DEA Analyse et Modélisation des Systèmes Biologiques

Introduction au logiciel S-PLUS©

D. Chessel

4 - Modèle linéaire généralisé

Résumé

La fiche contient le matériel nécessaire pour une séance de travaux dirigés sur S-PLUS consacrée au modèle linéaire généralisé. Elle illustre le cours de J. Estève, en particulier le fonctionnement de la régression logistique. La modélisation d'une variable quantitative (modèle linéaire), d'une fréquence et d'une variable binaire (présence-absence) sont mises en parallèle.

Plan

1 - Modéliser une probabilité

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

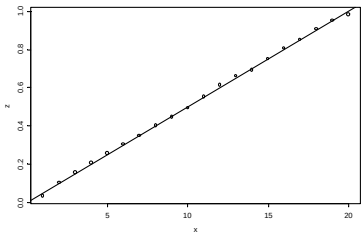
```
> x<-1:20
```

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
```

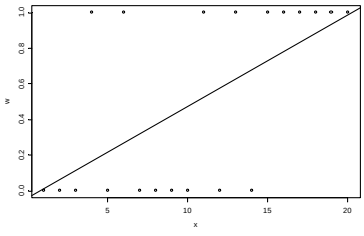
Mettons une petite erreur sur cette probabilité de gagner :

```
> 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 :

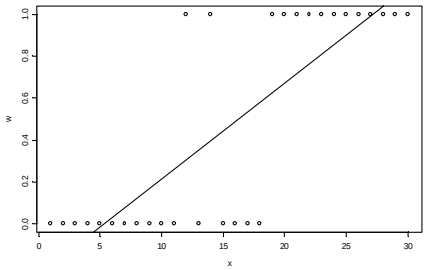
```
> lm(z~x)
Call:
lm(formula = z ~ x)

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
```

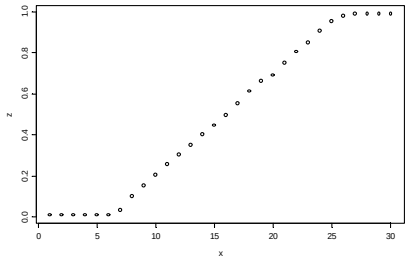
Le modèle y = 0.5*x tient encore si on remplace la probabilité par sa réalisation aléatoire. Moralité : les données peuvent être totalement fausses et le modèle accessible. C'est normal, la statistique considère toujours que les données sont « fausses ».

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 :

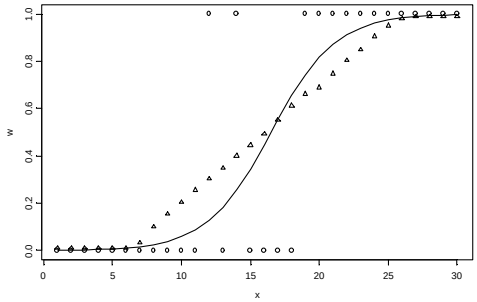
```
> x<-1:30
> z<-c(rep(0.01,le=6),z,rep(0.99,le=4))
> z
 [1] 0.01000 0.01000 0.01000 0.01000 0.01000 0.01000 0.03421 0.10178
 [9] 0.15495 0.20513 0.25622 0.30368 0.34967 0.40025 0.44609 0.49408
[17] 0.55102 0.61259 0.66149 0.69004 0.74892 0.80398 0.84907 0.90670
[25] 0.95086 0.98000 0.99000 0.99000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

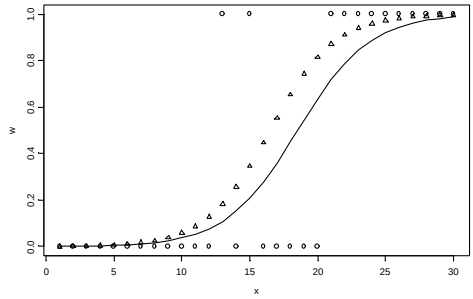



```
> plot(x,w)
> lines(x,predict(glm(w~x,family="binomial"),type="response"))
> points(x,z,pch=2)
```



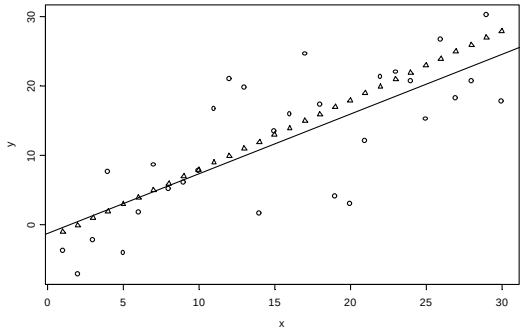
Le modèle qui a fourni la réalisation qui a fourni l'estimation du modèle met en évidence la contradiction. Faisons enfin :

```
> 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))
```



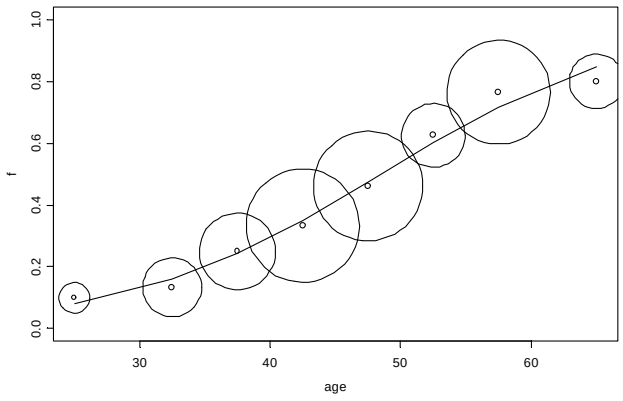
2 - Le fonctionnement de la régression logistique

2.1 - 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
```

2.2 - GLIM

Cette procédure est assurée par `glim` :

```
> glim0<-glim(age,Y,n,error="binomial",link="logit")
> glim0
$coef:
 Intercept    X1
    -5.091 0.105

$var:
          [,1]        [,2]
[1,]  0.120448 -0.00247404
[2,] -0.002474  0.00005325

$deviance:
[1] 287.015   5.242

$df:
[1] 7 6

$scale:
[1] 1

$intercept:
[1] T

> r<-glim0[[1]][1]+glim0[[1]][2]*age
> p<-exp(r)/(1+exp(r))
> p
[1] 0.07833 0.1574 0.2400 0.3481 0.4744 0.6041 0.7207 0.8501
```

Dans la documentation de la fonction glim :

DETAILS

An observation is considered missing if there are missing values for that observation in any of x, y, n, or wt. Two important choices when using glim are the probability distribution used to model the errors and the "link" function that describes the expected value of the response in

terms of the covariates. Of the error distributions the Poisson and Binomial distributions are discrete valued, the Gamma and Inverse Gaussian take on positive real values, and the Gaussian assumes all real values. The logit, probit and complementary log-log link functions are typically used when modeling proportions (with Binomial errors). The log link function is used in conjunction with Poisson errors as well as in other cases.

