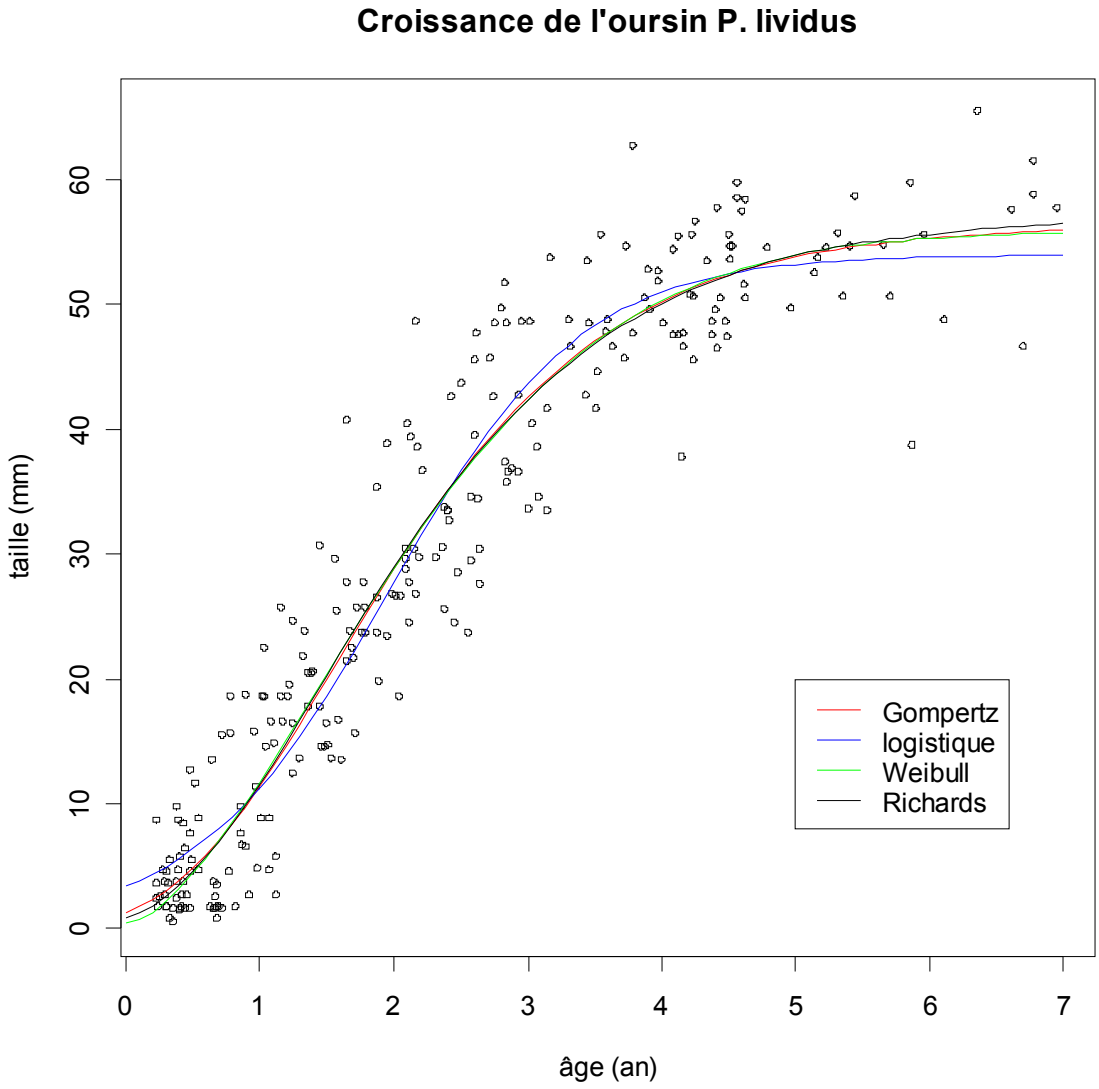
Université de Mons
Laboratoire d'Écologie
numérique des Milieux
aquatiques

Science des Données Biologiques



Philippe Grosjean, version 2.0 – novembre 2018

« A big computer, a complex algorithm and a long time does not equal science. »

-- Robert Gentleman

SSC 2003, Halifax (Juin 2003)

« Actually, I see it as part of my job to inflict R on people who are perfectly happy to have never heard of it. Happiness doesn't equal proficient and efficient. In some cases the proficiency of a person serves a greater good than their momentary happiness. »

-- Patrick Burns

R-help (April 2005)

Table of Contents

1. LA RÉGRESSION NON LINÉAIRE

Dans les cours précédents, vous avez étudié la régression linéaire. Celle-ci consiste à modéliser la variation d'une variable (dite variable réponse ou dépendante) par rapport à la variation d'une ou plusieurs autres variables (dites explicatives ou indépendantes). Le modèle utilisé pour représenter cette relation était la droite (y = a.x+b). Cette droite est, dans la technique la plus courante, ajustée en minimisant la somme des carrés des résidus (écart entre les observations y_i et les valeurs prédites par la droite, notées $\hat{y}_i$ - on dit aussi: les *valeurs estimées* par le modèle-).

Lorsque le nuage de point ne s'étire pas le long d'une droite, nous pouvons tenter de transformer les données afin de les linéariser. Malheureusement, il existe de nombreux cas où la relation n'est pas linéarisable. Nous allons étudier quelques uns de ces cas et nous verrons comment nous pouvons ajuster une fonction quelconque (non linéaire) sur ces données.

> *Il existe, en réalité, une autre raison pour laquelle nous pourrions être amené à ne pas transformer les données pour les linéariser. Il s'agit du cas où les résidus ont une distribution correcte avec les données non transformées (distribution normale, et variance homogène -homoscédasticité-) lorsqu'on utilise un modèle non linéaire. Dans ce cas précis, une transformation pour linéariser les données devrait permettre d'utiliser une régression linéaire. Mais ce faisant, on rend alors les résidus non normaux et/ou on perd la propriété d'homoscédasticité, ce qui constitue une violation des conditions d'application de la régression par les moindres carrés que nous utilisons ici. Ainsi, dans ce cas-là, il vaut alors mieux ne pas transformer et utiliser plutôt une régression non linéaire à la place d'une régression linéaire pourtant plus simple d'emploi.*

1.1. Principe de base: minimisation de la fonction objective

Nous appelons « fonction objective » la fonction qui quantifie la qualité de l'ajustement de sorte que plus le nombre renvoyé par cette fonction est petit, meilleur est l'ajustement. Cette fonction objective peut être définie librement, mais dans de nombreux cas, il s'agit du même critère que pour la régression linéaire par les moindres carrés, à savoir:

$$\sum_i (y_i - \hat{y}_i)^2$$

Donc, l'ajustement d'une courbe quelconque, notée y = f(x) peut se faire de manière itérative, en testant différentes valeurs des paramètres de la fonction, et en retenant au final la combinaison qui minimise le mieux la fonction objective. Par exemple, si notre courbe à ajuster est: $y = ax^b$, nous devrons estimer conjointement la valeur des deux paramètres (a et b) de la courbe. Bien qu'il s'agisse effectivement de la technique employée pour ajuster une courbe dans un nuage de point, il faut considérer divers problèmes potentiels.

- Le premier de ces problèmes est l'optimisation de la recherche des valeurs idéales des paramètres. Une recherche en aveugle est très peu efficace, évidemment. Il existe donc différentes approches. Leur description et développements mathématiques sort du cadre de ce cours. Je renvoie le lecteur intéressé à l'annexe C de Sen & Srivastava, (1990). Les algorithmes utilisés dans la fonction nls() de R, que nous utiliserons ici sont: **Gauss-Newton**, (par défaut), un algorithme utilisant la différentiation de la courbe et une expansion en série de Taylor pour approximer cette courbe par une série de termes additifs dont la solution peut être trouvée par régression linéaire multiple. The **Golub-Pereyra Plinear** algorithm, bien que peu répandu, est également implémenté. Il est utile en ce sens qu'il sépare les paramètres en deux sous-groupes: ceux qui sont linéaires dans la fonction (coefficients multiplicateurs de termes additifs) et ceux qui ne le sont pas. La recherche itérative ne se fait que sur les seconds. Les premiers étant estimés par régression linéaire. Ainsi, lorsque la fonction ne comporte que peu de paramètres non linéaires, l'algorithme converge beaucoup plus rapidement. Enfin, un troisième algorithme, nommé **Port** est également implémenté dans nls(). Il a la particularité, contrairement aux deux précédents, de permettre de définir des limites supérieures et inférieures acceptables pour chaque paramètres. Cela limite la recherche dans un domaine de validité, lorsque celui-ci peut être défini.

- La recherche de la solution optimale nécessite de partir de valeurs de départ plausibles pour les paramètres du modèle (c'est-à-dire, un ensemble de valeurs pour les paramètres telle que la courbe initiale est probablement proche de la solution recherchée). Définir les valeurs initiales n'est pas toujours chose aisée, et de plus, le résultat final peut dépendre fortement de ces valeurs initiales, à savoir que, si l'on choisi des valeurs initiales trop éloignées de la solution recherchée, on peut très bien se trouver coincé dans une fausse solution (un **minimum local de la fonction objective** qui est moins bas que le minimum absolu que l'on recherche).

- Le troisième problème est éventuellement lié à la complexité de la fonction non linéaire que l'on cherche à ajuster. Les propriétés mathématique de la courbe choisie peut très bien faire que cette courbe ne soit pas définie pour certaines valeurs des paramètres, ou qu'elle ait un comportement spécial dans un certain domaine (par exemple, courbe tendant vers l'infini). Ces cas sont difficilement traités par la plupart des algorithmes de minimisation de la fonction objective, et des erreurs de calcul de type « division par zéro », ou « résultat non défini » peuvent apparaître. Dans les meilleures implémentations (nls() en est une), des gardes-fous ont été ajoutés dans le code de la fonction pour éviter les erreurs dans pareils cas. Toutefois, il n'est pas possible de traiter

tous les cas d'erreur passibles. ainsi, il arrive parfois que le modèle choisi ne soit pas ajustable sur les données.

- Enfin, le dernier problème que l'on peut rencontrer, c'est une convergence trop lente, ou pas de convergence du tout vers la solution qui minimise la fonction objective. Ceci est fortement dépendant des 3 autres points précédents: choix de l'algorithme, choix des valeurs de départ pour les paramètres, et choix de la courbe à ajuster. En cas de non convergence, il faut essayer d'autres combinaisons avant de conclure que la courbe ne peut pas être ajustée sur les données.

Au final, l'ajustement d'un modèle non linéaire par les moindres carrés est une opération beaucoup plus délicate que l'ajustement par les moindres carrés d'un modèle linéaire sur les mêmes données. Dans le cas linéaire, la solution est trouvée de manière immédiate grâce à un petit calcul matriciel simple. Dans le cas non linéaire, il n'existe souvent pas de solution miracle, et le meilleur ajustement doit être recherché de manière itérative, et souvent à l'aveuglette. Nous verrons ci-dessous que R propose, pour certains modèles appelés 'SelfStart' une solution élégante pour calculer automatiquement les valeurs initiales et pour converger très rapidement vers la solution recherchée, mais la plupart du temps, il faut souvent tâtonner pour ajuster sa courbe.

