The VGAM Package for Categorical Data Analysis

Thomas W. Yee
University of Auckland

Abstract

Classical categorical regression models such as the multinomial logit and proportional odds models are shown to be readily handled by the vector generalized linear and additive model (VGLM/VGAM) framework. Additionally, there are natural extensions, such as reduced-rank VGLMs for dimension reduction, and allowing covariates that have values specific to each linear/additive predictor, e.g., for consumer choice modeling. This article describes some of the framework behind the **VGAM** R package, its usage and implementation details.

Keywords: categorical data analysis, Fisher scoring, iteratively reweighted least squares, multinomial distribution, nominal and ordinal polytomous responses, smoothing, vector generalized linear and additive models, **VGAM** R package.

1. Introduction

This is a **VGAM** vignette for categorical data analysis (CDA) based on **?**. Any subsequent features (especially non-backward compatible ones) will appear here.

The subject of CDA is concerned with analyses where the response is categorical regardless of whether the explanatory variables are continuous or categorical. It is a very frequent form of data. Over the years several CDA regression models for polytomous responses have become popular, e.g., those in Table 1. Not surprisingly, the models are interrelated: their foundation is the multinomial distribution and consequently they share similar and overlapping properties which modellers should know and exploit. Unfortunately, software has been slow to reflect their commonality and this makes analyses unnecessarily difficult for the practitioner on several fronts, e.g., using different functions/procedures to fit different models which does not aid the understanding of their connections.

This historical misfortune can be seen by considering R functions for CDA. From the Comprehensive R Archive Network (CRAN, http://CRAN.R-project.org/) there is `polr()` (in **MASS**; **?**) for a proportional odds model and `multinom()` (in **nnet**; **?**) for the multinomial logit model. However, both of these can be considered 'one-off' modeling functions rather than providing a unified offering for CDA. The function `lrm()` (in **rms**; **?**) has greater functionality: it can fit the proportional odds model (and the forward continuation ratio model upon preprocessing). Neither `polr()` or `lrm()` appear able to fit the nonproportional odds model. There are non-CRAN packages too, such as the modeling function `nordr()` (in **gnlm**; **?**), which can fit the proportional odds, continuation ratio and adjacent categories models; however it calls `nlm()` and the user must supply starting values. In general these R (**?**)

Quantity	Notation	**VGAM** family function
$P(Y = j+1)/P(Y = j)$	ζ_j	`acat()`
$P(Y = j)/P(Y = j+1)$	ζ_j^R	`acat(reverse = TRUE)`
$P(Y > j \mid Y \geq j)$	δ_j^*	`cratio()`
$P(Y < j \mid Y \leq j)$	δ_j^{*R}	`cratio(reverse = TRUE)`
$P(Y \leq j)$	γ_j	`cumulative()`
$P(Y \geq j)$	γ_j^R	`cumulative(reverse = TRUE)`
$\log\{P(Y = j)/P(Y = M+1)\}$		`multinomial()`
$P(Y = j \mid Y \geq j)$	δ_j	`sratio()`
$P(Y = j \mid Y \leq j)$	δ_j^R	`sratio(reverse = TRUE)`

Table 1: Quantities defined in **VGAM** for a categorical response Y taking values $1, \ldots, M+1$. Covariates $\boldsymbol{x}$ have been omitted for clarity. The LHS quantities are η_j or η_{j-1} for $j = 1, \ldots, M$ (not reversed) and $j = 2, \ldots, M+1$ (if reversed), respectively. All models are estimated by minimizing the deviance. All except for `multinomial()` are suited to ordinal Y.

modeling functions are not modular and often require preprocessing and sometimes are not self-starting. The implementations can be perceived as a smattering and piecemeal in nature. Consequently if the practitioner wishes to fit the models of Table 1 then there is a need to master several modeling functions from several packages each having different syntaxes etc. This is a hindrance to efficient CDA.

SAS (**?**) does not fare much better than R. Indeed, it could be considered as having an *excess* of options which bewilders the non-expert user; there is little coherent overriding structure. Its `proc logistic` handles the multinomial logit and proportional odds models, as well as exact logistic regression (see **?**, which is for Version 8 of SAS). The fact that the proportional odds model may be fitted by `proc logistic`, `proc genmod` and `proc probit` arguably leads to possible confusion rather than the making of connections, e.g., `genmod` is primarily for GLMs and the proportional odds model is not a GLM in the classical **?** sense. Also, `proc phreg` fits the multinomial logit model, and `proc catmod` with its WLS implementation adds to further potential confusion.

This article attempts to show how these deficiencies can be addressed by considering the vector generalized linear and additive model (VGLM/VGAM) framework, as implemented by the author's **VGAM** package for R. The main purpose of this paper is to demonstrate how the framework is very well suited to many 'classical' regression models for categorical responses, and to describe the implementation and usage of **VGAM** for such. To this end an outline of this article is as follows. Section 2 summarizes the basic VGLM/VGAM framework. Section 3 centers on functions for CDA in **VGAM**. Given an adequate framework, some natural extensions of Section 2 are described in Section 4. Users of **VGAM** can benefit from Section 5 which shows how the software reflects their common theory. Some examples are given in Section 6. Section 7 contains selected topics in statistial computing that are more relevant to programmers interested in the underlying code. Section 8 discusses several utilities and extensions needed for advanced CDA modeling, and the article concludes with a discussion. This document was run using **VGAM** 0.7-10 (**?**) under R 2.10.0.

Some general references for categorical data providing background to this article include **?**, **?**, **?**, **?**, **?**, **?** and **?**. An overview of models for ordinal responses is **?**, and a manual for fitting common models found in **?** to polytomous responses with various software is **?**. A package

for visualizing categorical data in R is **vcd** (**??**).

2. VGLM/VGAM overview

This section summarizes the VGLM/VGAM framework with a particular emphasis toward categorical models since the classes encapsulates many multivariate response models in, e.g., survival analysis, extreme value analysis, quantile and expectile regression, time series, bioassay data, nonlinear least-squares models, and scores of standard and nonstandard univariate and continuous distributions. The framework is partially summarized by Table 2. More general details about VGLMs and VGAMs can be found in **?** and **?** respectively. An informal and practical article connecting the general framework with the software is **?**.

2.1. VGLMs

Suppose the observed response $\boldsymbol{y}$ is a q-dimensional vector. VGLMs are defined as a model for which the conditional distribution of $\boldsymbol{Y}$ given explanatory $\boldsymbol{x}$ is of the form

$$f(\boldsymbol{y}|\boldsymbol{x}; \mathbf{B}, \phi) \;=\; h(\boldsymbol{y}, \eta_1, \ldots, \eta_M, \phi) \tag{1}$$

for some known function $h(\cdot)$, where $\mathbf{B} = (\boldsymbol{\beta}_1\, \boldsymbol{\beta}_2\, \cdots\, \boldsymbol{\beta}_M)$ is a $p \times M$ matrix of unknown regression coefficients, and the jth linear predictor is

$$\eta_j \;=\; \eta_j(\boldsymbol{x}) \;=\; \boldsymbol{\beta}_j^{\top} \boldsymbol{x} \;=\; \sum_{k=1}^{p} \beta_{(j)k}\, x_k, \quad\quad j = 1, \ldots, M. \tag{2}$$

Here $\boldsymbol{x} = (x_1, \ldots, x_p)^{\top}$ with $x_1 = 1$ if there is an intercept. Note that (2) means that *all* the parameters may be potentially modelled as functions of $\boldsymbol{x}$. It can be seen that VGLMs are like GLMs but allow for multiple linear predictors, and they encompass models outside the small confines of the exponential family. In (1) the quantity ϕ is an optional scaling parameter which is included for backward compatibility with common adjustments to overdispersion, e.g., with respect to GLMs.

In general there is no relationship between q and M: it depends specifically on the model or distribution to be fitted. However, for the 'classical' categorical regression models of Table 1 we have $M = q - 1$ since q is the number of levels the multi-category response Y has.