Two different types of residuals may be returned, Pearson residuals and deviance residuals. Pearson residuals are the difference between the observation and the fit divided by the square root of the estimated variance for the observation. For Poisson errors the sum of the squared Pearson residuals is the Pearson Chi-squared goodness-of-fit statistic. The deviance residual is defined as the square root of the deviance for each observation with the sign determined by the sign of the observation minus the fit. (The sum of the squares of the deviance residuals will equal the deviance for the final model in unweighted cases.)

Denote the predicted value f(xb+offset) by mu. The deviance for each point is computed as:
(y-mu)^2 for the Gaussian,
2*(y*log(ifelse(y==0,1,y/mu)) - (y-mu)) for the Poisson,
2*n*(y*log(ifelse(y==0,1,y/mu)) + (1-y)*log(ifelse(y==1,1,(1-y)/(1-mu)))) for the Binomial,
-2*(log(y/mu) - (y-mu)/mu) for the Gamma, and
(y-mu)^2 / (y*mu^2) for the Inverse Gaussian.
The deviance component of the output is the sum of wt times the deviance vector.

An iterated reweighted least squares algorithm is used to estimate the coefficients. Observations that have missing values in x, y, n or wt are deleted from the computations; residuals for such observations will be NA.

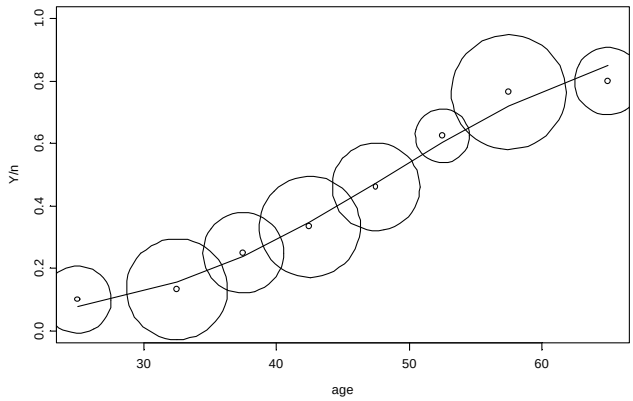
BACKGROUND

The name glim stands for Generalized Linear Models, that is, generalized linear regression. These models are linear in the sense that the parameters are linear in the explanatory variables (covariates). They are generalized by allowing the covariates to model a non-linear transformation of the expected value of the response, and by allowing the distribution of the errors to be non-Gaussian.

REFERENCE

McCullagh, P. and Nelder, J. A. (1983). Generalized Linear Models. London: Chapman and Hall.

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



Moralité : on peut lire un ouvrage de statistique (un grand) pour nombre de procédures de S-PLUS.

L'échantillon total est composé de témoins, de fumeurs et d'ex-fumeurs :

```
> ntem
[1] 50 72 56 68 57 34 69 38
> ytem
[1]  4  7 10 17 20 17 44 27
> > nfum
[1] 25 38 30 37 32 20 41 24
> yfum
[1]  3  6  8 14 16 14 35 22
> nex
[1] 25 40 34 45 41 26 60 38
> yex
[1]  3  7 12 19 24 19 51 31
```

Assembler :

```
> ntot
 [1] 50 72 56 68 57 34 69 38 25 38 30 37 32 20 41 24 25 40 34 45 41
[22] 26 60 38
> ytot
 [1]  4  7 10 17 20 17 44 27  3  6  8 14 16 14 35 22  3  7 12 19 24
[22] 19 51 31
> agetot
 [1] 25.0 32.5 37.5 42.5 47.5 52.5 57.5 65.0 25.0 32.5 37.5 42.5
[13] 47.5 52.5 57.5 65.0 25.0 32.5 37.5 42.5 47.5 52.5 57.5 65.0
> fumtot
 [1] 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

> glim(cbind(agetot,fumtot),ytot,ntot,error="binomial",link="logit",
       sequence=1:2)
$coef:
 Intercept agetot fumtot
    -5.609 0.1061   0.82

$var:
          [,1]       [,2]       [,3]
[1,]  0.140260 -0.0026494 -0.0199337
[2,] -0.002649  0.0000554  0.0001334
[3,] -0.019934  0.0001334  0.0236735

$deviance:
[1] 39.022  9.782

$df:
```

```
[1] 22 21

$scale:
[1] 1

$intercept:
[1] T
```

La variation de déviance (39.022 - 9.782 = 29.24) suit un khi2 à 1 ddl.

```
> 1-pchisq(29.24,1)
[1] 6.395e-008 # Selon la loi 91.32 Fumer nuit à la santé
> exp(0.82)
[1] 2.27 # Le risque est multiplié par 2.7
```

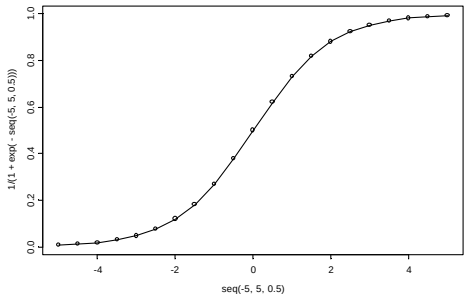
Moralité : on peut faire du GLIM avec S-PLUS (d'ailleurs, on peut tout faire, du bootstrap, de l'ARMA, de l'ANOVA, du glm, du non paramétrique, de la modélisation, du graphique, etc).

3 - Modéliser une présence-absence

3.1 - Le problème

On utilise encore la fonction logistique $y = \frac{1}{1+e^{-x}}$ en régression comme fonction de lien et on parle alors de régression logistique.