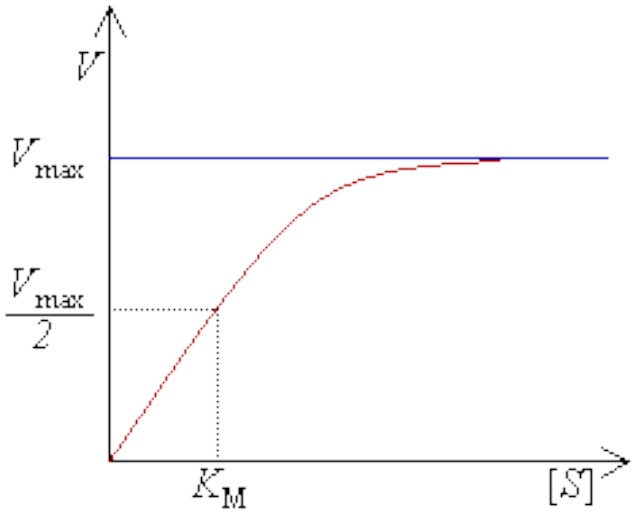
1.2. Les modèles 'selfStart' dans R

Dans certains cas, il existe des petites astuces de calcul pour converger directement vers la solution, ou du moins, pour calculer des valeurs de départ très proches de la solution recherchée, de sorte que l'algorithme de recherche pourra converger très rapidement. Par exemple, lorsqu'il est possible de linéariser le modèle en transformant les données. Dans ce cas, une bonne estimation des valeurs de départ des paramètres peut être obtenue en linéarisant la relation et en calculant les paramètres par régression linéaire. Ensuite, les paramètres obtenus sont retransformés dans leur forme initiale, et ils sont utilisés comme valeurs de départ.

Par exemple, si nous décidons d'ajuster un modèle allométrique de Huxley, de type $y = ax^b$, nous pouvons linéariser la relation en transformant les deux variables en log. En effet: $\log y = \log(ax^b)$, ce qui est équivalent à une droite en développant: $\log y = b \log x + \log a$. Le paramètre b devient donc la pente de la droite, et le log de a devient l'ordonnée à l'origine dans le modèle linéaire transformé. Une fois la droite ajustée, on a directement la valeur de b, et il suffit de calculer l'exponentielle de l'ordonnée à l'origine pour obtenir a. Toutefois, comme les résidus sont différents dans la relation non transformée initiale, il ne s'agit pas de la solution recherchée, mais d'une bon point de départ très proche de cette solution. Ainsi, on prendra les valeurs de a et de b, ainsi calculées par la droite en double log comme point de départ, et on laissera l'algorithme de recherche du minimum terminer le travail. En fait, c'est exactement de cette façon que les modèles dits 'selfStart' fonctionnent dans R.

1.3. Modèles non linéaires courants en biologie

Les domaines les plus courants où des modèles non linéaires sont utilisés en biologie concernent les cinétiques de réactions (chimiques, biochimiques), les courbes de type dose-réponse, les courbes de survie et les modèles de croissance. Nous verrons principalement divers modèles de croissance dans la section suivante. Certains de ces modèles, comme le modèle exponentiel, celui de Gompertz ou de Weibull sont aussi utilisés comme courbes de survie. De plus, la courbe logistique est un modèle classique de type dose-réponse. Ainsi, les différentes courbes de croissances recouvrent également une majorité des modèles utilisés dans d'autres domaines des biostatistiques. Nous mentionnerons ici un autre modèle courant qui n'est pourtant pas utilisé comme modèle de croissance: le modèle de Michaelis-Menten, fréquemment utilisé pour modéliser la cinétique de réactions chimiques, enzymatiques, en particulier.



Exemple de courbe cinétique de type Michaelis-Menten. La vitesse de la réaction augmente à un rythme donné par K_M *pour atteindre une vitesse maximale asymptotique* V_{max}.

Dans R, il existe un modèle 'selfStart' facile à utiliser pour ajuster une courbe de type Michaelis-Menten. Il s'agit de la fonction **SSmicmen()**.

1.4.Modèles de croissance

Parmi les phénomènes biologiques courants qui sont essentiellement non linéaires, les modèles de croissance occupent une part importante. Il s'agit de phénomènes complexes, résultat d'un ensemble de processus, eux-même très complexes: l' anabolisme, ou élaboration de matière organique et le catabolisme qui la détruit pour la transformer en énergie, en ce qui concerne la croissance somatique individuelle. Un modèle typique de croissance individuelle est le modèle de von Bertalanffy (von Bertalanffy, 1938, 1957).

Il existe un autre type de modèle de croissance: la croissance des populations. Ces modèles décrivent le nombre d'individus dans une

population au cours du temps. Dans ce cas particulier, il s'agit également du résultat de plusieurs processus: la natalité, la mortalité et, éventuellement, les migrations. Une courbe type de modèle de croissance de population est la courbe logistique. Un autre modèle couramment utilisé est celui de Weibull pour décrire un nombre décroissant d'individus suite à une mortalité prédominante dans celle-ci.

Il existe de nombreux modèles de croissance différents. Certains sont exclusivement des modèles de croissance individuelle, d'autres sont exclusivement des modèles de croissance de populations, mais beaucoup peuvent être utilisés indifféremment dans les deux cas. Tous ont comme particularité d'être l'une ou l'autre forme de modèle exponentiel, exprimant ainsi le fait que la croissance est fondamentalement un processus exponentiel (comprenez que l'augmentation de poids ou du nombre d'individus est proportionnelle au poids ou au nombre préexistant à chaque incrément temporel).

Nous allons voir quelques un des modèles de croissance principaux dans la suite de ce cours, et ensuite, nous les utiliserons pour ajuster une courbe de croissance dans un jeu de donnée réel.

1.4.1. La courbe exponentielle, un modèle de croissance Malthusien simple

En 1798, Thomas Malthus a décrit un modèle de croissance applicable pour décrire la croissance de la population humaine. Cependant, d'après Murray (1993), ce modèle a été suggéré en premier lieu par Euler. Quoi qu'il en soit, ce modèle n'est plus guère utilisé actuellement, mais son importance historique ne doit pas être négligée. Il s'agit, en effet, de la première modélisation mathématique d'une des caractéristiques les plus fondamentales de la croissance: son caractère exponentiel (positive ou négative). Malthus a observé que la population des Etat-Unis double tous les 25 ans. Il suggère alors que les populations humaines augmentent d'une proportion fixe *r* sur un intervalle de temps donné, lorsqu'elles ne sont pas affectées de contraintes environnementales ou sociales. Cette proportion *r* est par ailleurs indépendante de la taille initiale de la population:

$$Y_{t+1} = (1+r) \cdot Y_y = k\, Y_t$$

Une forme continue du modèle précédent (intervalle de temps infinitésimal) donne une équation différentielle:

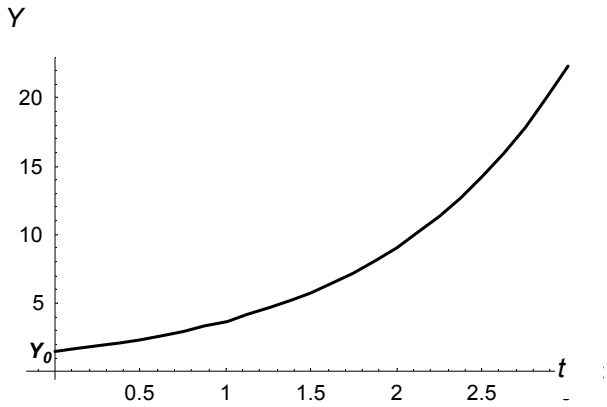
$$\frac{dY(t)}{dt} = Y'(t) = k \cdot Y(t)$$

Cette équation différentielle a la solution suivante:

$$Y(t) = Y_0 \cdot e^{k \cdot t}$$

avec Y_0, la taille initiale de la population au temps $t = 0$. Ce modèle à deux paramètres est également intéressant parce qu'il montre une bonne manière de construire un modèle de croissance: il suffit de décrire la croissance pour un accroissement infinitésimal de temps, par le biais d'une équation différentielle, et ensuite, de la résoudre (modélisation dynamique). Presque tous les modèles de croissance existants ont été élaborés de cette manière. Ainsi, la quasi-

totalité des modèles de croissance correspondent, en fait, à une équation différentielle relativement simple.



Exemple d'un modèle de croissance exponentiel avec $Y_0 = 1.5$ *et* $k = 0.9$.

1.4.2. La fonction logistique pour une croissance asymptotique

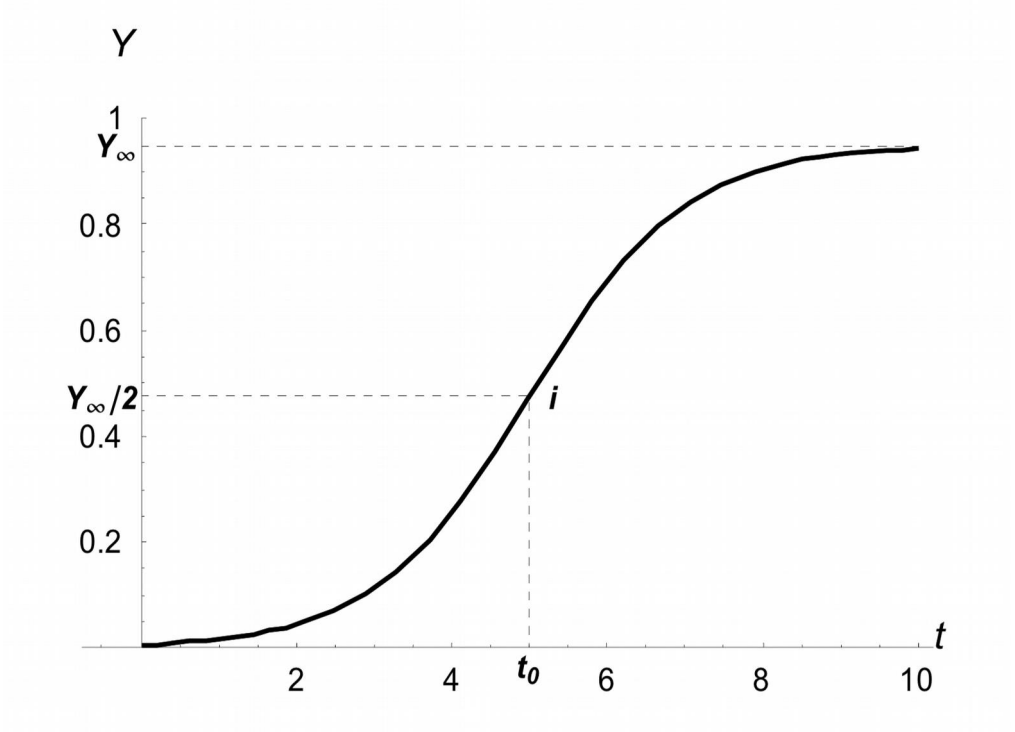
Le modèle exponentiel décrit une croissance infinie sans aucunes contraintes. Ce n'est pas une hypothèse réaliste. En pratique, la croissance est limitée par les ressources disponibles. Verhulst (1838), en travaillant aussi sur la croissance de populations, propose un modèle qui contient un terme d'auto-limitation $[Y_\infty - Y(t)]/Y_\infty$ qui représente une quelconque limite théorique des ressources disponibles:

$$\frac{dY(t)}{dt} = k \cdot \frac{Y_\infty - Y(t)}{Y_\infty} \cdot Y(t) = -\frac{k}{Y_\infty} Y(t)^2 + k \cdot Y(t)$$

Lorsque l'on résout et simplifie cette équation différentielle, on obtient:

$$Y(t) = \frac{Y_\infty}{1 + e^{-k \cdot (t - t_0)}}$$

Ceci est une des formes de la courbe logistique. Cette fonction a deux asymptotes horizontales en $Y(t) = 0$ and $Y(t) = Y_\infty$ (voir schéma ci-dessous) et c'est une sigmoïde symétrique autour du point d'inflexion (les deux courbes du S sont identiques). Le modèle 'selfStart' correspondant dans R s'appelle **SSlogis()**.



