VGAM Family Functions for Nonlinear Regression

Beta Version 0.5-15

© Thomas W. Yee

Department of Statistics,
University of Auckland,
New Zealand
yee@stat.auckland.ac.nz
http://www.stat.auckland.ac.nz/∼yee

January 16, 2004

Contents

[Important note: This document and code is not yet finished, but should be completed one day . . .]

Nb. Many of the VGAM **family functions documented here were written by Ziming Guan, a Science Faculty summer scholar during 2001–2002. T. Yee hasn't had time to check them out, but hopes to do so soon.**

1 Introduction

This document describes in detail VGAM family functions for nonlinear regression. Many of VGAM's features come from `glm()` and `gam()` so that readers unfamiliar with these functions are referred to Chambers and Hastie (1993). Additionally, the VGAM *User Manual* should be consulted for general instructions about the software.

Consider the model

$$Y_i \;=\; f(\boldsymbol{u}_i;\boldsymbol{\theta}) + \varepsilon_i, \qquad i = 1,\ldots,n, \tag{1}$$

where $\boldsymbol{\theta}$ is an M-vector of unknown parameters, the ε_i are assumed to be i.i.d. $N(0,\sigma^2)$ and the relationship $f(\boldsymbol{u}_i;\boldsymbol{\theta})$ is nonlinear[1]. Eqn (1) is the general form of a nonlinear regression model, and may be fitted using VGAM. See Table 1 for examples that have been implemented in VGAM. Nonlinear regression is a big topic and often requires specialist skills and knowledge. Books in this area include Seber and Wild (1989) and Bates and Watts (1988). We shall only look at a very few topics in this area.

1.1 Why use VGAM?

The primary S-PLUS function for nonlinear regression is `nls()` (for *nonlinear least squares*), which chooses $\hat{\boldsymbol{\theta}}$ to maximize

$$\ell(\boldsymbol{\theta}) \;=\; -\frac{1}{2}\sum_{i=1}^{n} w_i \left(y_i - f(\boldsymbol{u}_i;\boldsymbol{\theta})\right)^2 \tag{2}$$

where the w_i are known positive weights(so that $\mathrm{Var}(\varepsilon_i) = \sigma^2/\mathbf{W}_i$). The deviance

$$D \;=\; \sum_{i=1}^{n} w_i \left(y_i - f(\boldsymbol{u}_i;\boldsymbol{\theta})\right)^2, \tag{3}$$

the (weighted) residual sum of squares. `nls()`, which uses a Gauss-Newton algorithm, is very convenient as it can make use of `deriv()` for computing derivatives. A typical usage is (it fits the Michaelis-Menten model)

```
> Treated = Puromycin[Puromycin$state == "treated", ]
> Treated[1:3,]
  conc rate   state
1 0.02   76 treated
2 0.02   47 treated
3 0.06   97 treated
> Purfit1 = nls(rate ~ Vm*conc/(K + conc), Treated, list(Vm=200, K=0.1))
> summary(Purfit1)

Formula: rate ~ Vm * conc/(K + conc)

Parameters:
    Estimate Std. Error t value Pr(>|t|)
Vm 2.127e+02  6.947e+00  30.615 3.24e-11 ***
K  6.412e-02  8.281e-03   7.743 1.57e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.93 on 10 degrees of freedom
```

[1] In this section it is more convenient to use $\boldsymbol{u}$ to denote the regressors rather than the usual $\boldsymbol{x}$.

```
Correlation of Parameter Estimates:
      Vm
K 0.7651
```

So what advantages has VGAM over `nls()`? Here are a few:

1. VGAM allows one to write a specialized family function for a specific nonlinear regression model. Consequently it can be made more robust and efficient, as well as make use of more clever techniques for obtaining initial values etc, i.e., they are *self-starting*. Sometimes several different algorithms for obtaining starting values are implemented in he same family function.

2. `deriv()` only works for a restricted range of functions. It has not the capabilities of, e.g., MATLAB, MAPLE, etc., which one who programs VGAM family functions can exploit.

3. VGAM can fit models that `nls()` can't, e.g., when the response is multivariate. Here are some examples.

   (a) Galambos and Cornell (1966) considered the compartmental model

   $$\begin{pmatrix} y_{i1} \\ y_{i2} \end{pmatrix} = \begin{pmatrix} \theta_1 \exp(-\theta_2 x_i) + (1-\theta_1)\exp(-\theta_3 x_i) \\ 1 - (\theta_1+\theta_4)\exp(-\theta_2 x_i) + (\theta_1+\theta_4-1)\exp(-\theta_3 x_i) \end{pmatrix} + \begin{pmatrix} \varepsilon_{i1} \\ \varepsilon_{i2} \end{pmatrix},$$

   $E(\boldsymbol{\varepsilon}_i) = \mathbf{0}$, $\mathrm{Var}(\boldsymbol{\varepsilon}_i) = \boldsymbol{\Sigma}$ i.i.d., involving radioactive sulfate as a tracer.