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



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) :

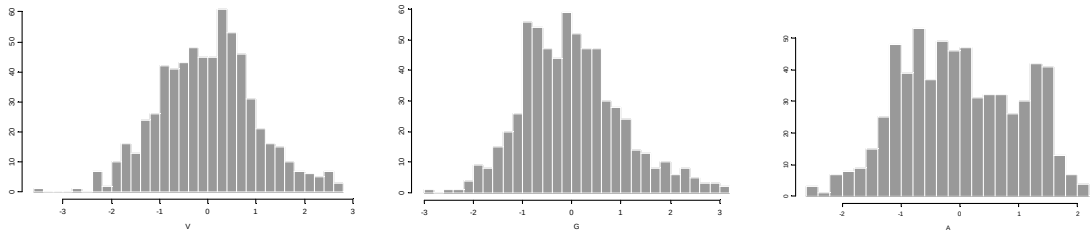
```
> levels(gardonmi$S)
[1] "Atla" "Garo" "Loir" "Manc" "Medi" "Nord" "Rhon" "Sein"
```

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)
>
```



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

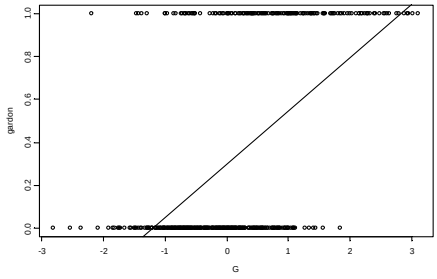
La particularité du problème «modéliser la présence du Gardon par les variables environnementales» tient à la variable à expliquer (0-1) dont les valeurs sont des réalisations d'un événement qui avait une certaine probabilité de survenir. On ne modélise donc pas le résultat mais la probabilité de ce 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)
```

[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.

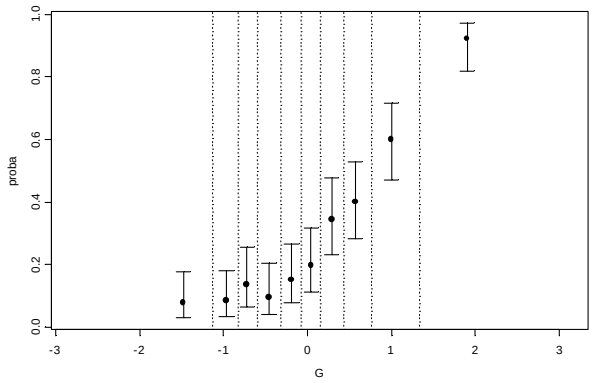
```
> abline(lm(gardon~G)
```



Commentaires

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

```
freqgrad<-function(datami=gardonmi,nomvar="G",nomy="gardon", ncla=10) {
  x0<-datami[,nomvar]
  yobs<-datami[,nomy]
  if (is.numeric(x0)==F) return ("Numeric data need")
  q0<-quantile(x0,probs = seq(0,1,1/ncla), na.rm = F)
  c.cla<-quantile(x0,probs=seq(0+1/2/ncla,1-1/2/ncla,1/ncla),na.rm=F)
  xmin<-min(x0)
  xmax<-max(x0)
  q1<-cut(x0,q0, include.lowest = T)
  t0<-table(q1,yobs)
  t0[, 1] <- (t0[, 1] + t0[, 2])
  freq<-t0[,2]/t0[,1]
  toto1<-rep(0,ncla)
  toto2<-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
            toto1[i] <- a0[1]
            toto2[i] <- a0[2]
            if(a0[1] > freq[i]) toto1[i] <- NA
            if(a0[2] < freq[i]) toto2[i] <- NA
       } else {
            toto1[i] <- NA
            toto2[i] <- NA
       }
  }
  ymin<- min(toto1,na.rm=T)
  ymax<- max(toto2,na.rm=T)
  plot(xmin,0,ylim=c(ymin,ymax),xlim=c(xmin,xmax),
  ylab="proba",xlab=nomvar,type="n")
  points(c.cla,freq,pch=16)
  for (i in 1:ncla) {
       if (!is.na(toto1[i])) {
            error.bar(c.cla[i],freq[i],toto1[i],toto2[i],
            add=T,incr=F,gap=F)
       }
       if (i>1) {
            abline(v=q0[i],lty=2)
       }
  }
}
> freqgrad()
```



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 S-PLUS, un seul paramètre dans gml suffit à se mettre dans cette situation.

3.2 - Courbes de réponse

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

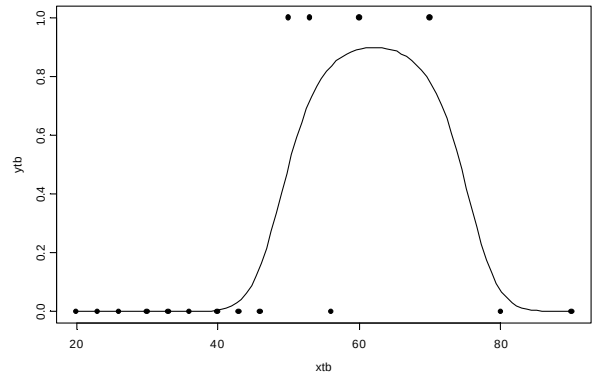
family a family object - a list of functions and expressions for defining the link and variance functions, initialization and iterative weights. Families supported are gaussian, binomial, poisson, Gamma, inverse.gaussian and quasi. Functions like binomial produce a family object, but can be given without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

Commençons par un exemple [2] (p. 5) :

```
> xtb
 [1] 20 23 26 30 33 36 40 43 46 50 53 56 60 70 80 90
> ytb
 [1] 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0

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

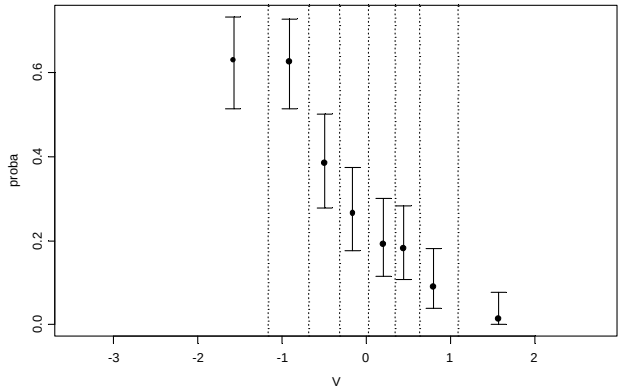
[2] Ter Braak, C.J.F. & Looman, C.W.N. (1986) Weighted averaging, logistic regression and the Gaussian response model. *Vegetatio* : 65, 3-11.