The η_j of VGLMs may be applied directly to parameters of a distribution rather than just to a mean for GLMs. A simple example is a univariate distribution with a location parameter ξ and a scale parameter $\sigma > 0$, where we may take $\eta_1 = \xi$ and $\eta_2 = \log\, \sigma$. In general, $\eta_j = g_j(\theta_j)$ for some parameter link function g_j and parameter θ_j. For example, the adjacent categories models in Table 1 are ratios of two probabilities, therefore a log link of ζ_j^R or ζ_j is the default. In **VGAM**, there are currently over a dozen links to choose from, of which any can be assigned to any parameter, ensuring maximum flexibility. Table 5 lists some of them.

VGLMs are estimated using iteratively reweighted least squares (IRLS) which is particularly suitable for categorical models (**?**). All models in this article have a log-likelihood

$$\ell \;=\; \sum_{i=1}^{n} w_i\, \ell_i \tag{3}$$

$\boldsymbol{\eta}$	Model	Modeling function	Reference
$\mathbf{B}_1^\top \boldsymbol{x}_1 + \mathbf{B}_2^\top \boldsymbol{x}_2$ $(= \mathbf{B}^\top \boldsymbol{x})$	VGLM	`vglm()`	?
$\mathbf{B}_1^\top \boldsymbol{x}_1 + \sum_{k=p_1+1}^{p_1+p_2} \mathbf{H}_k \, \boldsymbol{f}_k^*(x_k)$	VGAM	`vgam()`	?
$\mathbf{B}_1^\top \boldsymbol{x}_1 + \mathbf{A}\,\boldsymbol{\nu}$	RR-VGLM	`rrvglm()`	?
See ?	Goodman's RC	`grc()`	?

Table 2: Some of the package **VGAM** and its framework. The vector of latent variables $\boldsymbol{\nu} = \mathbf{C}^\top \boldsymbol{x}_2$ where $\boldsymbol{x}^\top = (\boldsymbol{x}_1^\top, \boldsymbol{x}_2^\top)$.

where the w_i are known positive prior weights. Let $\boldsymbol{x}_i$ denote the explanatory vector for the ith observation, for $i = 1, \ldots, n$. Then one can write

$$\begin{aligned} \boldsymbol{\eta}_i \;=\; \boldsymbol{\eta}(\boldsymbol{x}_i) \;&=\; \begin{pmatrix} \eta_1(\boldsymbol{x}_i) \\ \vdots \\ \eta_M(\boldsymbol{x}_i) \end{pmatrix} \;=\; \mathbf{B}^\top \boldsymbol{x}_i \;=\; \begin{pmatrix} \boldsymbol{\beta}_1^\top \boldsymbol{x}_i \\ \vdots \\ \boldsymbol{\beta}_M^\top \boldsymbol{x}_i \end{pmatrix} \\ &=\; \begin{pmatrix} \beta_{(1)1} & \cdots & \beta_{(1)p} \\ \vdots & & \\ \beta_{(M)1} & \cdots & \beta_{(M)p} \end{pmatrix} \boldsymbol{x}_i \;=\; \Big(\boldsymbol{\beta}_{(1)} \; \cdots \; \boldsymbol{\beta}_{(p)}\Big)\, \boldsymbol{x}_i. \end{aligned} \tag{4}$$

In IRLS, an adjusted dependent vector $\boldsymbol{z}_i = \boldsymbol{\eta}_i + \mathbf{W}_i^{-1} \boldsymbol{d}_i$ is regressed upon a large (VLM) model matrix, with $\boldsymbol{d}_i = w_i \, \partial \ell_i / \partial \boldsymbol{\eta}_i$. The working weights $\mathbf{W}_i$ here are $w_i \mathrm{Var}(\partial \ell_i / \partial \boldsymbol{\eta}_i)$ (which, under regularity conditions, is equal to $-w_i \, E[\partial^2 \ell_i / (\partial \boldsymbol{\eta}_i \, \partial \boldsymbol{\eta}_i^\top)]$), giving rise to the Fisher scoring algorithm.

Let $\mathbf{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^\top$ be the usual $n \times p$ (LM) model matrix obtained from the `formula` argument of `vglm()`. Given $\boldsymbol{z}_i$, $\mathbf{W}_i$ and $\mathbf{X}$ at the current IRLS iteration, a weighted multivariate regression is performed. To do this, a *vector linear model* (VLM) model matrix $\mathbf{X}_{\mathrm{VLM}}$ is formed from $\mathbf{X}$ and $\mathbf{H}_k$ (see Section 2.2). This is has nM rows, and if there are no constraints then Mp columns. Then $\left(\boldsymbol{z}_1^\top, \ldots, \boldsymbol{z}_n^\top\right)^\top$ is regressed upon $\mathbf{X}_{\mathrm{VLM}}$ with variance-covariance matrix $\mathrm{diag}(\mathbf{W}_1^{-1}, \ldots, \mathbf{W}_n^{-1})$. This system of linear equations is converted to one large WLS fit by premultiplication of the output of a Cholesky decomposition of the $\mathbf{W}_i$.

Fisher scoring usually has good numerical stability because the $\mathbf{W}_i$ are positive-definite over a larger region of parameter space than Newton-Raphson. For the categorical models in this article the expected information matrices are simpler than the observed information matrices, and are easily derived, therefore all the families in Table 1 implement Fisher scoring.

2.2. VGAMs and constraint matrices

VGAMs provide additive-model extensions to VGLMs, that is, (2) is generalized to

$$\eta_j(\boldsymbol{x}) \;=\; \beta_{(j)1} + \sum_{k=2}^{p} f_{(j)k}(x_k), \qquad j = 1, \ldots, M, \tag{5}$$

a sum of smooth functions of the individual covariates, just as with ordinary GAMs (**?**). The $\boldsymbol{f}_k = (f_{(1)k}(x_k), \ldots, f_{(M)k}(x_k))^\top$ are centered for uniqueness, and are estimated simultaneously using *vector smoothers*. VGAMs are thus a visual data-driven method that is well suited to exploring data, and they retain the simplicity of interpretation that GAMs possess.

An important concept, especially for CDA, is the idea of 'constraints-on-the functions'. In practice we often wish to constrain the effect of a covariate to be the same for some of the η_j and to have no effect for others. We shall see below that this constraints idea is important for several categorical models because of a popular parallelism assumption. As a specific example, for VGAMs we may wish to take

$$\begin{aligned} \eta_1 &= \beta_{(1)1} + f_{(1)2}(x_2) + f_{(1)3}(x_3), \\ \eta_2 &= \beta_{(2)1} + f_{(1)2}(x_2), \end{aligned}$$

so that $f_{(1)2} \equiv f_{(2)2}$ and $f_{(2)3} \equiv 0$. For VGAMs, we can represent these models using

$$\boldsymbol{\eta}(\boldsymbol{x}) \;=\; \boldsymbol{\beta}_{(1)} + \sum_{k=2}^{p} \boldsymbol{f}_k(x_k) \;=\; \mathbf{H}_1\, \boldsymbol{\beta}^*_{(1)} + \sum_{k=2}^{p} \mathbf{H}_k\, \boldsymbol{f}^*_k(x_k) \tag{6}$$

where $\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_p$ are known full-column rank *constraint matrices*, $\boldsymbol{f}^*_k$ is a vector containing a possibly reduced set of component functions and $\boldsymbol{\beta}^*_{(1)}$ is a vector of unknown intercepts. With no constraints at all, $\mathbf{H}_1 = \mathbf{H}_2 = \cdots = \mathbf{H}_p = \mathbf{I}_M$ and $\boldsymbol{\beta}^*_{(1)} = \boldsymbol{\beta}_{(1)}$. Like the $\boldsymbol{f}_k$, the $\boldsymbol{f}^*_k$ are centered for uniqueness. For VGLMs, the $\boldsymbol{f}_k$ are linear so that

$$\mathbf{B}^\top \;=\; \left(\mathbf{H}_1 \boldsymbol{\beta}^*_{(1)} \;\middle|\; \mathbf{H}_2 \boldsymbol{\beta}^*_{(2)} \;\middle|\; \cdots \;\middle|\; \mathbf{H}_p \boldsymbol{\beta}^*_{(p)} \right) \tag{7}$$

for some vectors $\boldsymbol{\beta}^*_{(1)}, \ldots, \boldsymbol{\beta}^*_{(p)}$.

