Complément formateur :
Du modèle linéaire à GLM au travers de calculs de DL50

Ce document, basé sur un exemple très simple traité sous Splus, est destiné au formateur. Après la présentation des données, nous comparerons la pratique du modèle linéaire généralisé avec la régression linéaire simple, la transformation arcsin(racine()), la régression linéaire pondérée sur variable transformée, la régression non linéaire par moindres carrés pondérés ainsi que la régression non linéaire par le maximum de vraisemblance, équivalent direct de la procédure glm.

Il ne s'agit pas de remettre à l'honneur des procédures plus anciennes voire de mauvaises recettes de cuisine, mais de mieux faire comprendre le principe de fonctionnement de glm et de l'algorithme utilisé en montrant à la fois sa proximité et sa distance par rapport à ces dernières. Il s'agit aussi de donner les commandes S+ permettant de refaire "à la main" les calculs de vraisemblances et de déviances ainsi qu'une procédure de calcul d'intervalles de confiance.

On suppose que l'utilisateur connaisse les grandes lignes de la théorie du glm, et possède quelques rudiments sur Splus, mais nous expliciterons les commandes au maximum, et pour commencer :
- Lancer S+ : Splus On obtient le ">", qui est le symbole d'attente de commande
- Quitter S+ : > q()
- Aide en ligne : > help.start(gui="motif")

1- Les données.

Nous reprenons l'exemple traité par Collett (1991, p. 75) et repris par Venables et Ripley (1994, p.189). Il s'agit d'exposer pendant trois jours des lots de 20 mites du tabac *Heliothis virescens* à 6 doses croissantes de pesticide pour évaluer leur résistance (1 dose par lot). On cherche en particulier à définir la dose qui provoque 50 % de mortalité (DL50) avec son intervalle de confiance. Nous reprenons ces données, mais en ne retenant que celles qui concernent les mâles, et en modifiant la dernière valeur : au lieu de 20 morts et 0 vivant, nous prendrons 19 morts et 1 vivant pour la dose maximale de pesticide, car ceci nous empêcherait de tester une régression sur données transformées par la fonction logit.

Créer les données

Création du vecteur du nombre d'individus morts (commande "c"= column), et affectation du résultat par le symbole "_" dans l'objet "mort" (6 doses) :
```
> mort_c(1,4,9,13,18,19)
```

Vecteur du nombre d'individus vivants et affectation dans l'objet "viv" :
```
> viv_c(19,16,11,7,2,1)
```

Liaison des deux vecteurs en une matrice (commande "cbind"=lien de colonnes) notée MV
```
> MV_cbind(mort,viv)
```

Si l'on tape le nom de l'objet tout seul, il s'affiche, ce qui permet de vérifier que tout va bien :
```
> MV
     mort viv
[1,]    1  19
[2,]    4  16
[3,]    9  11
[4,]   13   7
[5,]   18   2
[6,]   19   1
```

Vecteur des 6 doses, noté dose :
```
> dose_c(1,2,4,8,16,32)
```

Affichage d'un dessin

Ouverture d'une fenêtre de dessin :
> motif()

Dessin des proportions observées (morts / (morts+vivants)) :
> plot(dose,mort/(mort+viv))

Les données brutes montrent une relation non linéaire. On constate cependant qu'il faut doubler la dose à chaque fois pour obtenir une progression sensible de la mortalité des mites. Une transformation logique consiste à passer au logarithme des doses (quelle que soit la base, la base 2 est ici la plus lisible, les concentrations ayant été choisies comme des multiples de 2).

Création du vecteur log2 des doses et affectation dans l'objet "ldose" :
> ldose_log(dose,2)
> ldose
[1] 0 1 2 3 4 5

La relation apparaît plus linéaire, quoique encore légèrement sigmoïdale, et surtout plus symétrique.

2- Le modèle linéaire généralisé

Commençons par traiter ce cas par le modèle linéaire généralisé. Sous l'hypothèse d'indépendance entre les mites vis-à-vis de la mort, et si l'on considère que le nombre de décès dans un lot donné est la somme de n=20 processus binaires de probabilité p, alors cette somme suit une loi binomiale de variance np(1-p). Les proportions observées (nombre de décès / n) suivent donc une loi binomiale de variance p(1-p)/n. Nous avons peu de points d'observation (6) pour discuter de la forme générale de la réponse en fonction de la dose. Elle semble cependant symétrique et le lien canonique logit apparaît adapté. Avec Splus, utilisation de la fonction glm : on cherche à prédire la proportion de morts (donnée par la matrice MV) par le vecteur du log2 des doses, en appliquant le modèle standard de la famille binomiale (lien logit). Le résultat est stocké dans un objet que nous appellerons res.glm. La fonction "summary" sert à en extraire les principaux résultats.

> res.glm_glm(MV~ldose,family=binomial)
> summary(res.glm)

```
Call: glm(formula = MV ~ ldose, family = binomial)
Deviance Residuals:
          1         2         3          4         5          6
 -0.3257214 0.1625217 0.2868404 -0.4202333 0.3196241 -0.1614282

Coefficients:
               Value Std. Error   t value
(Intercept) -2.626062   0.516758 -5.081802
     ldose  1.148072   0.194472  5.903532

(Dispersion Parameter for Binomial family taken to be 1 )

    Null Deviance: 63.49754 on 5 degrees of freedom

Residual Deviance: 0.5195998 on 4 degrees of freedom
```

Notons simplement qu'avec 6 observations de 20 réalisations binaires, le nombre de réalisations est suffisant pour considérer que la déviance résiduelle suit approximativement un χ^2 à 6-2 = 4 degrés de liberté. Une valeur résiduelle de 0.52 est à peine supérieure au quantile à 5 % d'un test bilatéral. Si le modèle de variance binomial est valide et le lien logit adéquat, cela signifie qu'il n'y avait qu'environ 6 chances sur cent d'observer des résidus aussi faibles. Il est donc possible que les données soient sous-dispersées par rapport au modèle binomiale (corrélation, données artificielles...).

