Fiche d'utilisation du logiciel 

5-Modèle linéaire généralisé

D. Chessel & J. Thioulouse

Résumé

La fiche contient le matériel nécessaire pour des séances de travaux dirigés sur R consacrées aux modèles linéaires généralisés.

Plan

1. Introduction : modéliser une probabilité

Supposons qu'on joue à un jeu bizarre dont on fait l'apprentissage au cours d'une série de 20 essais :

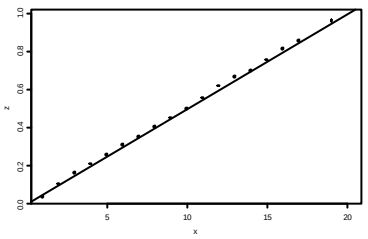
```
> x<-1:20
```

Supposons que la probabilité de gagner croît linéairement de 0.05 le premier coup jusqu'à 1 :

```
> y<-x/20
> y
 [1] 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60
[13] 0.65 0.70 0.75 0.80 0.85 0.90 0.95 1.00
```

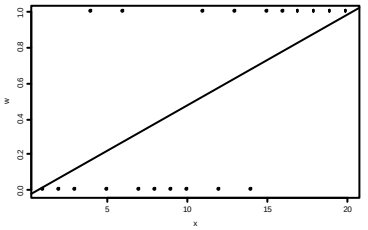
Mettons une petite erreur sur cette probabilité de gagner :

```
> z<-rnorm(rep(1,le=20),y,rep(0.01,le=20))
> z
 [1] 0.03421 0.10178 0.15495 0.20513 0.25622 0.30368 0.34967 0.40025
 [9] 0.44609 0.49408 0.55102 0.61259 0.66149 0.69004 0.74892 0.80398
[17] 0.84907 0.90670 0.95086 1.00449
> z[20]<-0.98 Une probabilité est comprise entre 0 et 1 !
> plot(x,z)
> abline(lm(z~x))
```



Jusqu'à présent, il n'y a rien de bien extraordinaire. Personne ne connaît la probabilité de gagner. On ne peut qu'observer le résultat (gagné ou perdu). Fabriquons donc un résultat observable de ce modèle :

```
> w<-rbinom(rep(1,le=20),rep(1,le=20),z)
> w
 [1] 0 0 0 1 0 1 0 0 0 0 1 0 1 0 1 1 1 1 1 1
> plot(x,w)
> abline(lm(w~x))
```



C'est déjà plus étonnant :

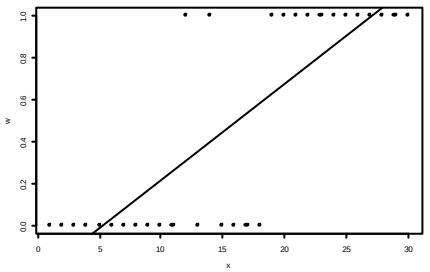
```
> lm(z~x)
Call:
lm(formula = z ~ x)

Coefficients:
 (Intercept)       x
    0.001383 0.04987

Degrees of freedom: 20 total; 18 residual
```

```
Residual standard error: 0.008331
> lm(w~x)
Call:
lm(formula = w ~ x)

Coefficients:
 (Intercept)       x
    -0.03684 0.05113

Degrees of freedom: 20 total; 18 residual
Residual standard error: 0.4257
```

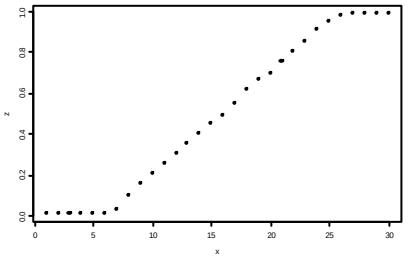
Le modèle y = 0.05*x tient encore si on remplace la probabilité par sa réalisation aléatoire. Moralité : les données peuvent être totalement fausses et le modèle accessible. C'est normal, la statistique considère toujours que les données sont « fausses ».

Compliquons un peu. Avant de commencer à apprendre quoi que ce soit, on prend en général quelques baffes. Pendant 6 parties, on ne comprend rien et la probabilité de gagner vaut 0.01. Après l'apprentissage on a fait le tour de la question et ce n'est plus drôle. Pendant encore 4 parties la probabilité de gagner vaut 0.99 puis on se lasse :

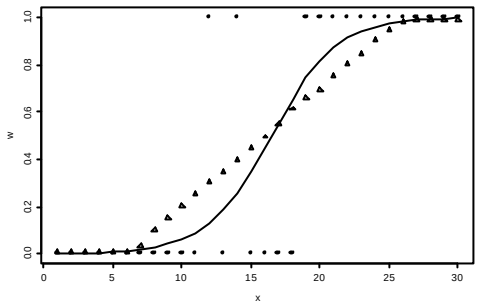
```
> x<-1:30
> z<-c(rep(0.01,le=6),z,rep(0.99,le=4))
> z
 [1] 0.01000 0.01000 0.01000 0.01000 0.01000 0.01000 0.03421 0.10178
 [9] 0.15495 0.20513 0.25622 0.30368 0.34967 0.40025 0.44609 0.49408
[17] 0.55102 0.61259 0.66149 0.69004 0.74892 0.80398 0.84907 0.90670
[25] 0.95086 0.98000 0.99000 0.99000 0.99000 0.99000
> w<-rbinom(rep(1,le=30),rep(1,le=30),z)
> w
 [1] 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
```



Le modèle simple est évidemment invalide sur la réalisation puisque le modèle lui-même n'est pas linéaire :

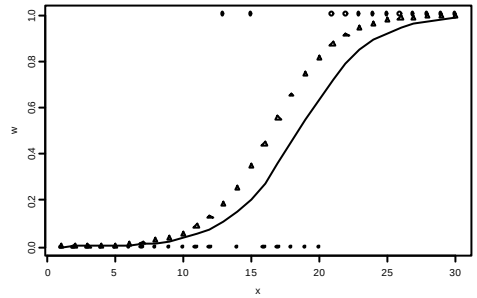



```
> plot(x,w)
> lines(x,predict(glm(w~x,family="binomial"),type="response"))
> points(x,z,pch=2)
```



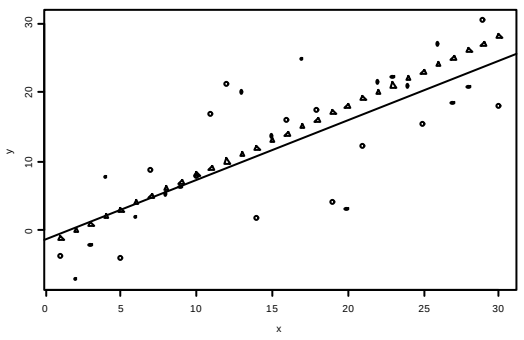
Le modèle qui a fourni la réalisation qui a fourni l'estimation du modèle met en évidence la contradiction. Faisons enfin :

```
> z<- predict(glm(w~x,family="binomial"),type="response")
> w<-rbinom(rep(1,le=20),rep(1,le=20),z)
> plot(x,w)
> lines(x,predict(glm(w~x,family="binomial"),type="response"))
> points(x,z,pch=2)
```



Cet exercice est le même que celui qui a ouvert le débat (fiche 1) :

```
> x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
[22] 22 23 24 25 26 27 28 29 30

> y<-x-2

> y
 [1]  -1   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
[17]  15  16  17  18  19  20  21  22  23  24  25  26  27  28
> y<-y +rnorm(30,0,6)

> plot(x,y)
> points(x,x-2,pch=2)
> abline(lm(y~x))
```

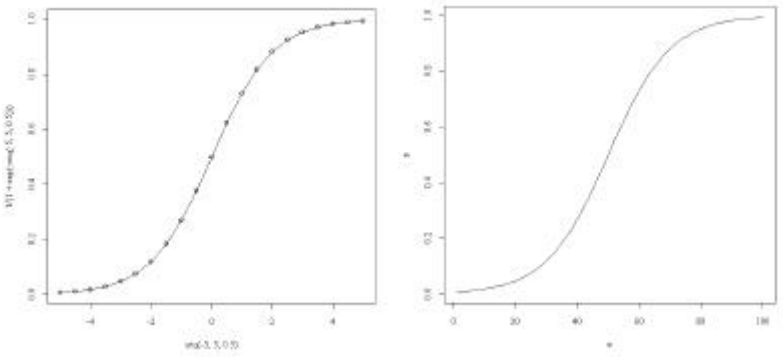


2. Erreurs, liens et déviance

2.1. Erreur de Bernouilly

Considérons une variable de milieu x qui varie de 1 à 100 et p la probabilité de rencontrer une espèce donnée qui varie en fonction de x de manière monotone. Il est impossible d'écrire $p = ax + b$ puisque seules les valeurs de l'intervalle $[0,1]$ ont un sens pour une probabilité. On contourne la difficulté en utilisant la fonction logistique $y = \frac{1}{1+e^{-x}}$:

```
> plot(seq(-5,5,.5),1/(1+exp(-seq(-5,5,.5))))
> lines(seq(-5,5,.5),1/(1+exp(-seq(-5,5,.5))))
```



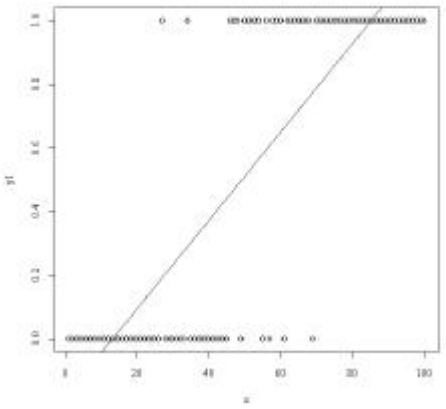
Considérons donc une variable de milieu x qui varie de 1 à 100 et p la probabilité de rencontrer une espèce donnée qui varie en fonction de x de manière monotone :

$$p = \frac{1}{1+e^{-0.10x+5}} \Leftrightarrow \log\left(\frac{p}{1-p}\right) = 0.10x - 5$$

```
> x_1:100
> p_1/(1+exp(-0.10*x+5))
> plot(x,p,type="l")
```

Supposons qu'on échantillonne la présence-absence de l'espèce le long du gradient et que chaque échantillon soit indépendant des autres. Pour une valeur x, la mesure suit une loi de Bernouilly de paramètre $p(x)$. On peut simuler un tel résultat :

```
> y1_rbinom(100,1,p)
> y1
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0
 [38] 0 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1
 [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```



```
> plot(x,y1)
> abline(lm(y1~x))
```

L'estimation directe de la probabilité à partir de x n'a pas de sens. Pour estimer les paramètres du modèle avec l'échantillon :

```
> glm1_glm(y1~x,family=binomial)
> glm1

Call:  glm(formula = y1 ~ x, family = binomial)

Coefficients:
(Intercept)            x
     -6.557        0.135

Degrees of Freedom: 99 Total (i.e. Null);  98 Residual
Null Deviance:      138
Residual Deviance: 48.4         AIC: 52.4
```

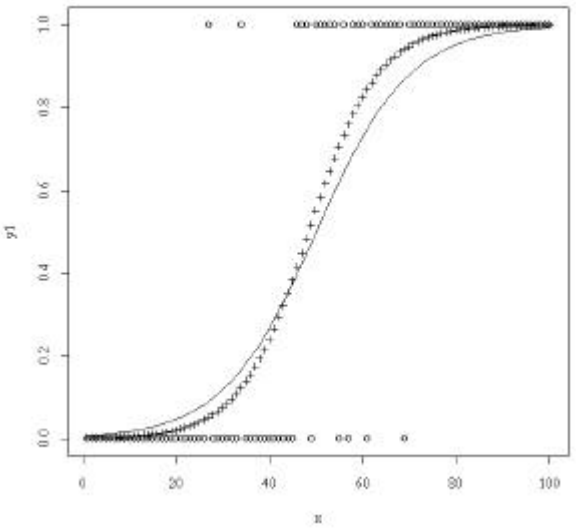
Le terme **family=binomial** signifie deux choses. La première est que y1 suit une loi binomiale (pour n = 1, donc une loi de Bernouilly), la seconde que ce n'est pas p mais log(p/(1-p)) qui est une fonction linéaire de x.

Le « vrai » modèle est défini par a = 0.10 et b = -5. Le modèle estimé est défini par a = 0.135 et b = -6.56.

