indo.1 <- subset(Indometh, Subject == 1)
plot(conc ~ time, data = Indo.1, pch = 19,
     xlab = "Temps [h]", las = 1,
     ylab = "Concentration [mcg/ml]", main = "SSbiexp")
```

```
resnls <- nls(conc ~ SSbiexp(time, A1, lrc1, A2, lrc2), data = Indo.1)
time <- seq(0, 8, length.out = 101)
lines(time, predict(resnls, list(time = time)))
```



6 Modèle à compartiment d'ordre 1 `SSfol()`

C'est un modèle à 3 paramètres (`lKe`, `lKa` et `lCl`) dont la valeur est `Dose * exp(lKe + lKa - lCl) * (exp(-exp(lKe) * input) - exp(-exp(lKa) * input)) / (exp(lKa) - exp(lKe))`. La constante `Dose` représente la dose initiale injectée. En utilisant les notations données en annexe page 20 :

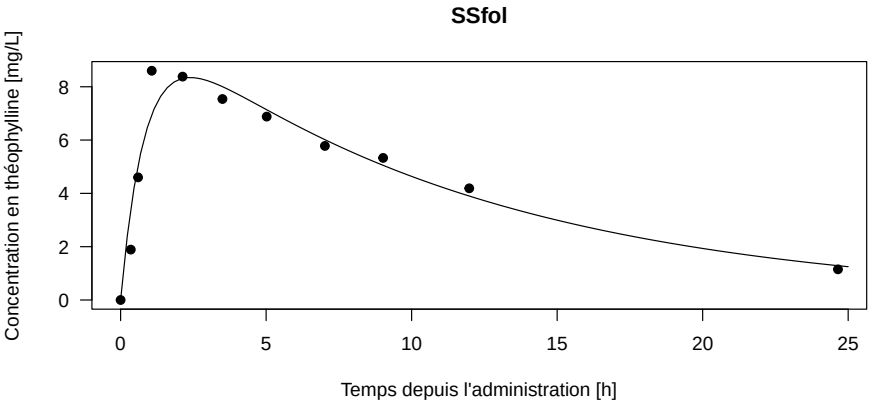
$$y = f(x) = De^{\ln K_e + \ln K_a - \ln Cl} \frac{e^{-e^{\ln K_e} x} - e^{-e^{\ln K_a} x}}{e^{\ln K_a} - e^{\ln K_e}} \tag{5}$$

Soit :

$$y = f(x) = D \frac{K_e K_a}{Cl} \frac{e^{-K_e x} - e^{-K_a x}}{K_a - K_e} \tag{6}$$

La documentation de la fonction `SSfol()` utilise le jeu de données standard `Theoph` [2, 4, 12] pour illustrer un exemple d'utilisation. Ce sont des données de pharmacocinétique de la Théophylline dont on ne conserve que celles du quatrième patient.

```
data("Theoph") ; Theoph.4 <- subset(Theoph, Subject == 4)
plot(conc ~ Time, data = Theoph.4,
     xlab = "Temps depuis l'administration [h]",
     ylab = "Concentration en théophylline [mg/L]",
     las = 1, pch = 19, main = "SSfol")
resnls <- nls(conc ~ SSfol(Dose, Time, lKe, lKa, lCl), data = Theoph.4)
Time <- seq(0, 25, length.out = 10*nrow(Theoph.4))
lines(Time, predict(resnls, list(Time = Time)))
```



Une remarque en passant : pour des raisons que je n'ai pas bien comprises, dans le cas de l'utilisation de `SSfol()` quand on veut tracer le modèle il faut que le vecteur des valeurs des abscisses pour la courbe (`xx` ci-dessus) soit d'une longueur multiple du nombre de lignes du tableau utilisé pour l'estimation des paramètres. Ce n'est pas bloquant : on aura juste un message d'avis du type :