Calcul des intervalles de confiance :

Pour calculer l'intervalle de confiance de la dose qui provoque 50 % de mortalité une méthode simple est de calculer la variance sur l'estimation $\hat{x}$ de cette dose et de supposer que $\hat{x}$ suit une loi normale (ou approchée par la loi de Student selon les cas). Il peut parfois être préférable d'utiliser une autre méthode, basée sur la vraisemblance par exemple (voir le chapitre 6), notamment si on s'intéresse à des proportions proches de 0 ou de 1. Dans notre cas de la DL50, pour p = 0.5 la distribution binomiale est symétrique et avec 20 résolutions binaires on peut considérer réaliste l'approximation normale.

x étant le régresseur, y les observées et β le vecteur de paramètres, d'une manière générale après ajustement à un modèle f on obtient la relation :

$y = f(x,\beta) + \varepsilon_y$ ε_y l'erreur résiduelle. Si f est adéquat ε_y est d'espérance nulle

La vraie valeur de β est inconnue, mais estimée par l'ajustement avec une certaine erreur :

$\hat{\beta} = \beta + (\hat{\beta} - \beta)$ et la valeur prédite par le modèle s'écrit : $\hat{y} = f(x, \hat{\beta})$

Nous nous intéressons à la prédiction inverse, à partir de la proportion de morts retrouver la dose :

(0) $\hat{x} = f^{-1}(y, \hat{\beta})$

Si f est linéaire par rapport aux paramètres et si on note $\varepsilon_\beta = (\hat{\beta} - \beta)$ on peut écrire :

(1) $\hat{x} = f^{-1}(y, \hat{\beta}) = f^{-1}(y, \beta + \varepsilon_\beta) = f^{-1}(y, \beta) + \frac{\partial f^{-1}}{\partial \alpha}\varepsilon_\beta$ et il vient :

(2) $\mathrm{var}(\hat{x}) = \mathrm{var}(\frac{\partial f^{-1}}{\partial \beta}\varepsilon_\beta) = \frac{\partial f^{-1}}{\partial \beta} Var(\varepsilon_\beta) \frac{\partial f^{-1}}{\partial \beta}^T = \frac{\partial f^{-1}}{\partial \beta} Var(\hat{\beta}) \frac{\partial f^{-1}}{\partial \beta}^T$

avec $\mathrm{Var}(\varepsilon_\beta) = Var(\hat{\beta})$, matrice de variance-covariance du vecteur paramètre. Pour un estimateur des moindres carrés elle s'écrit :

(3) $Var(\hat{\beta}) = \sigma_r^2 (X^T W X)^{-1}$

Où σ_r^2 est un terme de variance résiduelle, W une éventuelle matrice diagonale de poids et X est le Jacobien J de f. J est une matrice nxp dont le terme général $J_{i,j} = \partial f(y_i) / \partial \beta_j$. Dans le cas d'un modèle linéaire J est la classique matrice X issue des variables explicatives (matrice d'incidence).

Dans le cas d'une régression par les moindres carrés ordinaires la matrice de poids est l'identité, et σ_r^2 est la variance résiduelle (variance constante des observations). Une estimation non biaisée en est donnée par :

(4) $\sigma_r^2 = \frac{1}{n-p} \Sigma (y_i - \hat{y}_i)^2$ p, nombre de paramètres du modèle

Dans le cas de glm, on suppose que les observées y_i sont les réalisations de variables aléatoires Y_i et une manière d'obtenir une estimation de la variance de $\hat{x}$ est de faire l'approximation quadratique du maximum de vraisemblance (estimateur des moindres carrés pondérés par l'inverse de la variance théorique des observations, voir le document formateur "l'algorithme IRLS"). On obtient alors des résidus normés dont la variance résiduelle théorique σ_r^2 est constante et vaut 1. Et la matrice de variance-covariance se simplifie :

(5) $Var\cov(\hat{\beta}) = (X^T WX)^{-1}$ avec W matrice diagonale de terme: $W_{i,i} = 1/var(Y_i)$

La fonction proposée par Venables et Ripley (p.192) , notée dose.p, met en œuvre ces calculs : Eta représente y et dose représente x :

```
> dose.p_function(obj, cf = 1:2, p = 0.5)
+{
+          eta <- family(obj)$link(p)
+          b <- coef(obj)[cf]
+          dose <- (eta - b[1])/b[2]
+          names(dose) <- paste("p =", format(p), ":", sep = "")
+          pd <-  - cbind(1, dose)/b[2]
+          vr <- ((pd %*% vcov(obj)[cf, cf]) * pd) %*% c(1, 1)
+          attr(dose, "SE") <- sqrt(vr)
+          attr(dose, "p") <- p
+          class(dose) <- "glm.dose"
+          dose }
```

obj: objet résultant d'une régression.
cf: vecteur permettant de retrouver les deux paramètres du régresseur considéré. Initialisé par défaut pour un seul régresseur.
p : vecteur des probabilités pour lesquelles on cherche le résultat (par défaut 0,5).
eta: transformation de la probabilité souhaitée (p) dans l'échelle linéaire (dans notre cas le lien logit)
dose: valeur de la variable explicative correspondant à la probabilité p (=réciproque du prédicteur linéaire sur éta).
pd: dérivées partielles de la réciproque du prédicteur linéaire par rapport à ses paramètres et données ci-dessous par (6).
vr: variance de $\hat{x}$. Son calcul général est donné par (2). Le produit matriciel par c(1,1), joint au produit terme à terme entre vcov[] et pd sert à extraire la diagonale qui contient les variances.
vcov: matrice de variance-covariance des paramètres. Elle est redéfinie pour remplacer le terme de variance résiduelle σ^2 par le terme dit de "dispersion" dans glm, et normalement égal à 1 si le modèle de variance choisi est valide. La commande "cov" extrait directement $(X^TWX)^{-1}$ de obj. Notons que si facteur de dispersion = 1, vcov est inutile et l'on pourrait directement appeler cov.

```
> vcov
function(obj) {
          so <- summary(obj)
          so$dispersion * so$cov.unscaled   }
```

Application à notre cas avec

$y = a + bx$ et $y = \text{logit}(p)$
$x = f^{-1}(y,a,b) = (y-a)/b$ donc

(6) $pd[1] = df^{-1}/da = -1/b$ $pd[2] = df^{-1}/db = -(y-a)/b^2$

La fonction dose.p calcule les doses et les écarts-type pour le vecteur de probabilités fourni, dans l'échelle logarithmique des doses. Calcul pour trois probabilités (1/4, 1/2, 3/4) :

```
> dose.p(res.glm,1:2,(1:3)/4)
            Dose        SE
p =0.25: 1.330448 0.2742652
p =0.50: 2.287367 0.2154297
```

p =0.75: 3.244286 0.2648520

Sous l'approximation normale l'intervalle de confiance à 95% est calculé en multipliant cet écart type par le quantile à 97,5% de la loi normale (1.96). Et l'on revient à l'échelle des doses en appliquant 2^x à la valeur trouvée et aux bornes de son intervalle de confiance (on était en log2 des doses).