   (b) Gallant (1975) considered

   $$\begin{pmatrix} y_{i1} \\ y_{i2} \end{pmatrix} = \begin{pmatrix} \theta_{11} + \theta_{12}\, x_{i11} + \theta_{13} \exp(\theta_{14}\, x_{i12}) \\ \theta_{21} + \theta_{22} \exp(\theta_{23}\, x_{i2}) \end{pmatrix} + \begin{pmatrix} \varepsilon_{i1} \\ \varepsilon_{i2} \end{pmatrix},$$

   $E(\boldsymbol{\varepsilon}_i) = \mathbf{0}$, $\mathrm{Var}(\boldsymbol{\varepsilon}_i) = \boldsymbol{\Sigma}$ i.i.d.

   (c) Bates and Watts (1988) (p.166) considered

   $$\begin{pmatrix} y_{i1} \\ y_{i2} \end{pmatrix} = \begin{pmatrix} e^{-(\theta_1+\theta_2)t} \\ \dfrac{\theta_1}{\theta_1+\theta_2}\left[1 - e^{-(\theta_1+\theta_2)t}\right] \end{pmatrix} + \begin{pmatrix} \varepsilon_{i1} \\ \varepsilon_{i2} \end{pmatrix},$$

   $E(\boldsymbol{\varepsilon}_i) = \mathbf{0}$, $\mathrm{Var}(\boldsymbol{\varepsilon}_i) = \boldsymbol{\Sigma}$ i.i.d., for some thermal isomerization data.

   Multivariate responses are discussed in Section 3.

2 Nonlinear Least Squares and the Gauss-Newton Algorithm and VGAMs

2.1 Univariate Responses

It is possible to coerce VGAM to implement the Gauss-Newton algorithm. This algorithm is well-known for nonlinear least squares problems, and some of its details now follows. We first look at univariate response.

We have

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}} \;=\; \sum_{i=1}^{n} w_i \left(y_i - f(u_i;\boldsymbol{\theta})\right) \frac{\partial f_i}{\partial \boldsymbol{\theta}}, \tag{4}$$

$$-E\left[\frac{\partial^2 \ell}{\partial \boldsymbol{\theta}\, \partial \boldsymbol{\theta}^T}\right] \;=\; \sum_{i=1}^{n} w_i \frac{\partial f_i}{\partial \boldsymbol{\theta}} \frac{\partial f_i}{\partial \boldsymbol{\theta}^T} \tag{5}$$

the latter due to $E(\varepsilon_i) = 0$. Letting

$$\mathbf{F}_{\bullet} \;=\; \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}^T} \;=\; \begin{pmatrix} \frac{\partial f_1}{\partial \boldsymbol{\theta}^T} \\ \vdots \\ \frac{\partial f_n}{\partial \boldsymbol{\theta}^T} \end{pmatrix},$$

$\mathbf{W} = \mathrm{Diag}(w_1, \ldots, w_n)$ and $\boldsymbol{f} = (f_1, \ldots, f_n)^T$, the Fisher-scoring algorithm for maximizing $\ell(\boldsymbol{\theta})$ is of the form

$$\boldsymbol{\theta}^{(a+1)} \;=\; \boldsymbol{\theta}^{(a)} + \left(\mathbf{F}_{\bullet}^{(a)T}\, \mathbf{W}\, \mathbf{F}_{\bullet}^{(a)}\right)^{-1} \mathbf{F}_{\bullet}^{(a)T}\, \mathbf{W} \left(\boldsymbol{y} - \boldsymbol{f}^{(a)}\right). \tag{6}$$

This is the Gauss-Newton algorithm. It can be seen that $\boldsymbol{\theta}^{(a+1)}$ is the solution to the least squares problem

$$\mathbf{F}_{\bullet}^{(a)}\, \boldsymbol{\theta}^{(a)} + \left(\boldsymbol{y} - \boldsymbol{f}^{(a)}\right) \;=\; \mathbf{F}_{\bullet}^{(a)}\, \boldsymbol{\theta}^{(a+1)} + \boldsymbol{\varepsilon}, \quad \mathrm{Var}(\boldsymbol{\varepsilon}) = \mathbf{W}^{-1}. \tag{7}$$

Thus Gauss-Newton involves regressing residuals on the design matrix $\mathbf{F}_{\bullet}^{(a)}$. Note that only first order derivatives are needed for this method.

For VGAM, extensions of the above are needed. One models $\boldsymbol{\theta}$ through $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^T, \ldots, \boldsymbol{\beta}_M^T)^T$ using

$$g_j(\boldsymbol{\theta}_j) \;=\; \eta_j \;=\; \boldsymbol{\beta}_j^T \boldsymbol{x}, \quad j = 1, \ldots, M,$$

where g_i are parameter link functions and $\boldsymbol{x}$ is often just a 1. That is, in general, each parameter may be modelled as a linear combination of some predictor variables.