The $\mathbf{X}_{\mathrm{VLM}}$ matrix is constructed from $\mathbf{X}$ and the $\mathbf{H}_k$ using Kronecker product operations. For example, with trivial constraints, $\mathbf{X}_{\mathrm{VLM}} = \mathbf{X} \otimes \mathbf{I}_M$. More generally,

$$\mathbf{X}_{\mathrm{VLM}} \;=\; \left((\mathbf{X}\, \boldsymbol{e}_1) \otimes \mathbf{H}_1 \;\middle|\; (\mathbf{X}\, \boldsymbol{e}_2) \otimes \mathbf{H}_2 \;\middle|\; \cdots \;\middle|\; (\mathbf{X}\, \boldsymbol{e}_p) \otimes \mathbf{H}_p \right) \tag{8}$$

($\boldsymbol{e}_k$ is a vector of zeros except for a one in the kth position) so that $\mathbf{X}_{\mathrm{VLM}}$ is $(nM) \times p^*$ where $p^* = \sum_{k=1}^{p} \mathrm{ncol}(\mathbf{H}_k)$ is the total number of columns of all the constraint matrices. Note that $\mathbf{X}_{\mathrm{VLM}}$ and $\mathbf{X}$ can be obtained by `model.matrix(vglmObject, type = "vlm")` and `model.matrix(vglmObject, type = "lm")` respectively. Equation 7 focusses on the rows of $\mathbf{B}$ whereas 4 is on the columns.

VGAMs are estimated by applying a modified vector backfitting algorithm (cf. **?**) to the $\boldsymbol{z}_i$.

2.3. Vector splines and penalized likelihood

If (6) is estimated using a vector spline (a natural extension of the cubic smoothing spline to vector responses) then it can be shown that the resulting solution maximizes a penalized likelihood; some details are sketched in **?**. In fact, knot selection for vector spline follows the same idea as O-splines (see **?**) in order to lower the computational cost.

The usage of `vgam()` with smoothing is very similar to `gam()` (**?**), e.g., to fit a nonparametric proportional odds model (cf. p.179 of **?**) to the pneumoconiosis data one could try

```
R> pneumo <- transform(pneumo, let = log(exposure.time))
R> fit <- vgam(cbind(normal, mild, severe) ~ s(let, df = 2),
+    cumulative(reverse = TRUE, parallel = TRUE), pneumo)
```

Here, setting `df = 1` means a linear fit so that `df = 2` affords a little nonlinearity.

3. VGAM family functions

This section summarizes and comments on the **VGAM** family functions of Table 1 for a categorical response variable taking values $Y = 1, 2, \ldots, M + 1$. In its most basic invokation, the usage entails a trivial change compared to `glm()`: use `vglm()` instead and assign the `family` argument a **VGAM** family function. The use of a **VGAM** family function to fit a specific model is far simpler than having a different modeling function for each model. Options specific to that model appear as arguments of that **VGAM** family function.

While writing `cratio()` it was found that various authors defined the quantity "continuation ratio" differently, therefore it became necessary to define a "stopping ratio". Table 1 defines these quantities for **VGAM**.

The multinomial logit model is usually described by choosing the first or last level of the factor to be baseline. **VGAM** chooses the last level (Table 1) by default, however that can be changed to any other level by use of the `refLevel` argument.

If the proportional odds assumption is inadequate then one strategy is to try use a different link function (see Section 5.2 for a selection). Another alternative is to add extra terms such as interaction terms into the linear predictor (available in the S language; **?**). Another is to fit the so-called *partial* proportional odds model (**?**) which **VGAM** can fit via constraint matrices.

In the terminology of **?**, `cumulative()` fits the class of *cumulative link models*, e.g., `cumulative(link = probit)` is a cumulative probit model. For `cumulative()` it was difficult to decide whether `parallel = TRUE` or `parallel = FALSE` should be the default. In fact, the latter is (for now?). Users need to set `cumulative(parallel = TRUE)` explicitly to fit a proportional odds model—hopefully this will alert them to the fact that they are making the proportional odds assumption and check its validity (**?**; e.g., through a deviance or likelihood ratio test). However the default means numerical problems can occur with far greater likelihood. Thus there is tension between the two options. As a compromise there is now a **VGAM** family function called `propodds(reverse = TRUE)` which is equivalent to `cumulative(parallel = TRUE, reverse = reverse, link = "logit")`.

By the way, note that arguments such as `parallel` can handle a slightly more complex syntax. A call such as `parallel = TRUE ~ x2 + x5 - 1` means the parallelism assumption is only applied to X_2 and X_5. This might be equivalent to something like `parallel = FALSE ~ x3 + x4`, i.e., to the remaining explanatory variables.

4. Other models

Given the VGLM/VGAM framework of Section 2 it is found that natural extensions are readily proposed in several directions. This section describes some such extensions.

4.1. Reduced-rank VGLMs

Consider a multinomial logit model where p and M are both large. A (not-too-convincing) example might be the data frame `vowel.test` in the package **ElemStatLearn** (see **?**). The vowel recognition data set involves $q = 11$ symbols produced from 8 speakers with 6 replications of each. The training data comprises 10 input features (not including the intercept) based on digitized utterances. A multinomial logit model fitted to these data would have $\widehat{\mathbf{B}}$ comprising of $p \times (q-1) = 110$ regression coefficients for $n = 8 \times 6 \times 11 = 528$ observations. The ratio of n to the number of parameters is small, and it would be good to introduce some parsimony into the model.

A simple and elegant solution is to represent $\widehat{\mathbf{B}}$ by its reduced-rank approximation. To do this, partition $\boldsymbol{x}$ into $(\boldsymbol{x}_1^\top, \boldsymbol{x}_2^\top)^\top$ and $\mathbf{B} = (\mathbf{B}_1^\top \; \mathbf{B}_2^\top)^\top$ so that the reduced-rank regression is applied to $\boldsymbol{x}_2$. In general, $\mathbf{B}$ is a dense matrix of full rank, i.e., rank $= \min(M, p)$, and since there are $M \times p$ regression coefficients to estimate this is 'too' large for some models and/or data sets. If we approximate $\mathbf{B}_2$ by a reduced-rank regression

$$\mathbf{B}_2 \;=\; \mathbf{C}\,\mathbf{A}^\top \tag{9}$$

and if the rank R is kept low then this can cut down the number of regression coefficients dramatically. If $R = 2$ then the results may be biplotted (`biplot()` in **VGAM**). Here, $\mathbf{C}$ and $\mathbf{A}$ are $p_2 \times R$ and $M \times R$ respectively, and usually they are 'thin'.

More generally, the class of *reduced-rank VGLMs* (RR-VGLMs) is simply a VGLM where $\mathbf{B}_2$ is expressed as a product of two thin estimated matrices (Table 2). Indeed, **?** show that RR-VGLMs are VGLMs with constraint matrices that are unknown and estimated. Computationally, this is done using an alternating method: in (9) estimate $\mathbf{A}$ given the current estimate of $\mathbf{C}$, and then estimate $\mathbf{C}$ given the current estimate of $\mathbf{A}$. This alternating algorithm is repeated until convergence within each IRLS iteration.

Incidentally, special cases of RR-VGLMs have appeared in the literature. For example, a RR-multinomial logit model, is known as the *stereotype* model (**?**). Another is **?**'s RC model (see Section 4.2) which is reduced-rank multivariate Poisson model. Note that the parallelism assumption of the proportional odds model (**?**) can be thought of as a type of reduced-rank regression where the constraint matrices are thin ($\mathbf{1}_M$, actually) and known.

The modeling function `rrvglm()` should work with any **VGAM** family function compatible with `vglm()`. Of course, its applicability should be restricted to models where a reduced-rank regression of $\mathbf{B}_2$ makes sense.

4.2. Goodman's $R \times C$ association model

Let $\mathbf{Y} = [(y_{ij})]$ be a $n \times M$ matrix of counts. Section 4.2 of **?** shows that Goodman's RC(R) association model (**?**) fits within the VGLM framework by setting up the appropriate indicator variables, structural zeros and constraint matrices. Goodman's model fits a reduced-rank type

model to $\mathbf{Y}$ by firstly assuming that Y_{ij} has a Poisson distribution, and that

$$\log \mu_{ij} \;=\; \mu + \alpha_i + \gamma_j + \sum_{k=1}^{R} a_{ik}\, c_{jk}, \quad i = 1, \ldots, n; \;\; j = 1, \ldots, M, \tag{10}$$

where $\mu_{ij} = E(Y_{ij})$ is the mean of the i-j cell, and the rank R satisfies $R < \min(n, M)$.
The modeling function `grc()` should work on any two-way table $\mathbf{Y}$ of counts generated by (10) provided the number of 0's is not too large. Its usage is quite simple, e.g., `grc(Ymatrix, Rank = 2)` fits a rank-2 model to a matrix of counts. By default a `Rank = 1` model is fitted.