```
Messages d'avis :
1: Dans .expr4 * (.expr8 - .expr12) :
  la taille d'un objet plus long n'est pas multiple de la taille d'un objet plus court
2: Dans .expr4 * (.expr8 * (.expr5 * input)) :
  la taille d'un objet plus long n'est pas multiple de la taille d'un objet plus court
3: Dans .expr4 * (.expr12 * (.expr9 * input)) :
  la taille d'un objet plus long n'est pas multiple de la taille d'un objet plus court
```

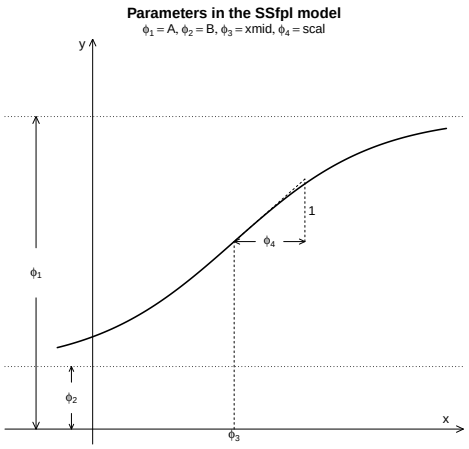
C'est un modèle à compartiment classique pour rendre compte de l'évolution de la concentration sanguine d'une substance administrée (hors intraveineuse directe). Le paramètre K_a représente le taux de diffusion entre le compartiment d'administration et le plasma sanguin, et le paramètre K_e le taux d'élimination du plasma sanguin (la fonction rénale en première approximation). Le modèle n'a de sens concret que pour $K_a > K_e$ sans quoi la concentration plasmatique serait toujours nulle, ce qui ne présente pas d'intérêt thérapeutique. Il n'y a pas d'interprétation graphique directe des paramètres du modèle.

7 Modèle de croissance logistique avec asymptote à gauche `SSfpl()`

C'est un modèle à 4 paramètres (`A`, `B`, `xmid` et `scal`) qui vaut `A + (B - A)/(1 + exp((xmid - input)/scal))`. Dans le cas où le paramètre `A` est nul on retrouve le modèle de croissance logistique de Verhulst `SSlogis()` étudié page 13. En utilisant les notations données en annexe page 20 :

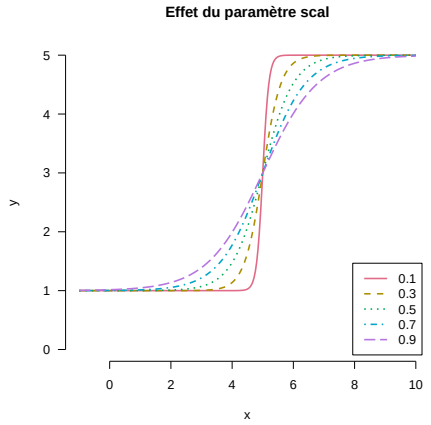
$$y = f(x) = y_{-\infty} + \frac{y_{\infty} - y_{-\infty}}{1 + e^{\frac{x_{0.5} - x}{\sigma}}} \tag{7}$$

L'exécution des exemples de la fonction `SSfpl()` produit un graphique qui met en évidence que l'on a comme courbe de réponse une sigmoïde :



LA SIGNIFICATIONS des paramètres est donc la suivante. `A` représente la valeur de l'asymptote horizontale à gauche, obtenue pour les très faibles valeurs de la variable de la fonction. `B` représente la valeur de l'asymptote horizontale à droite, obtenue pour les très grandes valeurs de la variable de la fonction. `xmid` est un paramètre de position qui donne la valeur du point d'inflexion. La valeur de la courbe au point d'inflexion est à mi-distance entre les deux asymptotes. `scale` contrôle la rapidité de la transition, plus il est petit, plus elle sera brutale :

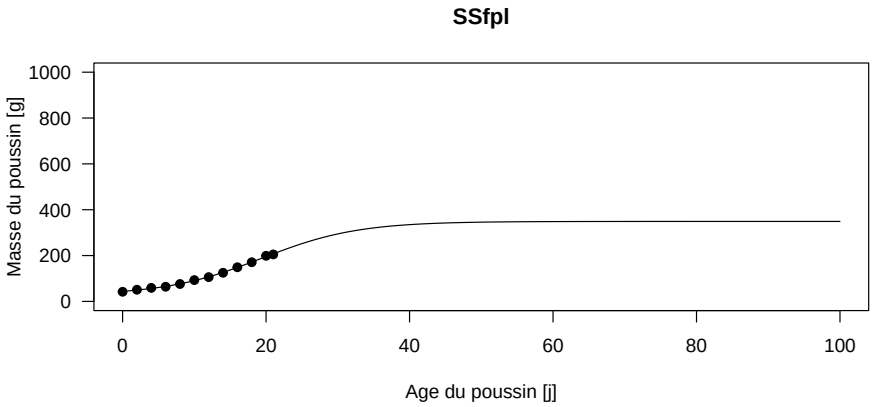
```
x <- seq(-1, 10, length = 255)
par(lwd = 2, lend = "butt") ; plot.new() ; plot.window(xlim = c(-1, 10), ylim = c(0, 5))
axis(1) ; axis(2, las = 1) ; title(main = "Effet du paramètre scal", xlab = "x", ylab = "y")
scaleseq <- seq(0.1, 1, by = 0.2) ; n <- length(scaleseq) ; mycol <- hcl.colors(n, "Dark 3")
for(i in seq_len(n)) points(x, SSfpl(x, A = 1, B = 5, xmid = 5, scal = scaleseq[i]), type = "l",
                            col = mycol[i], lty = i)
legend("bottomright", inset = 0.02, legend = scaleseq, lty = 1:n, col = mycol, box.lwd = 1)
```



LA documentation de la fonction `SSfpl()` utilise le jeu de données standard `ChickWeight` [3, 6, 12] pour illustrer un exemple d'utilisation. Ce sont des données de croissance pondérale de poussins dont on ne conserve que celles issues du premier individu.