Exemple d'une courbe logistique avec k = *1,* Y_∞ = *0.95,* t_0 = *5. Cette courbe sigmoïdale est asymptotique en 0 et* Y_∞*, et elle est également symétrique autour de son point d'inflexion* i *situé à {*t_0*,* $Y_\infty/2$*}.*

Il est possible de généraliser ce modèle en définissant une courbe logistique dont l'asymptote basse peut se situer n'importe où ailleurs qu'en 0. Si cette asymptote se situe en Y_0, nous obtenons l'équation:

$$Y(t) = Y_0 + \frac{Y_\infty - Y_0}{1 + e^{-k\cdot(t-t_0)}}$$

Ceci est le modèle logistique généralisé à 4 paramètres (modèle 'selfSart' **SSfpl()** dans R).

1.4.3. Le modèle de Gompertz, une courbe sigmoïde asymétrique

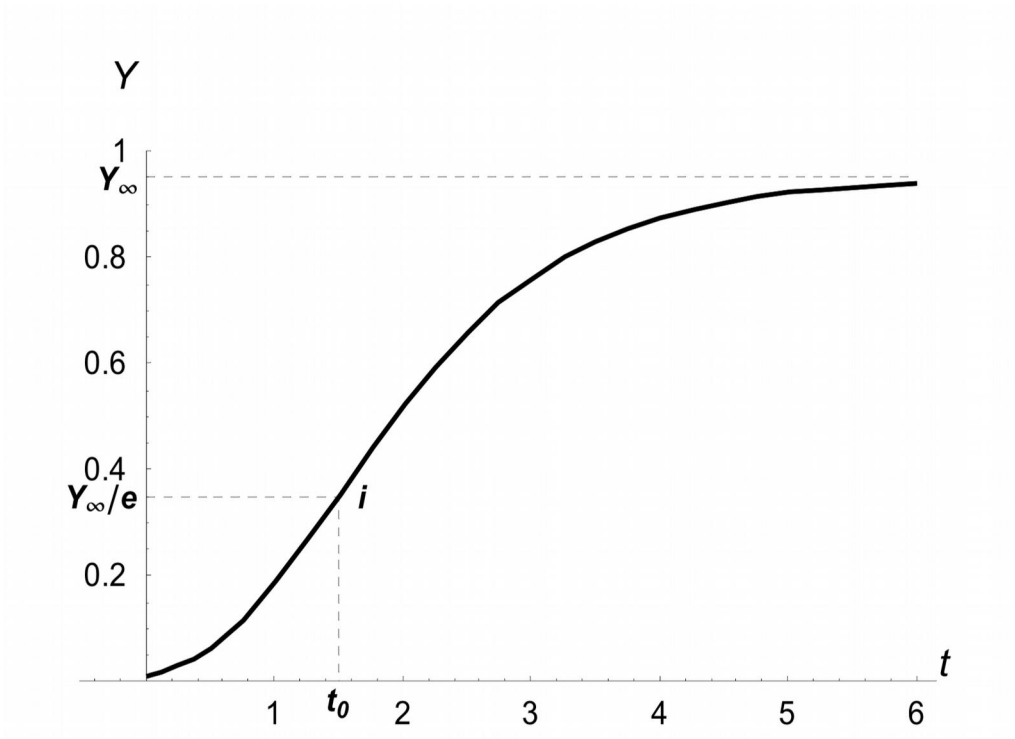
Gompertz (1825) a observé de manière empirique que le taux de survie décroît souvent de manière proportionnelle au logarithme du nombre d'animaux qui survivent survivent. Bien que ce modèle reste utilisé pour décrire des courbes de survie, elle trouve de nombreuses applications pour décrire également des données de croissance. L'équation différentielle du modèle de Gompertz est:

$$\frac{dY(t)}{dt} = k\cdot\left[\ln Y_\infty - \ln Y(t)\right]\cdot Y(t)$$

qui se résout et se simplifie en:

$$Y(t) = Y_\infty . e^{e^{-k.(t-t_0)}} = Y_\infty . a^{e^{-k.t}} = Y_\infty . a^{b^t}$$

La dernière forme apparaît plus simple et est le plus souvent utilisée. La première forme est directement dérivée de l'équation différentielle et donne une meilleure comparaison avec la courbe logistique, puisque *to* correspond aussi à l'abscisse du point d'inflexion, qui n'est plus en position symétrique ici (voir figure ci-dessous). Le modèle 'selfStart' correspondant dans R s'appelle **SSgompertz()**.



Exemple d'une courbe de Gompertz avec k = *1,* Y_∞ = *0.95,* t_0 = *1.5. Le point d'inflexion* i, *à* {t_0, Y_∞/e}, *est plus bas que pour la courbe logistique.*

1.4.4. Les modèles de von Bertalanffy

Le modèle de von Bertalanffy model, parfois appelé Brody-Bertalanffy (d'après les travaux de von Bertalanffy et Brody) ou modèle de Pütter (dans Ricker, 1979 notamment), est la première courbe de croissance qui a été élaborée spécifiquement pour décrire la croissance somatique individuelle. Il est basé sur une analyse bioénergétique simple. Un individu est vu comme un simple réacteur biochimique dynamique où les entrées (anabolisme) sont en compétition avec les sorties (catabolisme). Le résultat de ces deux flux étant la croissance. L'anabolisme est plus ou moins proportionnel à la respiration, et la respiration est proportionnelle à une surface pour beaucoup d'animaux: la surface développée des poumons ou des branchies. Le catabolisme est toujours proportionnel au volume ou au poids. Ces relations mécanistiques sont rassemblées dans l'équation différentielle suivante où *Y*(*t*) mesure l'évolution d'un volume ou d'un poids au cours du temps:

$$\frac{dY(t)}{dt} = a \cdot Y(t)^{2/3} - b \cdot Y(t) = 3k \cdot \left[Y_\infty{}^{1/3} \cdot Y(t)^{2/3} - Y(t) \right]$$

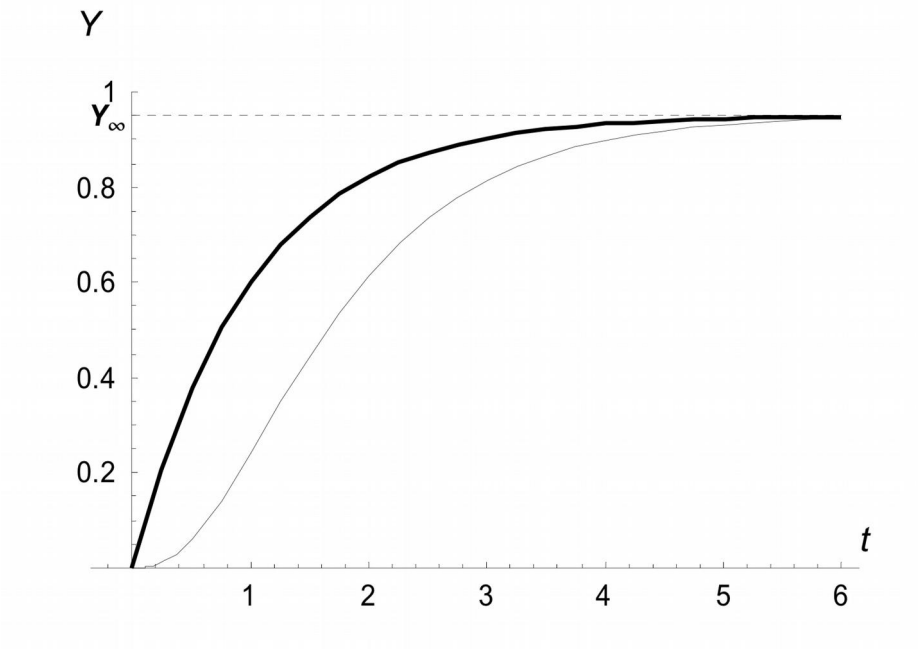
En résolvant cette équation, nous obtenons le modèle de croissance pondérale de von Bertalanffy:

$$Y(t) = Y_\infty \cdot \left(1 - e^{-k \cdot (t - t_0)} \right)^3$$

La forme la plus simple de ce modèle est obtenue lorsque nous mesurons des dimensions linéaires pour quantifier la taille de l'organisme, car une dimension linéaire est, en première approximation, la racine cubique d'un poids ou d'un volume (sans prendre en compte une possible allométrie). The modèle de von Bertalanffy pour des mesures linéaires est alors simplement:

$$Y(t) = Y_\infty \cdot \left(1 - e^{-k \cdot (t - t_0)} \right)$$

Un graphique des deux modèles est présenté ci-dessous. Le modèle de von Bertalanffy linéaire n'a pas de point d'inflexion. La croissance est la plus rapide à la naissance et ne fait que diminuer avec le temps pour finalement atteindre zéro lorsque la taille maximale asymptotique est atteinte. Avec ce modèle, la croissance est donc déterminée et elle ne peut dépasser cette asymptote horizontale située en $Y(t) = Y_\infty$. A cause de la puissance cubique de la forme pondérale du modèle von Bertalanffy, cette dernière est une sigmoïde asymétrique, comme l'est le modèle de Gompertz également. Trois modèles 'selfStart' existent dans R, selon que la courbe passe par o ou non: **SSasymp()**, **SSasympOff()** et **SSasympOrig()**.