4.3. Bradley-Terry models

Consider an experiment consists of n_{ij} judges who compare pairs of items T_i, $i = 1, \ldots, M+1$. They express their preferences between T_i and T_j. Let $N = \sum \sum_{i<j} n_{ij}$ be the total number of pairwise comparisons, and assume independence for ratings of the same pair by different judges and for ratings of different pairs by the same judge. Let π_i be the *worth* of item T_i,

$$P(T_i > T_j) \;=\; p_{i/ij} \;=\; \frac{\pi_i}{\pi_i + \pi_j}, \qquad i \neq j,$$

where "$T_i > T_j$" means i is preferred over j. Suppose that $\pi_i > 0$. Let Y_{ij} be the number of times that T_i is preferred over T_j in the n_{ij} comparisons of the pairs. Then $Y_{ij} \sim \text{Bin}(n_{ij}, p_{i/ij})$. This is a Bradley-Terry model (without ties), and the **VGAM** family function is `brat()`.
Maximum likelihood estimation of the parameters $\pi_1, \ldots, \pi_{M+1}$ involves maximizing

$$\prod_{i<j}^{M+1} \binom{n_{ij}}{y_{ij}} \left(\frac{\pi_i}{\pi_i + \pi_j}\right)^{y_{ij}} \left(\frac{\pi_j}{\pi_i + \pi_j}\right)^{n_{ij} - y_{ij}}.$$

By default, $\pi_{M+1} \equiv 1$ is used for identifiability, however, this can be changed very easily. Note that one can define linear predictors η_{ij} of the form

$$\text{logit}\left(\frac{\pi_i}{\pi_i + \pi_j}\right) \;=\; \log\left(\frac{\pi_i}{\pi_j}\right) \;=\; \lambda_i - \lambda_j. \tag{11}$$

The VGAM framework can handle the Bradley-Terry model only for intercept-only models; it has

$$\lambda_j \;=\; \eta_j \;=\; \log \pi_j = \beta_{(1)j}, \quad j = 1, \ldots, M. \tag{12}$$

As well as having many applications in the field of preferences, the Bradley-Terry model has many uses in modeling 'contests' between teams i and j, where only one of the teams can win in each contest (ties are not allowed under the classical model). The packaging function `Brat()` can be used to convert a square matrix into one that has more columns, to serve as input to `vglm()`. For example, for journal citation data where a citation of article B by article A is a win for article B and a loss for article A. On a specific data set,

```
R> journal <- c("Biometrika", "Comm.Statist", "JASA", "JRSS-B")
R> squaremat <- matrix(c(NA, 33, 320, 284, 730, NA, 813, 276,
+    498, 68, NA, 325, 221, 17, 142, NA), 4, 4)
R> dimnames(squaremat) <- list(winner = journal, loser = journal)
```

VGAM family function	Independent parameters
`ABO()`	p, q
`MNSs()`	m_S, m_s, n_S
`AB.Ab.aB.ab()`	p
`AB.Ab.aB.ab2()`	p
`AA.Aa.aa()`	p_A
`G1G2G3()`	p_1, p_2, f

Table 3: Some genetic models currently implemented and their unique parameters.

then `Brat(squaremat)` returns a 1×12 matrix.

Bradley-Terry model with ties

The **VGAM** family function `bratt()` implements a Bradley-Terry model with ties (no preference), e.g., where both T_i and T_j are equally good or bad. Here we assume

$$P(T_i > T_j) \;=\; \frac{\pi_i}{\pi_i + \pi_j + \pi_0}, \qquad P(T_i = T_j) \;=\; \frac{\pi_0}{\pi_i + \pi_j + \pi_0},$$

with $\pi_0 > 0$ as an extra parameter. It has

$$\boldsymbol{\eta} = (\log \pi_1, \ldots, \log \pi_{M-1}, \log \pi_0)^\top$$

by default, where there are M competitors and $\pi_M \equiv 1$. Like `brat()`, one can choose a different reference group and reference value.

Other R packages for the Bradley-Terry model include **BradleyTerry2** by H. Turner and D. Firth (with and without ties; **??**) and **prefmod** (**?**).

4.4. Genetic models

There are quite a number of population genetic models based on the multinomial distribution, e.g., **?**, **?**. Table 3 lists some **VGAM** family functions for such.

For example the ABO blood group system has two independent parameters p and q, say. Here, the blood groups A, B and O form six possible combinations (genotypes) consisting of AA, AO, BB, BO, AB, OO (see Table 4). A and B are dominant over bloodtype O. Let p, q and r be the probabilities for A, B and O respectively (so that $p + q + r = 1$) for a given population. The log-likelihood function is

$$\ell(p, q) \;=\; n_A \log(p^2 + 2pr) + n_B \log(q^2 + 2qr) + n_{AB} \log(2pq) + 2n_O \log(1 - p - q),$$

where $r = 1 - p - q$, $p \in (\,0, 1\,)$, $q \in (\,0, 1\,)$, $p + q < 1$. We let $\boldsymbol{\eta} = (g(p), g(r))^\top$ where g is the link function. Any g from Table 5 appropriate for a parameter $\theta \in (0, 1)$ will do.

A toy example where $p = p_A$ and $q = p_B$ is

```
R> abodat <- data.frame(A = 725, B = 258, AB = 72, O = 1073)
R> fit <- vglm(cbind(A, B, AB, O) ~ 1, ABO, abodat)
R> coef(fit, matrix = TRUE)
```

Genotype	AA	AO	BB	BO	AB	OO
Probability	p^2	$2pr$	q^2	$2qr$	$2pq$	r^2
Blood group	A	A	B	B	AB	O

Table 4: Probability table for the ABO blood group system. Note that p and q are the parameters and $r = 1 - p - q$.

```
            logit(pA) logit(pB)
(Intercept)     -1.33    -2.432

R> Coef(fit)

    pA     pB
0.2091 0.0808
```

The function `Coef()`, which applies only to intercept-only models, applies to $g_j(\theta_j) = \eta_j$ the inverse link function g_j^{-1} to $\widehat{\eta}_j$ to give $\widehat{\theta}_j$.

4.5. Three main distributions

? discusses three main distributions for categorical variables: binomial, multinomial, and Poisson (**?**). All these are well-represented in the **VGAM** package, accompanied by variant forms. For example, there is a **VGAM** family function named `mbinomial()` which implements a matched-binomial (suitable for matched case-control studies), Poisson ordination (useful in ecology for multi-species-environmental data), negative binomial families, positive and zero-altered and zero-inflated variants, and the bivariate odds ratio model (`binom2.or()`; see Section 6.5.6 of **?**). The latter has an `exchangeable` argument to allow for an exchangeable error structure:

$$\mathbf{H}_1 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{H}_k = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad k = 2, \ldots, p, \tag{13}$$

since, for data $(Y_1, Y_2, \boldsymbol{x})$, $\text{logit}\, P\Big(Y_j = 1 \Big| \boldsymbol{x}\Big) = \eta_j$ for $j = 1, 2$, and $\log \psi = \eta_3$ where ψ is the odds ratio, and so $\eta_1 = \eta_2$. Here, `binom2.or(zero = 3)` by default meaning ψ is modelled as an intercept-only (in general, `zero` may be assigned an integer vector such that the value j means $\eta_j = \beta_{(j)1}$, i.e., the jth linear/additive predictor is an intercept-only). See the online help for all of these models.