One has $\mathbf{F}_{\bullet\, j}$ whose ith row is

$$\frac{\partial f_i}{\partial \boldsymbol{\beta}_j^T} \;=\; \frac{\partial f_i}{\partial \boldsymbol{\theta}_j} \frac{\partial \boldsymbol{\theta}_j}{\partial \eta_j} \frac{\partial \eta_j}{\partial \boldsymbol{\beta}_j^T} \;=\; \frac{\partial f_i}{\partial \boldsymbol{\theta}_j} \frac{\partial \boldsymbol{\theta}_j}{\partial \eta_j}\, \boldsymbol{x}_i^T.$$

Thus $\mathbf{F}_{\bullet\, j} = \mathbf{W}_j^*\, \mathbf{X}_j$ where $\mathbf{W}_j^*$ is a diagonal matrix and $\mathbf{X}_j$ is the usual design matrix for η_j. This allows for constraints–on–the–functions. Then

$$-E\left[\frac{\partial^2 \ell}{\partial \boldsymbol{\beta}_j\, \partial \boldsymbol{\beta}_k^T}\right] \;=\; \sum_{i=1}^{n} w_i \frac{\partial f_i}{\partial \boldsymbol{\beta}_j} \frac{\partial f_i}{\partial \boldsymbol{\beta}_k^T}$$

so

$$\boldsymbol{\beta}^{(a+1)} \;=\; \boldsymbol{\beta}^{(a)} + \left(\mathbf{F}_{\bullet}^{(a)T}\, \mathbf{W}\, \mathbf{F}_{\bullet}^{(a)}\right)^{-1} \mathbf{F}_{\bullet}^{(a)T}\, \mathbf{W} \left(\boldsymbol{y} - \boldsymbol{f}^{(a)}\right),$$

where

$$\begin{aligned} \mathbf{F}_{\bullet} &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\beta}^T} = (\mathbf{F}_{\bullet\,1} \cdots \mathbf{F}_{\bullet\,M}) \\ &= (\mathbf{W}_1^* \, \mathbf{W}_2^* \ldots \mathbf{W}_M^*) \, \mathrm{Diag}(\mathbf{X}_1, \ldots, \mathbf{X}_M). \end{aligned} \tag{8}$$

Thus one can obtain $\boldsymbol{\beta}^{(a+1)}$ by solving the least squares problem

$$\mathbf{F}_{\bullet}^{(a)} \, \boldsymbol{\beta}^{(a)} + \left(\boldsymbol{y} - \boldsymbol{f}^{(a)} \right) = \mathbf{F}_{\bullet}^{(a)} \, \boldsymbol{\beta}^{(a+1)} + \boldsymbol{\varepsilon}, \quad \mathrm{Var}(\boldsymbol{\varepsilon}) = \mathbf{W}^{-1}. \tag{9}$$

Substituting (8) into (9) gives

$$\begin{aligned} &(\mathbf{W}_1^* \, \mathbf{W}_2^* \ldots \mathbf{W}_M^*) \, \mathrm{Diag}(\mathbf{X}_1 \ldots \mathbf{X}_M) \, \boldsymbol{\beta}^{(a)} + \left(\boldsymbol{y} - \boldsymbol{f}^{(a)} \right) = \\ &\qquad (\mathbf{W}_1^* \, \mathbf{W}_2^* \ldots \mathbf{W}_M^*) \, \mathrm{Diag}(\mathbf{X}_1 \ldots \mathbf{X}_M) \, \boldsymbol{\beta}^{(a+1)} + \boldsymbol{\varepsilon}. \end{aligned} \tag{10}$$

This fits within the VGAM framework only for $M = 1$ because one can then premultiply both sides of (10) by $\mathbf{W}_1^{*-1}$. The result would then be to regress the adjusted dependent vector

$$\boldsymbol{z}_i = \mathbf{X}\boldsymbol{\beta}^{(a)} + \left(\mathbf{W}_1^{*-1} \, \mathbf{W}^{-1} \, \mathbf{W}_1^{*-1} \right) \left(\mathbf{W}_1^* \, \mathbf{W} \left(\boldsymbol{y} - \boldsymbol{f}^{(a)} \right) \right)$$

with weight $\mathbf{W}_1^* \, \mathbf{W} \, \mathbf{W}_1^*$ on $\mathbf{X}$. For $M > 2$ we employ a trick. Suppose $M = 2$. Then we can 'add' an additional equation by solving, say,

$$\begin{pmatrix} \mathbf{W}_1^* & \mathbf{W}_2^* \\ \mathbf{O} & \sqrt{\lambda}\mathbf{I}_n \end{pmatrix} \begin{pmatrix} \mathbf{X}_1 & \mathbf{O} \\ \mathbf{O} & \mathbf{X}_2 \end{pmatrix} \boldsymbol{\beta}^{(a)} + \mathbf{1}_2 \otimes \left(\boldsymbol{y} - \boldsymbol{f}^{(a)} \right) =$$

$$\begin{pmatrix} \mathbf{W}_1^* & \mathbf{W}_2^* \\ \mathbf{O} & \sqrt{\lambda}\mathbf{I}_n \end{pmatrix} \begin{pmatrix} \mathbf{X}_1 & \mathbf{O} \\ \mathbf{O} & \mathbf{X}_2 \end{pmatrix} \boldsymbol{\beta}^{(a+1)} + \boldsymbol{\varepsilon}, \quad \mathrm{Var}(\boldsymbol{\varepsilon}) = \mathbf{I}_2 \otimes \mathbf{W}^{-1}.$$

If we let $\lambda^{(a)} \to 0$ then this reduces to the ordinary Gauss–Newton equation we wish to solve. In practice we let $\lambda^{(a)}$ become very small but never identically zero.