```
> pred0.link_predict(glm1,type="link")
> pred0.rep_predict(glm1,type="response")
```

On peut donc prédire soit le lien soit la probabilité, l'un dérivant de l'autre :

```
> pred0.link[25]
    25
-3.177
> pred0.rep[25]
     25
0.04005
> log(0.04005/(1-0.04005))
[1] -3.177
> 1/(1+exp(3.177))
[1] 0.04004
> coefficients(glm1)
(Intercept)           x
    -6.5567      0.1352
> 0.1352*25-6.5567
[1] -3.177
```



Modèle vrai, données simulées, modèle estimé

```
plot(x,y1)
points(x,p,type="l")
points(x,predict(glm1,type="response"),pch="+")
```

Le modèle, les observations et l'estimation sont présents sur la figure. On est parti du modèle :

$$Y_i \to B(p_i) \quad p_i = \frac{1}{1+\exp(-0.10x_i+5)}$$

On a trouvé l'estimation $p_i = \frac{1}{1+\exp(-0.13x_i+6.56)}$.

glm1 est un objet de la classe glm :

```
> summary(glm1)

Call:
glm(formula = y1 ~ x, family = binomial)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3803  -0.2547   0.0481   0.2545   2.4329

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -6.5567     1.3934   -4.71  2.5e-06 ***
x             0.1352     0.0277    4.88  1.1e-06 ***
---
Signif. codes:  0  `***'  0.001  `**'  0.01  `*'  0.05  `.'  0.1  ` '  1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 138.469  on 99  degrees of freedom
Residual deviance:  48.352  on 98  degrees of freedom
AIC: 52.35

Number of Fisher Scoring iterations: 5
```

Que signifie la déviance du modèle nul (*Null deviance*) ? Le modèle nul est caractérisé par aucun effet du facteur, donc p = *constante*. L'estimation au maximum de vraisemblance de la probabilité se fait par la fréquence.

```
> sum(y1)
[1] 52
> sum(y1)/100
[1] 0.52
```

La probabilité pour que l'espèce soit présente dans un relevé est 0.52. Quelle est la probabilité de l'observation dans ce modèle. On a observé 52 présences de probabilité 0.52 et 48 absences de probabilité 0.48. La vraisemblance est donc :

$$P(observation) = L(p) = \prod_{i=1}^{n} P(y_i) = 0.52^{52} 0.48^{48}$$

Le logarithme de la vraisemblance est donc :

$$\log\left(P(observation)\right) = LL(p) = 52\log(0.52) + 48\log(0.48) = -69.2347$$

La déviance du modèle nulle (`Null deviance: 138.469  on 99  degrees of freedom`) est par définition :

$$-2LL(p) = 138.47$$

On sait que 0.52 est la valeur qui minimise cette quantité. Pour toute autre valeur, on trouvera plus (estimation au maximum de vraisemblance) :

```
> -2*(52*log(0.54)+48*log(0.46))
[1] 138.6
```

Le terme déviance désigne une variation de la log-vraisemblance. Le modèle « parfait » est celui où la probabilité de rencontrer l'espèce vaut 1 là où on la rencontre et 0 dans le cas contraire. Ceci s'écrit $E(Y_i) = y_i$. La vraisemblance de l'échantillon est alors définie par $P(y_i) = 1$ et $2 * LL(H) = 0$. La variation de vraisemblance entre le modèle parfait et le modèle nul est la déviance résiduelle du modèle nul.

Quand on rajoute l'effet du facteur, la probabilité de présence dans la mesure de rang i vaut :

$$P(y_i) = \frac{1}{1 + \exp(-ax_i - b)}$$

```
> p.vec_1/(1+exp(-0.1352*x+6.5567))
> sum((p.vec-pred0.rep)^2)
[1] 2.157e-08
```

La vraisemblance de l'échantillon dans le nouveau modèle vaut donc :

$$-2LL(p) = -2\sum_{\substack{i=1 \\ y_i=1}}^{n} \log\left(P(y_i)\right) - 2\sum_{\substack{i=1 \\ y_i=0}}^{n} \log\left(1 - P(y_i)\right)$$

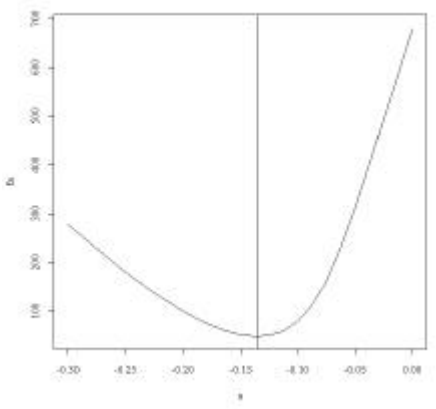
```
> -2*sum(log((y1==1)*p.vec+(y1==0)*(1-p.vec)))
[1] 48.35
```

On retrouve :

```
Residual deviance:  48.352  on 98  degrees of freedom
```

Toute autre valeur des paramètres donnerait plus (estimation au maximum de vraisemblance) :

```
> p.vec_1/(1+exp(-0.18*x+7.2))
> -2*sum(log((y1==1)*p.vec+(y1==0)*(1-p.vec)))
[1] 63.3
> f2
function()
{
        a_seq(-0.3,0,le=25)
        fa_rep(0,25)
        for (i in 1:25) {
                p.vec_1/(1+exp(a[i]*x+6.5567))
                fa[i]_ -2*sum(log((y1==1)*p.vec+(y1==0)*(1-p.vec)))
        }
        plot(a,fa,type="l")
        abline(v=-0.1352)
}
```



On a donc une variation de vraisemblance de 138.5 entre le modèle parfait et le modèle nul, une déviance résiduelle de 48.4 entre le modèle parfait et le modèle sur x et donc par différence une déviance de 90.1 entre le modèle nul et le modèle sur x. Cette quantité suit une loi Chi2 quand le modèle nul est vrai.

Le test de l'effet du facteur s'obtient par :

```
> anova(glm1,test="Chisq")
Analysis of Deviance Table

Model: binomial, link: logit

Response: y1

Terms added sequentially (first to last)


     Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                    99      138.5
x     1     90.1        98       48.4       0.0
```

Bien lire :

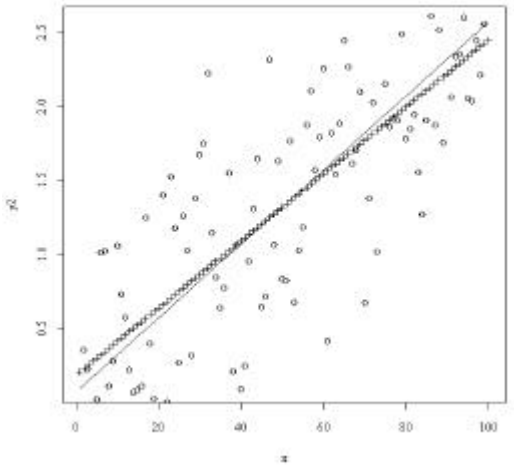
```
     Df  Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                    99      138.5
x     1      90.1        98       48.4       0.0

Df = Degree of freedom = DDL = degré de liberté
Deviance = deviance
Resid. Df = degré de liberté résiduel
Resd. Dev = déviance résiduelle
```

```
P(>|Chi|) = Proba(Khi2 à 1 ddl > 90.1)
```

2.2. Erreur normale

Le modèle linéaire généralisé contient deux éléments fondamentaux. Le premier est le type d'erreur. Dans un modèle linéaire, elle est normale de variance constante, par exemple :

```
> y2_0.025*x+0.075
> plot(x,y2,type = "l")
> y2_rnorm(100,sd=0.5)+y2
> y2
  [1] -0.598145  0.356381  0.224196 -0.353836  0.021856  1.009440
…
 [97]  2.442656  2.213208  2.559723  2.720097
```



Modèle vrai, données simulées, modèle estimé

```
> points(x,y2)
> points(x,predict(lm(y2~x)),pch="+")

> coefficients(lm(y2~x))
(Intercept)           x
    0.18727     0.02264
```

Le modèle linéaire est aussi un modèle linéaire généralisé :

```
> glm2_glm(y2~x,family=gaussian)
> glm2

Call:  glm(formula = y2 ~ x, family = gaussian)

Coefficients:
(Intercept)            x
     0.1873       0.0226

Degrees of Freedom: 99 Total (i.e. Null);  98 Residual
Null Deviance:      70.3
Residual Deviance: 27.6         AIC: 161

> lm2_lm(y2~x)
> lm2

Call:
lm(formula = y2 ~ x)
```

```
Coefficients:
(Intercept)            x
     0.1873       0.0226

> anova(lm2)

Analysis of Variance Table

Response: y2
          Df Sum Sq Mean Sq F value Pr(>F)
x          1   42.7    42.7     152 <2e-16 ***
Residuals 98   27.6     0.3
---
Signif. codes:  0  `***'  0.001  `**'  0.01  `*'  0.05  `.'  0.1  ` '  1

> anova(glm2,test="Chisq")
Analysis of Deviance Table

Model: gaussian, link: identity

Response: y2

Terms added sequentially (first to last)


     Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                    99       70.3
x     1     42.7        98       27.6   6.3e-11
```

Le second est la fonction de lien. Dans le modèle linéaire on cherche directement la liaison sous la forme $y = ax + b$. Dans le modèle linéaire généralisé on cherche à prédire la fonction de lien sous la forme :

$$\log\left(\frac{p}{1-p}\right) = ax + b \Leftrightarrow p = \frac{1}{1+e^{-(ax+b)}}$$

Un modèle linéaire est un modèle linéaire généralisé d'erreur normale et de lien identité.

2.3. Erreur binomiale

Si pour chaque valeur de x, on pouvait faire 5 mesures indépendantes :

```
> y3_rbinom(100,5,p)
> plot(x,y3/5)
> lines(x,p)
> glm3_glm(cbind(y3,5-y3)~x,family=binomial)
```

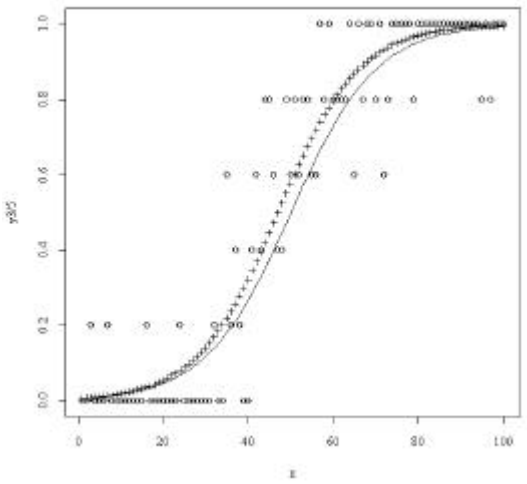
Bien noter la syntax :

```
glm(cbind(nsucces,nechec)~x,family=binomial)
```

```
> points(x,predict(glm3,type="response"),pch="+")
> glm3

Call:  glm(formula = cbind(y3, 5 - y3) ~ x, family = binomial)