```
> lines(xnou,ynou)
```

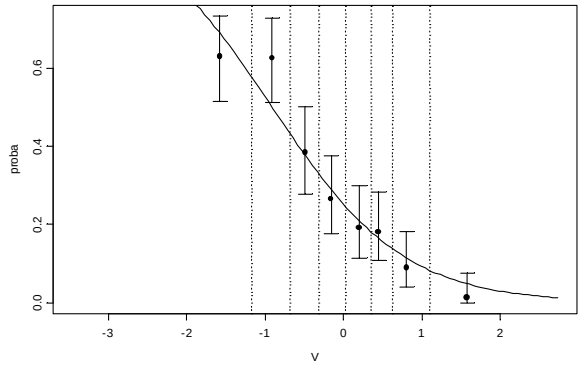


La probabilité de présence du Gardon dépend-elle de la vitesse ?

```
> freqgrad(nomvar="V",ncla=8)
```

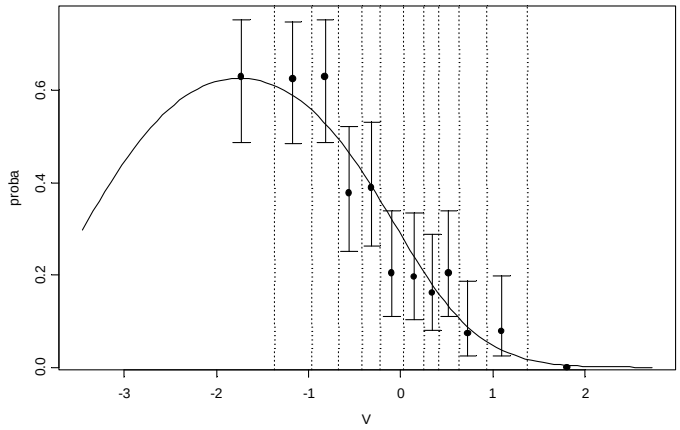



```
> glm1<-glm(gardon~V, family=binomial)
> glm1
Call:
glm(formula = gardon ~ V, family = binomial)

Coefficients:
 (Intercept)      V
      -1.088 -1.196

Degrees of Freedom: 645 Total; 643 Residual
Residual Deviance: 652
> anova(glm1,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

> 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 ?

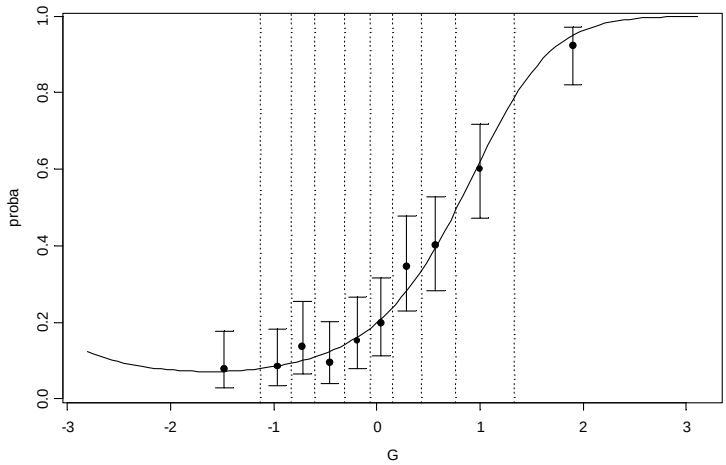
```
> glm2<-glm(gardon~V+V^2, family=binomial)
> anova(glm2,test="Chisq")
Analysis of Deviance Table

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
```




```
> freqgrad(nomvar="V",ncla=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+G^2,family=binomial)
> freqgrad(nomvar="G")
>  xnou<-seq(min(G),max(G),len=100)
> lines(xnou,predict(glm3,data.frame(G=xnou),type="response"))
```



3.3 - Surface de réponse

Écrire le modèle :

```
> glm3<-glm(gardon~V+V^2+G+G^2, family=binomial)
> anova(glm3,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 0.0000000
I(V^2)  1     16.8       642      635.3 0.0000425
     G  1    127.6       641      507.7 0.0000000
I(G^2)  1     11.2       640      496.4 0.0008122
```

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

DESCRIPTION

Creates a perspective plot given a matrix that represents heights on an evenly spaced grid. Axes and/or a box may be placed on the plot.

USAGE

```
persp(x, y, z, xlab = "X", ylab = "Y", zlab = "Z", axes = T,
      box = T, eye = <<see below>>, zlim = <<see below>>, ar = 1)
```

REQUIRED ARGUMENTS

x vector containing x coordinates of the grid over which z is evaluated. The values should be in sorted order; missing values are not accepted.

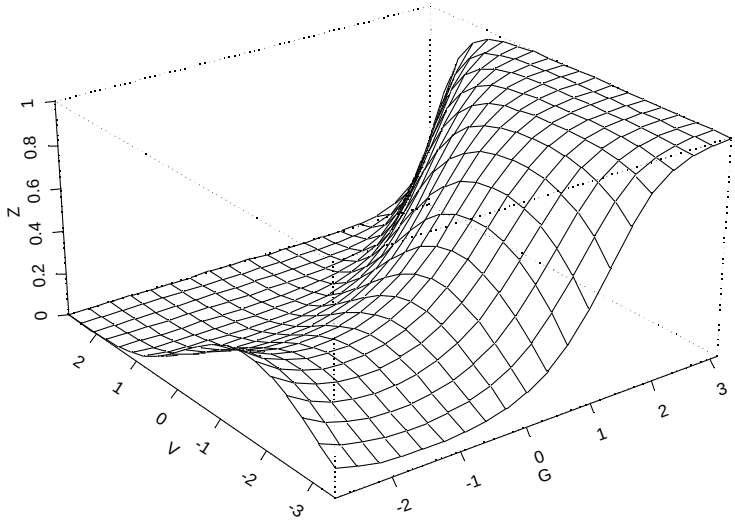
y vector of grid y coordinates. The values should be in sorted order; missing values are not accepted.

z matrix of size length(x) by length(y) giving the surface height at grid points, i.e., z[i,j] is evaluated at x[i], y[j]. The rows of z are indexed by x, and the columns by y. Missing values (NAs) are allowed, but should generally not occur in the convex hull of the non-missing values.

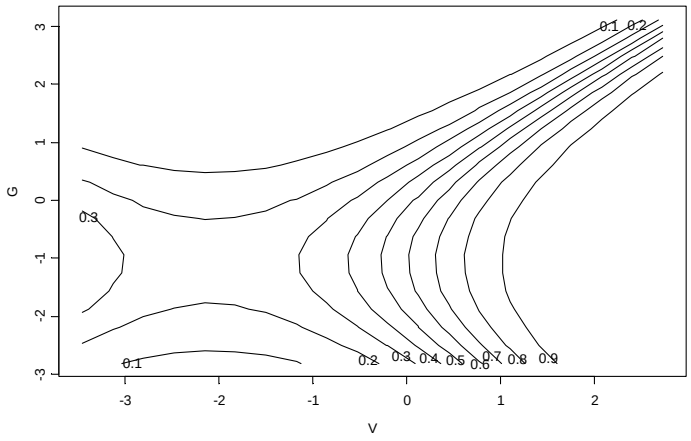
The first argument may be a list with components x, y, and z. In particular, the result of interp is suitable.