5. Some user-oriented topics

Making the most of **VGAM** requires an understanding of the general VGLM/VGAM framework described Section 2. In this section we connect elements of that framework with the software. Before doing so it is noted that a fitted **VGAM** categorical model has access to the usual generic functions, e.g., `coef()` for $\left(\widehat{\boldsymbol{\beta}}_{(1)}^{*T}, \ldots, \widehat{\boldsymbol{\beta}}_{(p)}^{*T}\right)^\top$ (see Equation 7), `constraints()`

Link function	$g(\theta)$	Range of θ
`cauchit()`	$\tan(\pi(\theta - \frac{1}{2}))$	$(0, 1)$
`cloglog()`	$\log_e\{-\log_e(1-\theta)\}$	$(0, 1)$
`fisherz()`	$\frac{1}{2}\log_e\{(1+\theta)/(1-\theta)\}$	$(-1, 1)$
`identity()`	θ	$(-\infty, \infty)$
`logc()`	$\log_e(1-\theta)$	$(-\infty, 1)$
`loge()`	$\log_e(\theta)$	$(0, \infty)$
`logit()`	$\log_e(\theta/(1-\theta))$	$(0, 1)$
`logoff()`	$\log_e(\theta + A)$	$(-A, \infty)$
`probit()`	$\Phi^{-1}(\theta)$	$(0, 1)$
`rhobit()`	$\log_e\{(1+\theta)/(1-\theta)\}$	$(-1, 1)$

Table 5: Some **VGAM** link functions pertinent to this article.

for $\mathbf{H}_k$, `deviance()` for $2\,(\ell_{\max} - \ell)$, `fitted()` for $\widehat{\boldsymbol{\mu}}_i$, `logLik()` for ℓ, `predict()` for $\widehat{\boldsymbol{\eta}}_i$, `print()`, `residuals(..., type = "response")` for $\boldsymbol{y}_i - \widehat{\boldsymbol{\mu}}_i$ etc., `summary()`, `vcov()` for $\widehat{\mathrm{Var}}(\widehat{\boldsymbol{\beta}})$, etc. The methods function for the extractor function `coef()` has an argument `matrix` which, when set `TRUE`, returns $\widehat{\mathbf{B}}$ (see Equation 1) as a $p \times M$ matrix, and this is particularly useful for confirming that a fit has made a parallelism assumption.

5.1. Common arguments

The structure of the unified framework given in Section 2 appears clearly through the pool of common arguments shared by the **VGAM** family functions in Table 1. In particular, `reverse` and `parallel` are prominent with CDA. These are merely convenient shortcuts for the argument `constraints`, which accepts a named list of constraint matrices $\mathbf{H}_k$. For example, setting `cumulative(parallel = TRUE)` would constrain the coefficients $\beta_{(j)k}$ in (2) to be equal for all $j = 1, \ldots, M$, each separately for $k = 2, \ldots, p$. That is, $\mathbf{H}_k = \mathbf{1}_M$. The argument `reverse` determines the 'direction' of the parameter or quantity.

Another argument not so much used with CDA is `zero`; this accepts a vector specifying which η_j is to be modelled as an intercept-only; assigning a `NULL` means none.

5.2. Link functions

Almost all **VGAM** family functions (one notable exception is `multinomial()`) allow, in theory, for any link function to be assigned to each η_j. This provides maximum capability. If so then there is an extra argument to pass in any known parameter associated with the link function. For example, `link = "logoff", earg = list(offset = 1)` signifies a log link with a unit offset: $\eta_j = \log(\theta_j + 1)$ for some parameter θ_j (> -1). The name `earg` stands for "extra argument". Table 5 lists some links relevant to categorical data. While the default gives a reasonable first choice, users are encouraged to try different links. For example, fitting a binary regression model (`binomialff()`) to the coal miners data set `coalminers` with respect to the response wheeze gives a nonsignificant regression coefficient for $\beta_{(1)3}$ with probit analysis but not with a logit link when $\eta = \beta_{(1)1} + \beta_{(1)2}\,\mathrm{age} + \beta_{(1)3}\,\mathrm{age}^2$. Developers and serious users are encouraged to write and use new link functions compatible with **VGAM**.

6. Examples

This section illustrates CDA modeling on three data sets in order to give a flavour of what is available in the package.

6.1. 2008 World Fly Fishing Championships

The World Fly Fishing Championships (WFFC) is a prestigious catch-and-release competition held annually. In 2008 it was held in New Zealand during the month of March. The data was released and appears in **VGAM** as the data frames `wffc`, `wffc.nc`, `wffc.indiv` and `wffc.teams`. Details about the competition are found in the online help, as well as `?`.

Briefly, we will model the abundance of fish caught during each three-hour session amongst the 90 or so competitors (from about 19 countries) who fished all their sessions. There were five sectors (locations) labelled I–V for the Whanganui River, Lake Otamangakau, Lake Rotoaira, Waihou River and Waimakariri River, respectively. The sessions were sequentially labelled 1–6 where odd and even numbers denote morning and afternoon respectively. There were three consecutive days of fishing during which each sector experienced a rest session.

`?` fitted Poisson and negative binomial regressions to the numbers caught at each competitor-session combination. The negative binomial regression had an intercept-only for its index parameter k and $\mathrm{Var}(Y) = \mu(1 + \mu/k)$. Both models had the log-linear relationship

$$\log \mu_{adsc} \quad = \quad \eta \quad = \quad \beta_{(1)1} + \alpha_s + \beta_a + \gamma_d + \delta_c. \tag{14}$$

where $\mu = E(Y)$ is the mean number caught, $\beta_{(1)1}$ is the intercept, α_s are the sector effects for $s = 1, \ldots, 5$ sectors, δ_c are the "competitor effects" for $c = 1, \ldots, 91$ competitors (8 competitors who did not fish all 5 sessions were excluded), β_a are the morning ($a = 1$) and afternoon ($a = 2$) effects, γ_d are the day effects for day $d = 1, 2, 3$. Recall for factors that the first level is baseline, e.g., $\alpha_1 = \beta_1 = 0$ etc. Not used here is $b = 1, \ldots, 19$ for which beat/boat was fished/used (e.g., fixed locations on the river). We will fit a proportional odds model with essentially the RHS of (14) as the linear predictor.

Here is a peek at the data frame used. Each row of `wffc.nc` is the number of captures by each sector-session-beat combination.

```
R> head(wffc.nc, 5)
```

```
  sector session beatboat numbers comid             iname country
1      4       1        1      20    42       BorisDzurek     SVK
2      4       1        2      20    26         JohnNishi     CAN
3      4       1        3       4    74    MichaelHeckler     WAL
4      4       1        4      32     3 TomasStarychfojtu     CZE
5      4       1        5      13    79      BoskoBarisic     CRO
```

We first process the data a little: create the regressor variables and restrict the analysis to anglers who fished all their sessions. Here, "`nc`" stands for numbers caught, and "`f`" stands for factor.

```
R> fnc <- transform(wffc.nc, finame = factor(iname), fsector = factor(sector),
+    fday = factor(ceiling(session/2)), mornaft = 1 - (session%%2),
```

```
+    fbeatboat = factor(beatboat))
R> fnc <- fnc[with(fnc, !is.element(comid, c(99, 72, 80, 93,
+    45, 71, 97, 78))), ]
R> fnc <- transform(fnc, ordnum = ifelse(numbers <= 2, "few",
+    ifelse(numbers <= 10, "more", "most")))
R> fnc$ordnum <- ordered(fnc$ordnum, levels = c("few", "more",
+    "most"))
```

The variable `ordnum` is ordinal with 3 levels. The cut-points chosen here were decided upon by manual inspection; they gave approximately the same numbers in each level:

```
R> with(fnc, table(ordnum))

ordnum
 few more most
 135  167  153
```

Now we are in a position to fit a proportional odds model to mimic (14).

```
R> fit.pom <- vglm(ordnum ~ fsector + mornaft + fday + finame,
+    family = cumulative(parallel = TRUE, reverse = TRUE),
+    data = fnc)
```

Here, we set `reverse = TRUE` so that the coefficients have the same direction as a logistic regression. It means that if a regression coefficient is positive then an increasing value of an explanatory variable is associated with an increasing value of the response. One could have used `family = propodds` instead.

Before interpreting some output let's check that the input was alright.

```
R> head(fit.pom@y, 3)

  few more most
1   0    0    1
2   0    0    1
3   0    1    0

R> colSums(fit.pom@y)

 few more most
 135  167  153
```

The checking indicates no problems with the input.