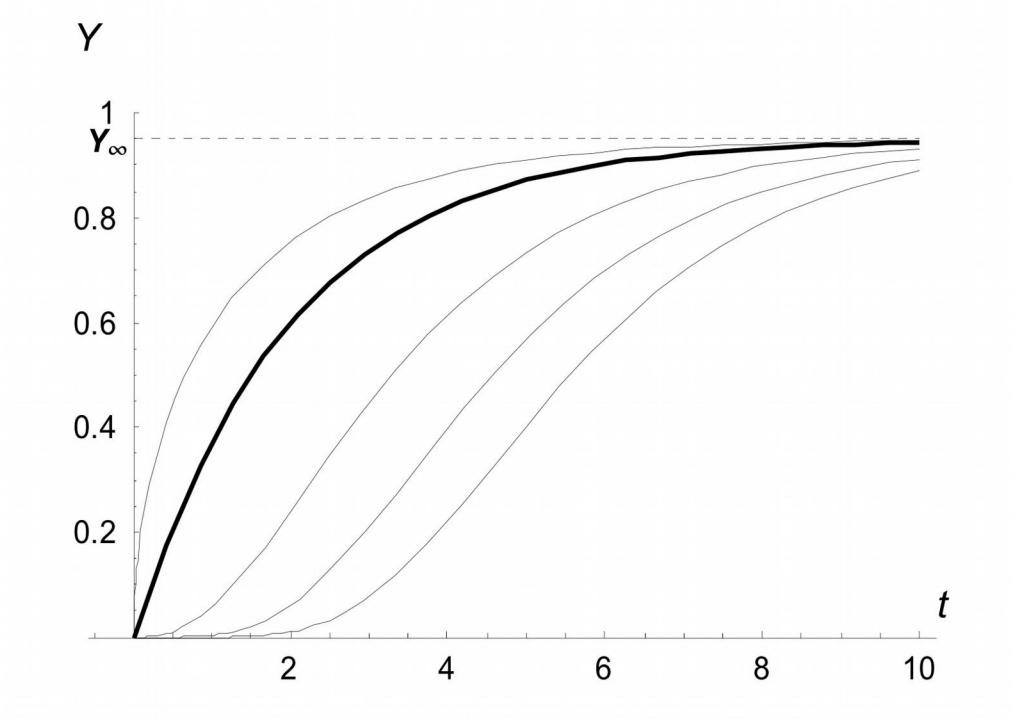
Exemples de modèles de croissance de von Bertalanffy linéaire (courbe en gras) et pondérale (courbe en trait fin) avec k = *1,* Y_∞ = *0.95 et* t_0 = *0. Les deux modèles décrivent une croissance asymptotique, mais la première courbe n'a pas de point d'inflexion alors que la seconde est sigmoïdale.*

1.4.5. Le modèle de Richards, une courbe flexible qui en contient d'autres

La forme généralisée du modèle de von Bertalanffy est:

$$Y(t) = Y_\infty \cdot \left(1 - e^{-k\cdot(t-t_0)}\right)^m$$

Von Bertalanffy (1938, 1957) a fixé m à 1 ou à 3. Richards (1959) permet à m de varier librement, et donc, son modèle a un paramètre de plus. Cette dernière courbe est très flexible (voir schéma ci-dessous) et il est possible de démontrer que plusieurs autres modèles de croissance ne sont que des cas particuliers de ce modèle généraliste avec différentes valeurs de m. Nous avons déjà observé que le modèle de Richards se réduit aux deux modèles de von Bertalanffy quand $m = 1$ et $m = 3$. Il se réduit aussi à une courbe logistique lorsque $m = -1$ et il est possible de montrer qu'il converge vers le modèle de Gompertz lorsque $|m| \to \infty$. Il n'existe aucun modèle 'selfStart' pour la courbe de Richards dans R. Par ailleurs, il s'agit d'un modèle particulièrement délicat à ajuster, comme nous le verrons dans un exemple concret plus loin.



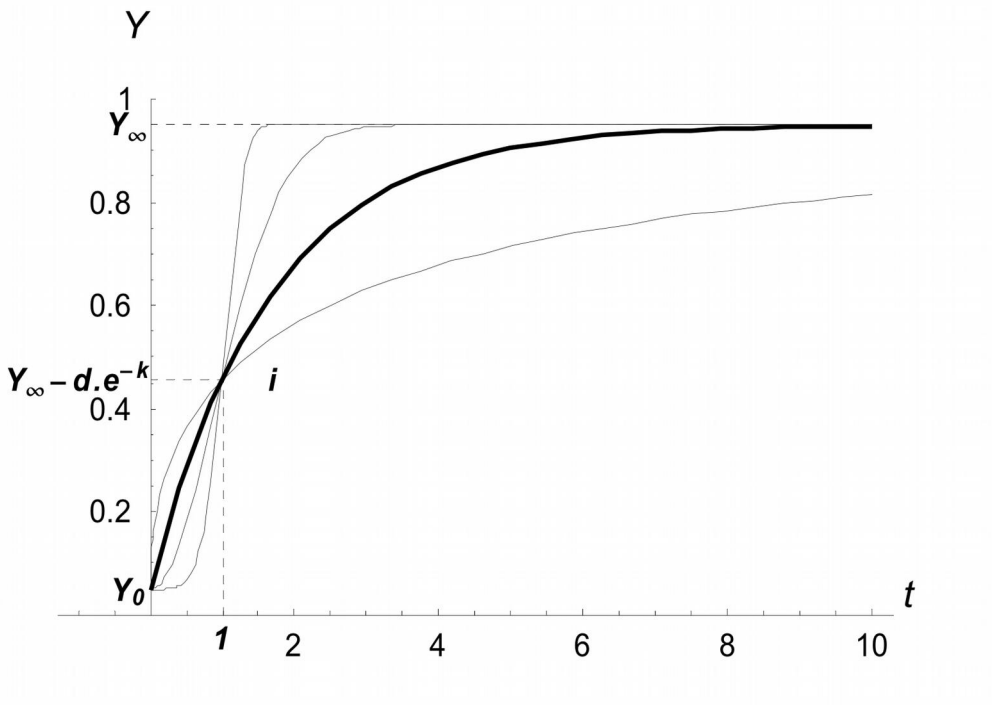
Allure de différentes courbes de Richards en fonction de la valeur de m. *De gauche à droite:* m = *0.5, 1, 3, 6 et 10; avec* k = *0.5,* Y_∞ = *0.95 et* t_0 = *0 pour toutes les courbes. La courbe en gras, avec* m = *1, est équivalente au modèle de von Bertalanffy linéaire.*

1.4.6. Le modèle de Weibull, une fonction polyvalente et flexible

Depuis son introduction en 1951 par Weibull, ce modèle est présenté comme polyvalent. Il a été décrit à l'origine comme une distribution statistique. Il trouve de nombreuses applications en croissance de population (éventuellement négative), et il est utilisé également pour décrire la courbe de survie en cas de maladies, ou dans des études de dynamique de populations. Il a parfois été utilisé comme un modèle de croissance. La forme la plus générale de ce modèle est:

$$Y(t) = Y_\infty - d \cdot e^{-k \cdot t^m} \quad \text{with} \quad d = Y_\infty - Y_0$$

Un modèle à 3 paramètres est également utilisé où $Y_0 = 0$. La fonction est sigmoïdale lorsque $m > 1$, sinon, elle ne possède pas de point d'inflexion (voir graphique ci-dessous). Dans R, le modèle 'selfStart' s'appelle **SSweibull()**.



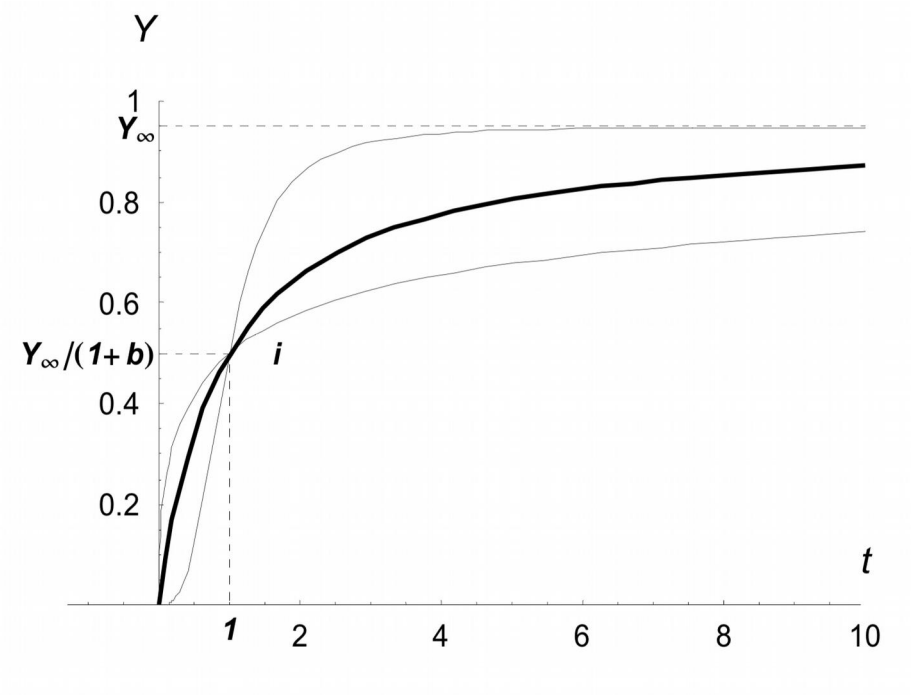
Exemples de courbes de Weibull pour respectivement m = *5, 2, 1 et 0.5, avec* k = *0.6,* Y_∞ = *0.95 et* Y_0 = *0.05. En gras, la courbe avec* m = *1, équivalente à un modèle de von Bertalanffy linéaire. Toutes les courbes démarrent en* Y_0 *et passent par* $Y_\infty - d \cdot e^{-k}$ *qui est également le point d'inflexion pour les sigmoïdes lorsque* m > *1.*

1.4.7. La courbe de Jolicoeur, un autre modèle flexible

Une autre courbe qui est tantôt sigmoïdale, tantôt non est le modèle de Jolicoeur (1985). Elle est dérivée d'une courbe logistique, mais en utilisant le logarithme du temps au lieu du temps lui-même:

$$Y(t) = \frac{Y_\infty}{1 + b \cdot t^{-m}}$$

Comme la fonction de Weibull, elle est sigmoïdale lorsque $m > 1$, mais avec l'asymétrie entre les deux courbures de part et d'autre du point d'inflexion qui peuvent varier indépendamment l'une de l'autre, en fonction de la valeur du paramètre *b*. Lorsque $m \leq 1$, il n'y a aucun point d'inflexion (voir schéma ci-dessous).