3- Régression linéaire simple sur l'ensemble des observées

L'utilisation de ce modèle suppose que :

- les valeurs du régresseur (les doses) sont parfaitement connues,
- le modèle linéaire est valide et les erreurs sont indépendantes,
- les erreurs sont de variance constante et d'espérance nulle.

Les deux dernières hypothèses ne sont pas valides. Même dans cette échelle logarithmique des doses les données montrent une courbe légèrement sigmoïde et les résidus afficheraient une légère structure. Et considérer une variance constante des écarts par rapport au modèle, quelle que soit la dose, paraît ici peu réaliste. Cela revient à supposer que les proportions observées sont connues avec la même incertitude dans toute la gamme du toxique. Or il est clair que la mortalité des mites est en principe nulle aux doses basses, et totale aux fortes doses. On peut donc s'attendre à ce que la variance soit nulle à ces deux extrêmes.

Deux hypothèses de la régression linéaire simple, indépendance et homogénéité des variances, ne sont pas vérifiées. Ce modèle doit être rejeté.

4- Régression linéaire sur la partie centrale

On peut chercher à lever la deuxième contrainte, et à limiter la dernière, en limitant l'ajustement à la partie la plus linéaire (la plus centrale) de la courbe, c'est à dire aux points 2 à 5 sur notre exemple. On constate en effet empiriquement que bien souvent, dans le cas de données sigmoïdes, la variance reste assez homogène dans la zone centrale, même si sa loi est mal connue. A titre d'exemple, dans le cas binomiale et pour n=20, la variance est maximale pour p=0.5 et vaut 0.0125. Elle vaut encore 0.0105 pour p=0.3 ou 0.7 mais n'est déjà plus que 0.008 pour p=0.2 ou 0.8 et 0.0045 pour 0.1 ou 0.9. Dans la gamme 0.3 - 0.7 il n'est donc pas irréaliste de supposer la variance grossièrement constante. Sur notre exemple, les points 2 à 5 correspondent à une gamme 0.2 - 0.8 ce qui est déjà discutable.

Sous Splus, exemples d'extraction de sous-matrices (points 2 à 5) :

```
> MV[2:5,]
     mort viv
[1,]    4  16
[2,]    9  11
[3,]   13   7
[4,]   18   2
```

```
> ldose[2:5]
[1] 1 2 3 4
```

Ajustement d'un modèle linéaire sur les points centraux avec la fonction lm() et stockage des résultats dans res.lmc :

```
> res.lmc_lm(mort[2:5,]/(mort[2:5]+viv[2:5])~ldose[2:5])
```

Le R2 est 0,998 et les résidus sont très faibles. On peut imaginer répondre aux questions posées.

Mais il nous faut modifier la fonction dose.p car nous travaillons directement sur les probabilités observées (pas de passage par "eta"), et nous devons réintroduire une estimation de la variance résiduelle dans le calcul de la matrice de variance-covariance des paramètres puisque nous sommes dans le cadre standard de l'estimateur des moindres carrés. Créons la fonction dose.lm inspirée de dose.p :

```
> dose.lm_function(obj, cf = 1:2, p = 0.5)  {
```

```
+         b <- coef(obj)[cf]
+         dose <- (p - b[1])/b[2]
+         names(dose) <- paste("p =", format(p), ":", sep = "")
+         pd <-  - cbind(1, dose)/b[2]
+         vr <- ((pd %*% vcov.lin(obj)) * pd) %*% c(1, 1)
+         attr(dose, "SE") <- sqrt(vr)
+         attr(dose, "p") <- p
+         class(dose) <- "glm.dose"
+         dose  }
```

avec une nouvelle fonction de variance covariance notée vcov.lin() dans laquelle nous réintroduisons le terme de variance résiduelle estimé par (3). Splus nous fournit en fait l'écart-type qu'il faut donc porter au carré :

```
> vcov.lin
function(obj)  {
         so <- summary(obj)
         so$sigma^2 * so$cov  }
```

Comme précédemment, calcul des doses et de leurs écarts type

```
> dose.lm(res.lmc,1:2,(1:3)/4)
         Dose       SE
p =0.25: 1.195652 0.05281601
p =0.50: 2.282609 0.03501632
p =0.75: 3.369565 0.04354499
```

Les doses calculées, et en particulier la DL50, sont très proches de celles obtenues par glm. La variance sur la dose prédite ayant été calculée à l'aide d'une variance résiduelle estimée par (3), on utilise les quantiles d'une loi de Student à n-p = 4 - 2 = 2 degrés de liberté et non directement ceux de la loi normale. Donc ici 4,3 au lieu de 1,96.

Remarques :

- Le domaine d'utilisation du modèle est réduit (2 à 16 dans l'échelle des doses). Attention de bien rester dans le domaine de validité du modèle (ce qui est d'ailleurs vrai pour tout type de modèle) car il va manifestement sous-estimer la mortalité aux doses < 2 et les surestimer aux doses > 16

- Les écarts-types sont très petits, 6 fois plus petits que ceux calculés avec glm, et conduisent, compte tenu des 2 degrés de liberté restants, à des intervalles de confiance environ 3 fois plus petits qu'avec glm. On retombe logiquement sur la constatation de sous-dispersion vue plus haut : les données, prises ici sans considération d'un modèle de variance particulier, s'ajustent très bien au modèle. Elles conduisent à des intervalles de confiance notablement plus étroits que si l'on avait pris en compte et accepté le caractère binomiale, comme nous venons de le faire avec glm.