For general M we replace the matrix $(\mathbf{W}_1^* \, \mathbf{W}_2^* \ldots \mathbf{W}_M^*)$ in (10) by 'adding' a $\sqrt{\lambda^{(a)}}\mathbf{I}_{n(M-1)}$ matrix below it. The result is a (square) invertible matrix, $\mathbf{W}_{**}$, say. Then, if we replace $\boldsymbol{y} - \boldsymbol{f}^{(a)}$ by $\mathbf{1}_M \otimes \left(\boldsymbol{y} - \boldsymbol{f}^{(a)} \right)$, we can premultiply (10) by $\mathbf{W}_{**}^{-1}$ to obtain

$$\mathrm{Diag}(\mathbf{X}_1, \ldots, \mathbf{X}_M). \begin{pmatrix} \boldsymbol{\beta}_1^{(a)} \\ . \\ . \\ \boldsymbol{\beta}_M^{(a)} \end{pmatrix} + \mathbf{W}_{**}^{-1} \left(\mathbf{1}_M \otimes (\boldsymbol{y} - \boldsymbol{f}^{(a)}) \right) = \boldsymbol{\eta}^{(a+1)} + \boldsymbol{\varepsilon}_{**}^{(a)},$$

$$\mathrm{Var}\left(\boldsymbol{\varepsilon}_{**}^{(a)} \right) = \left[\mathbf{W}_{**}^T \, (I_M \otimes \mathbf{W}) \, \mathbf{W}_{**} \right]^{-1}.$$

It is left as an exercise to the reader to verify that

$$\frac{\partial \ell_i}{\partial \boldsymbol{\eta}} = w_i \left(y_i - f_i^{(a)}\right) \begin{pmatrix} w_{i1}^* + \sqrt{\lambda^{(a)}} \\ w_{i2}^* + \sqrt{\lambda^{(a)}} \\ \vdots \\ w_{iM}^* + \sqrt{\lambda^{(a)}} \end{pmatrix}$$

(which is returned by `@deriv`) and

$$-E\left(\frac{\partial^2 \ell_i}{\partial \boldsymbol{\eta}\, \partial \boldsymbol{\eta}^T}\right)_{jk} \;=\; w_i \left(w_{ij}^* w_{ik}^* + \delta_{jk} \lambda^{(a)} I(j > 1)\right),$$

which is returned by `@weight`. That is, the outer-product of a $\boldsymbol{w}_i^*$ vector with $\lambda^{(a)}$ on all the diagonals except for the 1–1 element. All this can be simply programmed into a VGAM family function for a nonlinear regression model.

Here are some notes:

1. 'Adding' a $\sqrt{\lambda}\mathbf{I}$ to the $(\mathbf{W}_1^*, \ldots \mathbf{W}_M^*)$ matrix can be generalized to 'adding' *any* diagonal block matrices so that the result is invertible. The choice above was made for simplicity.

2. It appears that that choice produces similar results to the Levenberg–Marquardt method. It is, however, not the same.

3. $\lambda^{(a+1)} = \max(\lambda^{(a)}/$`divisor`$,$ `.Machine$double.eps`$)$ in VGAM family functions for nonlinear regression models. A simple method is to allow $\lambda^{(1)} = 0.01$, `divisor`$= 10$. On many machines, `.Machine$double.eps` is approximately 1×10^{-16}. Both $\lambda^{(1)}$ and `divisor` are arguments in a VGAM family function for nonlinear regression.

4. Exercise Another choice is to replace $(\mathbf{W}_1^* \ldots \mathbf{W}_M^*)$ in (10) by $\mathbf{1}_M \otimes (\mathbf{W}_1^* \ldots \mathbf{W}_M^*) + \lambda \mathbf{I}_{nM}$. Then, as an exercise, show that

$$\left(\frac{\partial \ell_i}{\partial \boldsymbol{\eta}_i}\right)_j \;=\; w_i \left(y_i - f_i^{(a)}\right) (M w_{ij}^* + \lambda)$$

and

$$\begin{aligned} (\mathbf{W}_i)_{ij} &= w_i \left[(w_{ij}^* + \lambda) w_{ik}^* + (w_{ik}^* + \lambda) w_{ij}^* + (M-2) w_{ij}^* w_{ik}^*\right], \;\; j \neq k, \; M \geq 2, \\ (\mathbf{W}_i)_{jj} &= w_i \left[(w_{ij}^* + \lambda)^2 + (M-1) {w_{ij}^*}^2\right]. \end{aligned}$$

This is more complicated than the first choice.

In general, the standard errors are

$$\mathrm{Var}(\hat{\boldsymbol{\beta}}^{(a)}) \;=\; \sigma^2 \left(\mathbf{F}_\bullet^T \, \mathbf{W} \, \mathbf{F}_\bullet\right)^{-1}$$

evaluated at $\hat{\boldsymbol{\theta}}$.

2.2 Levenberg-Marquardt Modification

Here are some miscellaneous notes about the Levenberg-Marquardt modification. The unmodified G-N algorithm is rarely used. To improve the chances of convergence in the basic Gauss-Newton algorithm, one popular method is the Levenberg-Marquardt modification. The Levenberg-Marquardt modification involves adding a $\lambda^{(a)}\mathbf{I}_d$ term to the inverse term in (6), viz.,

$$\boldsymbol{\theta}^{(a+1)} \;=\; \boldsymbol{\theta}^{(a)} + \left(\mathbf{F}_{\bullet}^{(a)T}\,\mathbf{W}\,\mathbf{F}_{\bullet}^{(a)} + \lambda^{(a)}\mathbf{I}\right)^{-1}\mathbf{F}_{\bullet}^{(a)T}\,\mathbf{W}\left(\boldsymbol{y} - \boldsymbol{f}^{(a)}\right),$$

where $\lambda^{(a)} > 0$ is a *ridge parameter* that may change with the iteration number a. Sometimes

$$\lambda^{(a+1)} \;=\; \lambda^{(a)}/\text{divisor}.$$

It is used to handle ill-conditioning due to the columns of $\partial f/\partial\boldsymbol{\theta}$ being collinear, or nearly so, bad initial values. A sequence that decreases to zero corresponds to the ordinary Gauss-Newton step, while a sequence that increases to infinity corresponds to steepest descents.