Now let's look at some output. Note that the Whanganui River, Mornings and Day 1 are the baseline levels of the factors. Also, the variable `mornaft` is 0 for morning and 1 for afternoons. Likewise, the factor `fday` has values `1`, `2` and `3`.

```
R> head(coef(fit.pom, matrix = TRUE), 10)
```

```
                       logit(P[Y>=2]) logit(P[Y>=3])
(Intercept)                   5.9734         1.6599
fsector2                     -5.0724        -5.0724
fsector3                     -5.8884        -5.8884
fsector4                     -0.1800        -0.1800
fsector5                     -0.1667        -0.1667
mornaft                      -0.4667        -0.4667
fday2                        -0.1558        -0.1558
fday3                        -1.1558        -1.1558
finameAlessandroSgrani       -1.9135        -1.9135
finameAndreSteenkamp         -1.9884        -1.9884
```

verifies the parallelism assumption. Standard errors and Wald statistics may be obtained by

```
R> head(coef(summary(fit.pom)), 10)
```

```
                          Value Std. Error  t value
(Intercept):1            5.9734     1.0766   5.5481
(Intercept):2            1.6599     0.9932   1.6712
fsector2                -5.0724     0.5069 -10.0065
fsector3                -5.8884     0.5437 -10.8295
fsector4                -0.1800     0.3585  -0.5020
fsector5                -0.1667     0.3539  -0.4709
mornaft                 -0.4667     0.2400  -1.9443
fday2                   -0.1558     0.3084  -0.5052
fday3                   -1.1558     0.2934  -3.9389
finameAlessandroSgrani  -1.9135     1.4805  -1.2925
```

Not surprisingly, these results agree with the Poisson and negative binomial regressions (reported in **?**). The most glaring qualitative results are as follows. We use the rough rule of thumb that if the absolute value of the t statistic is greater than 2 then it is 'statistically significant'.

- The two lakes were clearly less productive than the rivers. However, neither of the other two rivers were significantly different from the Whanganui River.
- There is a noticeable day effect: the second day is not significantly different from the opening day but it is for the third day. The decreasing values of the fitted coefficients show there is an increasing catch-reduction (fish depletion if it were catch-and-keep) as the competition progressed. Replacing `fday` by a variable `day` and entering that linearly gave a t statistic of -4.0: there is a significant decrease in catch over time.
- Mornings were more productive than afternoons. The p value for this would be close to 5%. This result is in line with the day effect: fishing often results in a 'hammering' effect over time on fish populations, especially in small streams. Since the morning and afternoon sessions were fixed at 9.00am–12.00pm and 2.30–5.30pm daily, there was only $2\frac{1}{2}$ hours for the fish to recover until the next angler arrived.

Let us check the proportional odds assumption with respect to the variable `mornaft`.

```
R> fit.ppom <- vglm(ordnum ~ fsector + mornaft + fday + finame,
+    cumulative(parallel = FALSE ~ 1 + mornaft, reverse = TRUE),
+    data = fnc)
R> head(coef(fit.ppom, matrix = TRUE), 8)

            logit(P[Y>=2]) logit(P[Y>=3])
(Intercept)         5.8950         1.7259
fsector2           -5.0588        -5.0588
fsector3           -5.9275        -5.9275
fsector4           -0.1505        -0.1505
fsector5           -0.1670        -0.1670
mornaft            -0.2849        -0.6326
fday2              -0.1617        -0.1617
fday3              -1.1821        -1.1821
```

As expected, all rows but `(Intercept)` and `mornaft` are identical due to the parallelism. Then

```
R> pchisq(deviance(fit.pom) - deviance(fit.ppom), df = df.residual(fit.pom) -
+    df.residual(fit.ppom), lower.tail = FALSE)

[1] 0.4401
```

gives a likelihood ratio test p value which is non-significant. Repeating the testing for each variable separately indicates that the parallelism assumption seems reasonable here except with `fday` (p value ≈ 0.012). For this model

```
R> fit2.ppom <- vglm(ordnum ~ fsector + mornaft + fday + finame,
+    family = cumulative(parallel = FALSE ~ 1 + fday, reverse = TRUE),
+    data = fnc)
R> head(coef(fit2.ppom, matrix = TRUE), 8)

            logit(P[Y>=2]) logit(P[Y>=3])
(Intercept)         6.0783         2.2622
fsector2           -5.5024        -5.5024
fsector3           -6.3192        -6.3192
fsector4           -0.2654        -0.2654
fsector5           -0.2903        -0.2903
mornaft            -0.5331        -0.5331
fday2               1.0450        -1.0586
fday3              -0.9694        -1.1951
```

Some miscellaneous output is as follows.

```
R> head(fitted(fit2.ppom), 3)
```

```
       few    more   most
1 0.002681 0.10615 0.8912
2 0.000691 0.02976 0.9695
3 0.024747 0.51072 0.4645
```

are the fitted probabilities $\widehat{P}(Y = j)$ which sum to unity for each row. The ith row of

```
R> head(predict(fit2.ppom), 3)

  logit(P[Y>=2]) logit(P[Y>=3])
1          5.919         2.1028
2          7.277         3.4606
3          3.674        -0.1421
```

is $\widehat{\boldsymbol{\eta}}(\boldsymbol{x}_i)^\top$. The dimensions of the LM and VLM design matrices are

```
R> dim(model.matrix(fit2.ppom, type = "lm"))

[1] 455  98

R> dim(model.matrix(fit2.ppom, type = "vlm"))

[1] 910 101
```

which shows the VLM matrix grows quickly with respect to M. Lastly,

```
R> constraints(fit2.ppom)[c(1, 2, 5, 6)]

$`(Intercept)`
     [,1] [,2]
[1,]    1    0
[2,]    0    1

$fsector2
     [,1]
[1,]    1
[2,]    1

$fsector5
     [,1]
[1,]    1
[2,]    1

$mornaft
     [,1]
[1,]    1
[2,]    1
```

shows some of the constraint matrices, $\mathbf{H}_1 = \mathbf{I}_2$ and $\mathbf{H}_2 = \mathbf{H}_5 = \mathbf{H}_6 = \mathbf{1}_2$ (see Equations 6–7).

6.2. Marital status data

We fit a nonparametric multinomial logit model to data collected from a self-administered questionnaire administered in a large New Zealand workforce observational study conducted during 1992–3. The data were augmented by a second study consisting of retirees. For homogeneity, this analysis is restricted to a subset of 6053 European males with no missing values. The ages ranged between 16 and 88 years. The data can be considered a reasonable representation of the white male New Zealand population in the early 1990s, and are detailed in **?** and **?**. We are interested in exploring how Y = marital status varies as a function of x_2 = age. The nominal response Y has four levels; in sorted order, they are divorced or separated, married or partnered, single and widower. We will write these levels as $Y = 1, 2, 3, 4$, respectively, and will choose the married/partnered (second level) as the reference group because the other levels emanate directly from it.

Suppose the data is in a data frame called `nzmarital` and looks like

```
R> head(nzmarital, 4)
```

```
  age ethnicity            mstatus
1  29  European             Single
2  55  European  Married/Partnered
3  44  European  Married/Partnered
4  53  European Divorced/Separated
```

```
R> summary(nzmarital)
```

```
      age            ethnicity                   mstatus
 Min.   :16.0   European  :6053   Divorced/Separated: 349
 1st Qu.:33.0   Maori     :   0   Married/Partnered :4778
 Median :43.0   Other     :   0   Single            : 811
 Mean   :43.7   Polynesian:   0   Widowed           : 115
 3rd Qu.:52.0
 Max.   :88.0
```

We fit the VGAM

```
R> fit.ms <- vgam(mstatus ~ s(age, df = 3), multinomial(refLevel = 2),
+    data = nzmarital)
```

Once again let's firstly check the input.

```
R> head(fit.ms@y, 4)
```

```
  Divorced/Separated Married/Partnered Single Widowed
1                  0                 0      1       0
2                  0                 1      0       0
3                  0                 1      0       0
4                  1                 0      0       0
```

```
R> colSums(fit.ms@y)

Divorced/Separated  Married/Partnered             Single
               349               4778                811
           Widowed
               115
```

This seems ok.