Cependant, dans le cas où l'on ne s'intéresse qu'à la partie centrale de la courbe (calcul d'une DL50), où l'on a "suffisamment" d'observations dans cette zone linéaire (ce qui n'est pas le cas ici, seuls les deux points centraux sont vraiment dans la zone linéaire à variance pratiquement constante), que les résidus apparaissent non corrélés et que l'on puisse faire l'hypothèse d'une variance résiduelle constante dans la zone considérée, cette démarche d'un ajustement linéaire sur la portion linéaire des données est parfaitement recevable. Elle ne doit cependant être employée que lorsqu'il est impossible de faire du glm, et en particulier lorsque il est impossible de définir la loi que suivent les observations. On peut d'ailleurs éventuellement se contenter de modéliser la variance comme une fonction de l'espérance, sans nécessairement se rattacher à une distribution connue. On fait alors ce que l'on appelle de la quasi-vraisemblance, ce qui est toujours préférable à enlever des données *a posteriori* comme nous l'avons fait ici.

5- Linéariser en transformant les observées

L'idée est de chercher une transformation des proportions observées qui élargisse la zone linéaire et qui homogénéise simultanément la variance. Pour cette deuxième contrainte, il faut que la transformation ait tendance à réduire les distances dans les zones où la variance est forte, et au contraire à dilater les distances, dans les zones où la variance est faible.

5.1- La transformation arcsinus(racine())

Dans le cas où les observées sont des proportions, la transformation arcsinus(racine(proportions observées)) est classiquement employée pour à la fois linéariser les données et homogénéiser les variances. Dans notre cas de données binomiales, cette variance est alors approximativement égale à 1/(4n), n étant le nombre de répétitions binaires.

Puisque nous connaissons maintenant *a priori* la variance des observations, nous allons effectuer une régression linéaire sur les données transformées en pondérant le carré des écarts au modèle par l'inverse de cette variance (normalisation des écarts). Ceci parce que minimiser ce critère, qui est alors la distance du X^2 entre données et modèle avec une variance connue, revient à maximiser la vraisemblance (document formateur "L'algorithme IRLS"). Notons bien qu'il s'agit d'une approximation du X^2 puisque 1/(4n) n'est pas la valeur exacte de la variance, mais l'intérêt est que cette approximation est connue à priori et est indépendante des paramètres du modèle.

Le fait de pondérer, alors que les poids sont identiques pour tous les points (n=20 pour tous les lots), est simplement plus pratique pour ensuite calculer la matrice de variance covariance : la variance résiduelle théorique des résidus normés est alors 1 (analogue au facteur de dispersion de 1 de glm). L'alternative est de ne pas pondérer et de réintroduire la variance résiduelle théorique dans la fonction vcov()[1]. Dans notre cas, n=20, l'inverse de la variance est donc 4*n = 80. L'instruction "rep(80,6)" créé un vecteur contenant 6 fois 80. La commande ci-dessous effectue à la fois la transformation des données (asin(sqrt...) et la régression linéaire pondérée (lm avec un argument supplémentaire "weights" = poids). Les résultats de l'ajustement sont stockés dans res.asin :

```
> res.asin_lm(asin(sqrt(mort/(mort+viv)))~ldose,weights=rep(80,6))
> summary(res.asin)

Call: lm(formula = asin(sqrt(mort/(mort + viv))) ~ ldose, weights = rep(80, 6))
Residuals:
       1       2      3        4     5       6
 -0.1601 -0.1148 0.2304 -0.04367 0.656 -0.5678

Coefficients:
             Value Std. Error t value Pr(>|t|)
(Intercept)  0.2434  0.0372     6.5386  0.0028
      ldose  0.2331  0.0123    18.9554  0.0000

Residual standard error: 0.4601 on 4 degrees of freedom
Multiple R-Squared: 0.989
F-statistic: 359.3 on 1 and 4 degrees of freedom, the p-value is 4.562e-05

Correlation of Coefficients:
      (Intercept)
ldose -0.8257
```

La linéarisation est même plutôt meilleure que la linéarisation logit que nous verrons plus bas (R2 légèrement supérieur, que confirmerait un graphe).

---

[1] Si variance sur les observations constante et modèle adéquat alors :
Régression non pondérée : var(résidusNP) = var(observation) = σ^2 = 1/4n dans notre cas.
Régression pondérée: var(résidusP) = var(résidusNP/ σ) = var(résidusNP)/ σ^2 = 1.

Pour calculer la DL 50, il faut encore modifier la fonction dose.p pour entrer explicitement la valeur de eta (valeur de la probabilité désirée dans l'échelle linéaire) que nous calculerons directement à l'appel de la fonction. Soit dose.lm2 cette nouvelle fonction. Au passage, on en profite pour mettre l'argument "cf" qui ne nous sert à rien en dernier pour ne plus avoir à s'en soucier. Et nous appelons directement la matrice de variance covariance de la régression, en prenant donc implicitement une variance résiduelle de 1.

```
> dose.lm2_function(obj, p = 0.5, eta = 0, cf = 1:2)  {
        b <- coef(obj)[cf]
        dose <- (eta - b[1])/b[2]
        names(dose) <- paste("p =", format(p), ":", sep = "")
        pd <-  - cbind(1, dose)/b[2]
        vr <- ((pd %*% summary(obj)$cov[cf, cf]) * pd) %*% c(1, 1)
        attr(dose, "SE") <- sqrt(vr)
        attr(dose, "p") <- p
        class(dose) <- "glm.dose"
        dose  }
```

Calculons ces DL dans une large gamme de probabilités, en appliquant la fonction arcsinus à la racine des probabilités recherchées (calcul explicite de eta) :

```
> dose.lm2(res.asin,(1:9)/10,asin(sqrt((1:9)/10)))
              Dose        SE
p =0.1: 0.3361028 0.3161040
p =0.2: 0.9449185 0.2648582
p =0.3: 1.4425879 0.2303344
p =0.4: 1.8934361 0.2078207
p =0.5: 2.3254037 0.1968564
p =0.6: 2.7573713 0.1980470
p =0.7: 3.2082194 0.2120068
p =0.8: 3.7058889 0.2397345
p =0.9: 4.3147045 0.2857511

à comparer au glm :
> dose.p(res.glm,1:2,(1:9)/10)
              Dose        SE
p =0.1: 0.3735282 0.3957031
p =0.2: 1.0798691 0.3024032
p =0.3: 1.5493488 0.2529727
p =0.4: 1.9341964 0.2256664
p =0.5: 2.2873668 0.2154297
p =0.6: 2.6405373 0.2214776
p =0.7: 3.0253849 0.2451147
p =0.8: 3.4948646 0.2916229
p =0.9: 4.2012054 0.3826633
```

Les résultats sont très proches, surtout pour la DL50 car les courbes sont symétriques et se croisent toutes peu ou prou autour de p=0.5. Et les écarts type sont bien estimés, au moins dans la zone de probabilités 0.25-0.75. Il sont inférieurs d'environ 10% sur cet exemple. Ceci est logique : le modèle s'ajuste légèrement mieux qu'avec une transformation logit et la matrice de variance-covariance des paramètres, est donc plus faible.