A VGAM family function with a `rpar` argument corresponds to the initial and `divisor` is λ.

The effect of adding $\lambda\mathbf{I}$ is to make the matrix whose inverse is required less singular, and it shortens the step $\boldsymbol{\theta}^{(a+1)} - \boldsymbol{\theta}^{(a)}$. The result interpolates between the G-N step ($\lambda^{(a)} \to 0$) and the steepest–descent direction ($\lambda^{(a)} \to \infty$). If $\lambda^{(a)}$ is too large for too many iterations the algorithm will take too many steps and thus make little progress.

For $\lambda^{(a)} > 0$, $(\mathbf{F}_{\bullet}^{(a)^T}\,\mathbf{W}\,\mathbf{F}_{\bullet}^{(a)} + \lambda^{(a)}\mathbf{I})$ is positive–definite, so the step will be a descent direction.

The success of the L-M modification depend, nearly on the appropriate choice of the diagonal matrix that is to be added.

3 Multivariate Responses

In this section we state the results for the multivariate extension of Section 2.1. The model is now

$$\boldsymbol{Y}_i \;=\; \boldsymbol{f}(\boldsymbol{u}_i;\boldsymbol{\theta}) + \boldsymbol{\varepsilon}_i, \qquad i = 1,\ldots,n, \quad \boldsymbol{\varepsilon}_i \sim N_d(\mathbf{0},\boldsymbol{\Sigma}_i) \;\text{ independently.}$$

Then

$$\ell(\boldsymbol{\theta}) \;=\; -\frac{1}{2}\sum_{i=1}^{n}\left(\boldsymbol{y}_i - \boldsymbol{f}(\boldsymbol{u}_i;\boldsymbol{\theta})\right)^T\boldsymbol{\Sigma}_i^{-1}\left(\boldsymbol{y}_i - \boldsymbol{f}(\boldsymbol{u}_i;\boldsymbol{\theta})\right),$$

the deviance $D = \sum_{i=1}^{n}\left(\boldsymbol{y}_i - \boldsymbol{f}(\boldsymbol{u}_i;\boldsymbol{\theta})\right)^T\boldsymbol{\Sigma}_i^{-1}\left(\boldsymbol{y}_i - \boldsymbol{f}(\boldsymbol{u}_i;\boldsymbol{\theta})\right)$, and Equations (4) and (5) generalize to

$$\begin{aligned}\frac{\partial\ell}{\partial\boldsymbol{\theta}} &= \sum_{i=1}^{n}\frac{\partial\boldsymbol{f}_i^T}{\partial\boldsymbol{\theta}}\boldsymbol{\Sigma}_i^{-1}\left(\boldsymbol{y}_i - \boldsymbol{f}(u_i;\boldsymbol{\theta})\right),\\ -E\left[\frac{\partial^2\ell}{\partial\boldsymbol{\theta}\,\partial\boldsymbol{\theta}^T}\right] &= \sum_{i=1}^{n}\frac{\partial\boldsymbol{f}_i^T}{\partial\boldsymbol{\theta}}\boldsymbol{\Sigma}_i^{-1}\frac{\partial\boldsymbol{f}_i}{\partial\boldsymbol{\theta}^T}.\end{aligned}$$

Letting $\mathbf{W}_i = \boldsymbol{\Sigma}_i^{-1}$,

$$\mathbf{F}_\bullet \;=\; \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}^T} \;=\; \begin{pmatrix} \frac{\partial \boldsymbol{f}_1}{\partial \boldsymbol{\theta}^T} \\ \vdots \\ \frac{\partial \boldsymbol{f}_n}{\partial \boldsymbol{\theta}^T} \end{pmatrix},$$

$\mathbf{W} = \mathrm{Diag}(\mathbf{W}_1, \ldots, \mathbf{W}_n)$, $\boldsymbol{y} = (\boldsymbol{y}_1^T, \ldots, \boldsymbol{y}_n^T)^T$ and $\boldsymbol{f} = (\boldsymbol{f}_1^T, \ldots, \boldsymbol{f}_n^T)^T$, then the Fisher-scoring algorithm for maximizing $\ell(\boldsymbol{\theta})$ is of the form

$$\boldsymbol{\theta}^{(a+1)} \;=\; \boldsymbol{\theta}^{(a)} + \left(\mathbf{F}_\bullet^{(a)T}\, \mathbf{W}\, \mathbf{F}_\bullet^{(a)}\right)^{-1} \mathbf{F}_\bullet^{(a)T}\, \mathbf{W} \left(\boldsymbol{y} - \boldsymbol{f}^{(a)}\right).$$