Coefficients:
(Intercept)            x
     -4.948        0.105
```

```
Degrees of Freedom: 99 Total (i.e. Null);  98 Residual
Null Deviance:      477
Residual Deviance: 91.1         AIC: 168
```

En chaque point de x, on fait 5 essais. La probabilité de succès est $p(x)$. On obtient un résultat entre 0 à 5, tirage d'une loi binomiale de paramètre 5 et $p(x)$, qui donne des fréquences observées possibles 0, 0.2, 0.4, 0.6, 0.8 et 1. On utilise le même lien mais l'erreur est binomiale.



Modèle vrai, données simulées, modèle estimé

2.4. Erreur de Poisson

Supposons enfin que pour chaque valeur de x, on compte un nombre d'individus. Ce nombre suit une loi de Poisson de paramètre vérifiant $\log(m) = ax + b$. L'erreur a changé, elle est poissonienne. On change aussi le lien :

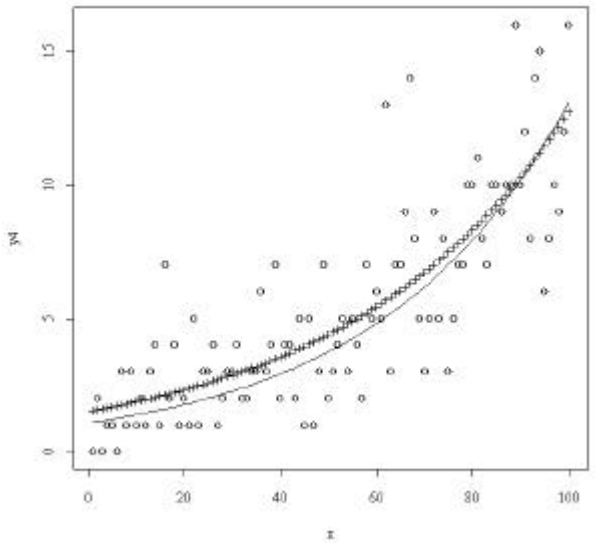
```
> m_exp(0.025*x+0.075)
> y4_rpois(100,m)
> y4
  [1]  0  2  0  1  1  0  3  1  3  1  2  1  3  4  1  7  2  4  1  2  1  5
 [23]  1  3  3  4  1  2  3  3  4  2  2  3  3  6  3  4  7  2  4  4  2  5
 [45]  1  5  1  3  7  2  3  4  5  3  5  4  2  7  5  6  5 13  3  7  7  9
 [67] 14  8  5  3  5  9  5  8  3  5  7  7 10 10 11  8  7 10 10  9 10 10
 [89] 16 10 12  8 14 15  6  8 10  9 12 16
> plot(x,y4)
> lines(x,m)
> glm4_glm(y4~x,family=poisson)
```

- `points(x,predict(glm4,type="response"),pch="+")`

En chaque point de x, le résultat est un entier réalisation d'une loi de Poisson de paramètre $m(x)$.

```
> glm4

Call:  glm(formula = y4 ~ x, family = poisson)
```

```
Coefficients:
(Intercept)            x
     0.4164       0.0213

Degrees of Freedom: 99 Total (i.e. Null);  98 Residual
Null Deviance:      277
Residual Deviance: 96   AIC: 423
```



Modèle vrai, données simulées, modèle estimé

On a obtenu 0.021 et 0.416 pour 0.025 et 0.75. Les résultats sont toujours bons parce que les hypothèses du modèle sont satisfaites et que l'échantillonnage est régulier sur le gradient. R permettra de passer de la régression simple à la régression logistique ou poissonienne.

2.5. Retour sur les déviances

En général, un glm s'écrit $Y \to L(\Theta) \quad E(Y) = g(X)$. Le résultat de l'expérience est une variable aléatoire de loi L qui dépend des paramètres Θ dont la moyenne $E(Y)$ est une fonction des variables de contrôle X. On fait *n* expériences indépendantes qui donnent des observations $(y_1,...,y_n)$ réalisations des variables $Y_1,...,Y_n$.

On compare plusieurs modèles. Ce qu'on a appelé modèle «parfait », qui se dit modèle complet (*full model*) ou modèle saturé (*satured model*), est celui où la moyenne de la variable est définie par l'observation elle-même $E(Y_i) = y_i$. La log-vraisemblance de ce modèle n'est pas nulle sauf pour le cas de Bernouilly.

Dans le cas de Bernouilly :

$$Y \to B(1,p) \quad E(Y) = p = g(X)$$

$E(Y_i) = y_i$ signifie p = 1 si 1 est observé et p = 0 si 0 est observé. La probabilité de l'observation dans le modèle saturé est toujours 1, la logvraisemblance est nulle.

Dans le cas binomial :

$$Y \to B(n,p) \quad E(Y) = np = g(X)$$

$E(Y_i) = y_i$ signifie $p_i = \frac{y_i}{n}$ et la probabilité d'observer l'observation ne vaut pas 1 mais $\binom{n}{y_i}\left(\frac{y_i}{n}\right)^{y_i}\left(1-\frac{y_i}{n}\right)^{n-y_i}$. La logvraisemblance du modèle saturé LL_f n'est pas nulle. On peut la calculer simplement :

```
> llf_sum(log(dbinom(y3,5,y3/5)))
> llf
[1] -36.3
```

Pour le modèle nul $E(Y_i) = p_0$ et cette constante est estimée au maximum de vraisemblance par la fréquence totale. Vérification :

```
> glm(cbind(y3,5-y3)~1,family=binomial)

Call:  glm(formula = cbind(y3, 5 - y3) ~ 1, family = binomial)

Coefficients:
(Intercept)
      0.136

Degrees of Freedom: 99 Total (i.e. Null);  99 Residual
Null Deviance:      477
Residual Deviance: 477  AIC: 551
> 1/(1+exp(-0.136))
[1] 0.534
> sum(y3/500)
[1] 0.534
```

La logvraisemblance de modèle nul LL_0 peut être calculée simplement :

```
> ll0_sum(log(dbinom(y3,5,0.534)))
> ll0
[1] -274.6
```

Par définition, la déviance résiduelle du modèle nul est le double de la variation de vraisemblance entre le modèle nul et le modèle saturé soit :

$$D_0 = -2\log\left(\frac{L_o}{L_f}\right) = 2\left(LL_f - LL_0\right)$$

soit :

```
> 2*(llf-ll0)
[1] 476.5             Null Deviance:      477
```

Quand on introduit la variable x, les paramètres sont estimés au maximum de vraisemblance ce qui permet de calculer la nouvelle logvraisemblance LL_x

```
> llx_sum(log(dbinom(y3,5,predict(glm3,type="response"))))
> llx
```

```
[1] -81.84
```

Par définition, la déviance résiduelle du modèle x est le double de la variation de vraisemblance entre le modèle x et le modèle saturé soit :

$$D_x = -2\log\left(\frac{L_x}{L_f}\right) = 2\left(LL_f - LL_x\right)$$

```
> 2*(llf-llx)
[1] 91.1
```

par différence on a l'effet du facteur :

$$D_0 - D_x = -2\log\left(\frac{L_o}{L_x}\right) = 2\left(LL_x - LL_0\right)$$

```
> anova(glm3,test="Chisq")
Analysis of Deviance Table

Model: binomial, link: logit

Response: cbind(y3, 5 - y3)

Terms added sequentially (first to last)


     Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                    99        477
x     1      385        98         91         0
```

Dans le cas poissonien :

$$Y \to P(m) \quad E(Y) = m = g(X)$$

$E(Y_i) = y_i$ signifie $m_i = y_i$ et la probabilité d'observer l'observation ne vaut pas 1 mais $\exp(-y_i)\frac{y_i^{y_i}}{y_i!}$. La logvraisemblance du modèle saturé LL_f n'est pas nulle. On peut la calculer simplement :

```
> llf_sum(log(dpois(y4,y4)))
> llf
[1] -161.4
```

Pour le modèle nul $E(Y_i) = m$ et cette constante est estimée au maximum de vraisemblance par la moyenne. Vérification :

```
> > glm(y4~1,family=poisson)

Call:  glm(formula = y4 ~ 1, family = poisson)

Coefficients:
(Intercept)
       1.67

Degrees of Freedom: 99 Total (i.e. Null);  99 Residual
Null Deviance:      277
Residual Deviance: 277  AIC: 602
> mean(y4)
[1] 5.33
> exp(1.67)
[1] 5.312
> exp(coefficients(glm(y4~1,family=poisson))[1])
```

```
(Intercept)
       5.33                Non mais !
```

La logvraisemblance de modèle nul LL_0 peut être calculée simplement :

```
> ll0_sum(log(dpois(y4,y4)))
> ll0
[1] -161.4
```

Par définition, la déviance résiduelle du modèle nul est le double de la variation de vraisemblance entre le modèle nul et le modèle saturé soit :

$$D_0 = -2\log\left(\frac{L_o}{L_f}\right) = 2\left(LL_f - LL_0\right)$$

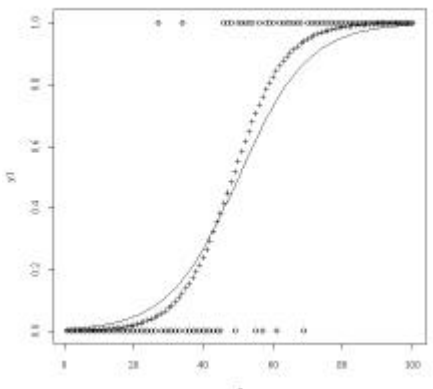
soit :

```
> ll0_sum(log(dpois(y4,mean(y4))))
> ll0
[1] -299.8
> 2*(llf-ll0)
[1] 276.9            Null Deviance:      277
```

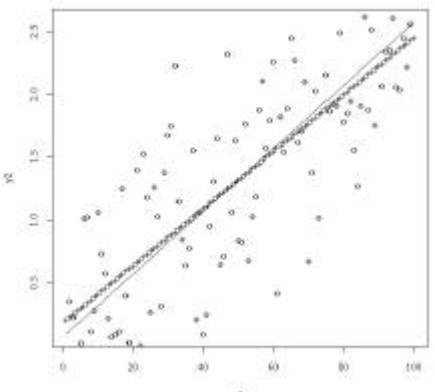
$$D_x = -2\log\left(\frac{L_x}{L_f}\right) = 2\left(LL_f - LL_x\right) \quad D_0 - D_x = -2\log\left(\frac{L_o}{L_x}\right) = 2\left(LL_x - LL_0\right)$$

```
> llx_sum(log(dpois(y4,predict(glm4,type="response"))))
> llx
[1] -209.4
> 2*(llf-llx)
[1] 96.01
> anova(glm4,test="Chisq")

     Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                    99        277
x     1      181        98         96         0
```

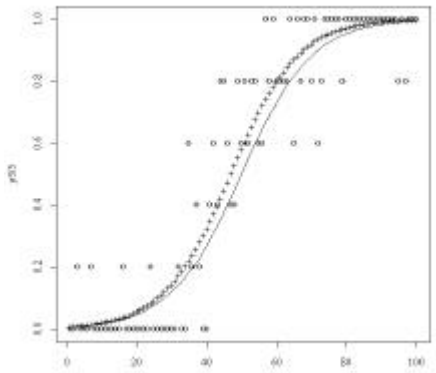
2.6. Pour mémoire

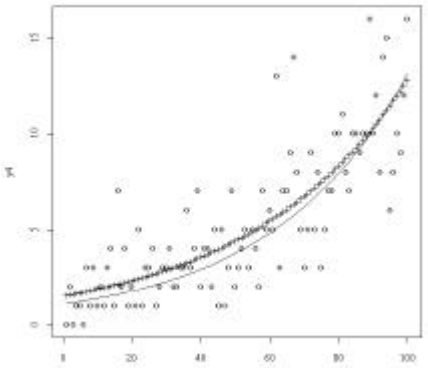


A



B





C D

A Données en presence-absence - Erreur de Bernouilly - Lien logit
B Données en abondance - Erreur normale - Lien identité
C Données en fréquence – Erreur binomiale – Lien logit
D Données en dénombrements – Erreur poissonienne – Lien logarithme

La fonction **glm** peut servir ! Il y a d'autres liens possibles et d'autres erreurs possibles. Voir help(family) pour en avoir une idée.

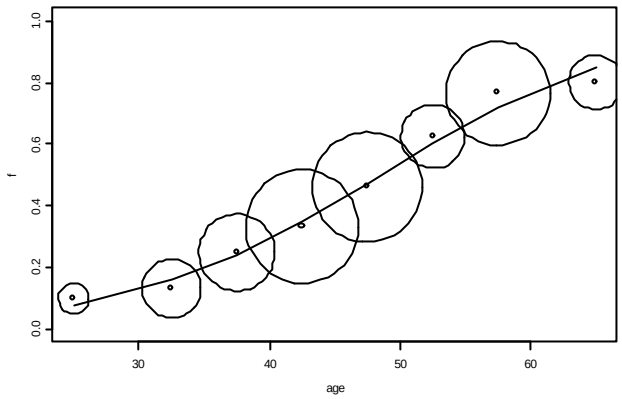
3. Le fonctionnement de la régression logistique

Age, Tabac et probabilité de succès !