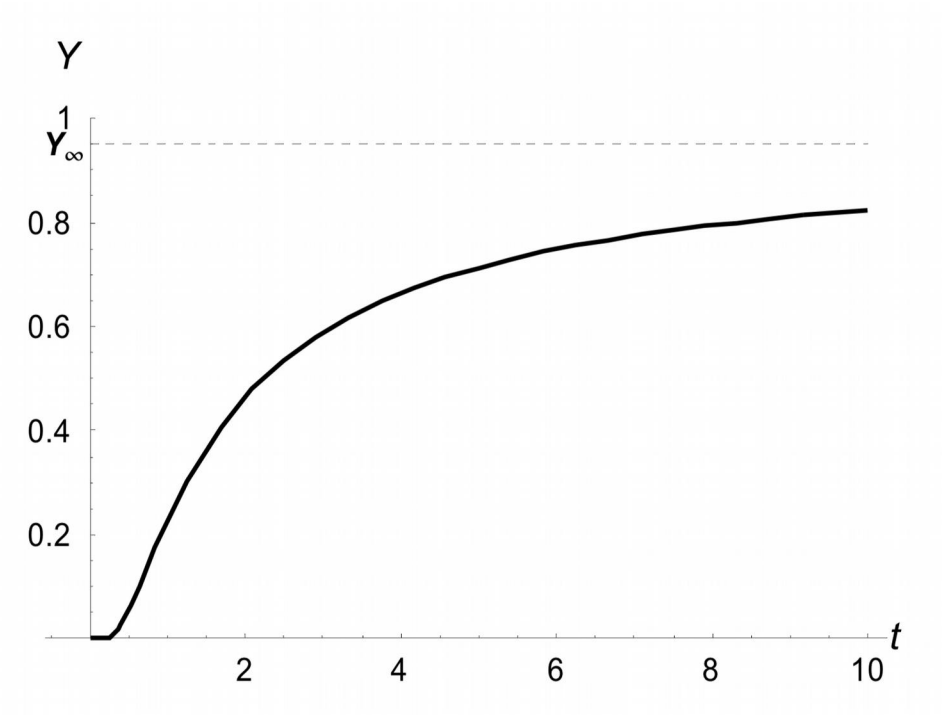


Exemples de courbes de Jolicoeur avec m = *3, 1 (trait gras) et 0.5 respectivement depuis la courbe supérieure vers la courbe inférieure;* Y_∞ = *0.95 et* b = *0.9. Lorsque* m > *1, la courbe est sigmoïdale.*

1.4.8. Le modèle de Johnson, une sigmoïde très asymétrique

La courbe de croissance de Johnson (voir Ricker, 1979) utilise $1/t$ à la place de t:

$$Y(t) = Y_\infty \cdot e^{1/k\cdot(t-t_0)}$$



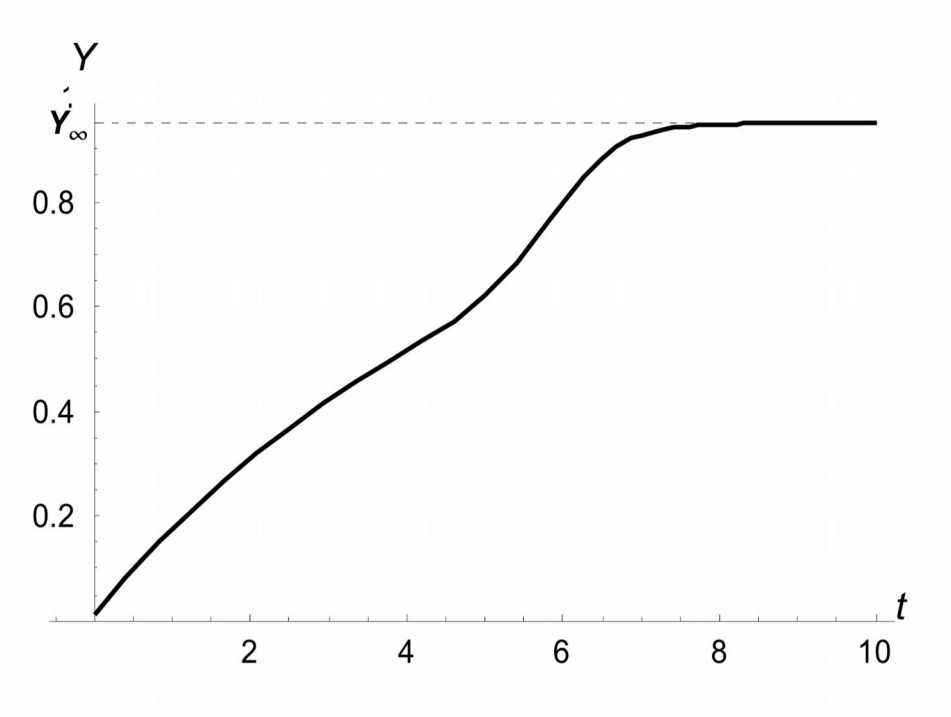
Exemple d'une courbe de Johnson, avec k = *0.7,* Y_∞ = *0.95 et* t_0 = *0.*

Ce modèle est sigmoïdal également mais une asymétrie tellement forte que le point d'inflexion est très bas, presque à 0 sur l'axe des ordonnées (il est donc difficilement visible sur le graphe). Cette courbe ressemble donc à une courbe de croissance non sigmoïdale, mais pour laquelle on observerait un temps de latence avant le début de la croissance sur l'axe temporel.

1.4.9. Le modèle Preece-Baines 1pour la croissance humaine

Preece et Baines (1978) ont décrit plusieurs modèles spécifiques à la croissance humaine. Ces modèles combinent deux phases de croissance exponentielle pour représenter la croissance graduelle d'enfants suivie par une courte phase de croissance accélérée à l'adolescence, mais qui atteint rapidement un plateau correspondant à la taille adulte définitive (voir graphique ci-dessous). Ce type de modèle est naturellement très utile pour tous les mammifères, mais certains, comme le modèle 1, ont aussi été utilisés dans d'autres circonstances, profitant de sa grande flexibilité. Son équation est:

$$Y(t) = Y_\infty - \frac{2 \cdot (Y_\infty - d)}{e^{k_1 \cdot (t - t_0)} + e^{k_2 \cdot (t - t_0)}}$$



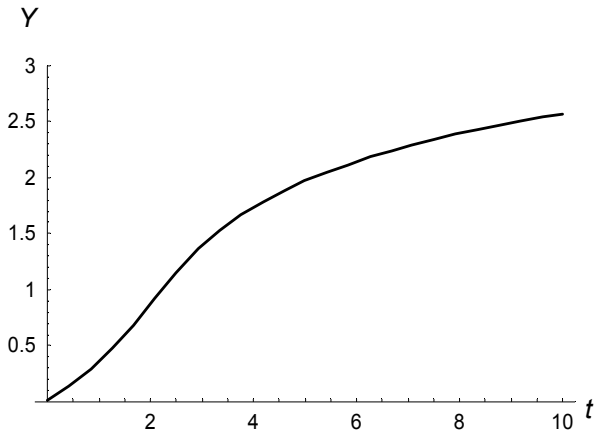
Exemple d'une courbe Preece-Baines 1 avec $k_1 = 0.19$, $k_2 = 2.5$, $Y_\infty = 0.95$, $d = 0.8$ *et* $t_0 = 6$.

1.4.10. Le modèle de Tanaka pour une croissance indéterminée

Tous les modèles précédents sont asymptotiques, à l'exception de la courbe exponentielle (mais cette dernière ne modélise valablement que la phase de croissance initiale). Tous ces modèles décrivent donc une croissance déterminée qui n'excédera jamais une taille maximale représentée par une asymptote horizontale en $Y(t) = Y_\infty$. Knight (1968) s'est demandé s'il s'agit d'une réalité biologique ou simplement d'un artefact mathématique. Dans le second cas, la croissance n'apparaîtrait déterminée que parce que les modèles choisis pour la représenter sont asymptotiques. Pour s'affranchir d'une telle contrainte, Tanaka (1982, 1988) a élaboré un nouveau modèle de croissance qui décrit une croissance non déterminée:

$$Y(t) = \left(\frac{1}{\sqrt{b}}\right) \cdot \ln\left|2b \cdot (t - t_0) + 2\sqrt{b^2 \cdot (t - t_0)^2 + a \cdot b}\right| + d$$

Ce modèle complexe à 4 paramètres a une période initiale de croissance lente, suivie d'une période de croissance exponentielle qui se poursuit par une croissance continue mais plus faible tout au long de la vie de l'animal (voir graphique ci-dessous).

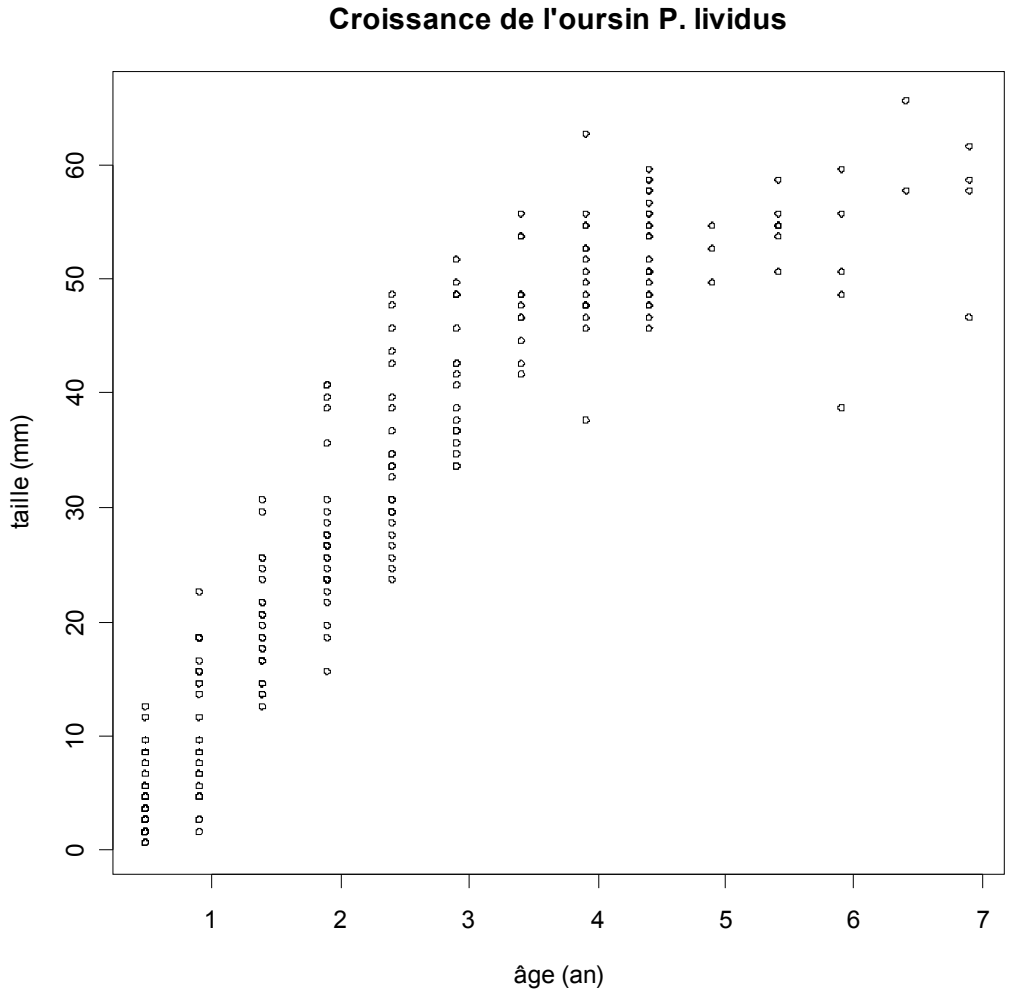


Exemple d'une courbe de Tanaka avec a = *3,* b = *2.5*, d = *-0.2 et* t_0 = *2*.