If the first argument is a matrix, it is assumed to be the z matrix and vectors x and y are generated (x is 1:nrow(z), y is 1:ncol(z)). The result of predict together with expand.grid is suitable as an argument to persp.

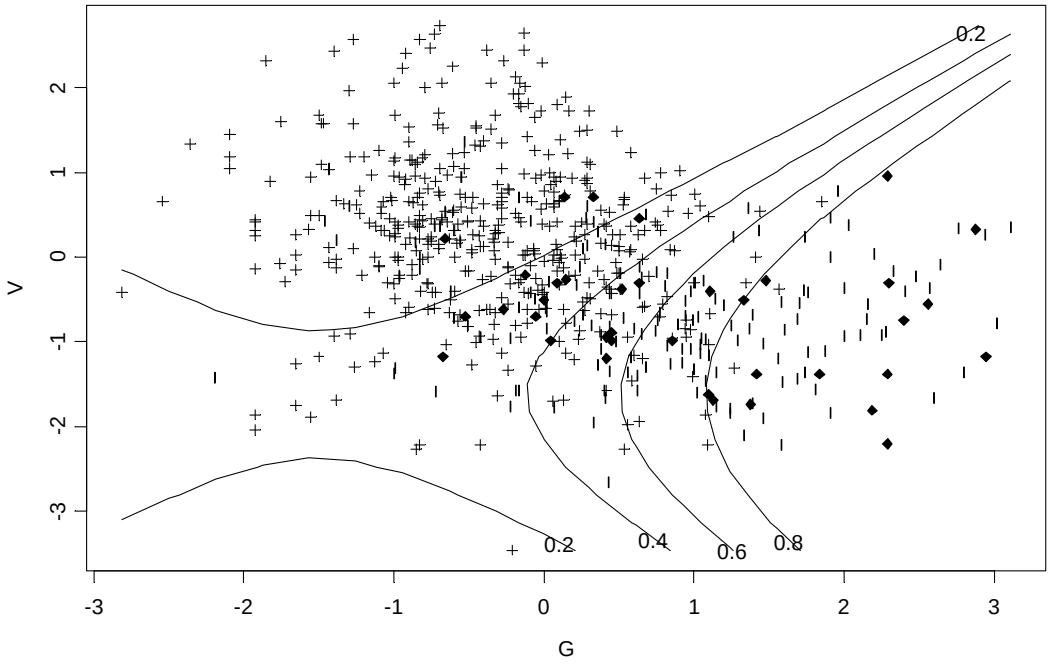
```
> newresult<-matrix(newresult,nrow=20,ncol=20,byrow=T)
> newresult[1:5,1:5]
        [,1]   [,2]   [,3]   [,4]   [,5]
[1,] 0.13390 0.1942 0.2562 0.3111 0.3519
[2,] 0.10496 0.1545 0.2072 0.2552 0.2917
[3,] 0.08746 0.1300 0.1760 0.2187 0.2518
[4,] 0.07782 0.1162 0.1583 0.1978 0.2286
[5,] 0.07412 0.1109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


> persp(newG,newV,newresult,xlab="G",ylab="V")
```




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

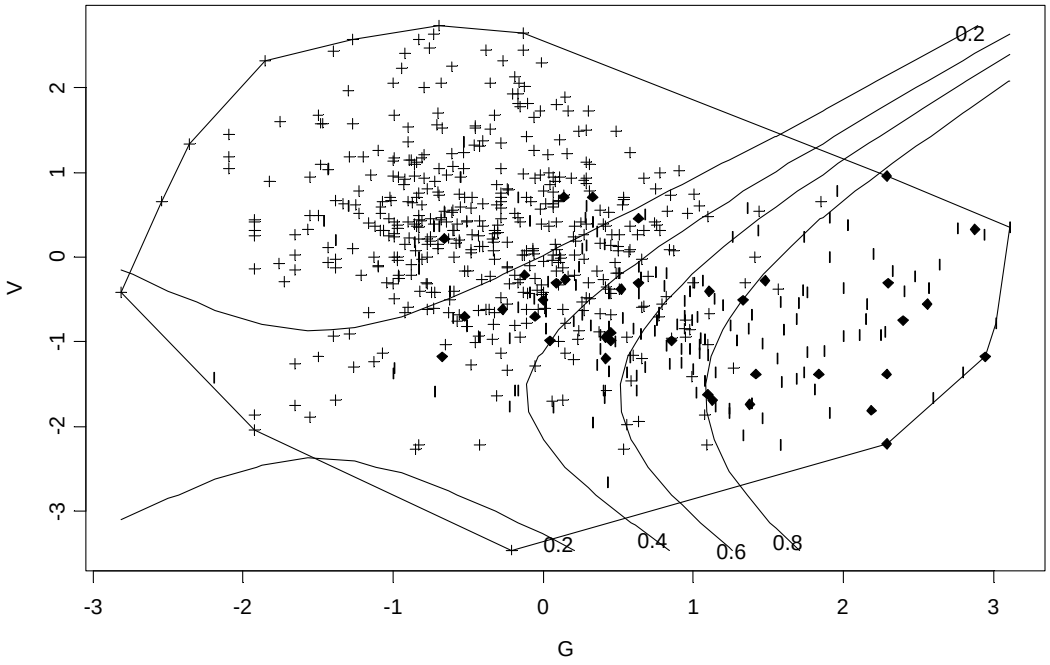



```
> contour(newG,newV,newresult,xlab="G",ylab="V")
> points(G[gardon==0],V[gardon==0],pch=3)
> points(G[gardon==1],V[gardon==1],pch=18)
```