Now the estimated component functions $\widehat{f}_{(s)2}(x_2)$ may be plotted with

```
R> mycol <- c("red", "darkgreen", "blue")
R> par(mfrow = c(2, 2))
R> plot(fit.ms, se = TRUE, scale = 12, lcol = mycol, scol = mycol)
R> plot(fit.ms, se = TRUE, scale = 12, overlay = TRUE, llwd = 2,
+    lcol = mycol, scol = mycol)
```

to produce Figure 1. The `scale` argument is used here to ensure that the y-axes have a common scale—this makes comparisons between the component functions less susceptible to misinterpretation. The first three plots are the (centered) $\widehat{f}_{(s)2}(x_2)$ for η_1, η_2, η_3, where

$$\eta_s \;=\; \log(P(Y=t)/P(Y=2)) \;=\; \beta_{(s)1} + f_{(s)2}(x_2), \tag{15}$$

$(s,t) = (1,1),(2,3),(3,4)$, and x_2 is `age`. The last plot are the smooths overlaid to aid comparison.

It may be seen that the ± 2 standard error bands about the `Widowed` group is particularly wide at young ages because of a paucity of data, and likewise at old ages amongst the `Single`s. The $\widehat{f}_{(s)2}(x_2)$ appear as one would expect. The log relative risk of being single relative to being married/partnered drops sharply from ages 16 to 40. The fitted function for the `Widowed` group increases with `age` and looks reasonably linear. The $\widehat{f}_{(1)2}(x_2)$ suggests a possible maximum around 50 years old—this could indicate the greatest marital conflict occurs during the mid-life crisis years!

The methods function for `plot()` can also plot the derivatives of the smooths. The call

```
R> plot(fit.ms, deriv = 1, lcol = mycol, scale = 0.3)
```

results in Figure 2. Once again the y-axis scales are commensurate.

The derivative for the `Divorced/Separated` group appears linear so that a quadratic component function could be tried. Not surprisingly the `Single` group shows the greatest change; also, $\widehat{f}'_{(2)2}(x_2)$ is approximately linear till 50 and then flat—this suggests one could fit a piecewise quadratic function to model that component function up to 50 years. The `Widowed` group appears largely flat. We thus fit the parametric model

```
R> foo <- function(x, elbow = 50) poly(pmin(x, elbow), 2)
R> clist <- list(`(Intercept)` = diag(3), `poly(age, 2)` = rbind(1,
+    0, 0), `foo(age)` = rbind(0, 1, 0), age = rbind(0,
+    0, 1))
R> fit2.ms <- vglm(mstatus ~ poly(age, 2) + foo(age) + age,
+    family = multinomial(refLevel = 2), constraints = clist,
+    data = nzmarital)
```

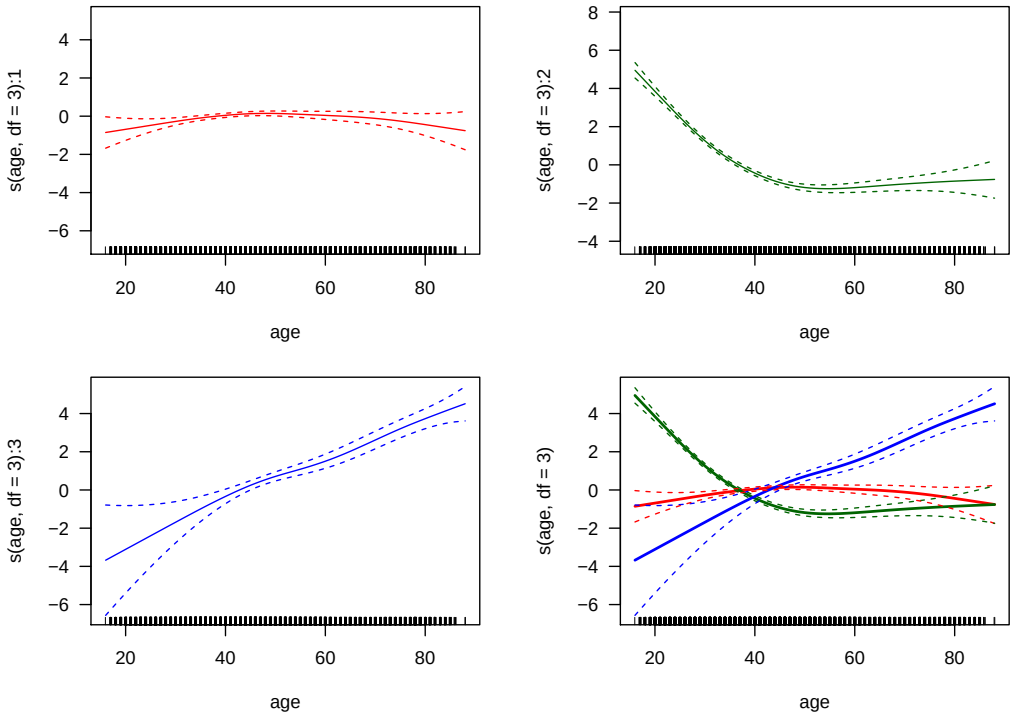

Figure 1: Fitted (and centered) component functions $\widehat{f}_{(s)2}(x_2)$ from the NZ marital status data (see Equation 15). The bottom RHS plot are the smooths overlaid.

Then

```
R> coef(fit2.ms, matrix = TRUE)
```

```
                log(mu[,1]/mu[,2]) log(mu[,3]/mu[,2]) log(mu[,4]/mu[,2])
(Intercept)                 -2.692             -2.469            -9.5048
poly(age, 2)1                7.678              0.000             0.0000
poly(age, 2)2              -19.566              0.000             0.0000
foo(age)1                    0.000           -103.820             0.0000
foo(age)2                    0.000             36.198             0.0000
age                          0.000              0.000             0.1025
```

confirms that one term was used for each component function. The plots from

```
R> par(mfrow = c(2, 2))
R> plotvgam(fit2.ms, se = TRUE, scale = 12, lcol = mycol[1],
+    scol = mycol[1], which.term = 1)
R> plotvgam(fit2.ms, se = TRUE, scale = 12, lcol = mycol[2],
+    scol = mycol[2], which.term = 2)
R> plotvgam(fit2.ms, se = TRUE, scale = 12, lcol = mycol[3],
+    scol = mycol[3], which.term = 3)
```

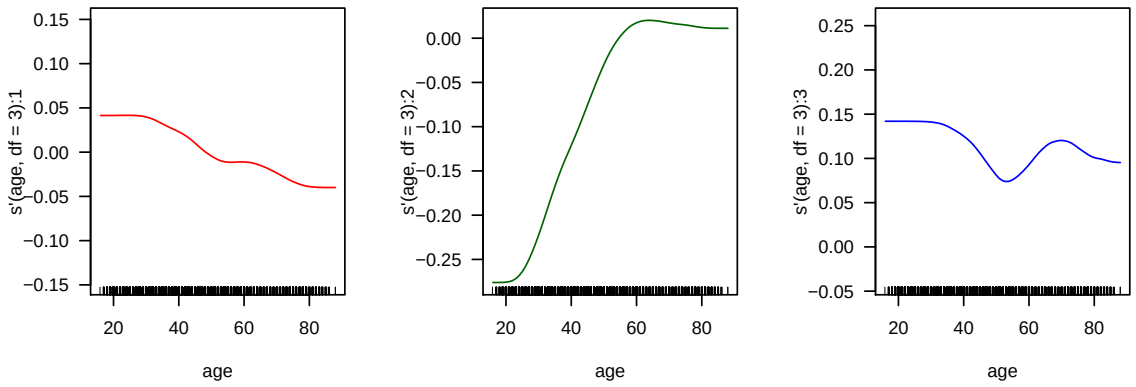

Figure 2: Estimated first derivatives of the component functions, $\widehat{f}'_{(s)2}(x_2)$, from the NZ marital status data (see Equation 15).

are given in Figure 3 and appear like Figure 1.

It is possible to perform very crude inference based on heuristic theory of a deviance test:

```
R> deviance(fit.ms) - deviance(fit2.ms)

[1] 0.06733
```

is small, so it seems the parametric model is quite reasonable against the original nonparametric model. Specifically, the difference in the number of 'parameters' is approximately

```
R> (dfdiff <- df.residual(fit2.ms) - df.residual(fit.ms))

[1] 3.994
```

which gives an approximate p value of

```
R> 1 - pchisq(deviance(fit.ms) - deviance(fit2.ms), df = dfdiff)

[1] 0.9994
```

Thus `fit2.ms` appears quite reasonable.