```
> age
[1] 25.0 32.5 37.5 42.5 47.5 52.5 57.5 65.0
> n
[1] 100 150 120 150 130  80 170 100
> Y
[1]  10  20  30  50  60  50 130  80
```

Y est le nombre de succès, n est le nombre d'essais et age une variable explicative quantitative. L'approximation numérique de la régression logistique est :

```
> f<-Y/n
> f
[1] 0.1000 0.1333 0.2500 0.3333 0.4615 0.6250 0.7647 0.8000
> g<-log(f/(1-f)) # Transformation des données
> w<-n*f*(1-f) # Pondération des données
> r<-predict(lm(g~age,weights=w)) # Régression pondérée
> p<-exp(r)/(1+exp(r)) # Transformation inverse
> p
       1      2      3      4      5      6      7      8
 0.08007 0.1594 0.2416 0.3486 0.4735 0.6017 0.7174 0.8468

> plot(age,f,ylim=c(0,1))
> lines(age,p)
> symbols(age,f,circles=w,add=T)
```



On peut itérer cette régression pondérée :

```
> p
       1      2      3      4      5      6      7      8
 0.08007 0.1594 0.2416 0.3486 0.4735 0.6017 0.7174 0.8468
> w<-n*p*(1-p)
> gu<-r+(f-p)/p/(1-p)
> r<-predict(lm(gu~age,weights=w))
> r
      1      2      3       4       5      6      7     8
 -2.465 -1.678 -1.152 -0.6274 -0.1023 0.4227 0.9478 1.735
> p<-exp(r)/(1+exp(r))
> p
       1      2    3      4      5      6      7      8
 0.07834 0.1574 0.24 0.3481 0.4744 0.6041 0.7207 0.8501
> w<-n*p*(1-p)
> gu<-r+(f-p)/p/(1-p)
> r<-predict(lm(gu~age,weights=w))
> r
      1      2      3       4       5      6      7     8
 -2.465 -1.678 -1.153 -0.6274 -0.1023 0.4228 0.9479 1.736
> p<-exp(r)/(1+exp(r))
> p
       1      2    3      4      5      6      7      8
 0.07833 0.1574 0.24 0.3481 0.4744 0.6041 0.7207 0.8501
> w<-n*p*(1-p)
> gu<-r+(f-p)/p/(1-p)
> r<-predict(lm(gu~age,weights=w))
> r
      1      2      3       4       5      6      7     8
 -2.465 -1.678 -1.153 -0.6274 -0.1023 0.4228 0.9479 1.736
> p<-exp(r)/(1+exp(r))
> p
       1      2    3      4      5      6      7      8
 0.07833 0.1574 0.24 0.3481 0.4744 0.6041 0.7207 0.8501
```

On peut montrer qu'on obtient ainsi les estimations au maximum de vraisemblance (cours de J. Estève p. 26)

```
> coefficients(lm(gu~age,weights=w))
 (Intercept)   age
      -5.091 0.105
```

```
> glm(cbind(Y,n-Y)~age,family=binomial)

Call:  glm(formula = cbind(Y, n - Y) ~ age, family = binomial)

Coefficients:
(Intercept)          age
     -5.091        0.105
```

```
Degrees of Freedom: 7 Total (i.e. Null);  6 Residual
Null Deviance:      287
Residual Deviance: 5.24         AIC: 48.3
```

4. Modéliser une présence-absence

4.1. Le problème

Installer le data.frame gardonmi :

```
> gardonmi[1:10,]
      S     V     G     A gardon
 1 Nord -0.49  0.31 -1.26      0
 2 Nord -0.56  0.79 -1.65      0
 3 Nord -0.59 -0.17 -0.22      1
 4 Nord -1.30 -1.26 -1.17      0
 5 Nord  0.30  0.36  0.15      1
 6 Nord -0.65  0.77  0.06      1
 7 Nord -0.03 -1.38  0.18      0
 8 Nord -1.12  1.76  0.17      1
 9 Nord -0.22 -1.21  0.40      0
10 Nord -0.52  0.94  0.17      1
> dim(gardonmi)
[1] 645   5
```

Dans 645 stations de référence, le Conseil Supérieur de la Pêche (CSP) a enregistré la présence ou l'absence du Gardon (variable gardon en 0-1)[1]. Chaque station appartient à un bassin (variable S) :

```
> levels(gardonmi$S)
[1] "Atla" "Garo" "Loir" "Manc" "Medi" "Nord" "Rhon" "Sein"
```

On connaît pour chaque station un indice caractérisant les conditions hydrauliques locales basé sur la vitesse du courant, la pente et la largeur (variable V, variable normalisée), la position de la station dans le gradient Amont-Aval basée sur la distance à la source et la surface du bassin draîné (variable G, normalisée) et l'altitude (variable A, transformée et normalisée). Attacher le tableau :

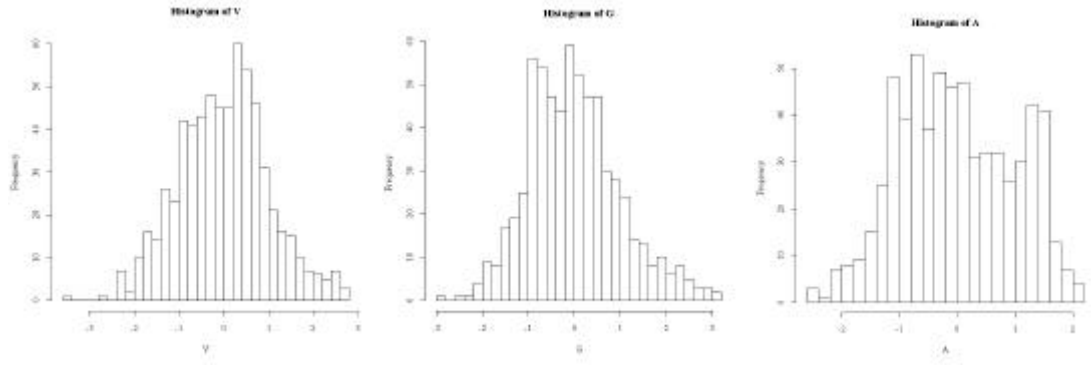
```
> search()
 [1] "D:\\Data\\Dea3\\_Data"
 [2] "d:\\asplus\\splus\\_Functio"
...
> attach(gardonmi,pos=1)
> search()
 [1] "gardonmi"
 [2] "D:\\Data\\Dea3\\_Data"
 [3] "d:\\asplus\\splus\\_Functio"
...
```

Les distributions des variables explicatives sont convenables :

```
> hist(V,nclass=25)
> hist(G,nclass=25)
> hist(A,nclass=25)
```

[1] Données extraites du Programme National « Indice Poisson ». GIP Hydrosystèmes, CSP, Agences de Bassin. Mise au point d'un indice Poisson applicable sur le territoire natrional : Convention n° 1302 Conseil Supérieur de la pêche / Agence de l'eau Adour-Garonne. Août 1996 - Décembre 2000. Responsable scientifique : T. Oberdorff.

```
>
```

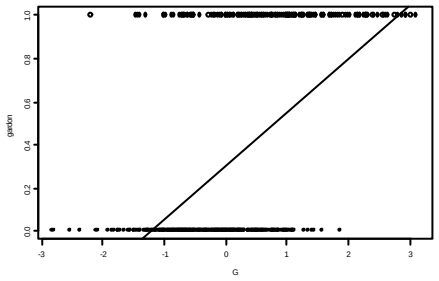


```
> summary(S)
 Atla Garo Loir Manc Medi Nord Rhon Sein
   56  103   84   74   69   42  102  115
```

La particularité du problème «modéliser la présence du Gardon par les variables environnementales» tient à la variable à expliquer (0-1) dont les valeurs sont des réalisations d'un événement qui avait une certaine probabilité de survenir. On ne modélise donc pas le résultat mais la probabilité de ce résultat, de même qu'on ne modélise pas une valeur observée mais la moyenne des valeurs observées dans les mêmes conditions :

$$X = E(X) + erreur \quad et \quad E(X) = f(J)$$

```
> plot(G,gardon)
> abline(lm(gardon~G)
```



Commentaires

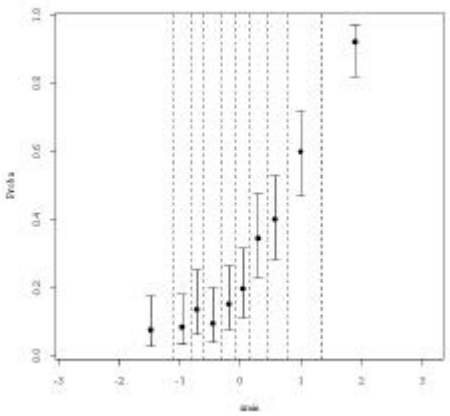
Ce dont on a besoin s'exprime clairement par la fonction suivante dont on détaillera les constituants :

```
function (xobs , yobs , ncla = 10)
{
    if (is.numeric(xobs) == F) return("Numeric data need")
     # Le gradient est découpé en ncla classes
    q0 <- quantile(xobs, probs = seq(0, 1, 1/ncla), na.rm = F)
    c.cla <- quantile(xobs, probs = seq(0 + 1/2/ncla, 1 - 1/2/ncla, 1/ncla),
na.rm = F)
    xmin <- min(xobs) ; xmax <- max(xobs)
     # On calcule les fréquences par classe
    q1 <- cut(xobs, q0, include.lowest = T)
    t0 <- table(q1, yobs)
    t0[, 1] <- (t0[, 1] + t0[, 2])
    freq <- t0[, 2]/t0[, 1]
     # On calcule les intervalles de confiance des fréquences par classes
    basbar <- rep(0, ncla) ; haubar <- rep(0, ncla)
    for (i in 1:ncla) {
        succes <- t0[i, 2] ;  essai <- t0[i, 1]
```