1.4.11. Exemple: ajustement d'une courbe de croissance à un jeu de données

Nous allons illustrer l'ajustement d'une courbe non linéaire par le choix et l'ajustement d'un modèle de croissance dans un jeu de données en modélisant la croissance somatique de l'oursin *Paracentrotus lividus* (Grosjean 2001). Ce jeu de données est disponible dans le package `UsingR`, sous le nom `urchin.growth`. (installez le package depuis CRAN s'il n'est pas déjà installé... Faites: **`Packages -> Installer le(s) package(s)...`**, sélectionnez un site près de chez vous, et sélectionnez ensuite la package `UsingR` dans la liste). Ensuite:

```
> library(UsingR)
> data(urchin.growth)
> plot(urchin.growth, xlab = "âge (an)", ylab = "taille (mm)")
> title("Croissance de l'oursin P. lividus")
```



Comme vous pouvez le voir, différents oursins ont été mesurés: diamètre à l'ambitus du test (zone la plus large) en mm à différents âges (en années). Les mesures ont été effectuées tous les 6 mois pendant 7 ans, ce qui donne un bon aperçu de la croissance de cet animal, y compris la taille maximale asymptotique qui est atteinte vers les 4 à 5 ans (pour ce genre de modèle, il est très important de continuer à mesurer les animaux afin de bien quantifier cette taille maximale asymptotique). Ainsi, l'examen du graphique nous permet d'emblée de choisir un modèle à croissance finie (pas le modèle de Tanaka, donc), et de forme sigmoïdale, tel que la courbe logistique, Weibull, Gompertz, par exemple. Nous pouvons à ce stade, essayer différents modèles et choisir celui qui nous semble le plus adapté.

Le choix du meilleur modèle se fait grâce à deux critères:

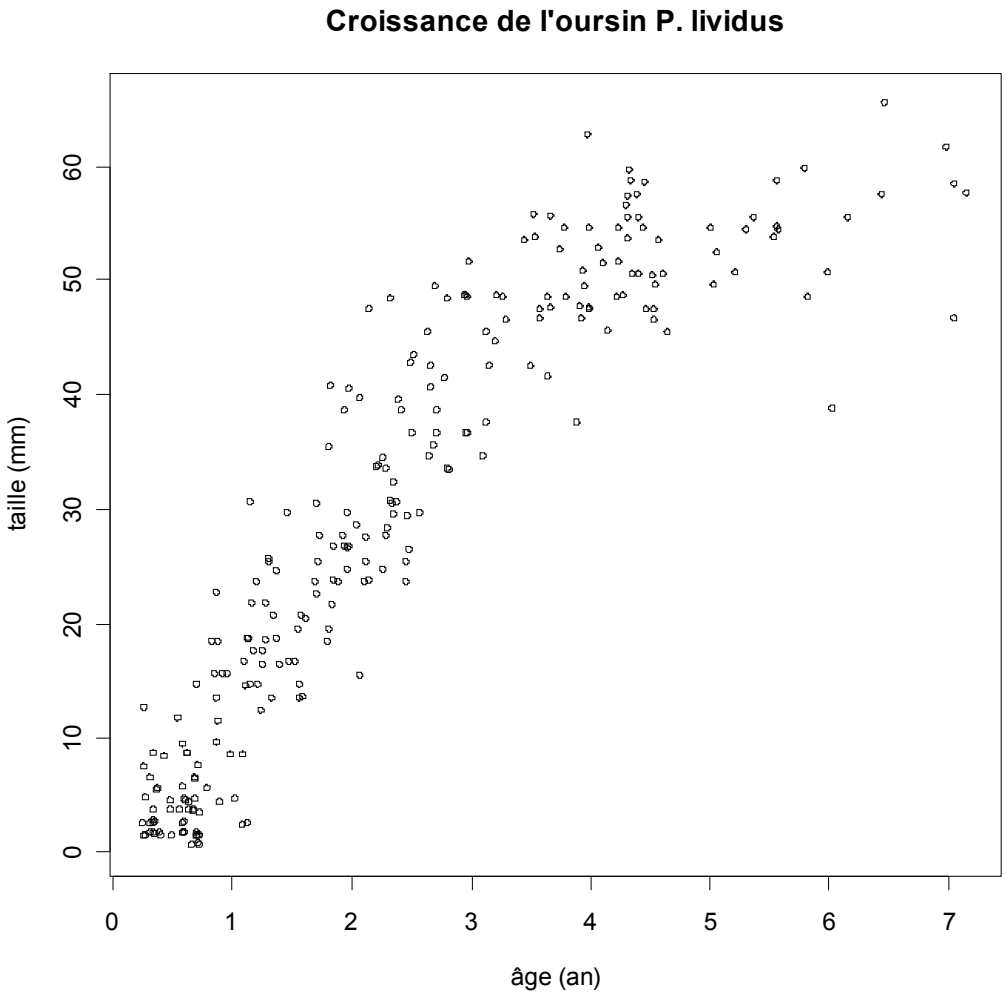
1. Les connaissances théoriques et *a priori* du modèle que l'on ajuste. En effet, il n'existe qu'un seul modèle linéaire, amis une infinité de modèles curvilinéaires qui peuvent s'ajuster dans les données. Le choix du meilleur modèle se fait en fonction de considérations sur le phénomène sous-jacent qui doivent se refléter dans les propriétés mathématiques de la courbe choisie. Par exemple, si on sait que la croissance est asymptotique vers une taille maximal, nous devrons choisir une courbe mathématique qui présente

une asymptote horizontale à son maximum pour représente au mieux le phénomène étudié.

2. Le coefficient de détermination R^2 n'est pas calculé par R pour une régression non linéaire, car sa validité est sujette à discussion entre les statisticiens (d'autres logiciels statistiques le calculent). Nous n'avons, donc, pas d'estimation de la qualité de l'ajustement par ce biais, comme dans le cas de la régression linéaire. Par contre, il est possible de calculer un autre critère, plus fiable, appelé **critère d'information d'Akaike** (fonction `AIC()` dans R). Ce critère tient compte à la fois de la qualité d'ajustement, et de la complexité du modèle, exprimée par le nombre de paramètres qu'il faut estimer. Plus le modèle est complexe, plus on peut s'attendre à ce qu'il s'ajuste bien aux données, var il est plus flexible. Cependant, en ce domaine, la complexité n'est pas forcément un gage de qualité. On recherche plutôt un compromis entre meilleur ajustement et simplicité. Le critère d'information d'Akaiké quantifie précisément ce compromis, c'est-à-dire que le modèle qui a un AIC le plus faible est considéré comme le meilleur. Appliquons donc ce concept pour sélectionner le meilleur modèle de croissance pour décrire la croissance somatique de nos oursins, après avoir sélectionné les candidtats les plus judicieux (point 1, modèle sigmoïdal avec asymptote horizontale au maximum).)

   > *Comme les animaux sont mesurés au mêmes âges, tous les 6 mois, certains points se superposent. Afin d'afficher tous les points, il est utile d'utiliser la fonction `jitter()` qui décale les points d'une valeur aléatoire pour éviter les superpositions. Voici ce que cela donne:*

```
> plot(jitter(size) ~ jitter(age, 3), data = urchin.growth,
+     xlab = "âge (an)", ylab = "taille (mm)")
> title("Croissance de l'oursin P. lividus")
```



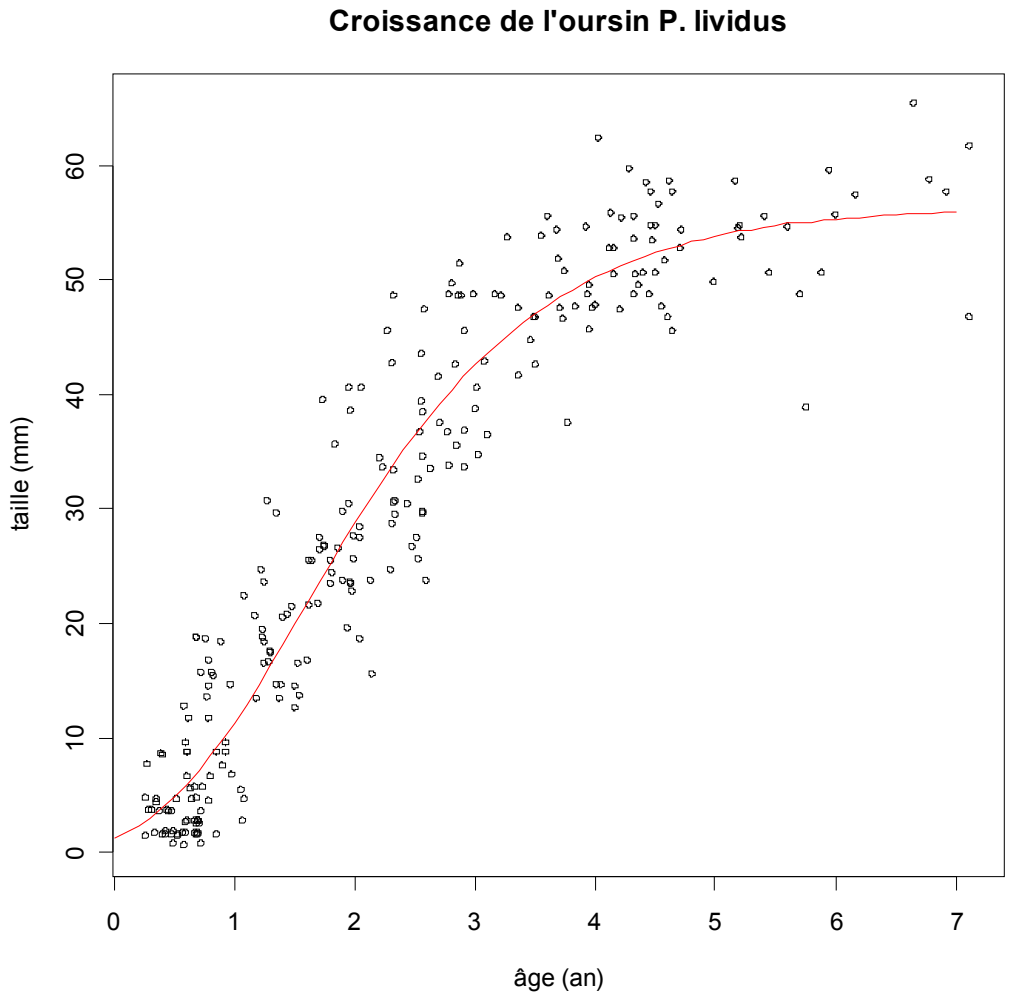
Ajustons un modèle de Gompertz (modèle 'SelfStart'):

```
> reg.gomp <- nls(size ~ SSgompertz(age, Asym, b2, b3),
+     data = urchin.growth)
> summary(reg.gomp)

Formula: size ~ SSgompertz(age, Asym, b2, b3)

Parameters:

     Estimate Std. Error t value Pr(>|t|)
Asym 56.50398    1.11393   50.73   <2e-16 ***
b2    3.82500    0.22976   16.65   <2e-16 ***
b3    0.41971    0.01888   22.23   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.298 on 247 degrees of freedom

> AIC(reg.gomp)
[1] 1548.146
```

Pour ajouter cette courbe sur le graphique, il faut créer un vecteur de valeurs x (Age) espacé régulièrement et prédire les valeurs de size obtenues avec ce modèle:

```
> Pred <- list(age = seq(from = 0, to = 7, by = 0.1))
```