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+V^2+G+G^2+A+A^2, family=binomial)
> anova(glm4,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 0.0000000
I(V^2)  1     16.8       642      635.3 0.0000425
     G  1    127.6       641      507.7 0.0000000
I(G^2)  1     11.2       640      496.4 0.0008122
     A  1     21.4       639      475.0 0.0000037
I(A^2)  1     12.3       638      462.7 0.0004435
```

Et le bassin ?

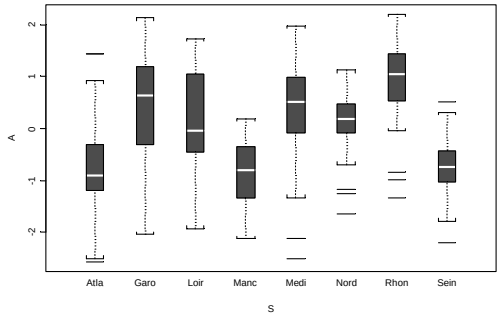
```
> glm5<-glm(gardon~V+V^2+G+G^2+A+A^2+S, family=binomial)
> anova(glm4,glm5,test="Chisq")
Analysis of Deviance Table

Response: gardon

                            Terms Resid. Df Resid. Dev Test Df
1     V + V^2 + G + G^2 + A + A^2       638      462.7
2 V + V^2 + G + G^2 + A + A^2 + S       631      439.7   +S  7
  Deviance  Pr(Chi)
1
2    22.97 0.001726
```

La difficulté :

```
> glm6<-glm(gardon~V+V^2+G+G^2+S+A+A^2, family=binomial)
> anova(glm6,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 0.00000
I(V^2)  1     16.8       642      635.3 0.00004
     G  1    127.6       641      507.7 0.00000
I(G^2)  1     11.2       640      496.4 0.00081
     S  7     12.4       633      484.0 0.08741
     A  1     35.9       632      448.1 0.00000
I(A^2)  1      8.4       631      439.7 0.00373
```

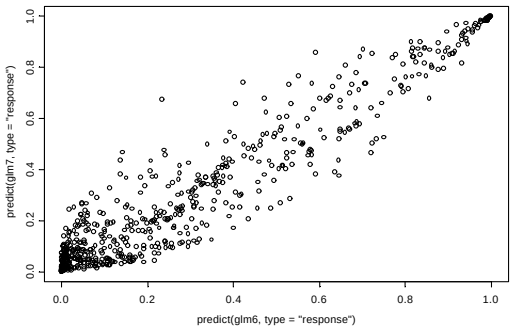
S est un effet significatif derrière A mais pas devant. Les deux variables sont liées :

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

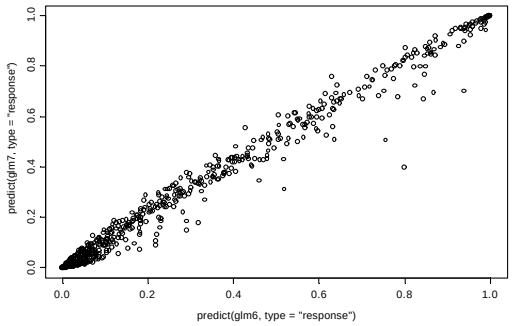


Mais introduire l'une ou l'autre des variables ne donnent pas les mêmes prédictions :

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




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



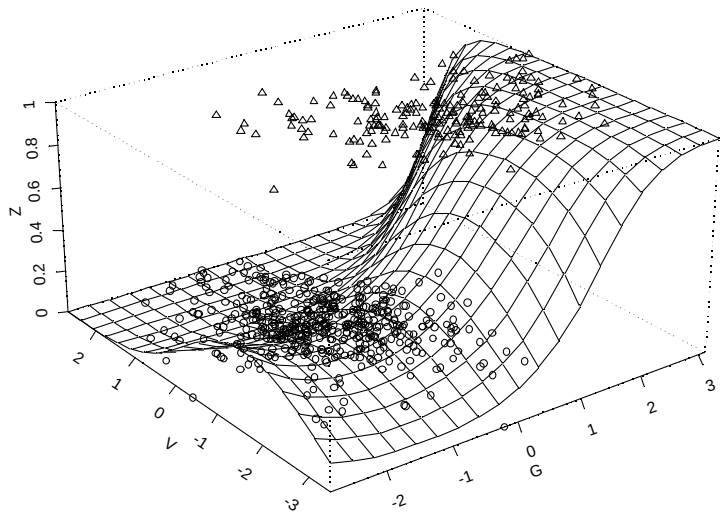
Le terme A^2 a donc beaucoup moins d'effet que le terme S.

Voir aussi :

step, add1, drop1

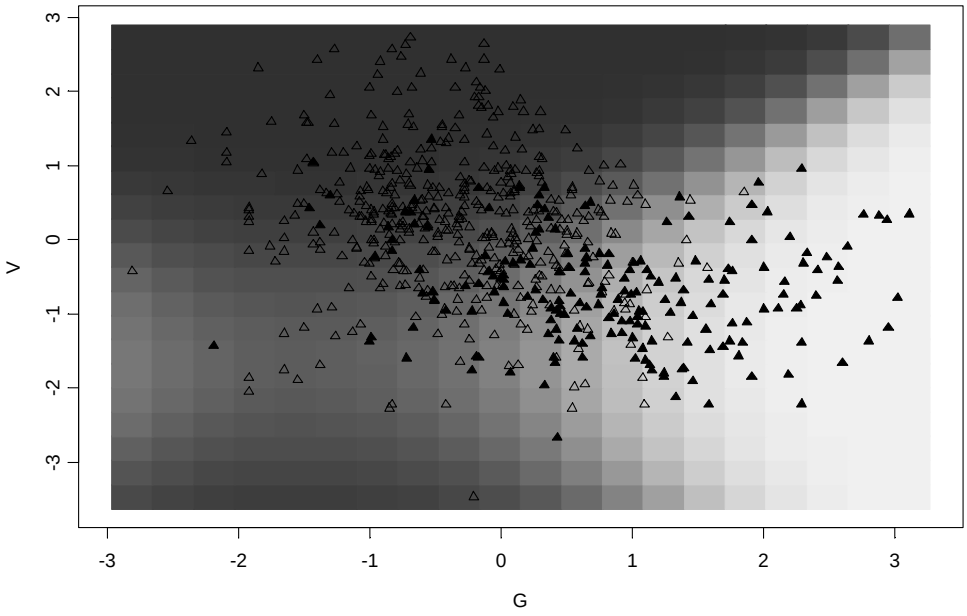
Essayer encore :