```
data("ChickWeight") ; Chick.1 <- subset(ChickWeight, Chick == 1)
plot(weight ~ Time, data = Chick.1, pch = 19,
     xlab = "Age du poussin [j]", las = 1, xlim = c(0, 100), ylim = c(0, 1000),
     ylab = "Masse du poussin [g]", main = "SSfpl")
```

```
resnls <- nls(weight ~ SSfpl(Time, A, B, xmid, scal), data = Chick.1)
Time <- seq(0, 100, length.out = 101)
lines(Time, predict(resnls, list(Time = Time)))
```



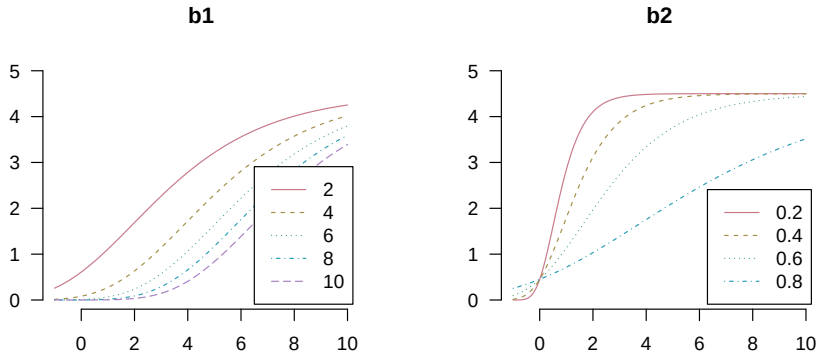
8 Le modèle de croissance de Gompertz `SSgompertz()`

C'est un modèle à 3 paramètres (`Asym`, `b2` et `b3`) dont la valeur est `Asym*exp(-b2*b3^x)`. Il s'agit du modèle de croissance de Gompertz [5]. En utilisant les notations données en annexe page 20 :

$$y = f(x) = y_\infty e^{-b_2 b_3^x} \tag{8}$$

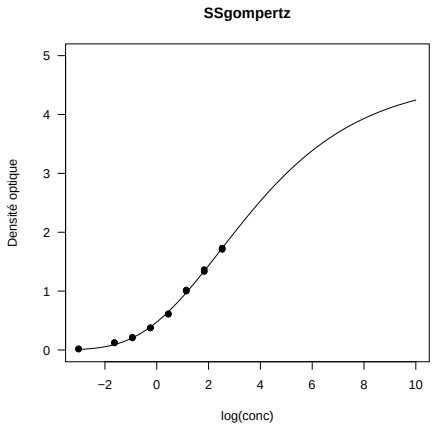
La signification des paramètres est la suivante. `Asym` représente la valeur de l'asymptote horizontale à droite, obtenue pour les très grandes valeurs de la variable de la fonction. `b1` est un paramètre lié à la valeur de la fonction à l'origine. `b2` est un paramètre lié à l'échelle de l'axe des abscisses.

```
par(mfrow = c(1, 2))
plot.new() ; plot.window(xlim = range(x), ylim = c(0, 5)) ; axis(1) ; axis(2, las = 1)
title(main = "b1")
b1seq <- seq(2, 10, by = 2) ; n <- length(b1seq) ; mycol <- hcl.colors(n, "Dark 2")
for(i in seq_len(n))
  lines(x, SSgompertz(x, 4.5, b1seq[i], 0.7), lty = i, col = mycol[i])
legend("bottomright", inset = 0.02, legend = b1seq, lty = 1:n, col = mycol, bg = "white")
plot.new() ; plot.window(xlim = range(x), ylim = c(0, 5)) ; axis(1) ; axis(2, las = 1)
title(main = "b2")
b2seq <- seq(0.2, 0.8, by = 0.2) ; n <- length(b1seq) ; mycol <- hcl.colors(n, "Dark 2")
for(i in seq_len(n))
  lines(x, SSgompertz(x, 4.5, 2.3, b2seq[i]), lty = i, col = mycol[i])
legend("bottomright", inset = 0.02, legend = b2seq, lty = 1:n, col = mycol, bg = "white")
```



La documentation de la fonction `SSgompertz()` utilise le jeu de données standard `DNase` [4, 12] pour illustrer un exemple d'utilisation. Ce sont les données d'un test ELISA dont on ne conserve que celles issues de la première expérience.