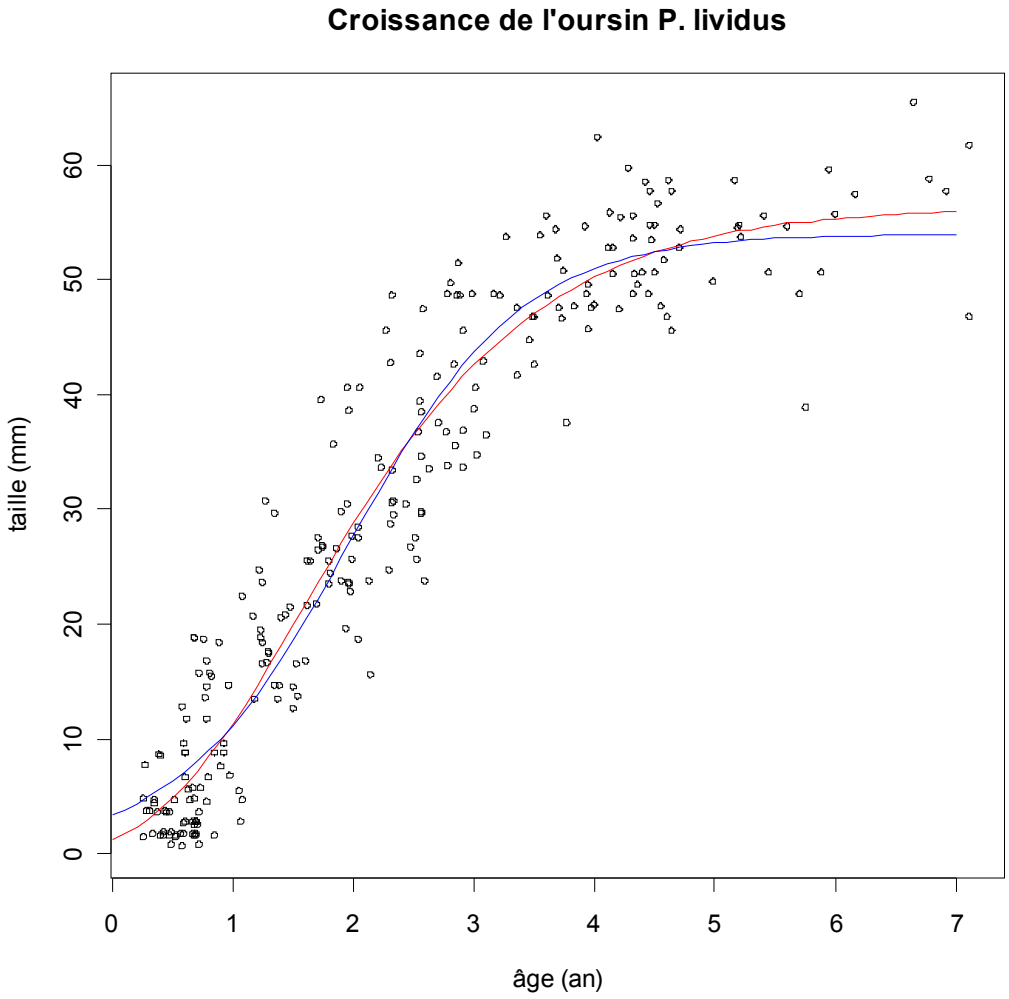
```
> Pred
$age
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4
[16] 1.5 1.6 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9
[31] 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1 4.2 4.3 4.4
[46] 4.5 4.6 4.7 4.8 4.9 5.0 5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9
[61] 6.0 6.1 6.2 6.3 6.4 6.5 6.6 6.7 6.8 6.9 7.0

> lines(Pred$age, predict(reg.gomp, newdata = Pred), col = "red")
```



L'ajustement de cette fonction semble très bon, à l'oeil. Voyons ce qu'il en est d'autres modèles:

```
> reg.logis <- nls(size ~ SSlogis(age, Asym, xmid, scal),
+ data = urchin.growth)
> summary(reg.logis)

Formula: size ~ SSlogis(age, Asym, xmid, scal)

Parameters:
     Estimate Std. Error t value Pr(>|t|)
Asym 53.90343    0.86507   62.31   <2e-16 ***
xmid  1.95792    0.04387   44.63   <2e-16 ***
scal  0.71801    0.03484   20.61   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 5.436 on 247 degrees of freedom
```

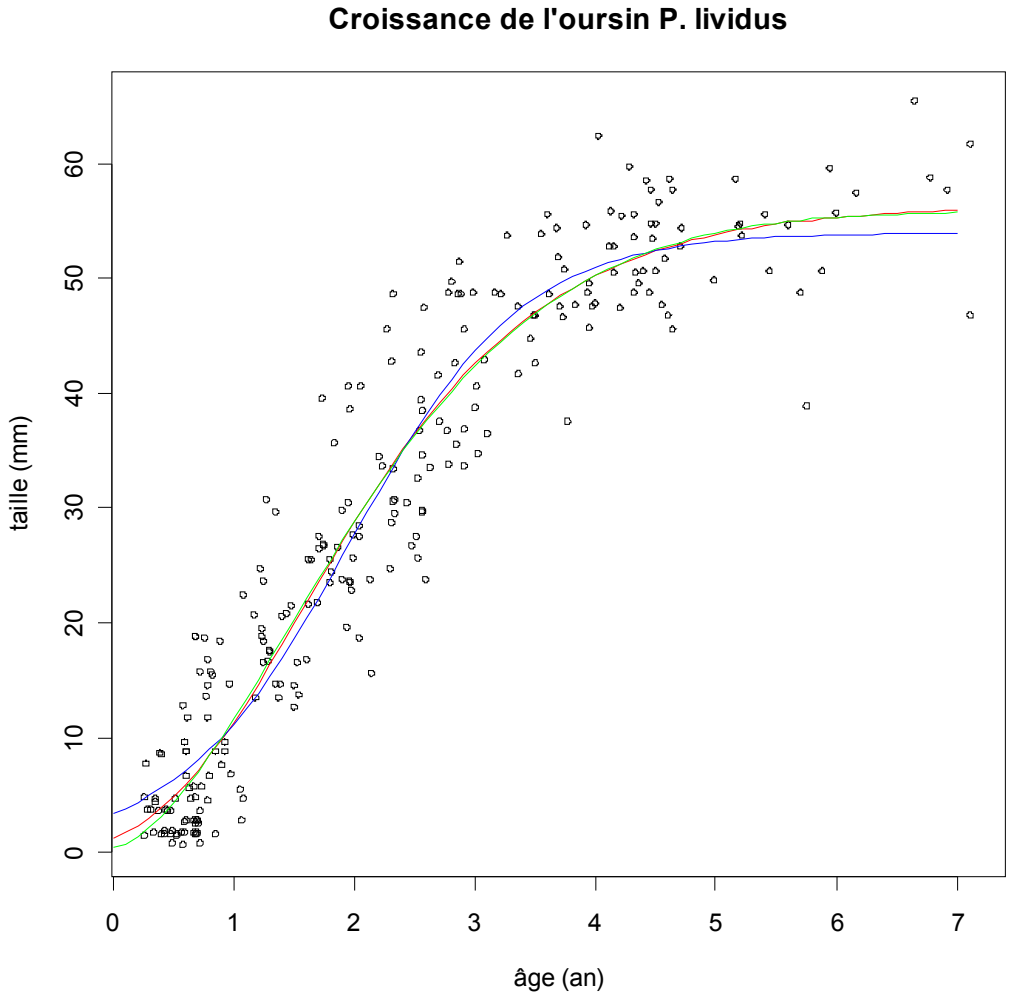
```
> AIC(reg.logis)
[1] 1560.963

> lines(Pred, predict(reg.logis, newdata = Pred), col = "blue")
```



Comme on peut le voir clairement sur le graphe, la courbe logistique donne une autre solution, cette dernière est à peine moins bonne que le modèle de Gompertz, selon le critère d'Akaiké. Pourtant, la courbe est assez bien démarquée de celle de Gompertz. Essayons maintenant un modèle de Weibull. Ce modèle est plus complexe car il a 4 paramètres au lieu de 3 pour les deux modèles précédents:

```
> reg.weib <- nls(size ~ SSweibull(age, Asym, Drop, lrc, pwr),
+     data = urchin.growth)
> summary(reg.weib)

Formula: size ~ SSweibull(age, Asym, Drop, lrc, pwr)

Parameters:
     Estimate Std. Error t value Pr(>|t|)
Asym  55.9745     1.4680   38.13   <2e-16 ***
Drop  55.5705     2.5145   22.10   <2e-16 ***
lrc   -1.4840     0.1256  -11.81   <2e-16 ***
pwr    1.6617     0.1427   11.65   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 5.286 on 246 degrees of freedom
```

```
> AIC(reg.weib)
[1] 1547.956

> lines(Pred$age, predict(reg.weib, newdata = Pred), col = "green")
```



Ce modèle fait presque jeu égal avec le modèle de Gompertz en terme de critère d'Akaiké; juste un tout petit peu mieux. En fait, les deux courbes sont pratiquement superposées l'une à l'autre, mais le modèle de Weibull à un démarrage de croissance plus lent au début, ce qui se reflète dans les données. Par contre, il est pénalisé par le fait que c'est un modèle plus complexe qui possède un paramètre de plus. L'un dans l'autre, le critère d'information d'Akaiké considère donc les deux modèles sur pratiquement le même plan du point de vue de la qualité de leurs ajustements respectifs.

A ce stade, nous voudrions également essayer un autre modèle flexible à 4 paramètres: le modèle de Richards. Malheureusement, il n'existe pas de fonction 'SelfStart' dans R pour ce modèle. Nous sommes donc réduit à mettre les mains dans le cambouis, à définir la fonction nous même, à trouver de bonnes valeurs de départ, etc. Voici comment définir la fonction:

```
> Richards <- function(x, Asym, k, x0, m) Asym*(1-exp(-k*(x-x0)))^m
```