Conclusion, cette approche, très utilisée avant les développements de la théorie du glm, donne de bons résultats, surtout si comme ici dans le cas binomiale, on peut estimer la variance théorique attendue de la transformation et qu'elle est en outre indépendante des paramètres. Etant en outre simple, elle est encore très employée. Mais pour les probabilités proches de 0 ou de 1, l'approche arcsinus(racine(p)) montre tout de même ses limites.

5.2- La transformation logit

Il existe bien d'autres fonctions sigmoïdes et en particulier la fonction logit classiquement employée par glm. Mais elle n'est applicable que si les données ne montrent aucun 0 ou 1.

Par commodité, créons le vecteur p des proportions observées :
```
> p_mort/(mort+viv)
 [1] 0.05 0.20 0.45 0.65 0.90 0.95
```

Transformation logit pour linéariser les données et affectation dans le vecteur noté logitp :
```
> logitp_log(p/(1-p))
 [1] -2.9444390 -1.3862944 -0.2006707  0.6190392  2.1972246  2.9444390
```

Rem : on aurait pu écrire logitp_family(binomial)$link(p) pour récupérer le lien défaut de la famille binomiale.

Nous souhaitons, comme dans glm, tenir compte du fait que la variance de nos données est binomiale. Il faut donc connaître la variance de logit(p) mais la loi que suit logit(p) est inconnue. Par analogie avec glm, nous prendrons var(logit(p) ≈ 1/(np(1-p)). p est le vecteur inconnu des probabilités théoriques. On peut estimer p par les proportions observées car cette estimation tend bien vers p quand n augmente. **Mais cela n'implique pas que la variance estimée avec p observé tend vers la vraie variance quand n augmente, du fait du terme n dans la variance**. Cette procédure d'estimation de la variance n'est donc même pas valable asymptotiquement et relève de la recette de cuisine.

Avec lm(), régression linéaire sur les données transformées (logitp) en pondérant par np(1-p).

```
> res.tlogit_lm(logitp~ldose,weights=20*p*(1-p))
> summary(res.tlogit)

Call: lm(formula = logitp ~ ldose, weights = 20 * p * (1 - p))
Residuals:
      1      2      3      4      5       6
 -0.331 0.1432 0.2828 -0.409 0.3326 -0.1398

Coefficients:
              Value Std. Error  t value Pr(>|t|)
(Intercept)  -2.6049   0.1892   -13.7687   0.0002
      ldose   1.1386   0.0710    16.0399   0.0001

Residual standard error: 0.3562 on 4 degrees of freedom
Multiple R-Squared: 0.9847
F-statistic: 257.3 on 1 and 4 degrees of freedom, the p-value is 8.834e-05

Correlation of Coefficients:
      (Intercept)
ldose -0.8854
```

Les paramètres de pente et d'origine sont identiques à 1% près à ceux de glm et les courbes sont quasi confondues dans l'échelle des probabilités.

Calcul de la DL 50

La fonction à utiliser est très proche de dose.p, à ceci près qu'il faut forcer le lien logit (calcul de "eta") car lm ne propose évidemment pas de fonction de lien, et toujours prendre directement la matrice de covariance comme dans la transformation arcsinus(racine(p)), puisque nous avons normé les écarts. On modifie donc encore la fonction dose.p du Venables, que nous appelons dose.lmt

```
> dose.lmt  function(obj, cf = 1:2, p = 0.5)   {
         eta <- family(binomial)$link(p)
         b <- coef(obj)[cf]
```

```
dose <- (eta - b[1])/b[2]
names(dose) <- paste("p =", format(p), ":", sep = "")
pd <-  - cbind(1, dose)/b[2]
vr <- ((pd %*% summary(obj)$cov[cf, cf]) * pd) %*% c(1, 1)
attr(dose, "SE") <- sqrt(vr)
attr(dose, "p") <- p
class(dose) <- "glm.dose"
dose   }
```

Exécution pour les trois probabilités et comparaison avec glm :

```
> dose.lmt(res.lm,1:2,(1:3)/4)
        Dose      SE
p =0.25:  1.322962  0.2827948
p =0.50:  2.287876  0.2172461
p =0.75:  3.252790  0.2673436

> dose.p(res.glm,1:2,(1:3)/4)
        Dose      SE
p =0.25:  1.330448  0.2742652
p =0.50:  2.287367  0.2154297
p =0.75:  3.244286  0.2648520
```

Les doses prédites et les intervalles de confiance sont très proches du glm. Il peuvent même être jugés identiques si l'on considère qu'il s'agit de toute façon d'approximations dans les deux cas.

Calcul de la déviance résiduelle

Puisque les observations suivent une loi binomiale, nous pouvons calculer la vraisemblance de ce modèle et sa déviance par rapport au modèle saturé. Pour calculer la vraisemblance, produit des densités de probabilité, il faut calculer les densités.

On utilise la fonction Splus "dbinom" qui calcule la densité de probabilité binomiale à partir des 3 arguments définissant la réalisation : arg1= nb de succès, arg2 = nb essais, arg3 = probabilité de succès.

On a aussi besoin de calculer les probabilités théoriques (prédites) de mortalité en fonction de la dose. Pour simplifier les expressions, écrivons la fonction de lien inverse de la logit (notée prob.logit), qui sera plus facile à utiliser qu'en invoquant directement le lien inverse fourni par la famille binomiale (family(binomial)$inverse()). Cette fonction attend en arguments la dose pour laquelle on souhaite calculer la probabilité, et les paramètres a (pente) et b (ordonnée à l'origine) définissant le prédicteur linéaire.

```
> prob.logit_function(dose, a, b) {  exp(a * dose + b)/(1 + exp(a * dose + b))   }
```

La vraisemblance de notre modèle, dit modèle candidat (Lc), est calculée en fournissant à dbinom la probabilité théorique donnée par prob.logit sur les paramètres estimés par la procédure lm.