```
data("DNase") ; DNase.1 <- subset(DNase, Run == 1)
plot(density ~ log(conc), data = DNase.1, pch = 19,
     xlab = "log(conc)", las = 1, xlim = c(-3, 10), ylim = c(0, 5),
     ylab = "Densité optique", main = "SSgompertz")
resnls <- nls(density ~ SSgompertz(log(conc), Asym, b2, b3), data = DNase.1)
conc <- exp(seq(-3, 10, length.out = 256))
lines(log(conc), predict(resnls, list(conc = conc)))
```

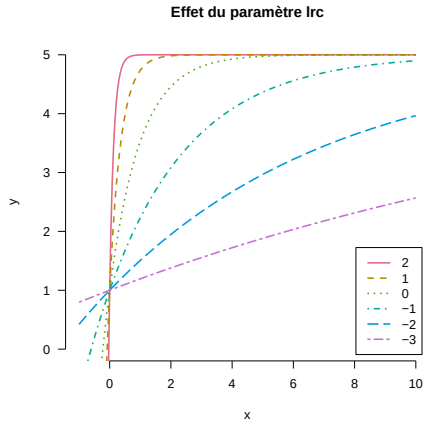


9 Modèle de croissance logistique de Verhulst `SSlogis()`

C'est un modèle à 3 paramètres (`Asym`, `xmid` et `scal`) dont la valeur est `Asym/(1+exp((xmid-input)/scal))`. Il s'agit du modèle de croissance de Verhulst [15, 16, 17, 18]. En utilisant les notations données en annexe page 20 :

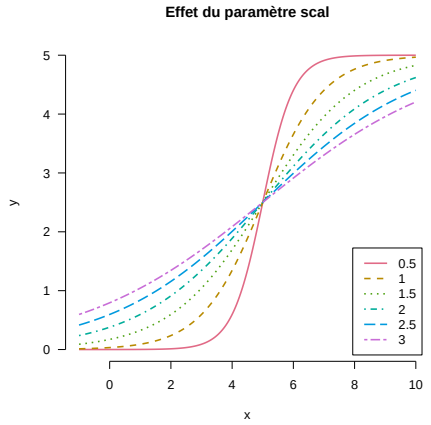
$$y = f(x) = \frac{y_\infty}{1 + e^{\frac{x_{0.5} - x}{\sigma}}} \tag{9}$$

L'EXÉCUTION des exemples de la fonction `SSlogis()` produit un graphique qui met en évidence que l'on a comme courbe de réponse une sigmoïde :



LA SIGNIFICATION des paramètres est donc la suivante. `Asym` représente la valeur de l'asymptote horizontale à droite, obtenue pour les très grandes valeurs de la variable de la fonction. `xmid` est un paramètre de position qui donne la valeur du point d'inflexion. La valeur de la courbe au point d'inflexion vaut la moitié de la valeur de l'asymptote horizontale à droite. `scale` contrôle la rapidité de la transition, plus il est petit, plus elle sera brutale :

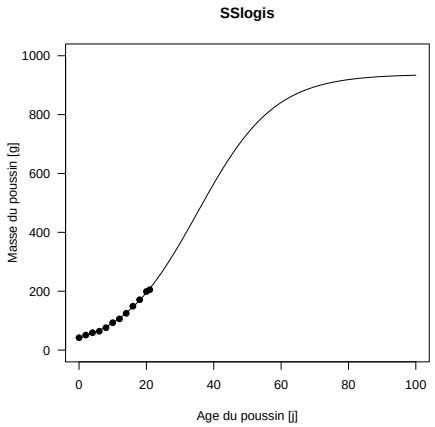
```
x <- seq(-1, 10, length = 255)
par(lwd = 2, lend = "butt") ; plot.new() ; plot.window(xlim = c(-1, 10), ylim = c(0, 5))
axis(1) ; axis(2, las = 1) ; title(main = "Effet du paramètre scal", xlab = "x", ylab = "y")
scaleseq <- seq(0.5, 3, by = 0.5) ; n <- length(scaleseq) ; mycol <- hcl.colors(n, "Dark 3")
for(i in seq_len(n)) points(x, SSlogis(x, Asym = 5, xmid = 5, scal = scaleseq[i]), type = "l",
                            col = mycol[i], lty = i)
legend("bottomright", inset = 0.02, legend = scaleseq, lty = 1:n, col = mycol, box.lwd = 1)
```



LA documentation de la fonction `SSlogis()` utilise le jeu de données standard `ChickWeight` [3, 6, 12] pour illustrer un exemple d'utilisation. Ce sont des

données de croissance pondérale de poussins dont on ne conserve que celles issues du premier individu.

```
data("ChickWeight") ; Chick.1 <- subset(ChickWeight, Chick == 1)
plot(weight ~ Time, data = Chick.1, pch = 19,
     xlab = "Age du poussin [j]", las = 1, xlim = c(0, 100), ylim = c(0, 1000),
     ylab = "Masse du poussin [g]", main = "SSlogis")
resnls <- nls(weight ~ SSlogis(Time, Asym, xmid, scal), data = Chick.1)
Time <- seq(0, 100, length.out = 101)
lines(Time, predict(resnls, list(Time = Time)))
```



Même si la valeur de l'asymptote n'est pas complètement aberrante (de l'ordre du kilogramme pour un adulte) il faut bien réaliser que l'estimation ne peut être que très imprécise quand on a comme ici que des valeurs en amont du point d'inflexion. L'erreur standard sur l'estimation de la valeur du paramètre `Asym` est par conséquent très élevée :

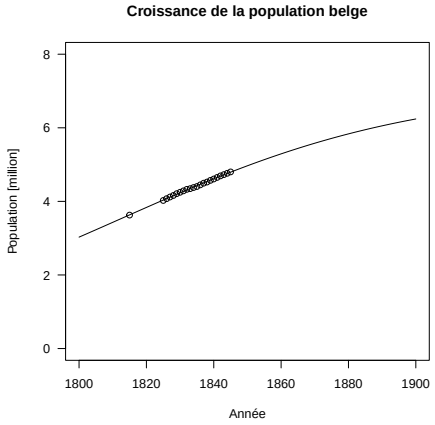
```
summary(resnls)
```