```
> persp(newG,newV,newresult,xlab="G",ylab="V")
> ppp<-persp(newG,newV,newresult,xlab="G",ylab="V")
> G0<-G[gardon==0]
> V0<-V[gardon==0]
> R0<-gardon[gardon==0]
> points(perspp(G0,V0,R0,ppp),pch=1)
> G1<-G[gardon==1]
> V1<-V[gardon==1]
> R1<-gardon[gardon==1]
> points(perspp(G1,V1,R1,ppp),pch=2)
```



Essayer encore :

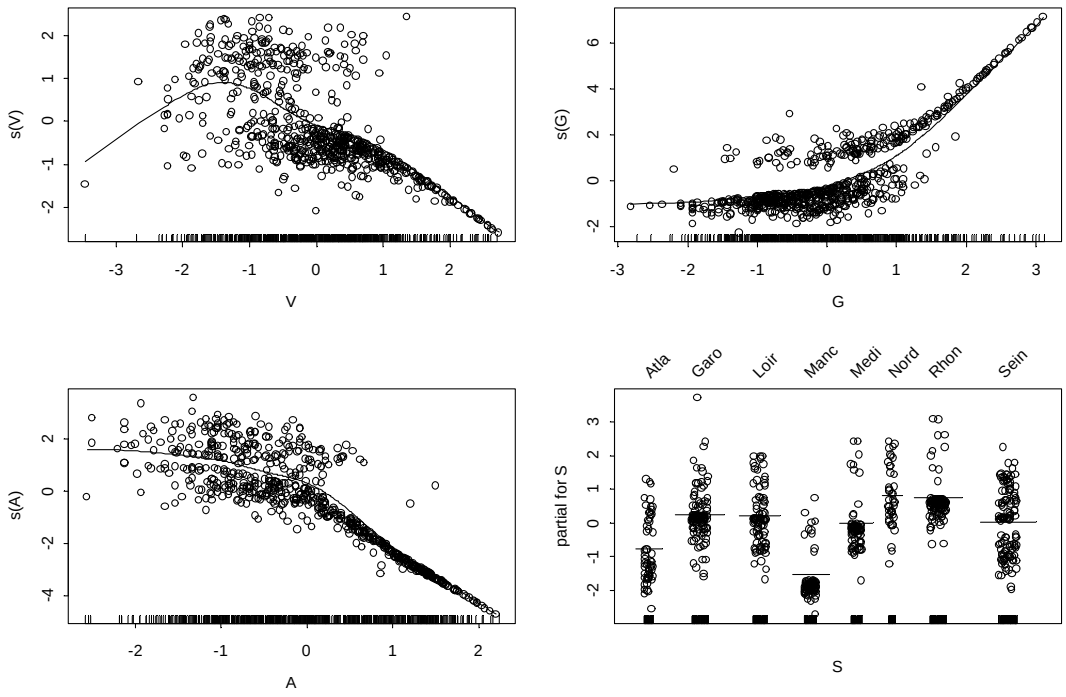
```
> image(newG,newV,newresult,xlab="G",ylab="V")
> points(G1,V1,pch=17)
> points(G0,V0,pch=2)
```



4 - Modèles additifs généralisés

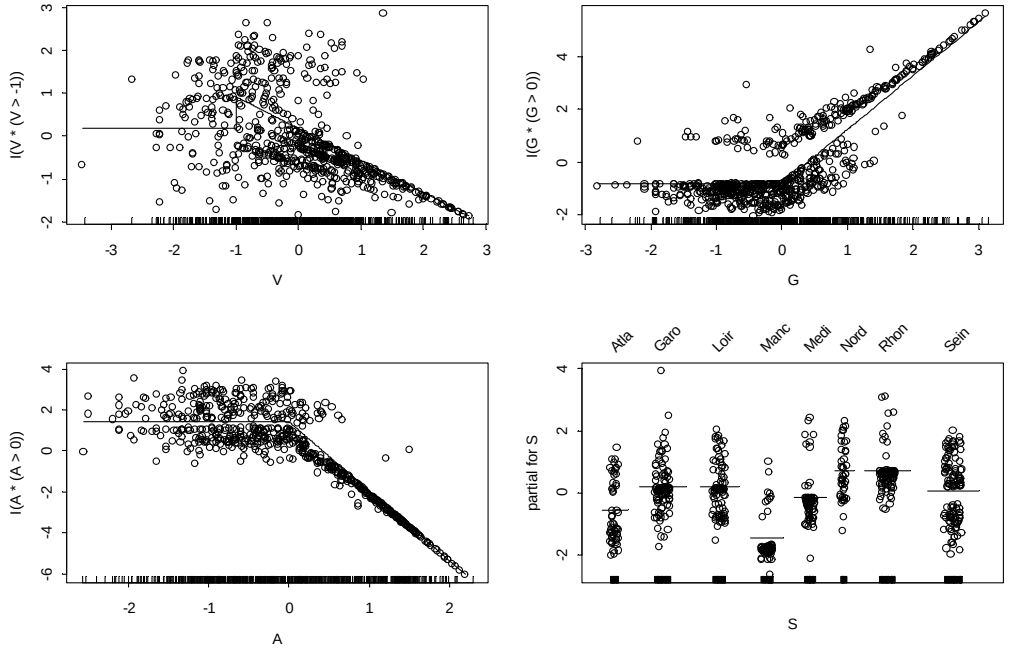
gam

```
> glms<-gam(gardon~s(V)+s(G)+s(A)+S,family=binomial,data=gardonmi)
> plot(glms,residuals=T)
> par(mfrow=c(2,2))
> plot(glms,residuals=T)
```



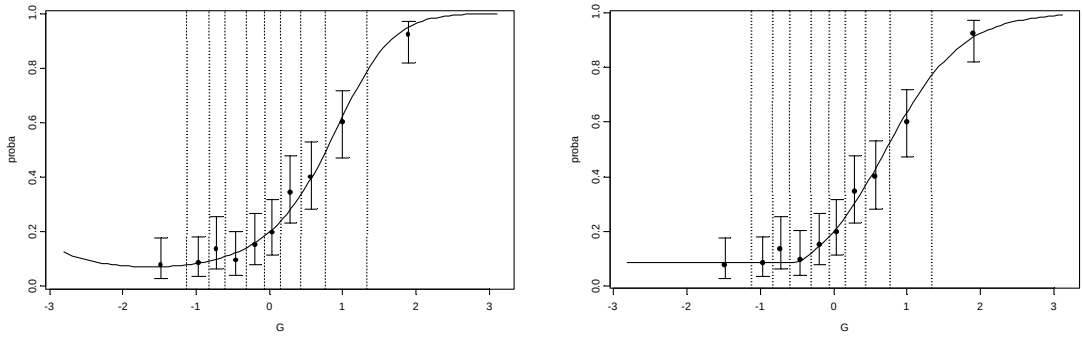
On travaille ici sur le lien et les résidus partiels. Les termes carrés ne sont pas indispensables. Il vaut mieux s'orienter vers des fonctions localement linéaire :