La vraisemblance du modèle saturée (Ls) est calculée en fournissant à dbinom les probabilités observées p.

La déviance résiduelle est 2 fois la différence entre les log(Lc) et log(Ls), sachant que le passage au log des produits de densité donne des sommes des log des densités :

```
> 2*(sum(-log(dbinom(mort,20,prob.logit(ldose,res.tlogit$coef[2],res.tlogit$coef[1]))))
+ -sum(-log(dbinom(mort,20,p))))
[1]  0.5220206
```

L'expression est un peu longue à écrire, mais ouf, on est proche la déviance résiduelle de glm (0.5195998).

Et on vérifie que la somme des carrés des résidus donne l'approximation quadratique de la déviance :

DevAproch = (n-p)* σ^2 = 4*0.3562^2 = 0.5074364

On peut vérifier que la variance résiduelle sigma2 est bien le carré du " residual standard error: 0.3562 " donné dans le summary(res.tlogfit))

```
> (summary(res.tlogit)$sigma)^2
[1] 0.1268591
```

et qu'il est bien bâti sur les résidus pondérés

```
> sum(summary(res.tlogit)$residual^2)/4
 [1] 0.1268591
```

(alors que attention res.tlogit$residual donne les résidus non pondérés !!!)

La variance résiduelle est beaucoup plus petite que 1, variance théorique attendue compte tenu de nos données binomiales dont nous avons tenu compte en pondérant par une estimation de l'inverse des variances. Comme pour le glm, le modèle apparaît sous-dispersé, ce qui se teste en considérant que dans le cas binomial la somme des carrés des écarts suit approximativement un X^2 à n-p degrés de libertés.

Remarques importantes :

1- Transformer les variables n'est pas toujours licite, ici il ne faut ni 0 ni 1. Une alternative est alors la transformation dite "logistique empirique" qui consiste à prendre comme variable transformée $z_i = \log(y_i+1/2 \ / \ (n_i-y_i+1/2)$ et comme estimation de sa variance $var(Z_i) = (y_i+1/2)^{-1} + (n_i-y_i+1/2)^{-1}$ avec $y_i=n_ip_i$ (voir par exemple McCullagh et Nelder page 107). L'ajout de la constante permet de rendre possible cette transformation quel que soit p. On remarquera que si cette constante vaut 0, alors la transformation est logit(y) et l'estimation de la variance vaut 1/(np(1-p)). C'est ce que nous venons de faire. Testons cette transformation dans notre cas :

```
> emplogit_log((mort+1/2) /(20-mort+1/2))
 [1] -2.5649494 -1.2992830 -0.1910552  0.5877867  2.0014800  2.5649494
```

à comparer avec la transformation logit

```
> logitp
[1] -2.9444390 -1.3862944 -0.2006707  0.6190392  2.1972246  2.9444390
```

La différence est tout de même sensible. Estimation de la variance pour cette transformation :

```
> varemplogit_1/(mort+1/2) + 1/(20 -mort+1/2)
 [1] 0.7179487 0.2828283 0.1922197 0.2074074 0.4540541 0.7179487
```

à comparer avec l'estimation 1/np(1-p)

```
> 1/20/p/(1-p)
[1] 1.0526316 0.3125000 0.2020202 0.2197802 0.5555556 1.0526316
```

les valeurs sont assez différentes lorsqu'on s'approche de 0 ou de 1.

Regression linéaire sur cette transformation

```
> res.emplogit_lm(emplogit~ldose, weights=1/varemplogit)
Coefficients:
 (Intercept)    ldose
   -2.359015 1.027835
```

Les résultats sont donc logiquement un peu différents. Estimation de la DL50

```
> dose.lmt(res.emplogit,1:2,(1:3)/4)
         Dose       SE
p =0.25: 1.226270 0.2997480
p =0.50: 2.295131 0.2263923
p =0.75: 3.363992 0.2811883
```

A comparer avec les résultats de glm. On est logiquement moins proche qu'avec la transformation directe via la logit. Mais cette transformation empirique est toujours possible, quelles ques soient les valeurs des observations. Elle accumule cependant les approximations plus ou moins licites et devrait être définitivement abandonnée au profit de glm.

2- On a fait deux approximations concernant la variance : utiliser 1/np(1-p) par analogie avec glm, et remplacer p par les probabilités observées, non par les théoriques inconnues. Car rappelons que dans glm, 1/np(1-p) est la variance de la variable de travail, et non pas celle de logit(p). On peut vérifier que même si on remplace p par l'estimation de sa valeur théorique trouvée par glm, la régression ne donne pas les bons résultats.

```
>lm(logitp~ldose,weights=20*prob.logit(ldose,1.148072,-2.626062)*(1-prob.logit(ldose,1.148072,-2.626062)))
(Intercept)    ldose
   -2.647154 1.157609
```

Par contre, si l'on fait vraiment la régression linéaire de la variable de travail de IRLS pondérée par np(1-p), la formule est un peu longue mais on retrouve exactement les valeurs obtenues par glm (se référer au document formateur "l'algorithme IRLS") :

```
>  lm(1.148072*ldose-2.626062+(p-prob.logit(ldose,1.148072,-2.626062))/
+    prob.logit(ldose,1.148072,-2.626062)/(1-prob.logit(ldose,1.148072,-2.626062))~ldose,
+  weights=20*prob.logit(ldose,1.148072,-2.626062)*
+  (1-prob.logit(ldose,1.148072,-2.626062)))
(Intercept)    ldose
   -2.626062 1.148072
```

Mais la vraie valeur de np(1-p) est inconnue, c'est pourquoi IRLS travaille de manière itérative, en partant d'une valeur de p estimée par les proportions observées.

Conclusion

Faire des moindres carrés pondérés par l'inverse de la variance revient bien à maximiser la vraisemblance, à condition de connaître a priori cette variance. Or ce n'est le cas que pour la transformation arcsin(racin()), et encore, il s'agit d'une approximation. Remplacer cette variance par une estimation basée sur les observées peut suffire, comme sur cet exemple, mais peut conduire à des résultats biaisés dans la mesure où l'on est pas sûr que cette estimation converge vers la vraie valeur de la variance quand le nombre d'observations augmente. Cela dépend de la loi, mais c'est déjà faux dans le cas binomiale, et c'est encore pire avec une loi de Poisson par exemple. **Ces procédures empiriques qui ont pu être utilisées par le passé sont donc à proscrire, surtout qu'elles reposent sur le fait que l'on se donne une loi sur les observées et que glm est donc pleinement utilisable**.