```
        if (essai > 10) {
            a0 <- prop.test(succes, essai)$conf.int
            basbar[i] <- a0[1] ; haubar[i] <- a0[2]
            if (a0[1] > freq[i]) basbar[i] <- NA
            if (a0[2] < freq[i]) haubar[i] <- NA
        } else {
            basbar[i] <- NA ; haubar[i] <- NA
        }
    }
      # On trace le cadre
    ymin <- min(basbar, na.rm = T)
    ymax <- max(haubar, na.rm = T)
     plot(xmin, 0, ylim = c(ymin, ymax), xlim = c(xmin, xmax), ylab = "Proba",
type = "n")
      # On trace les fréquences par classes
   points(c.cla, freq, pch = 16)
   for (i in 1:ncla) {
        if (!is.na(basbar[i])) {
      # On trace les intervalles de confiance
           size.bar <- par("cxy")[1]/2
            segments(c.cla[i], basbar[i] , c.cla[i], haubar[i] )
            segments(c.cla[i]  -  size.bar,  haubar[i]  ,  c.cla[i]  +  size.bar,
haubar[i] )
            segments(c.cla[i]  -  size.bar,  basbar[i]  ,  c.cla[i]  +  size.bar,
basbar[i] )
        }
      # On trace les séparation de classes
        if (i > 1) abline(v = q0[i], lty = 2)
    }
}
```

Pour se servir de la fonction récupérer le fichier plot.freq.grad.R, le mettre dans le dossier de travail et utiliser :

```
> plot.freq.grad_dget("plot.freq.grad.R")
> plot.freq.grad(gardonmi$G,gardonmi$gardon)
```



Ce dont on a besoin

Si la probabilité de rencontrer du Gardon dans une station de milieu J s'écrit :

$$P(X=1) = E(X) = f(J)$$

on se trompe toujours fortement dans l'observation ! On demande à estimer cette probabilité de manière qu'en moyenne pour les stations de probabilité 0.15 on ait du Gardon dans 15% des cas. Autour de la prévision (probabilité), l'observation (oui/non) réalise une erreur très particulière dite de type binomiale (cas particulier de Bernouilly, n = 1). La fonction d'erreur

est la première généralisation du modèle linéaire (erreur non gaussienne). La seconde fait que cette probabilité (comprise entre 0 et 1) ne peut être une fonction linéaire des paramètres). On ne prédit pas linéairement *p* mais une fonction inversible de *p*. Le modèle s'écrit :

$$g\big(P(X=1)\big) = f_{lin}(J)$$
$$P(X=1) = g^{-1}\big(f_{lin}(J)\big)$$

g, dite fonction de lien est la seconde généralisation du modèle linéaire. On parle alors de modèles linéaires généralisés (**glm**). On peut utiliser le lien logit qui par inversion renvoie à la fonction logistique :

$$g(p) = \log\left(\frac{p}{1-p}\right) = f(J) \Leftrightarrow p = \frac{1}{1+e^{-f(J)}}$$

Dans R, un seul paramètre dans glm suffit à se mettre dans cette situation.

4.2. Un exemple

glm

DESCRIPTION

Produces an object of class "glm" which is a generalized linear fit of the data.

USAGE

```
glm(formula, family = gaussian, data = <<see below>>,
    weights = <<see below>>, subset = <<see below>>, na.action = na.fail,
    start = <<see below>>, control, trace = F, model = F, x = F, y = T,
    contrasts = NULL, qr = F, ...)
```

REQUIRED ARGUMENTS

formula a formula expression as for other regression models, of the form response ~ predictors. See the documentation of lm and formula for details.

OPTIONAL ARGUMENTS

family a family object - a list of functions and expressions for defining the link and variance functions, initialization and iterative weights. Families supported are gaussian, binomial, poisson, Gamma, inverse.gaussian and quasi. Functions like binomial produce a family object, but can be given without the parentheses. Family functions can take arguments, as in binomial(link=probit).
...

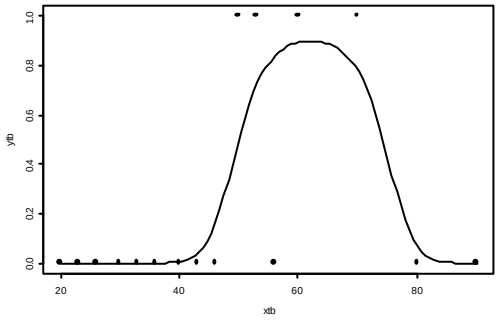
Commençons par un exemple **2** (p. 5) :

xtb est un gradient, ytb donne la présence d'une espèce :

```
> xtb
 [1] 20 23 26 30 33 36 40 43 46 50 53 56 60 70 80 90
> ytb
```

2 Ter Braak, C.J.F. & Looman, C.W.N. (1986) Weighted averaging, logistic regression and the Gaussian response model. *Vegetatio* : 65, 3-11.

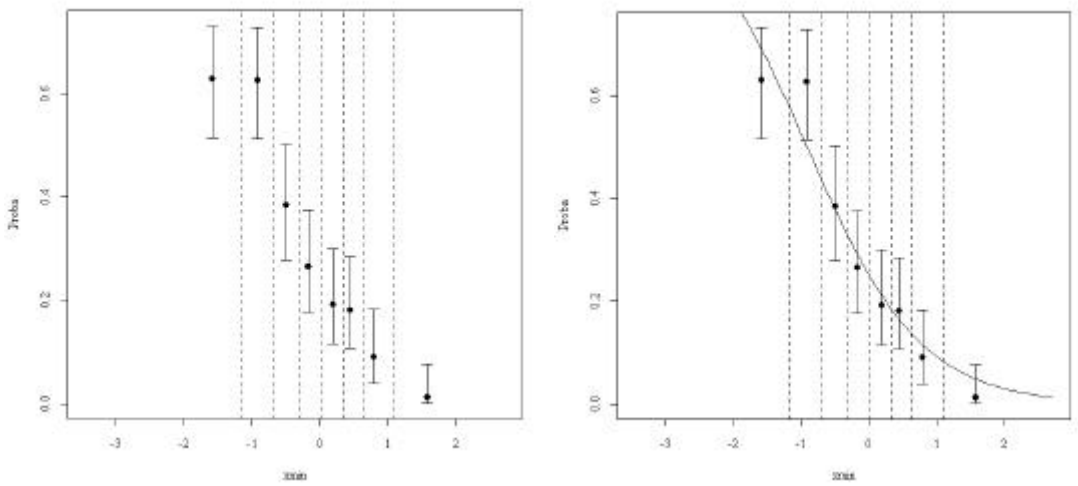
```
 [1] 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0

> mod1<-glm(ytb~xtb+I(xtb^2), family=binomial)
> mod1
Call:
glm(formula = ytb ~ xtb + I(xtb^2), family = binomial)

Coefficients:
 (Intercept)   xtb I(xtb^2)
      -55.45 1.855 -0.01492

Degrees of Freedom: 16 Total; 13 Residual
Residual Deviance: 6.96
> class(mod1)
[1] "glm" "lm"

> summary(mod1)

Call: glm(formula = ytb ~ xtb + xtb^2, family = binomial)
Deviance Residuals:
    Min      1Q    Median     3Q   Max
 -1.903 -0.1495 -0.004676 0.1186 1.181

Coefficients:
                Value Std. Error t value
(Intercept) -55.45496  34.291015  -1.617
        xtb   1.85500   1.148155   1.616
   I(xtb^2)  -0.01492   0.009359  -1.594

...

> xnou<-seq(min(xtb),max(xtb),len=100)
> ynou<-predict(mod1,data.frame(xtb=xnou),type="response")
> plot(xtb,ytb,pch=16)
> lines(xnou,ynou)
```



4.3. Courbes de réponse

La probabilité de présence du Gardon dépend-elle de la vitesse ?

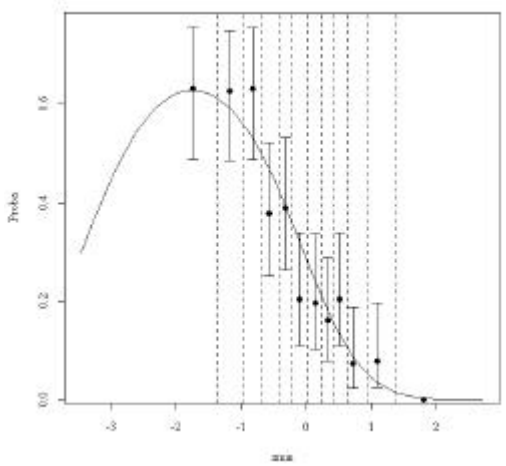
```
> attach(gardonmi)
> plot.freq.grad(V,gardon,8)
```



```
> glm1<-glm(gardon~V, family=binomial)
> glm1

Call:  glm(formula = gardon ~ V, family = binomial)

Coefficients:
(Intercept)            V
      -1.09        -1.20

Degrees of Freedom: 644 Total (i.e. Null);  643 Residual
Null Deviance:      787
Residual Deviance: 652  AIC: 656

> anova(glm1,test="Chisq")
Analysis of Deviance Table

Model: binomial, link: logit

Response: gardon

Terms added sequentially (first to last)


      Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                    644        787
V      1      135       643        652         0

> xnou<-seq(min(V),max(V),len=100)
> ynou<-predict(glm1,data.frame(V=xnou),type="response")
> lines(xnou,ynou)
```

Doit-on ajouter un terme carré ?

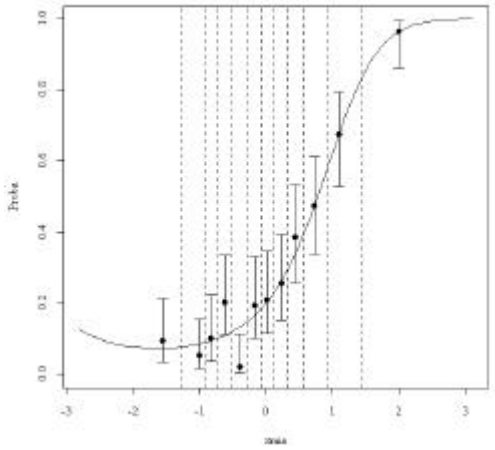
```
> glm2<-glm(gardon~V+I(V^2), family=binomial)
> anova(glm2,test="Chisq")

Analysis of Deviance Table

Binomial model
Response: gardon
Terms added sequentially (first to last)
       Df Deviance Resid. Df Resid. Dev    Pr(Chi)
  NULL                  644      787.2
     V  1    135.1       643      652.0 0.00000000
I(V^2)  1     16.8       642      635.3 0.00004251
```



```
>  plot.freq.grad(V,gardon,12)
> lines(xnou,predict(glm2,data.frame(V=xnou),type="response"))
```

La probabilité de présence du Gardon dépend-elle aussi du gradient Amont-Aval ?