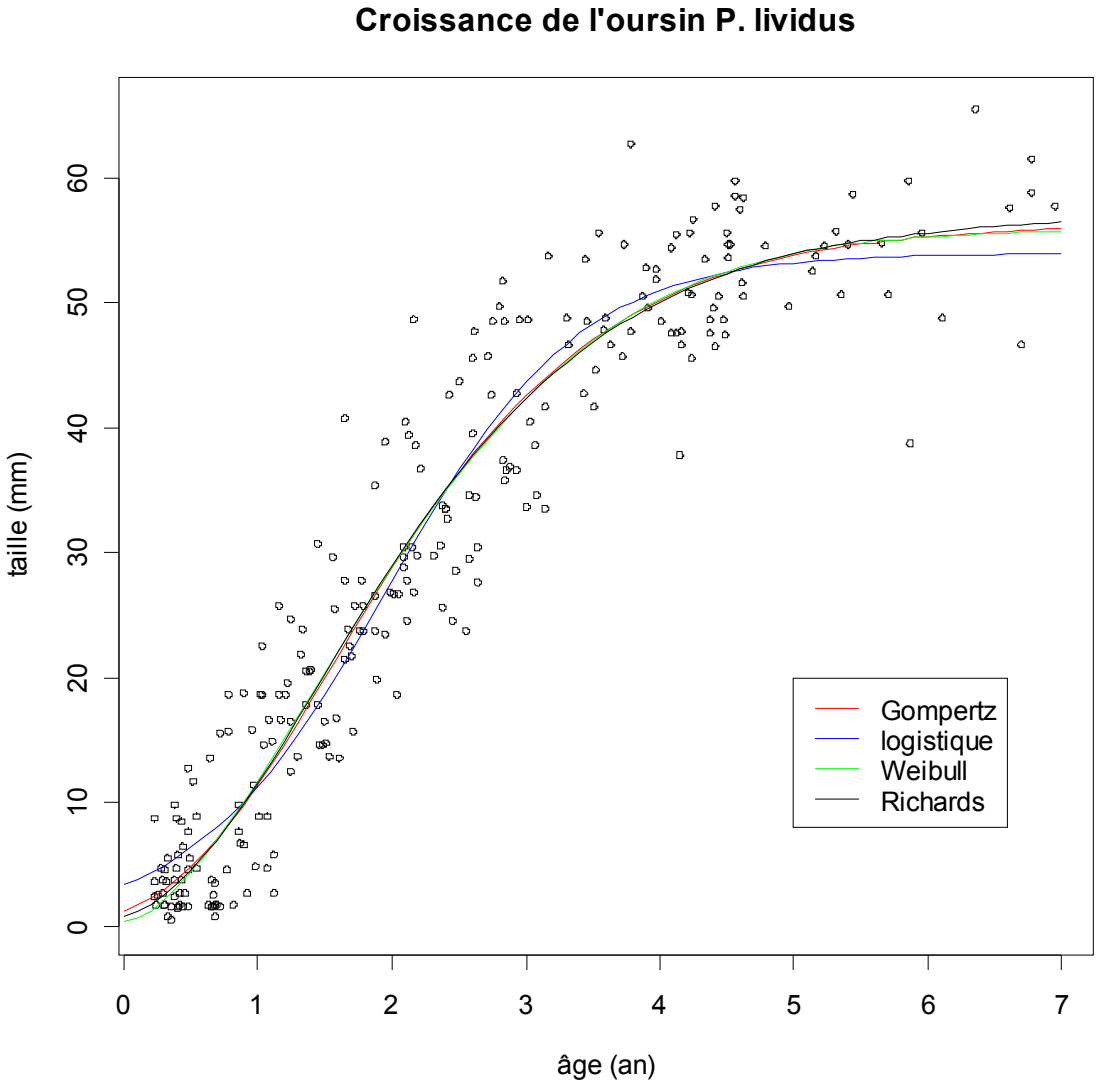
Pour les valeurs de départ, là c'est pas facile. Asym est l' asymptote horizontale à la taille maximum. On voit qu'elle se situe aux environ de 55 mm sur le graphique. Pour les autres paramètres, c'est plus difficile à évaluer. Prenons par exemple 1 comme valeur de départ pour ces 3 autres paramètres, ce qui donne:

```
> reg.rich <- nls(size ~ Richards(age, Asym, k, x0, m), data =
urchin.growth, start = c(Asym = 55, k = 1, x0 = 1, m = 1))
Erreur dans numericDeriv(form[[3]], names(ind), env) :
        Valeur manquante ou infinie obtenue au cours du calcul du modèle
```

... et voilà! Un excellent exemple de plantage de l'algorithme de minimisation de la fonction objective, suite à un comportement inadéquat de la fonction avec les valeurs testées (ici, la fonction renvoie l'infini, et l'algorithme ne peut donc effectuer la minimisation). La fonction de Richards est effectivement connue pour être difficile à ajuster pour cette raison.

Il nous faut donc, soit tester d'autres valeurs de départ, soit utiliser un autre algorithme de minimisation, soit les deux. Après différents essais, il apparaît que le changement des valeurs de départ suffit dans le cas présent:

```
> reg.rich <- nls(size ~ Richards(age, Asym, k, x0, m),
+     data = urchin.growth, start = c(Asym = 55, k = .5, x0 = 0, m = 1))
> summary(reg.rich)

Formula: size ~ Richards(age, Asym, k, x0, m)

Parameters:
     Estimate Std. Error t value Pr(>|t|)
Asym  57.2649     1.7167  33.358  < 2e-16 ***
k      0.7843     0.1236   6.343 1.07e-09 ***
x0    -0.8587     1.2377  -0.694    0.488
m      6.0636     8.2101   0.739    0.461
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 5.304 on 246 degrees of freedom

> AIC(reg.rich)
[1] 1549.703

> lines(Pred$age, predict(reg.rich, newdata = Pred), col = "black")
> legend(5, 20, c("Gompertz", "logistique", "Weibull", "Richards"),
+     col =c("red", "blue", "green", "black"), lwd = 2)
```



La courbe est intermédiaire entre Gompertz et Weibull aux jeunes âges, mais l'asymptote maximale est légèrement plus haute que pour les deux autres modèles (57 au lieu de 56). Les trois courbes sont très, très proches l'une de l'autre. Le critère d'information d'Akaiké est un peu moins bon pour le modèle de Richards. En outre les écarts types pour deux de ses paramètres sont très élevés (x0 et m), ce qui démontre une certaine instabilité de la fonction par rapport à ces paramètres, et par conséquent, une grande incertitude dans leur évaluation). Pour toutes ces raisons, le modèle de Richards sera écarté dans notre cas, au benefice du modèle de Gompertz, ou de celui de Weibull.

Le choix final entre Gompertz ou Weibull dépend de l'usage que l'on veut faire du modèle. Si la simplicité du modèle est primordiale, nous garderons Gompertz. Si la croissance des petits oursins est un aspect important de l'analyse, nous garderons Weibull qui semble mieux s'ajuster aux données à ce niveau.

2. BIBLIOGRAPHIE

Brostaux, Y. Introduction à l'environnement de programmation statistique R. 22 pp. [en ligne : http://cran.r-project.org].

Chambers, J. M., 1998. Programming with data. A guide to the S language. Springer, New York. 469 pp.

Dalgaard, P., 2002. Introductory statistics with R. Springer, New York. 267 pp.

Fox, J., 2003. R Commander [en ligne : http://cran.r-project.org]

Grosjean, Ph., 2001. Growth model of the reared sea urchin *Paracentrotus lividus* (Lamarck, 1816). Thèse de doctorat en Sciences Agronomiques et Ingénierie Biologique, ULB, Bruxelles, 285pp (http://www.sciviews.org/_phgrosjean/These.pdf).

Ihaka R. & R. Gentleman, 1996. R : a language for data analysis and graphics. *J. Comput. Graphic. Stat.*, 5:299-314.

Legendre, L. & P. Legendre, 1984. Ecologie numérique. Tome 2: La structure des données écologiques. Masson, Paris. 335 pp.

Paradis, E. R pour débutants. [en ligne : http://cran.r-project.org]

Pinhero, J. C. & D. M. Bates, 2000. Mixed-effects models in S and S-PLUS. Srpinger, New York. 528 pp.

Sen, A. & M. Srivastava, 1990. Regression analysis. Theory, methods, and applications. Springer-Verlag, New York, 348pp.

Sokal, R.R. & F.J. Rohlf, 1981. Biometry. Freeman & Co, San Francisco. 860 pp.

Venables W.N. & B.D. Ripley, 2000. S programming. Springer, New York, 264 pp.

Venables W.N. & B.D. Ripley, 2002. Modern applied statistics with S-PLUS (4^{th} ed.). Springer, New York, 495 pp.

Verzani, J., 2005. Using R for introductory statistics. Chapman & Hall, London. 414 pp.

3. ANNEXE: SCRIPTS DES ANALYSES DANS R

```
# Régression non linéaire. Exemple.
library(MASS)
data(wtloss)
plot(wtloss)
wtlost <- function(t, b0, b1, th) b0 + b1 * 2^(-t/th)
wtlost.start <- c(b0 = 100, b1 = 85, th = 100)
wtloss.reg <- nls(Weight ~ wtlost(Days, b0, b1,th), data = wtloss,
  start = wtlost.start, trace = TRUE)
summary(wtloss.reg)
# Superposer la courbe sur le graphe
wtlost2 <- list(Days = seq(from = 0, to = 250, by = 1))
lines(wtlost2$Days, predict(wtloss.reg, newdata = wtlost2), col = "red")

# Utilisation de modèles 'SelfStart'
# Courbe logistique: SSlogis(input, Asym, xmid, scal)
#Y = Asym/(1+exp((xmid-X)/scal))
data(ChickWeight)
coplot(weight ~ Time | Chick, data = ChickWeight,
         type = "b", show = FALSE)
Chick.1 <- ChickWeight[ChickWeight$Chick == 1, c("Time", "weight")]
plot(Chick.1)
Chick.1.reg <- nls(weight ~ SSlogis(Time, Asym, xmid, scal), data =
Chick.1, trace = TRUE)
summary(Chick.1.reg)
Chick.2 <- list(Time = seq(from = 0, to = 21, by = 0.1))
lines(Chick.2$Time, predict(Chick.1.reg, newdata = Chick.2), col = "red")

# Modèles 'SelfStart'?
apropos("SS")

# Modèles de croissance
# Croissance de l'oursin Paracentrotus lividus
library(UsingR)
data(urchin.growth)
plot(urchin.growth, xlab = "âge (an)", ylab = "taille (mm)")
title("Croissance de l'oursin P. lividus")
# Separation des points sur le graphe
plot(jitter(size) ~ jitter(age, 3), data = urchin.growth,
  xlab = "âge (an)", ylab = "taille (mm)")
title("Croissance de l'oursin P. lividus")

# Gompertz
reg.gomp <- nls(size ~ SSgompertz(age, Asym, b2, b3),
  data = urchin.growth)
summary(reg.gomp)
AIC(reg.gomp)
Pred <- list(age = seq(from = 0, to = 7, by = 0.1))
lines(Pred$age, predict(reg.gomp, newdata = Pred), col = "red")

# Courbe logistique
reg.logis <- nls(size ~ SSlogis(age, Asym, xmid, scal),
  data = urchin.growth)
summary(reg.logis)
AIC(reg.logis)
lines(Pred$age, predict(reg.logis, newdata = Pred), col = "blue")

# Weibull
reg.weib <- nls(size ~ SSweibull(age, Asym, Drop, lrc, pwr),
  data = urchin.growth)
summary(reg.weib)
AIC(reg.weib)
lines(Pred$age, predict(reg.weib, newdata = Pred), col = "green")

# Richards (pas de 'SelfStart'!)
Richards <- function(x, Asym, k, x0, m) Asym*(1-exp(-k*(x-x0)))^m
reg.rich <- nls(size ~ Richards(age, Asym, k, x0, m),
  data = urchin.growth, start = c(Asym = 55, k = 1, x0 = 1, m = 1))
#Autres valeurs de départ
reg.rich <- nls(size ~ Richards(age, Asym, k, x0, m),
  data = urchin.growth, start = c(Asym = 55, k = .5, x0 = 0, m = 1))
summary(reg.rich)
AIC(reg.rich)
```

```
lines(Pred$age, predict(reg.rich, newdata = Pred), col = "black")


# Légende
legend(5, 20, c("Gompertz", "logistique", "Weibull", "Richards"),
  col =c("red", "blue", "green", "black"), lwd = 2)
```