6- Moindres carrés non linéaire pondérés

Pour éviter l'écueil de la transformation de la variable, on peut minimiser directement la somme des carrés pondérés sans chercher à se retrouver dans une échelle linéaire. Mais les variances théoriques nous sont toujours inconnues. Comme précédemment, une solution approchée est d'utiliser les proportions observées, avec tous les inconvénients que nous venons de soulever.

La procédure nls (non linear least squares) de S+ minimise la somme des carrés de la fonction passée en argument. Dans notre cas, il faut donc lui donner les écarts normés (divisés par la racine de la variance). Les paramètres sont définis dans l'option "start", ainsi que leur valeur de départ (nls utilise un processus itératif de minimisation). p est toujours le vecteur des probabilités observées. L'objet résultant est stocké dans res.nls :

```
> res.nls_nls(~(p-prob.logit(ldose,a,b))/ sqrt(p*(1-p)/20), start=list("a"=1, "b"=-2.5))

Residual sum of squares : 0.5337929
parameters:
        a         b
 1.166327 -2.665728
```

On est effectivement très proche des résultats de glm, et comme prévu la somme des carrés résiduelle est maintenant l'approximation quadratique de la déviance résiduelle.

Pour construire des intervalles de confiance, on peut alors procéder de la même manière qu'avec glm ou lm pondéré, en considérant que (1) devient une approximation de (0) par un développement limité du premier ordre au voisinage des paramètres estimés, et sachant que nls fournit une approximation de la matrice de variance covariance des paramètres fondée sur la même approximation. Dans la fonction dose.p, il faut également redéfinir "pd" comme la matrice des dérivées partielles du lien logit par rapport à ses paramètres.

Conclusion :

Cette procédure est très proche de la transformation logit des données que nous venons de voir mais elle ne donne pas exactement le même résultat dans la mesure où les pondérations sont dans les deux cas fondées sur des variances approchées. Le critère n'est donc pas rigoureusement le même. Son seul avantage serait de permettre un ajustement à des observées égales à 0 ou 1 sans avoir recours à des astuces de transformation qui nous éloignent quand même de la solution. Il reste néanmoins que la pondération serait infinie pour ces valeurs. On les remplace alors respectivement par (0.5/n et 1-0.5/n) dans le calcul des poids approchés. Mais les remarques faites précédemment sur les dangers d'approcher la variance par une estimation basée sur les observées sont également valables ici et cette procédure n'est donnée qu'à titre didactique.

Remarque importante :

nls étant une procédure itérative, on peut être tenté de faire entrer dans le critère, non plus un poids approché, mais un poids qui dépende de la valeur des paramètres en cours, comme dans la commande ci-dessous :

```
> res.nls_nls(~(p-prob.logit(ldose,a,b))/ sqrt(prob.logit(ldose,a,b)*
+ (1-prob.logit(ldose,a,b))/20), start=list("a"=1, "b"=-2.5))

Residual sum of squares : 0.5097134
parameters:
        a        b
 1.139218 -2.60676
```

Prob.logit() étant évaluée à chaque itération de nls on peut alors penser qu'à la convergence, les variances entrant dans le calcul des pondérations seront bien les variances théoriques recherchées et que la procédure est alors équivalente à celle utilisée par glm.

Pourquoi cette procédure qui cherche à minimiser la distance du χ^2 entre modèle et données est-elle fausse, et ne retrouve-t-on pas exactement les résultats de glm ? La vraie procédure consiste à se fixer une variance approchée, puis à minimiser le critère, ce qui donne une nouvelle estimation des paramètres et donc de la variance approchée, et à renouveler ce processus jusqu'à convergence. Or nous avons minimisé un autre critère, une fonction complexe des écarts dont les poids ne sont pas constants. Techniquement, les dérivées partielles du critère par rapport aux paramètres sont donc différentes et conduisent à un gradient différent qui ne mène pas au même minimum. De fait, si l'on pondère par les "vraies" variances (données par exemple par glm) nls converge vers le bon résultat :

```
> nls(~(p-prob.logit(ldose,a,b))/ sqrt(prob.logit(ldose,1.148072,-2.626062)*
+ (1-prob.logit(ldose,1.148072,-2.626062))/20), start=list("a"=1, "b"=-2.5))
Residual sum of squares : 0.5117821
parameters:
    a      b
 1.14805 -2.626023
```

7- Minimiser directement -log(L)

Pour finir, on peut chercher à faire vraiment du maximum de vraisemblance, c'est à dire minimiser directement -log(Lc), c'est à dire $\sum$-log(densités). On utilise pour cela la procédure ms (minimisation de somme) de S+ qui minimise le vecteur fourni. Comme précédemment les densités de probabilités sont calculées avec dbinom. Le résultat est stocké dans res.ms.

>res.ms_ms(~-log(dbinom(mort,20,prob.logit(ldose,a,b))),start=list(a=1,b=-2.5))

```
value: 8.407863
parameters:
        a         b
 1.148072 -2.626062
```

On obtient exactement les résultats de glm avec 7 chiffres significatifs. Une telle similitude s'explique par le fait qu'on peut montrer que l'algorithme utilisé par glm est strictement équivalent à l'algorithme de Fisher (Fisher scoring, Collett p.343 et document formateur "l'algorithme IRLS") utilisé par ms. Si on utilise le lien probit (utiliser la fonction Splus pnorm(), loi normale, inverse du lien logit, à la place de prob.logit() dans l'appel à ms), les résultats sont aussi bons (6 premiers chiffres significatifs égaux).