```
> glm3<-glm(gardon~G+I(G^2),family=binomial)
> plot.freq.grad(G,gardon,12)
> xnou<-seq(min(G),max(G),len=100)
> lines(xnou,predict(glm3,data.frame(G=xnou),type="response"))
```



Noter l'usage de I() :

While formulae usually involve just variable and factor names, they can also involve arithmetic expressions. The formula `log(y) ~ a + log(x)' is quite legal. When such arithmetic expressions involve operators which are also used symbolically in model formulae, there can be confusion between arithmetic and symbolic operator use.

To avoid this confusion, the function `I()' can be used to bracket those portions of a model formula where the operators are used in their arithmetic sense. For example, in the formula `y ~ a + I(b+c)', the term `b+c' is to be interpreted as the sum of `b' and `c'.

4.4. Surfaces de réponse

Écrire le modèle :

```
> glm3<-glm(gardon~V+I(V^2)+G+I(G^2), family=binomial)
> anova(glm3,test="Chisq")
Analysis of Deviance Table

Model: binomial, link: logit

Response: gardon

Terms added sequentially (first to last)


        Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                      644        787
V        1      135       643        652         0
I(V^2)   1       17       642        635   4.3e-05
G        1      128       641        508         0
I(G^2)   1       11       640        496   8.1e-04
```

Mettre en place une grille de valeurs des prédicteurs :

```
> newV<-seq(min(V),max(V),le=20)
> newG<-seq(min(G),max(G),le=20)
> newdata<-expand.grid(V=newV, G=newG)
> gardon.surf<-predict(glm3,newdata,type="response")
> newV[1:10]
 [1] -3.4600 -3.1342 -2.8084 -2.4826 -2.1568 -1.8311 -1.5053 -1.1795
 [9] -0.8537 -0.5279
> newG[1:10]
 [1] -2.810000 -2.498421 -2.186842 -1.875263 -1.563684 -1.252105
 [7] -0.940526 -0.628947 -0.317368 -0.005789
> newdata[1:5,]
       V     G
1 -3.460 -2.81
2 -3.134 -2.81
3 -2.808 -2.81
4 -2.483 -2.81
5 -2.157 -2.81
```

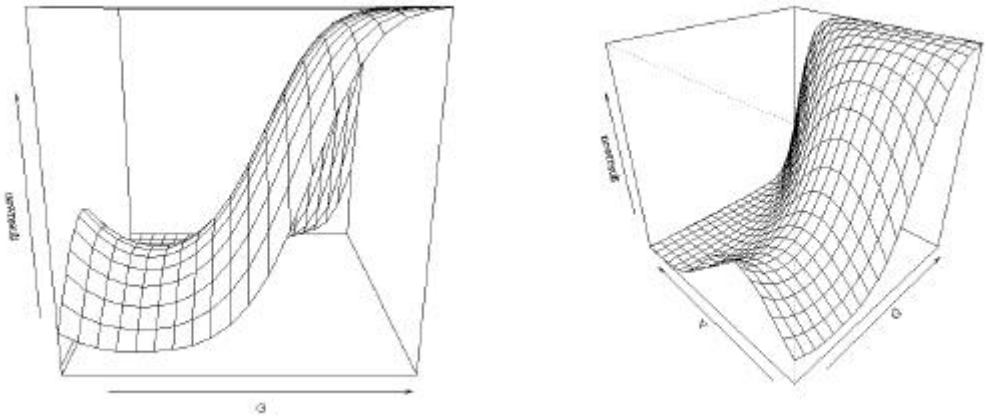
Estimer le modèle pour les valeurs de la grille :

```
> newresult<-predict.glm(glm3,newdata,type="response")
> newresult[1:10]
      1      2      3      4      5      6      7      8      9
 0.1339 0.1942 0.2562 0.3111 0.3519 0.3745 0.3771 0.3596 0.3232
     10
 0.2714
```

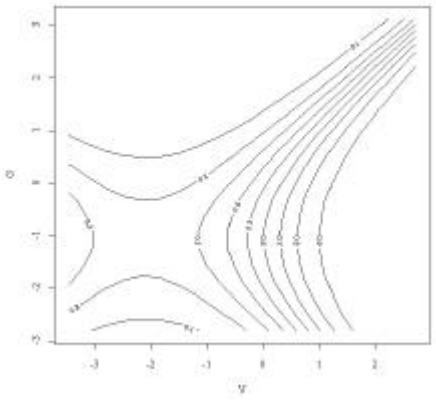
persp

```
persp                  package:base                  R Documentation

Perspective Plots

Description:

     This function draws perspective plots of surfaces over the x-y
     plane.

Usage:

     persp(x = seq(0, 1, len = nrow(z)), y = seq(0, 1, len = ncol(z)), z,
           xlim = range(x), ylim = range(y), zlim = range(z, na.rm = TRUE),
           xlab = NULL, ylab = NULL, zlab = NULL,
           theta = 0, phi = 15, r = sqrt(3), d = 1,
           scale = TRUE, expand = 1,
           col = NULL, border = NULL, ltheta = -135, lphi = 0,
```

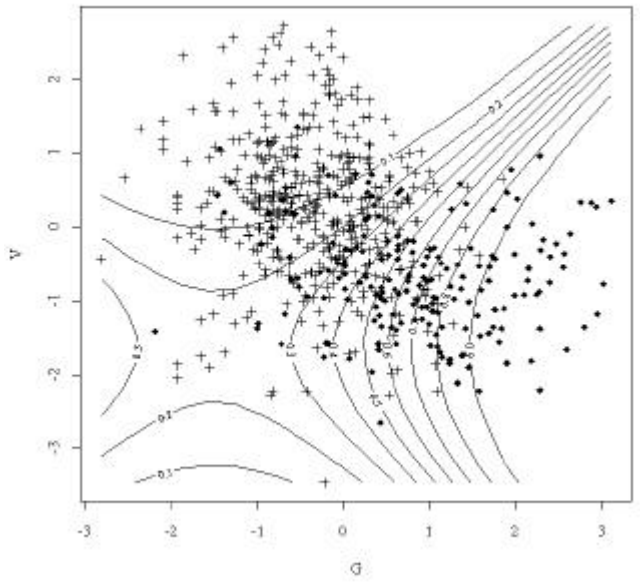
```
          shade = NA, box = TRUE, axes = TRUE, nticks = 5, ticktype = "simple",
          ...)

Arguments:

    x, y: locations of grid lines at which the values in `z' are
          measured.  These must be in ascending order.  By default,
          equally spaced values from 0 to 1 are used.  If `x' is a
          `list', its components `x$x' and `x$y' are used for `x' and
          `y', respectively.

       z: a matrix containing the values to be plotted (`NA's are
          allowed).  Note that `x' can be used instead of `z' for
          convenience.

…

theta, phi: angles defining the viewing direction. `theta' gives the
          azimuthal direction and `phi' the colatitude.

       r: the distance of the eyepoint from the centre of the plotting
          box.

       d: a value which can be used to vary the strength of the
          perspective transformation.  Values of `d' greater than 1
          will lessen the perspective effect and values less and 1 will
          exaggerate it.

   scale: before viewing the x, y and z coordinates of the points
          defining the surface are transformed to the interval [0,1].
          If `scale' is `TRUE' the x, y and z coordinates are
          transformed separately.  If `scale' is `FALSE' the
          coordinates are scaled so that aspect ratios are retained.
          This is useful for rendering things like DEM information.

  expand: a expansion factor applied to the `z' coordinates. Often used
          with `0 < expand < 1' to shrink the plotting box in the `z'
          direction.

…

> newresult<-matrix(newresult,nrow=20,ncol=20,byrow=T)
> newresult[1:5,1:5]
        [,1]   [,2]   [,3]   [,4]   [,5]
[1,] 0.13390 0.1942 0.2562 0.3111 0.3519
[2,] 0.10496 0.1545 0.2072 0.2552 0.2917
[3,] 0.08746 0.1300 0.1760 0.2187 0.2518
[4,] 0.07782 0.1162 0.1583 0.1978 0.2286
[5,] 0.07412 0.1109 0.1514 0.1895 0.2195

> persp(newG,newV,newresult,xlab="G",ylab="V")
> persp(newG,newV,newresult,xlab="G",ylab="V",theta=-45,phi=30)
```



```
> contour(newV,newG,newresult,xlab="V",ylab="G",nlevels=10)
```

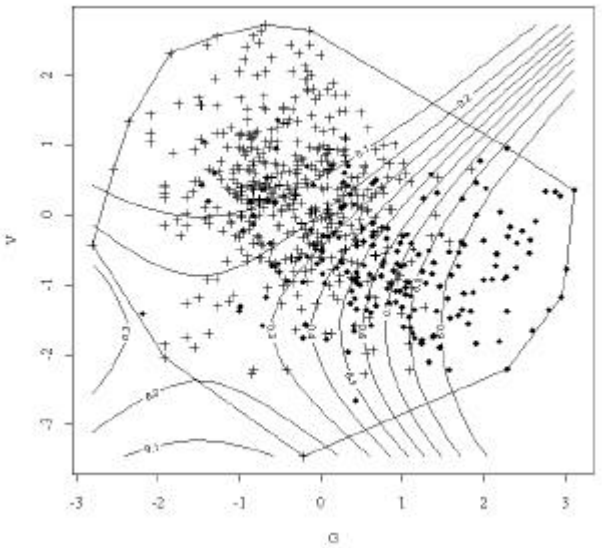


```
> contour(newG,newV,newresult,xlab="G",ylab="V")
> points(G[gardon==0],V[gardon==0],pch=3)
> points(G[gardon==1],V[gardon==1],pch=18)
```



Convex hull

```
> GV.chull<-chull(G,V)
> GV.chull
 [1] 424 506 570 492 491 455 571 553 547 129 127 339  15 428
> paste(G[GV.chull],V[GV.chull])
 [1] "-0.21 -3.46" "-1.92 -2.04" "-2.81 -0.42" "-2.54 0.66"
 [5] "-2.36 1.34"  "-1.85 2.32"  "-1.27 2.57"  "-0.69 2.73"
 [9] "-0.13 2.65"  "2.29 0.96"   "3.11 0.35"   "3.02 -0.78"
[13] "2.95 -1.18"  "2.29 -2.21"
> polygon(G[GV.chull],V[GV.chull],density=0)
```



Commentaires ?

Et l'altitude ?

```
> glm4<-glm(gardon~V+I(V^2)+G+I(G^2)+A+I(A^2), family=binomial)
> anova(glm4,test="Chisq")
Analysis of Deviance Table

Model: binomial, link: logit

Response: gardon

Terms added sequentially (first to last)


        Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                      644        787
V        1      135       643        652         0
I(V^2)   1       17       642        635   4.3e-05
G        1      128       641        508         0
I(G^2)   1       11       640        496   8.1e-04
A        1       21       639        475   3.7e-06
I(A^2)   1       12       638        463   4.4e-04
```

Et le bassin ?

```
> glm5<-glm(gardon~V+I(V^2)+G+I(G^2)+A+I(A^2)+S, family=binomial)
> anova(glm4,glm5,test="Chisq")

Analysis of Deviance Table
Response: gardon
                                          Resid. Df Resid. Dev  Df Deviance
V + I(V^2) + G + I(G^2) + A + I(A^2)            638        463
V + I(V^2) + G + I(G^2) + A + I(A^2) + S        631        440   7       23
                                          P(>|Chi|)
```

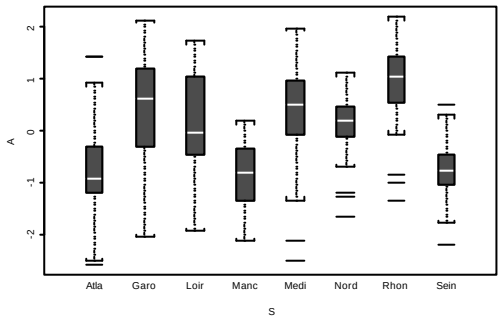
```
V + I(V^2) + G + I(G^2) + A + I(A^2)
V + I(V^2) + G + I(G^2) + A + I(A^2) + S    0.0017
```

La difficulté :

```
> glm6<-glm(gardon~V+I(V^2)+G+I(G^2)+S+A+I(A^2), family=binomial)
> anova(glm6,test="Chisq")
Analysis of Deviance Table

Model: binomial, link: logit

Response: gardon

Terms added sequentially (first to last)


        Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                      644        787
V        1      135       643        652         0
I(V^2)   1       17       642        635   4.3e-05
G        1      128       641        508         0
I(G^2)   1       11       640        496   8.1e-04
S        7       12       633        484   8.7e-02
A        1       36       632        448   2.1e-09
I(A^2)   1        8       631        440   3.7e-03
```

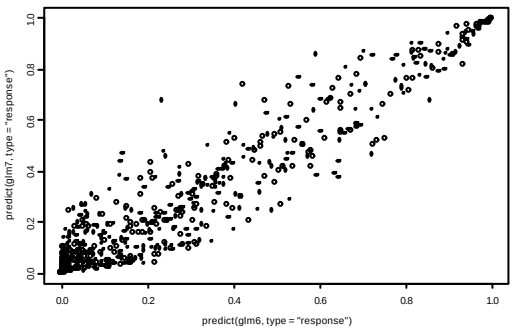
S est un effet significatif derrière A mais pas devant. Les deux variables sont liées :

```
> plot(S,A)
```

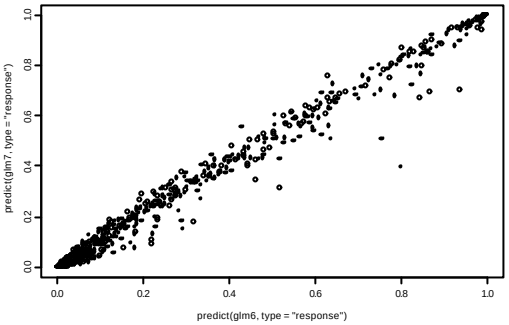


Mais introduire l'une ou l'autre des variables ne donnent pas les mêmes prédictions :

```
> glm6<-glm(gardon~V+I(V^2)+G+I(G^2)+A+I(A^2), family=binomial)
> glm7<-glm(gardon~V+I(V^2)+G+I(G^2)+S, family=binomial)
> plot(predict(glm6,type="response"),predict(glm7,type="response"))
```




```
> glm6<-glm(gardon~V+I(V^2)+G+I(G^2)+A+S, family=binomial)
> glm7<-glm(gardon~V+I(V^2)+G+I(G^2)+A+I(A^2)+S, family=binomial)
> plot(predict(glm6,type="response"),predict(glm7,type="response"))
```



Le terme A^2 a donc beaucoup moins d'effet que le terme S.

Voir aussi **step** (choix d'un modèle pas à pas), **add1** (ajouter une variable dans un modèle), **drop1** (enlever une variable dans un modèle).

Et les interactions ? On dit que la statistique est un art...

5. Modèle poissonien

On reprend l'exemple ecrin de la fiche 2.

```
> ecrin[1:5,]
   STA SEM HEU RIC
1    3   2   1   5
2    3   2   2   3
3    3   3   1   5
4    3   3   2   3
5    3   4   1   4

> ecrin_read.table("ecrin.txt",h=T)
> ric_ecrin$RIC
> heu_as.factor(ecrin$HEU)
> levels(heu)
[1] "1" "2"
> levels(heu)_c("matin","soir")

> sem_as.factor(ecrin$SEM)
> sta_as.factor(ecrin$STA)

> lm1_lm(ric~heu+sem+sta+sem:sta)
Error: heap memory (6144 Kb) exhausted [needed 7489 Kb more]
       See "help(Memory)" on how to increase the heap size.
```

Memory **package:base** **R Documentation**

Memory Available for Data Storage

Description:

Use command line options to set the memory available for R.

Usage:

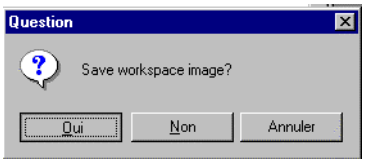
Rgui --vsize=v --nsize=n

…

```
      The `--nsize' option can be used to specify the number of cons
      cells (each occupying 20 bytes on a 32-bit machine) which R is to
      use (the default is 250000), and the `--vsize' option to specify
      the size of the vector heap in bytes (the default is 6 MB).  Both
      options must be integers or integers ending with `M', `K', or `k'
      meaning Mega (2^{20} = 1048576), (computer) Kilo (2^{10} = 1024),
       or regular kilo (1000). (The minimum allowed values are 200000
and
      2M.)
```

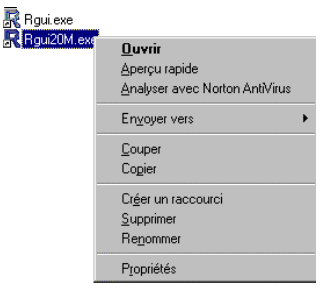
5.1. Augmenter la mémoire

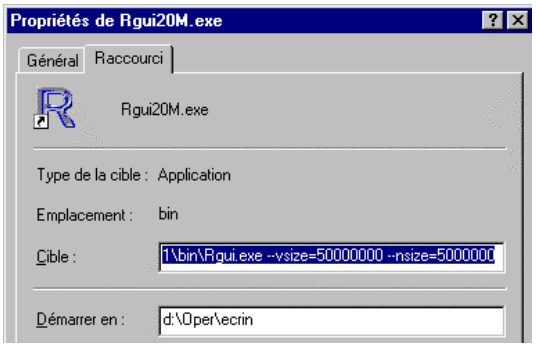
Quitter le programme en sauvant le dossier de travail :



Dupliquer le raccourci de départ, éditer ses propriétés et ajouter la chaîne :

```
E:\rw1001\bin\Rgui.exe --vsize=50000000 --nsize=5000000
```

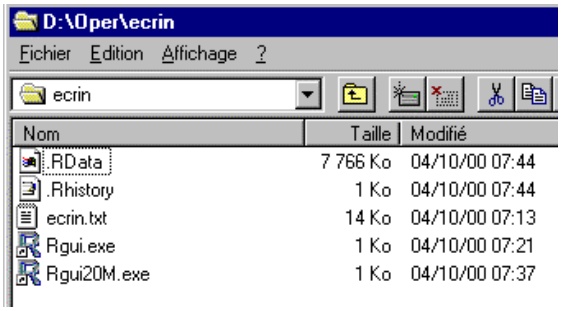


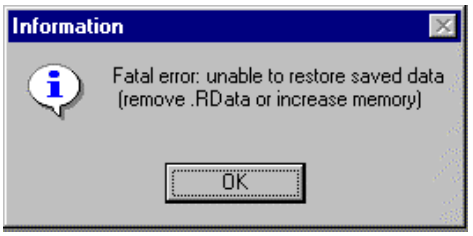


Relancer le programme à l'aide du nouveau raccourci :

```
> lm1_lm(ric~heu+sem+sta+sem:sta)
```

Quitter en sauvegardant le dossier de travail. Attention : il sera impossible de redémarrer avec l'ancien :



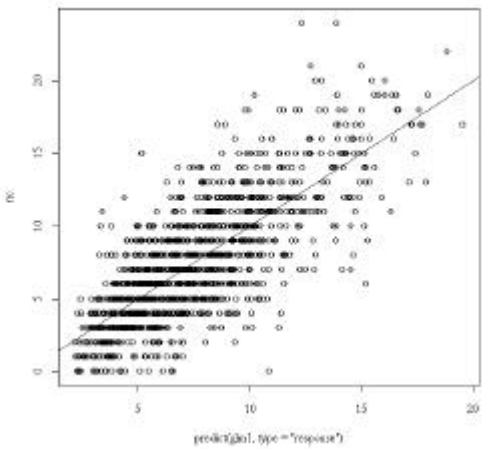


Relancer avec le nouveau.

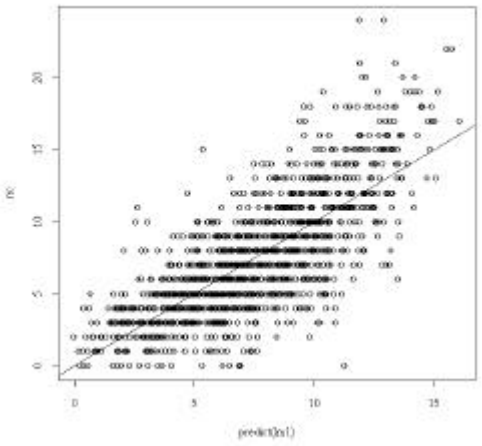
```
> glm1_glm(ric ~ heu + sem + sta + sem:sta,family=poisson)
Error: heap memory (48828 Kb) exhausted [needed 7489 Kb more]
       See "help(Memory)" on how to increase the heap size.
```

On peut encore augmenter la taille mais les temps de calculs suivront. Comparer les modèles sans interactions :

```
> lm1_lm(ric~heu+sem+sta)
> glm1_glm(ric~heu+sem+sta,family=poisson)

> plot(predict(glm1,type="response"),ric)
> abline(0,1)
> plot(predict(lm1),ric)
> abline(0,1)
```

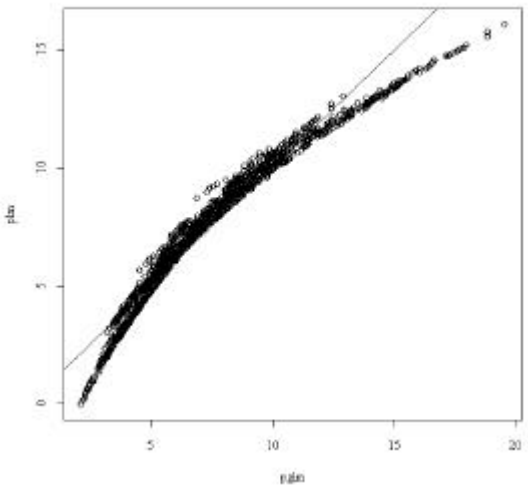


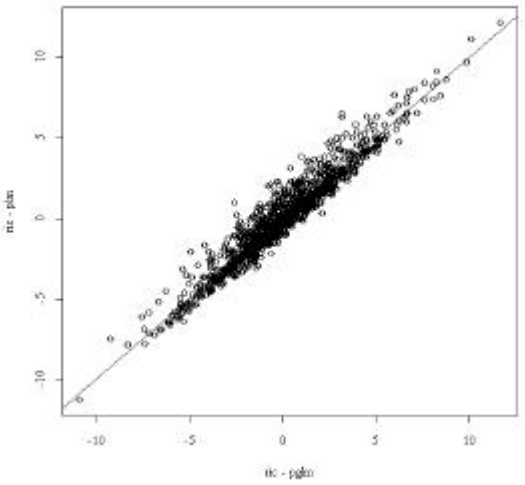
GLM



LM

```
> plm_predict(lm1)
> pglm_predict(glm1,type="response")
> plot(pglm,plm)
> plot(ric-pglm,ric-plm)
```



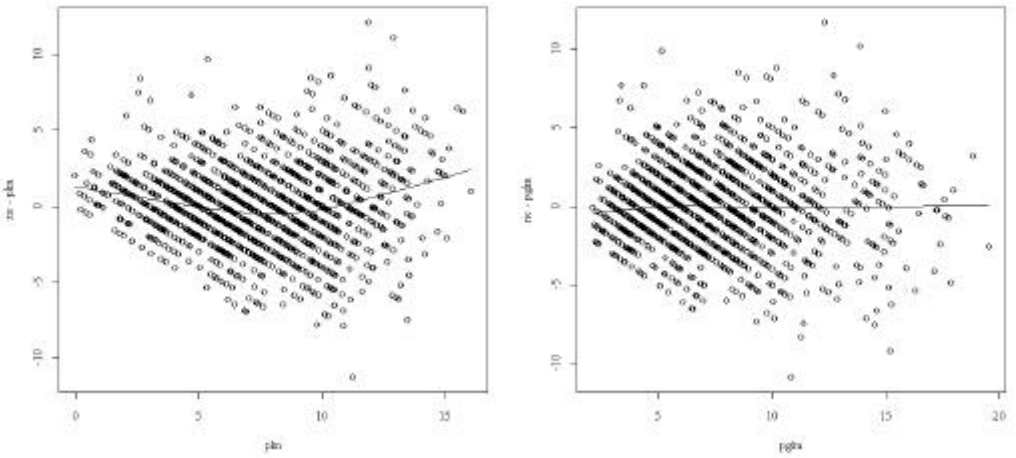


Les modifications sont donc sensibles. Le lien joue un rôle non négligeable. Mais les deux modèles sont de précision voisine, parce que l'erreur aléatoire dans ce modèle est considérable.