Usually, $\boldsymbol{\Sigma}$ is unknown, and has to be estimated. This may be done (Gallant, 1987) by

$$\hat{\boldsymbol{\Sigma}}^{(a)} \;=\; \frac{1}{n} \sum_{i=1}^{n} \left(\boldsymbol{y}_i - \boldsymbol{f}_i^{(a)}\right) \left(\boldsymbol{y}_i - \boldsymbol{f}_i^{(a)}\right)^T.$$

To save a long story, the algorithm is identical to (9) with the above extension of notation. VGAM uses a $n \times M$ matrix to represent $\boldsymbol{y} - \boldsymbol{f}^{(a)}$, and because $\boldsymbol{z}_i$ is computed as $\boldsymbol{\eta}_i + \mathbf{W}_i^{-1} \boldsymbol{d}_i$, `@deriv` premultiplies $\boldsymbol{d}_i = \boldsymbol{y}_i - \boldsymbol{f}_i^{(a)}$ by $\mathbf{W}_i$ in order to achieve the correct adjusted dependent variable.

4 Models

This section describes models currently implemented by VGAM that mainly Z. Guan has written. They are summarized in Table 1. Yet to do is to implement all the models listed in Ratkowsky (1990).

4.1 Michaelis-Menten function

This model is described on p.520 of Pinheiro and Bates (2000). The program works well.

For more about this model, see JASA 98: 679–686. H. Dette and S. Biedermann, 2003. Robust and efficient designs for the Michaelis-Menten model. This article contains the Fisher information matrix of a transformed model of the Michaelis-Menten model.

4.2 Shinozaki-Kira function

This model is described on p.362 of Seber and Wild (1989). The program works well.

4.3 Holliday function

This model is described on p.362 of Seber and Wild (1989). The program works well.

4.4 Bleasdale Simplified function

This model is described on p.362 of Seber and Wild (1989). The program works well if value of θ_3 is ≤ 2 apart from real value.

Table 1: *Summary of nonlinear regression models* $Y = f(u; \boldsymbol{\theta}) + \varepsilon$ *currently supported by* VGAM.

Name	$f(u; \boldsymbol{\theta})$	Range of y[a]	Range of $\boldsymbol{\theta}$[b]	VGAM family function	Working?	Initialization
Michaelis-Menten	$\frac{\theta_1 u}{\theta_2 + u}$			`micmen()`	Yes	Median of data
Shinozaki-Kira	$\frac{1}{\theta_1 + \theta_2 u}$			`skira()`	Yes	WLS[c]
Holliday	$(\theta_1 + \theta_2 u + \theta_3 u^2)^{-1}$			`holliday()`	Yes	WLS
Bleasdale Simplified	$(\theta_1 + \theta_2 u)^{-1/\theta_3}$			`bsimp()`	Yes	WLS
Farazdaghi-Harris	$(\theta_1 + \theta_2 u^{\theta_3})^{-1}$			`fharris()`	Yes	WLS
Bleasdale-Nelder	$\frac{1}{(\theta_1 + \theta_2 u^{\theta_4})^{\frac{1}{\theta_3}}}$			`bnelder()`	No	WLS
Nelder[1961]	$\frac{u}{\theta_1 + \theta_2 u + \theta_3 u^2}$			`nelder61()`	Yes	WLS
Gompertz	$\theta_1 \exp(-\exp(\theta_2(u - \theta_3)))$			`gomp()`	Yes	LSF
Monomolecular	$\theta_1(1 - \theta_2 \exp(-\theta_3 u))$			`monomo()`	Yes	LSF
Auto logistic	$\frac{\theta_1}{(1 + \theta_2 \exp(-\theta_3 u))}$			`autocata()`	Yes	WLS
Morgan-Mercer-Flodin	$\theta_1 - \frac{(\theta_1 - \theta_2)}{(1 + (\theta_3 u)^{\theta_4})}$			`mmf()`	No	WLS
Makeham's second	$\theta_1 + \theta_2 \theta_3^u$		$\theta_3 \in (0, 1)$	`makeham()`	Yes	LSF
Cook and Witmer	$\theta_1 u + \theta_2 (1 - u)$	$(-\infty, \infty)$	$(-\infty, \infty)$	`cwitmer()`	Yes	LSF
Weibull	$1 - \exp(-u^{\theta_1)}$	$(0, 1)$	$(-\infty, \infty)$	`weibull.nl()`	No	Undecided
Generalized logistic	$\frac{\exp(\theta_1 + u\theta_2)}{(1 + \exp(\theta_1 + u\theta_2))}$	$(0, 1)$	$(-\infty, \infty)$	`logi()`	No	Undecided
Asymp Reg thru O	$\theta_1(1 - \exp(-\exp(\theta_2)u))$	$(-\infty, \infty)$	$(-\infty, \infty)$	`asyreg0()`	Yes	WLS

[a] $0 < y < \infty$ and y real-valued is assumed unless otherwise stated.
[b] $0 < \theta_j < \infty$ and θ_j real-valued is assumed unless otherwise stated.
[c] WLS=weighted least squares; LSF=least squares fit.

4.5 Farazdaghi-Harris function

This model is described on p.362 of Seber and Wild (1989). The program works if initial value of θ_3 is close to real value, and real value of $\theta_3 \leq 2$.