```
Formula: weight ~ SSlogis(Time, Asym, xmid, scal)
Parameters:
     Estimate Std. Error t value Pr(>|t|)
Asym 937.0298   465.8675   2.011  0.07516 .
xmid  35.2230     8.3120   4.238  0.00218 **
scal  11.4052     0.9052  12.599 5.08e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.919 on 9 degrees of freedom

Number of iterations to convergence: 0
Achieved convergence tolerance: 7.244e-06
```

On aura le même problème avec les données originelles de Verhulst sur la croissance de la population belge. Elles sont extraites du tableau page 8 de son mémoire [18] lu à la séance de l'Académie royale de Belgique du 15 mai 1846 et reprises ici dans l'objet `ver` :

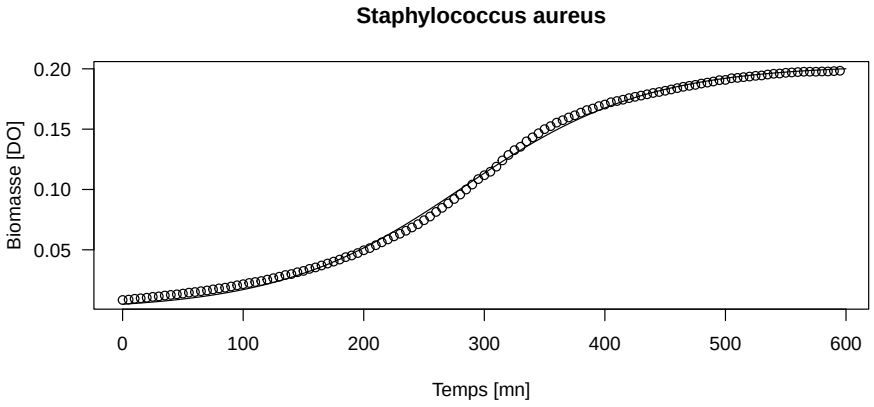
```
ver <- structure(list(t = c(1815L, 1825L, 1826L, 1827L, 1828L, 1829L,
1830L, 1831L, 1832L, 1833L, 1834L, 1835L, 1836L, 1837L, 1838L,
1839L, 1840L, 1841L, 1842L, 1843L, 1844L, 1845L), obs = c(3627253L,
4024855L, 4073751L, 4118840L, 4160279L, 4210128L, 4247113L, 4285969L,
4326697L, 4345940L, 4376214L, 4404220L, 4448769L, 4494688L, 4525687L,
4570708L, 4608776L, 4650400L, 4693190L, 4727391L, 4763246L, 4800861L
), theo = c(3627300L, 4048200L, 4088600L, 4128600L, 4168500L,
4207900L, 4247100L, 4286000L, 4324600L, 4362900L, 4400900L, 4438600L,
4476000L, 4513100L, 4549900L, 4586400L, 4622600L, 4658500L, 4694100L,
4729400L, 4764400L, 4799100L)), class = "data.frame", row.names = c(NA,
-22L))
```

```
ver$obs <- ver$obs/10^6
ans <- 1800:1900
plot(obs ~ t, data = ver, xlab = "Année", ylab = "Population [million]",
     las = 1, pch = 1, main = "Croissance de la population belge",
     xlim = c(1800, 1900), ylim = c(0, 8))
resnls <- nls(obs ~ SSlogis(t, Asym, xmid, scal), data = ver)
lines(ans, predict(resnls, list(t = ans)))
```



Pour pouvoir estimer correctement la valeur de l'asymptote il faut un jeu de données avec des valeurs qui s'en approchent. Voici un exemple avec la courbe de croissance de *Staphylococus aureus* (extrait de l'annexe 6.7 de [10]) :