Le minimum trouvé est 8.41. C'est -log(Lc), que l'on peut retrouver par res.ms$value. On se ramène à la déviance résiduelle en calculant la vraisemblance saturée est en faisant deux fois la différence. On retrouve bien la déviance résiduelle de glm :

> 2*(res.ms$value-sum(-log(dbinom(mort,20,p2))))
[1] 0.5195998

On peut également calculer l'approximation quadratique de la déviance, qui est la somme des carrés des résidus.
> v_prob.logit(ldose,res.ms$param[1],res.ms$param[2])*(1-prob.logit(ldose, res.ms$param[1], res.ms$param[2]))/20
> sum(((p-prob.logit(ldose,res.ms$param[1],res.ms$param[2]))/sqrt(v))^2)
[1] 0.5117821 Elle est effectivement légèrement différente de la vraie déviance.

Pour calculer les intervalles de confiance, on peut procèder comme en section 6, mais ms ne donne aucune approximation de la matrice de variance covariance (c'est logique, elle n'est pas censée faire des moindres carrés). Il faut donc la calculer explicitement selon (5).

La procédure nls peut également être utilisée car elle attend la racine du critère à minimiser qui n'est donc pas nécessairement une somme de carrés :

> nls(~sqrt(-log(dbinom(mort,20,prob.logit(ldose,a,b)))),start=list(a=1,b=-2.5))

```
Residual sum of squares : 8.407863
parameters:
        a         b
 1.148114 -2.626207
```

La somme résiduelle est strictement identique a celle donnée par ms (même minimum du critère heureusement), mais seuls les quatre premiers chiffres significatifs des paramètres a et b sont identiques. Pourtant, nls utilise par défaut l'algorithme de Gaus-Newton dont on peut montrer qu'il est bien équivalent à celui de Fisher dans le cas d'un estimateur des moindres carrés. Mais nous ne sommes pas ici sur des moindres carrés et la différence est sans doute due au fait que nls fait une estimation quadratique des dérivées secondes du critère par rapport aux paramètres, qui n'est alors pas tout à fait exacte. Compte tenu de la corrélation entre a et b (0,8) il est assez logique d'obtenir des estimations légèrement différentes pour ces paramètres mais conduisant au même minimum.

Remarque importante : Même si ces légères différences sur les algorithmes n'ont en général aucune influence pratique sur les résultats, il est impératif d'utiliser nls dans le cadre des moindres carrés et ms dans les autres cas, pour s'assurer une validité aussi large que possible. En particulier, la matrice de variance covariance fournie par nls n'a aucun sens ici et ne doit pas être utilisée pour d'éventuels calculs d'intervalles de confiances.

8- Graphe des résultats d'ajustement

Préparation du dessin, les doses vont de 1 à 32, échelle logarithmique, les probabilités de 0 à 1 :
> plot(c(1,32),c(0,1),type="n",xlab="dose",ylab="proba",log="x")

Affichage des probabilités observées, fonction text, en notant les points par des * :
> text(2^ldose,p,rep("*",6))

Affichage des résultats de glm. On génère une séquence sur les log2 des doses entre 0 (->1) et 5 (->32) avec un pas de 0.1 pour avoir une courbe lisse. La très utile fonction predict de glm donne directement les données modélisées quand on lui spécifie le type "response".
> lines(2^seq(0,5,0.1),predict(res.glm,data.frame(ldose=seq(0,5,0.1)),type="response"))

Affichage des résultats de lm. Predict fonctionne aussi avec lm, mais évidemment dans l'échelle linéaire. Le plus simple est de récupérer l'inverse du lien logit de la famille binomiale pour repasser de l'échelle linéaire à l'échelle des données :
> lines(2^seq(0,5,0.1),family(binomial)$inverse(predict(res.lm,
data.frame(ldose=seq(0,5,0.1)),type="response")))

Affichage des résultats de nls ou ms. Il n'y a pas de fonction predict, car pas de modèle défini à priori (logit et linéaire dans les cas précédents). On reconstruit donc les probabilités à partir des paramètres estimés via la fonction prob.logit définie plus haut (inverse du lien logit).
> lines(2^seq(0,5,0.1),prob.logit(seq(0,5,0.1),res.nls$parameters[1],res.nls$parameters[2]))

Résultats, les trois courbes sont pratiquement superposées, le trait est à peine plus épais que si glm était seul. Les (très légères) différences se font assez logiquement surtout sentir aux extrémités de la courbe.

Affichage des résultats de la transformation arcsinus(racine(p)). On reconstruit les probabilités en prenant les carrés des sinus du prédicteur linéaire :

> lines(2^seq(0,5,0.1),sin(predict(res.asin,data.frame(ldose=seq(0,5,0.1))))^2)

La courbe résultante est un peu moins creuse que la logit, et assez proche d'une probit. Il est impossible de trancher entre ces trois modèles puisque la déviance résiduelle (non calculable pour arcsin mais c'est elle qui a la plus faible somme des résidus) est de toute façon inférieure à 11, limite du rejet à 5% (bilatéral) d'un X^2 à 4 dl. Avec un aussi petit nombre d'observations, et compte tenu de la variabilité liée au processus binomial, ces trois modèles sont équivalents.

9 Conclusion générale :

Sur cet exemple, glm, lm pondéré sur variable transformée, moindres carrés pondérés des observations, et minimisation de -log(L) donnent des résultats extrêmement proches. La différence sur l'estimation des paramètres est toujours inférieure à 1%. De fait, cette différence est à peine perceptible sur les ajustements, et dans le cas d'une application au calcul d'une DL 50 les 4 procédures donnent des résultats strictement équivalents. La transformation arcsinus(racine(p)) tient fort bien la route et reste tout à fait adaptée à des calculs de DL50.

Il est bien entendu recommandé d'utiliser la procédure glm car aucune autre méthode ne cumule autant d'avantages :

- méthode valide sur le plan statistique, reposant sur des bases théoriques fiables,
- le modèle reste cohérent pour toutes les valeurs possibles du régresseur,
- les données observées peuvent prendre des valeurs 0 ou 1,
- on peut facilement introduire d'autres variables explicatives, régresseurs ou facteurs,

Ces méthodes "approchées" (elles le sont toutes, sauf section 7), ont été données dans un but purement didactique, pour mieux faire comprendre la procédure glm. Elles trouvaient leur justification lorsque l'utilisateur devait se débrouiller avec les moyens du bord, souvent un simple tableur, voire une calculatrice. Aujourd'hui, il serait regrettable de ne pas utiliser pleinement glm, une méthode plus valide sur le plan statistique et relativement facile à employer du fait des logiciels grand public que l'on trouve maintenant.