```
> glms1<-gam(gardon~I(V*(V>-1))
   +I(G*(G>0))+I(A*(A>0))+S,family=binomial,data=gardonmi)
> plot(glms1,residuals=T)
```

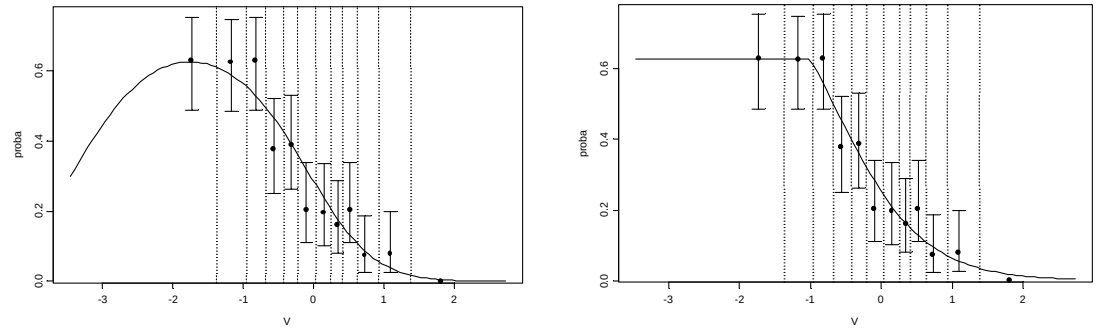



```
glm8<-glm(gardon~I((G+0.5)*(G>-0.5)),family=binomial,data=gardonmi)
xnou<-seq(min(G),max(G),len=100)
ynou<-predict(glm8,data.frame(G=xnou),type="response")
freqgrad()
```

```
lines(xnou,ynou)
```

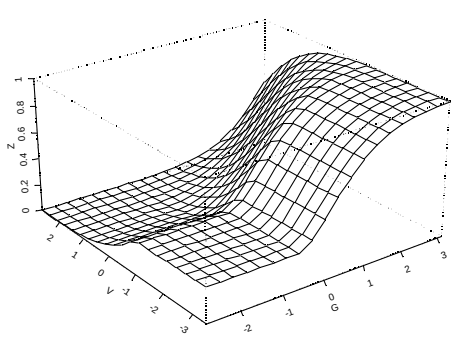


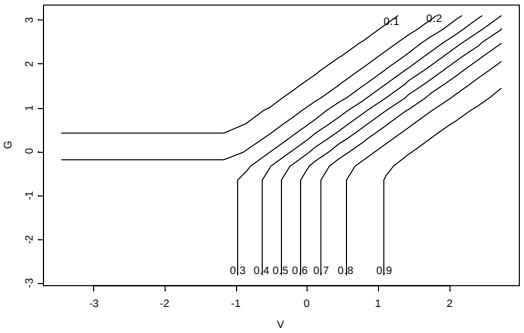

```
glm9<-glm(gardon~I((V+1)*(V>-1)),family=binomial,data=gardonmi)
freqgrad(nomvar="V",ncla=12)
xnou<-seq(min(V),max(V),len=100)
lines(xnou,predict(glm9,data.frame(V=xnou),type="response"))
```



Et les deux à la fois ?

```
> glm10<-glm(gardon~I((G + 0.5) * (G > -0.5))+
   I((V + 1) * (V > -1)),family = binomial, data = gardonmi)
> anova(glm10,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
I((G + 0.5) * (G > -0.5))  1    222.3       643      564.8 0.000e+000
    I((V + 1) * (V > -1))  1     63.4       642      501.4 1.665e-015

> newresult<-predict.glm(glm10,newdata,type="response")
> newresult<-matrix(newresult,nrow=20,ncol=20,byrow=T)
> persp(newG,newV,newresult,xlab="G",ylab="V")
> contour(newV,newG,newresult,xlab="V",ylab="G",nlevels=10)
```





Et l'interaction ? On dit que la statistique est un art...

5 - Le lien entre GLIM et glm

Nous avons vu qu'une régression logistique sur des données en présence-absence s'aborde par l'écriture d'un modèle `glm(y~x, family = binomial)` tout comme une régression simple s'aborde par l'écriture d'un modèle `lm(y~x)` qui équivaut à `glm(y~x, family=gaussian)`. Sur des données de fréquences, on a utilisé `glim(x, y, n, ...)` qui donne un objet ne supportant pas la fonction `predict`, ce qui semble curieux. Dans la version 5 sous Unix, la fonction glim est abandonnée, parce que la cohérence de l 'ensemble l'assure de manière élégante.

```
> glim0<-glim(age,Y,n,error="binomial",link="logit")
> r<-glim0[[1]][1]+glim0[[1]][2]*age
> r
[1] -2.4653 -1.6776 -1.1525 -0.6274 -0.1023  0.4228  0.9479  1.7355
> p<-exp(r)/(1+exp(r))
> p
[1] 0.07833 0.15741 0.24003 0.34810 0.47444 0.60415 0.72069 0.85012
```

Pour avoir l'équivalent avec un modèle linéaire généralisé il suffira de faire :

```
> glm0<-glm(cbind(Y,n-Y)~age,family=binomial)
> predict(glm0)
      1      2      3       4       5      6      7     8
 -2.465 -1.678 -1.153 -0.6274 -0.1023 0.4228 0.9479 1.736
> predict(glm0,type="response")
       1      2    3      4      5      6      7      8
 0.07833 0.1574 0.24 0.3481 0.4744 0.6041 0.7207 0.8501
```

`glm(`**`(n succès, n échecs)`**`~x, family = binomial)` redonne un statut identique à tous les modèles. On s'attend alors à retrouver l'analyse de covariance du modèle linéaire dans l'extension :

```
> glm1<-glm(cbind(ytot,ntot-ytot)~agetot+fumtot,family=binomial)
> anova(glm1,test="Chisq")
Analysis of Deviance Table

Binomial model

Response: cbind(ytot, ntot - ytot)

Terms added sequentially (first to last)
       Df Deviance Resid. Df Resid. Dev    Pr(Chi)
  NULL                    23      320.8
```

```
agetot  1    281.8        22       39.0 0.000e+000
fumtot  1     29.2        21        9.8 6.395e-008
> coefficients(glm1)
 (Intercept) agetot fumtot
      -5.609 0.1061   0.82
```

On a retrouvé ainsi l'intégralité des résultats avec le graphique en cadeau :

```
> plot(agetot,predict(glm1,type="response"))
```