```
saureus <- structure(list(t = c(0L, 5L, 10L, 15L, 20L, 25L, 30L, 35L, 40L,
45L, 50L, 55L, 60L, 65L, 70L, 75L, 80L, 85L, 90L, 95L, 100L,
105L, 110L, 115L, 120L, 125L, 130L, 135L, 140L, 145L, 150L, 155L,
160L, 165L, 170L, 175L, 180L, 185L, 190L, 195L, 200L, 205L, 210L,
215L, 220L, 225L, 230L, 235L, 240L, 245L, 250L, 255L, 260L, 265L,
270L, 275L, 280L, 285L, 290L, 295L, 300L, 305L, 310L, 315L, 320L,
325L, 330L, 335L, 340L, 345L, 350L, 355L, 360L, 365L, 370L, 375L,
380L, 385L, 390L, 395L, 400L, 405L, 410L, 415L, 420L, 425L, 430L,
435L, 440L, 445L, 450L, 455L, 460L, 465L, 470L, 475L, 480L, 485L,
490L, 495L, 500L, 505L, 510L, 515L, 520L, 525L, 530L, 535L, 540L,
545L, 550L, 555L, 560L, 565L, 570L, 575L, 580L, 585L, 590L, 595L
), x = c(0.0084, 0.0087, 0.0094, 0.0098, 0.0103, 0.011, 0.0114,
0.0121, 0.0126, 0.0131, 0.0137, 0.0145, 0.015, 0.0157, 0.0163,
0.0172, 0.018, 0.0186, 0.0196, 0.0206, 0.0215, 0.0225, 0.0233,
0.0241, 0.0253, 0.0265, 0.0277, 0.0291, 0.0299, 0.0314, 0.0325,
0.0342, 0.0354, 0.037, 0.0386, 0.0402, 0.0419, 0.0439, 0.0453,
0.0473, 0.0496, 0.0514, 0.0538, 0.0562, 0.0586, 0.0611, 0.0633,
0.0658, 0.0685, 0.0712, 0.0745, 0.0776, 0.0814, 0.0848, 0.0884,
0.0921, 0.0958, 0.1, 0.1041, 0.1086, 0.1118, 0.1147, 0.1189,
0.1241, 0.1286, 0.1326, 0.1354, 0.1399, 0.1432, 0.1466, 0.15,
0.1525, 0.1554, 0.1574, 0.1597, 0.1615, 0.1637, 0.1658, 0.1671,
0.1691, 0.1704, 0.1723, 0.1733, 0.1744, 0.1757, 0.1768, 0.1778,
0.1789, 0.18, 0.1808, 0.1818, 0.1829, 0.184, 0.1851, 0.1859,
0.1867, 0.1878, 0.1886, 0.1895, 0.1906, 0.1908, 0.1922, 0.1925,
0.1931, 0.1936, 0.1942, 0.1945, 0.1953, 0.1959, 0.1961, 0.1967,
0.197, 0.1973, 0.1976, 0.1976, 0.1976, 0.1978, 0.1978, 0.1981,
0.1984)), class = "data.frame", row.names = c(NA, -120L))
plot(x ~ t, data = saureus, xlab = "Temps [mn]", ylab = "Biomasse [DO]",
     las = 1, pch = 1, main = "Staphylococcus aureus")
resnls <- nls(x ~ SSlogis(t, Asym, xmid, scal), data = saureus)
t <- seq(0, 600, length.out = 255)
lines(t, predict(resnls, list(t = t)))
```

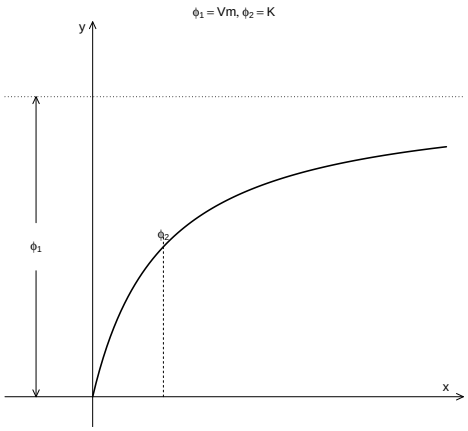


10 Le modèle de Michaelis et Menten `SSmicmen()`

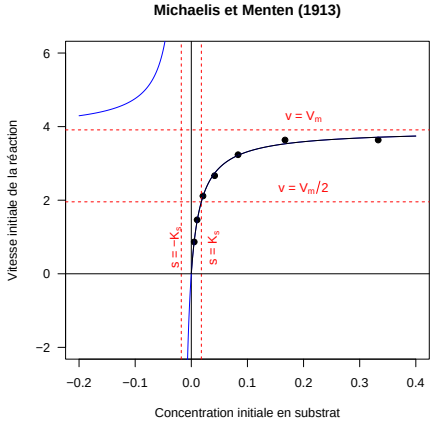
C'est un modèle à 2 paramètres (`Vm` et `K`) dont la valeur est `Vm*input/(K+input)`. Il s'agit du modèle de cinétique enzymatique de Michaelis et Menten [11] qui donne la vitesse initiale de la réaction en fonction de la concentration en substrat. En utilisant les notations données en annexe page 20 :

$$y = f(x) = \frac{y_\infty x}{K_s + x} \tag{10}$$

L'exécution des exemples de la fonction `SSmicmen()` produit un graphique qui met en évidence que l'on a comme courbe de réponse une branche d'hyperbole équilatère :

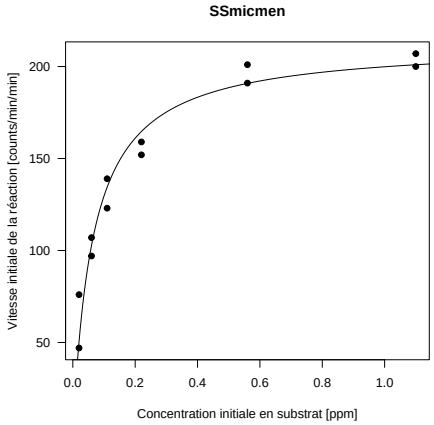


La signification des paramètres est donc la suivante. `Vm` représente la valeur de l'asymptote horizontale à droite, obtenue pour les très grandes valeurs de la variable de la fonction. C'est la vitesse maximale de la réaction (V_m). `K` est la valeur de la variable telle que la vitesse soit demi-maximale. C'est la constante de saturation K_s souvent appelée paramètre de Michaelis en enzymologie. Une autre façon de comprendre l'effet du paramètre K_s c'est de voir qu'il contrôle la position de asymptote verticale en $-K_s$:



La documentation de la fonction `SSmicmen()` utilise le jeu de données standard `Puromycin` [1, 14] pour illustrer un exemple d'utilisation. Ce sont des données de cinétique enzymatique dont on ne conserve que celles issues des cellules traitées à la puromycine.