4.6 Bleasdale-Nelder function

This model is described on p.362 of Seber and Wild (1989). The program has 4 parameters and does not converge.

4.7 Nelder[1961] function

This model is described on p.167 of Hunt (1982). The program works well.

4.8 Gompertz function

This model is described on p.331 of Seber and Wild (1989). The program works if initial value of $\theta_3 < 1.5$.

4.9 Monomolecular function

This model is described on p.123 of Hunt (1982). The program works.

4.10 Auto logistic (Autocatalytic) function

This model is described on p.330 of Seber and Wild (1989). The program works, but returned values are not quite accurate expecially θ_2.

4.11 Morgan-Mercer-Flodin function

This 4-parameter model is described on p.340 of Seber and Wild (1989). The program does not converge.

4.12 Makeham's second modification function

This model is described on p.10 of Seber and Wild (1989). The program works well.

4.13 Cook and Witmer's function

This model is described on p.136 of Seber and Wild (1989). The program works well.

4.14 Weibull function

This model is described on p.338 of Seber and Wild (1989). The function has only one parameter so cannot use WLS to get the initial value. Although manually allocate an initial value, the program doesn't work.

4.15 Generalized logistic function

This model is described on p.339 of Seber and Wild (1989). The program is not working as it does not converge. The initial values of this function are hard to obtain.

4.16 Asymptotic Regression function through the origin

This model is described on p.513 of Pinheiro and Bates (2000). This program works, but returned value of θ_2 is close to initial value rather than real value. There is no problem with returned value of θ_1.

5 Other Topics

5.1 Estimation of the Dispersion Parameter

VGAM family functions for nonlinear regression models have a component `$summary.dispersion = FALSE` which tells `summary.vglm()` and `summary.vgam()` that the general VGAM formula for estimating the dispersion parameter (involving the residual sum of squares) is not valid. Thus the dispersion parameter must be estimated by the family function itself during fitting. The code for this is found in `$last`, and

$$\hat{\sigma}^2 = \sum_{i=1}^{n} w_i (y_i - \hat{f}_i)^2 / (n - p^*)$$

where $p^* = \dim(\boldsymbol{\beta})$ is the number of parameters.

5.2 Richards Results

See the VGAM *User Manual* for more info.

5.3 `vgam()` and Nonlinear Regression

Can VGAM family functions for nonlinear regression models use `vgam()`? The answer is not yet! For example, one would want

```
vgam(y ~ s(x),     micmen, regressor=u)
vglm(y ~ bs(x, 4), micmen, regressor=u)
```

both fit

$$Y_i = \frac{f_1(x)\, u}{f_2(x) + u} + \varepsilon_i$$

where f_1 and f_2 are estimated by splines or a regression spline. Also,

```
vgam(y ~ s(x),     micmen, regressor=u, constraints=list(rbind(1,1)))
```

ought to fit constrain $f_1 = f_2$, i.e.,

$$Y_i = \frac{f(x)\, u}{f(x) + u} + \varepsilon_i$$

where f is estimated by a spline. But unfortunately, there is a bug in the program that needs to be fixed.

Table 2: *Enzyme velocity (y) and substrate concentration (u). Source: Watts (1981).*

u	y	u	y
2	0.0615	0.286	0.0129
2	0.0527	0.286	0.0183
0.667	0.0334	0.222	0.0083
0.667	0.0258	0.222	0.0169
0.40	0.0138	0.2	0.0129
0.40	0.0258	0.2	0.0087

5.4 Input

If $\boldsymbol{y}_i$ is multivariate then `y` should be a n-row matrix.

6 Tutorial Examples

In this section we illustrate fitting a nonlinear regression model or two.

6.1 The Michaelis-Menten model

Initial values and has been fitted to the enzyme data of Table 2. This model relates the rate of formation of product ($Y =$ enzyme velocity) in an enzyme-catalyzed chemical reaction to the concentration of substrate u. To keep things simple, an identity link will be used for the two parameters, and also they will be modelled as intercepts. One has

$$\frac{\partial f_i}{\partial \boldsymbol{\theta}_1} = \frac{u_i}{\boldsymbol{\theta}_2 + u_i}, \qquad \frac{\partial f_i}{\partial \boldsymbol{\theta}_2} = \frac{-\boldsymbol{\theta}_1\, u_i}{(\boldsymbol{\theta}_2 + u_i)^2}.$$

VGAM will fit the model to the data with

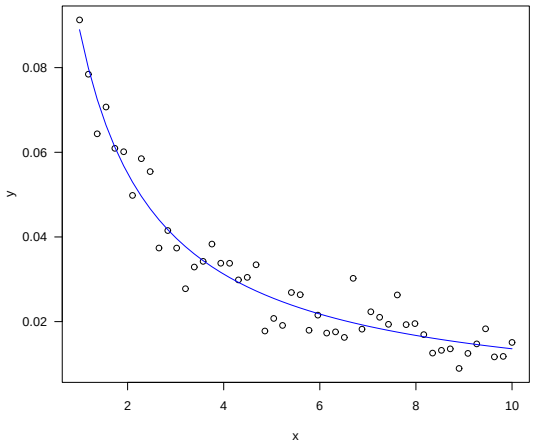

Figure 1: *Fitted `skira()` model to `sk.df`.*

```
data(enzyme)
fit = vglm(velocity ~ 1, micmen, regressor=enzyme$conc,
           data=enzyme, trace=TRUE, cri="coef")
summary(fit)
```

The true solution is $\hat{\boldsymbol{\theta}} = (0.10579, 1.7077)^T$.