```
> sum((ric-plm)^2)
[1] 9673
> sum((ric-pglm)^2)
[1] 9154
```

Les erreurs sont globalement du même ordre mais n'ont pas la même structure.

```
> plot(plm,ric-plm)
> lines(lowess(plm,ric-plm))
> plot(pglm,ric-pglm)
> lines(lowess(pglm,ric-pglm))
```

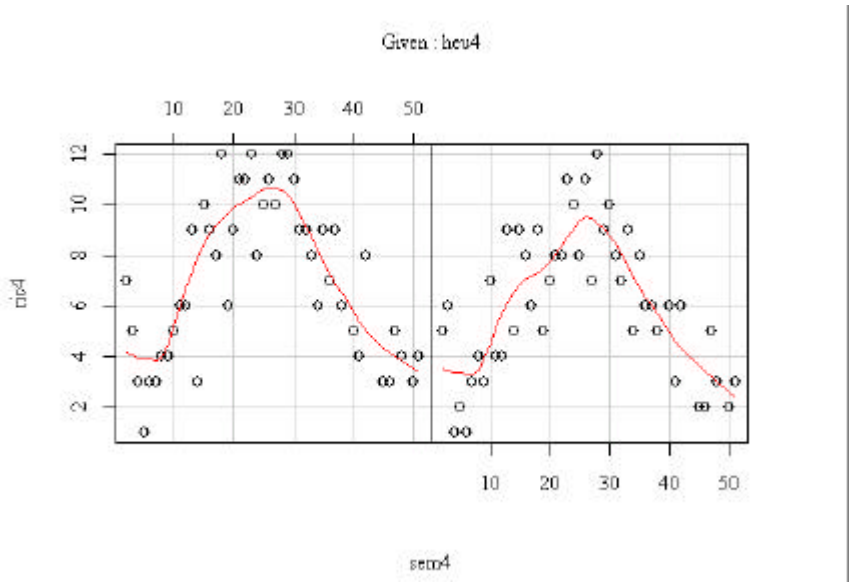


L'absence des interactions est de toute manière préjudiciable aux deux modèles.

5.2. Etude partielle

Extraire ce qui concerne la station 4 :

```
> ecrin4_ecrin[ecrin$STA==4,]
> ric4_ecrin4$RIC
> heu4_as.factor(ecrin4$HEU)
> sem4_ecrin4$SEM
>              coplot(ric4~sem4|heu4,show=F,panel=function                (x,y,...)
panel.smooth(x,y,span=0.3,...))
```




```
> lm4_lm(ric4~sem4+I(sem4^2)+I(sem4^3)+heu4)
> anova(lm4)
Analysis of Variance Table

Response: ric4
```

```
          Df Sum Sq Mean Sq F value Pr(>F)
sem4       1 0.0159  0.0159  0.0042  0.948
I(sem4^2)  1    474     474  125.01 <2e-16 ***
I(sem4^3)  1 0.1831  0.1831    0.05  0.827
heu4       1     22      22    5.80  0.018 *
Residuals 87    330       4
---
Signif. codes:  0  `***'  0.001  `**'  0.01  `*'  0.05  `.'  0.1  ` '  1
```

```
> glm4_glm(ric4~sem4+I(sem4^2)+I(sem4^3)+heu4,family=poisson)
> anova(glm4,test="Chisq")
Analysis of Deviance Table

Model: poisson, link: log

Response: ric4

Terms added sequentially (first to last)


          Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                        91      136.2
sem4       1  2.4e-03        90      136.2       1.0
I(sem4^2)  1     81.3        89       54.9       0.0
I(sem4^3)  1  2.7e-05        88       54.9       1.0
heu4       1      3.4        87       51.6       0.1
```

On introduit des différences sensibles. Continuer en étudiant d'autres stations.

5.3. Nombres d'accidents

On extrait de Eddy, P., Potter, E. & Page, B. (1976) Destination désastre. Grasset, Paris. 330-332 la date en nombre de jours entre le 01/01/1972 (x=1) et le 31/12/1975 (x=1461) de 142 catastrophes aériennes :

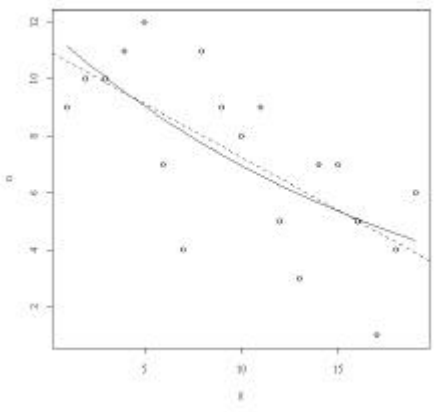
```
> date
  [1]    7   17   21   21   26   34   36   42   63   74   79  104  107  109  111
 [16]  126  129  139  142  150  166  167  170  176  181  181  181  211  211  224
 [31]  227  229  240  245  254  257  268  276  276  287  295  301  304  309  323
 [46]  333  338  343  343  355  358  364  388  395  416  418  418  428  430  444
 [61]  466  491  505  515  515  517  518  537  547  547  558  570  570  578  591
 [76]  600  605  607  617  620  637  640  652  662  672  673  687  708  716  721
 [91]  722  732  740  741  748  757  761  784  793  803  805  825  843  848  853
[106]  888  909  948  954  955  957  981  982  985  989 1033 1055 1066 1069 1087
[121] 1093 1094 1112 1126 1130 1154 1167 1209 1271 1308 1308 1311 1327 1338 1340
[136] 1363 1366 1369 1392 1399 1418 1422
> a_seq(0,1430,by=73)
> a
 [1]    0   73  146  219  292  365  438  511  584  657  730  803  876  949 1022
[16] 1095 1168 1241 1314 1387
```

On a partagé la période d'étude en unités de 73 jours. On compte le nombre d'accidents par période :

```
> n_as.vector(table(cut(date,a)))
> n
 [1]  9 10 10 11 12  7  4 11  9  8  9  5  3  7  7  5  1  4  6
```

Caractériser l'amélioration de la sécurité aérienne sur cette période. Dans un processus localement aléatoire on a affaire à une distribution poissonienne.

```
> length(n)
[1] 19
> x_1:19
> glmacci_glm(n~x,family=poisson)

> anova(glmacci,test="Chisq")
Analysis of Deviance Table

Model: poisson, link: log

Response: n

Terms added sequentially (first to last)


     Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                    18       26.1
x     1     11.2        17       14.9   0.00084

> plot(x,n)
> lines(x,predict(glmacci,type="response"))
> abline(lm(n~x),lty=2)
```



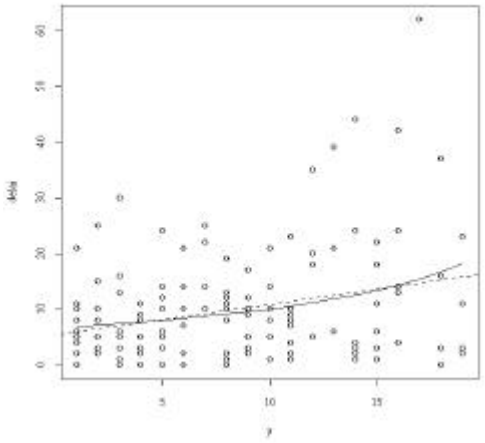
Exercice : prédire le nombre d'accident pour les périodes suivantes sous l'hypothèse de continuité du phénomène.

Pour les 138 premières catastrophes, on compte le nombre de jours d'attente de la catastrophe suivante :

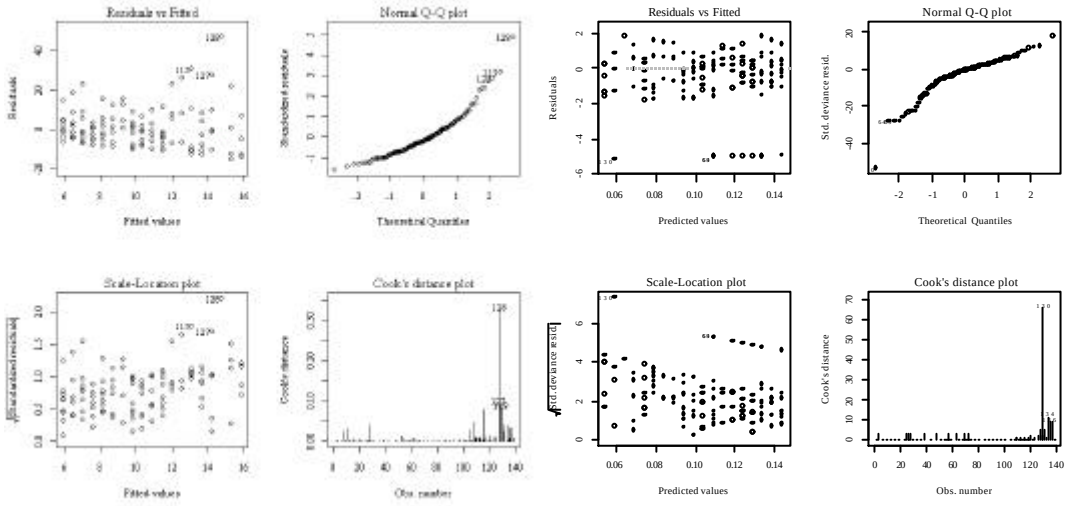
```
> y_as.numeric(cut(date,a))[1:138]
> delai_diff(date)[1:138]
> y
  [1]  1  1  1  1  1  1  1  1  1  2  2  2  2  2  2  2  2  2  2  3  3  3  3  3  3
 [26]  3  3  3  3  4  4  4  4  4  4  4  4  4  4  4  4  5  5  5  5  5  5  5  5  5  5
 [51]  5  5  6  6  6  6  6  6  6  7  7  7  7  8  8  8  8  8  8  8  8  8  8  8  9
 [76]  9  9  9  9  9  9  9  9 10 10 10 10 10 10 10 10 11 11 11 11 11 11 11 11 11
[101] 12 12 12 12 12 13 13 13 14 14 14 14 14 14 14 15 15 15 15 15 15 15 16 16 16
[126] 16 16 17 18 18 18 18 19 19 19 19 19 19
> delai
  [1] 10  4  0  5  8  2  6 21 11  5 25  3  2  2 15  3 10  3  8 16  1  3  6  5  0
 [26]  0 30  0 13  3  2 11  5  9  3 11  8  0 11  8  6  3  5 14 10  5  5  0 12  3
 [51]  6 24  7 21  2  0 10  2 14 22 25 14 10  0  2  1 19 10  0 11 12  0  8 13  9
 [76]  5  2 10  3 17  3 12 10 10  1 14 21  8  5  1 10  8  1  7  9  4 23  9 10  2
[101] 20 18  5  5 35 21 39  6  1  2 24  1  3  4 44 22 11  3 18  6  1 18 14  4 24
[126] 13 42 62 37  0  3 16 11  2 23  3  3 23
```

Caractériser l'amélioration de la sécurité aérienne sur cette période. Dans un processus localement aléatoire on a affaire à une distribution exponentielle.

```
> delai0_delai+1e-5
> glmexp_glm(delai0~y,family=Gamma)
> plot(y,delai)
> lines(y,predict(glmexp,type="response"))
> abline(lm(delai~y),lty=2)
```



```
> par(mfrow=c(2,2))
> plot(lm(delai~y))
> plot(glmexp)
```



Les modèles sont voisins mais les analyses sont différentes. La seconde repère les collisions.