```
data("Puromycin") ; PurTrt <- Puromycin[ Puromycin$state == "treated", ]
plot(rate ~ conc, data = PurTrt, pch = 19,
     xlab = "Concentration initiale en substrat [ppm]", las = 1,
     ylab = "Vitesse initiale de la réaction [counts/min/min]", main = "SSmicmen")
resnls <- nls(rate ~ SSmicmen(conc, Vm, K), data = PurTrt)
conc <- seq(0, 1.2, length.out = 256)
lines(conc, predict(resnls, list(conc = conc)))
```



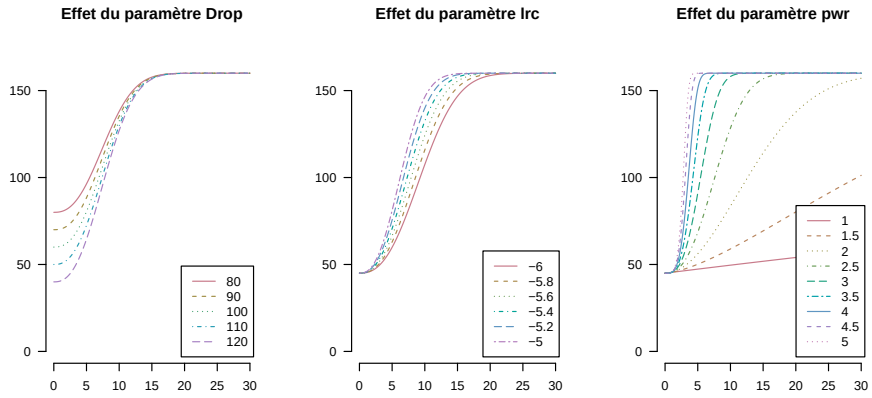
11 Le modèle de croissance de Weibull `SSweibull()`

C'est un modèle à 4 paramètres (`Asym`, `Drop`, `lrc` et `pwr`) dont la valeur est `Asym-Drop*exp(-exp(lrc)*x^pwr)`. Il s'agit du modèle de Weibull initialement publié en tant que distribution statistique [19]. En utilisant les notations données en annexe page 20 :

$$y = f(x) = y_\infty - \delta e^{-e^{\ln \lambda} x^n} = y_\infty - \delta e^{-\lambda x^n} \tag{11}$$

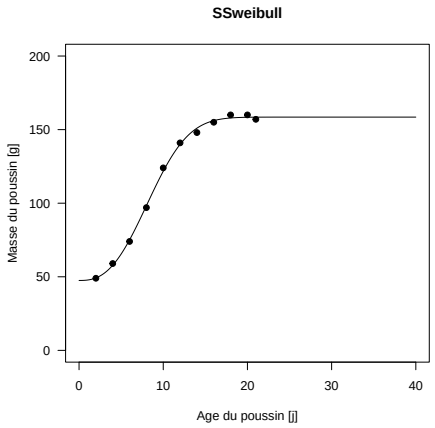
La signification des paramètres est illustrée dans la figure ci-après. `Asym` représente la valeur de l'asymptote horizontale à droite, obtenue pour les très grandes valeurs de la variable de la fonction. `Drop` contrôle la valeur de l'ordonnée à l'origine, puis il est élevé plus elle est faible. `lrc` contrôle la vitesse à laquelle on s'écrase sur l'asymptote. `pwr` ajoute de la flexibilité en modulant la forme de la courbe.

```
par(mfrow = c(1, 3))
Asym <- 160 ; Drop <- 115 ; lrc <- -5.5 ; pwr <- 2.5
x <- seq(0, 30, le = 256)
pseq <- seq(80, 120, by = 10) ; n <- length(pseq) ; col <- hcl.colors(n, "Dark 2")
plot.new() ; plot.window(xlim = range(x), ylim = c(0, 170))
axis(1) ; axis(2, las = 1) ; title(main = "Effet du paramètre Drop")
for(i in seq_len(n)) lines(x, SSweibull(x, Asym, pseq[i], lrc, pwr), col = col[i], lty = i)
legend("bottomright", inset = 0.02, legend = pseq, col = col, lty = 1:n)
#
pseq <- seq(-6, -5, by = 0.2) ; n <- length(pseq) ; col <- hcl.colors(n, "Dark 2")
plot.new() ; plot.window(xlim = range(x), ylim = c(0, 170))
axis(1) ; axis(2, las = 1) ; title(main = "Effet du paramètre lrc")
for(i in seq_len(n)) lines(x, SSweibull(x, Asym, Drop, pseq[i], pwr), col = col[i], lty = i)
legend("bottomright", inset = 0.02, legend = pseq, col = col, lty = 1:n)
#
pseq <- seq(1, 5, by = 0.5) ; n <- length(pseq) ; col <- hcl.colors(n, "Dark 2")
plot.new() ; plot.window(xlim = range(x), ylim = c(0, 170))
axis(1) ; axis(2, las = 1) ; title(main = "Effet du paramètre pwr")
for(i in seq_len(n)) lines(x, SSweibull(x, Asym, Drop, lrc, pseq[i]), col = col[i], lty = i)
legend("bottomright", inset = 0.02, legend = pseq, col = col, lty = 1:n, bg = "white")
```



La documentation de la fonction `SSweibull()` utilise le jeu de données standard `ChickWeight` [3, 6, 12] pour illustrer un exemple d'utilisation. Ce sont des données de croissance pondérale de poussins dont on ne conserve que celles issues du sixième individu pour les temps strictement positifs.