6.2 `skira()` Model

We apply `skira()` to some artificial data. We generate a data set called `sk.df`, which has variables of `skx` and `sky`:

```
n = 50
xxx = seq(1,10,length = n)
sk.t1 = 4
sk.t2 = 7
set.seed(12)
err = rnorm(n, sd = 0.005)
sk.df = data.frame(skx = xxx, sky = 1/(sk.t1+sk.t2*xxx)+err)
plot(sk.df$skx, sk.df$sky, xlab="x", ylab="y")
```

Then type in the following commands in R to call function `skira()`:

```
> sk.fit = vglm(sky ~ 1, skira, data = sk.df,
              tr = T, crit = "c", regressor = sk.df$skx, eps=1e-8)
VGLM    linear loop 1: coefficients=c(4.958520, 6.543171)
VGLM    linear loop 2: coefficients=c(5.035916, 6.499976)
VGLM    linear loop 3: coefficients=c(5.065323, 6.483631)
VGLM    linear loop 4: coefficients=c(4.669438, 6.715736)
VGLM    linear loop 5: coefficients=c(4.361027, 6.899345)
VGLM    linear loop 6: coefficients=c(4.312516, 6.929454)
VGLM    linear loop 7: coefficients=c(4.310033, 6.931002)
VGLM    linear loop 8: coefficients=c(4.309851, 6.931114)
VGLM    linear loop 9: coefficients=c(4.309802, 6.931144)
VGLM    linear loop 10: coefficients=c(4.309786, 6.931154)
VGLM    linear loop 11: coefficients=c(4.309783, 6.931156)
VGLM    linear loop 12: coefficients=c(4.309783, 6.931156)
> lines(xxx, fitted(sk.fit), col=4)
```

This produces Figure 1. One can input initial values for any of the parameters, for example, using an initial $\theta_2 = 5$,

```
> vglm(sky ~ 1, skira(init2=7), data=sk.df, regressor=sk.df$skx, tr=T, cri="c")
VGLM    linear loop 1: coefficients=c(4.131284, 7.024765)
VGLM    linear loop 2: coefficients=c(4.148627, 7.030046)
VGLM    linear loop 3: coefficients=c(4.146189, 7.031542)
VGLM    linear loop 4: coefficients=c(4.234216, 6.977449)
VGLM    linear loop 5: coefficients=c(4.301759, 6.936048)
VGLM    linear loop 6: coefficients=c(4.310205, 6.930901)
VGLM    linear loop 7: coefficients=c(4.310010, 6.931017)
VGLM    linear loop 8: coefficients=c(4.309851, 6.931114)
VGLM    linear loop 9: coefficients=c(4.309802, 6.931144)
VGLM    linear loop 10: coefficients=c(4.309786, 6.931154)
```

gives the same result.

7 Discussion

The connection between IRLS for maximum likelihood estimation and the Gauss-Newton method for solving least squares problems is described in Wedderburn (1974).

Unmodified Gauss-Newton and Newton-Raphson algorithms are unpractical for computing least squares estimates. There are a range of techniques to make the algorithms more reliable, e.g., Levenberg-Marquardt methods, and step-halving. VGAM only implements the latter, and further investigation is needed on whether the L-M algorithm can be implemented within the VGAM framework.

S-PLUS has the functions `ms()` and `nls()` for minimizing a sum of squares and nonlinear least squares.

Acknowledgements

Ziming Guan was supported by a University of Auckland Science Faculty summer scholarship during 2001–2002.

Exercises

1. Fix up the VGAM family functions described in this document that don't work, and improve those that do work.

2. Fix up the bug described in Section 5.3.

References

Bates, D. M., Watts, D. G., 1988. Nonlinear Regression Analysis and Its Applications. Wiley, New York.

Chambers, J. M., Hastie, T. J. (Eds.), 1993. Statistical Models in S. Chapman & Hall, New York.

Galambos, J. T., Cornell, R. G., 1966. Mathematical models for the study of the metabolic pattern of sulfate. J. Lab. and Clinical Med. 60, 53–63.

Gallant, A. R., 1975. Seemingly unrelated nonlinear regressions. Journal of Econometrics 3, 35–50.

Gallant, A. R., 1987. Nonlinear Statistical Models. Wiley.

Hunt, R., 1982. Plant Growth Curves. Edward Arnold Limited, London.

Pinheiro, J. C., Bates, D. M., 2000. Statistics and Computing–Mixed-Effects Models in S and S-PLUS. Springer-Verlag, New York.

Ratkowsky, D. A., 1990. Handbook of Nonlinear Regression Models. Marcel Dekker, New York.

Seber, G. A. F., Wild, C. J., 1989. Nonlinear Regression. Wiley, New York.

Watts, D. G., 1981. An introduction to nonlinear least squares. In: Endrenyi, L. (Ed.), Kinetic Data Analysis: Design and Analysis of Enzyme and Pharmacokinetic Experiments. lenum Press, New York, pp. 1–24.

Wedderburn, R. W. M., 1974. Quasi-likelihood functions, generalized linear models, and the Gauss-Newton method. Biometrika 61, 439–447.