```
Chick.6 <- subset(ChickWeight, (Chick == 6) & (Time > 0))
plot(weight ~ Time, data = Chick.6, pch = 19,
     xlab = "Age du poussin [j]", las = 1, xlim = c(0, 40), ylim = c(0, 200),
     ylab = "Masse du poussin [g]", main = "SSweibull")
resnls <- nls(weight ~ SSweibull(Time, Asym, Drop, lrc, pwr), data = Chick.6)
Time <- seq(0, 40, length.out = 256)
lines(Time, predict(resnls, list(Time = Time)))
```



12 Annexe : notations

A : $y_{-\infty}$

B : y_{∞}

b2, b3 : b_2, b_3

A1, A2 : α_1, α_2

Asym : y_{∞}

c0 : x_0

Dose : D

Drop : δ

input : x

K : K_s

lrc, lrc1, lrc2 : $\ln \lambda$, $\ln \lambda_1$, $\ln \lambda_2$

lKe, lKa, lCl : $\ln K_e$, $\ln K_a$, $\ln\ Cl$

pwr : n

R0 : y_0

scale : σ

Vm : y_{∞}

xmid : $x_{0.5}$

References

[1] D.M. Bates and D.G. Watts. *Nonlinear Regression Analysis and Its Applications*. Wiley, New York, USA, 1988.

[2] A.J. Boeckmann, L.B. Sheiner, and S.L. Beal. NONMEM users guide: Part V. Technical report, NONMEM Project Group, University of California, San Francisco, USA, 1994.

[3] M. Crowder and D. Hand. *Analysis of Repeated Measures*. Chapman and Hall, 1990.

[4] M. Davidian and D.M. Giltinan. *Nonlinear Models for Repeated Measurement Data*. Chapman and Hall, 1995.

[5] B. Gompertz. On the nature of the function expressive of the law of human mortality, and on a new mode of determining the value of life contingencies. *Philosophical transactions of the Royal Society of London*, 115:513–583, 1825.

[6] D. Hand and M. Crowder. *Practical Longitudinal Data Analysis*. Chapman and Hall, 1996.

[7] F.H. Kung. Fitting logistic growth curve with predetermined carrying capacity. *Proceedings of the Statistical Computing Section, American Statistical Association*, pages 340–343, 1986.

[8] K.C. Kwan, G.O. Breault, E.R. Umbenhauer, F.G. McMahon, and D.E. Duggan. Kinetics of indomethicin absorption, elimination and enterohepatic circulation in man. *Journal of Pharmacokinetics and Biopharmaceutics*, 4:255–280, 1976.

[9] H. Lineweaver and D. Burk. The determination of enzyme dissociation constants. *Journal of the American Chemical Society*, 56:658–666, 1934.

[10] 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.

[11] L. Michaelis and M.L. Menten. Die Kinetik der Invertinwirkung. *Biochemische Zeitschrift*, 49:333–369, 1913.

[12] J.C. Pinheiro and D.M. Bates. *Mixed-effects Models in S and S-PLUS*. Springer-Verlag, New York, USA, 2000.

[13] C. Potvin, M.J. Lechowicz, and S. Tardif. The statistical analysis of ecophysiological response curves obtained from experiments involving repeated measures. *Ecology*, 71:1389–1400, 1990.

[14] M.A. Treloar. *Effects of Puromycin on Galactosyltansferase of Golgi Membranes*. PhD thesis, University of Toronto, 1974.

[15] P.-F. Verhulst. Notice sur la loi que la population suit dans son accroissement. *Correspondance mathématique et physique*, 10:113–121, 1838.

[16] P.-F. Verhulst. Recherches mathématiques sur la loi d'accroissement de la population. *Nouveaux mémoires de l'académie royale des sciences, des lettres et des beaux-arts de Belgique*, 18:1–38, 1845. Lu à la séance du 30 novembre 1844.

[17] P.-F. Verhulst. Note sur la loi d'accroissement de la population. *Bulletins de l'Académie Royale des Sciences et Belles-Lettres de Bruxelles*, 13, 1846.

[18] P.-F. Verhulst. Deuxième mémoire sur la loi d'accroissement de la population. *Mémoires de l'académie royale des sciences, des lettres et des beaux-arts de Belgique*, 20:1–32, 1847. Lu à la séance de l'Académie royale du 15 mai 1846.

[19] W. Weibull. A statistical distribution function of wide applicability. *Journal of Applied Mechanics*, 8:293–297, 1951.