The estimated probabilities of the original fit can be plotted against `age` using

```
R> ooo <- with(nzmarital, order(age))
R> with(nzmarital, matplot(age[ooo], fitted(fit.ms)[ooo, ],
+    type = "l", las = 1, lwd = 2, ylim = 0:1, ylab = "Fitted probabilities",
+    xlab = "Age", col = c(mycol[1], "black", mycol[-1])))
R> legend(x = 52.5, y = 0.62, col = c(mycol[1], "black", mycol[-1]),
+    lty = 1:4, legend = colnames(fit.ms@y), lwd = 2)
R> abline(v = seq(10, 90, by = 5), h = seq(0, 1, by = 0.1),
+    col = "gray", lty = "dashed")
```

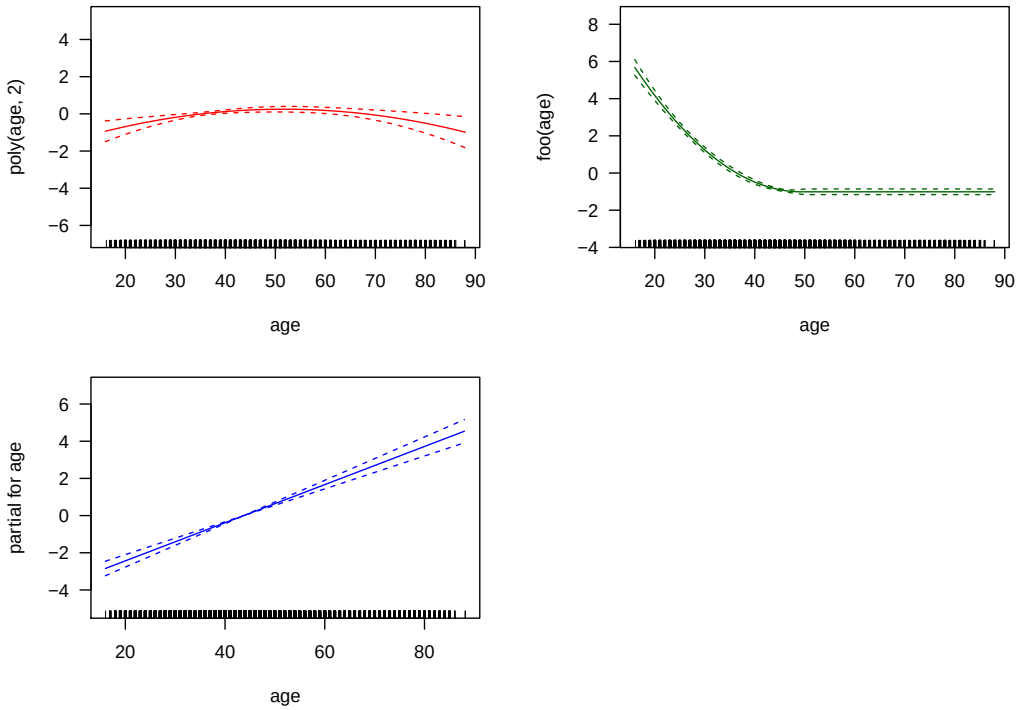

Figure 3: Parametric version of `fit.ms`: `fit2.ms`. The component functions are now quadratic, piecewise quadratic/zero, or linear.

which gives Figure 4. This shows that between 80–90% of NZ white males aged between their early 30s to mid-70s were married/partnered. The proportion widowed started to rise steeply from 70 years onwards but remained below 0.5 since males die younger than females on average.

6.3. Stereotype model

We reproduce some of the analyses of **?** regarding the progress of 101 patients with back pain using the data frame `backPain` from **gnm** (**??**). The three prognostic variables are length of previous attack ($x_1 = 1, 2$), pain change ($x_2 = 1, 2, 3$) and lordosis ($x_3 = 1, 2$). Like him, we treat these as numerical and standardize and negate them. The output

```
R> head(backPain, 4)
```

```
  x1 x2 x3                pain
1  1  1  1                same
2  1  1  1 marked.improvement
3  1  1  1    complete.relief
4  1  2  1                same
```

```
R> summary(backPain)
```

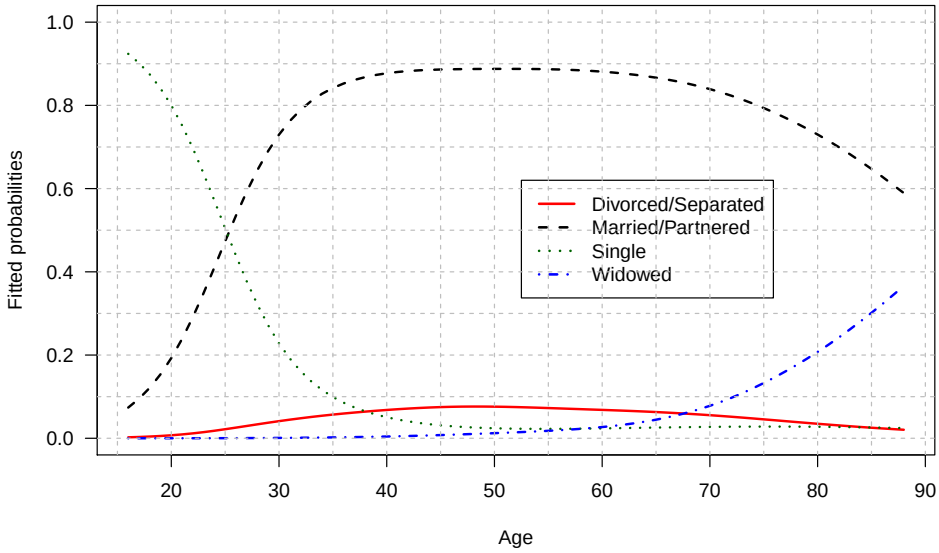

Figure 4: Fitted probabilities for each class for the NZ male European marital status data (from Equation 15).

```
      x1              x2              x3                          pain
 Min.   :1.00   Min.   :1.00   Min.   :1.00   worse                 : 5
 1st Qu.:1.00   1st Qu.:2.00   1st Qu.:1.00   same                  :14
 Median :2.00   Median :2.00   Median :1.00   slight.improvement    :18
 Mean   :1.61   Mean   :2.07   Mean   :1.37   moderate.improvement:20
 3rd Qu.:2.00   3rd Qu.:3.00   3rd Qu.:2.00   marked.improvement    :28
 Max.   :2.00   Max.   :3.00   Max.   :2.00   complete.relief       :16
```

```
R> backPain <- transform(backPain, sx1 = -scale(x1), sx2 = -scale(x2),
+    sx3 = -scale(x3))
```

displays the six ordered categories. Now a rank-1 stereotype model can be fitted with

```
R> bp.rrmlm1 <- rrvglm(pain ~ sx1 + sx2 + sx3, multinomial,
+    backPain)
```

Then

```
R> Coef(bp.rrmlm1)

A matrix:
                        lv
log(mu[,1]/mu[,6]) 1.0000
log(mu[,2]/mu[,6]) 0.3094
log(mu[,3]/mu[,6]) 0.3467
log(mu[,4]/mu[,6]) 0.5099
```

```
log(mu[,5]/mu[,6]) 0.1415

C matrix:
        lv
sx1 -2.628
sx2 -2.146
sx3 -1.314

B1 matrix:
            log(mu[,1]/mu[,6]) log(mu[,2]/mu[,6]) log(mu[,3]/mu[,6])
(Intercept)             -2.914             0.2945             0.5198
            log(mu[,4]/mu[,6]) log(mu[,5]/mu[,6])
(Intercept)             0.3511             0.9026
```

are the fitted $\mathbf{A}$, $\mathbf{C}$ and $\mathbf{B}_1$ (see Equation 9) and Table 2) which agrees with his Table 6. Here, what is known as "corner constraints" is used ($(1,1)$ element of $\mathbf{A} \equiv 1$), and only the intercepts are not subject to any reduced-rank regression by default. The maximized log-likelihood from `logLik(bp.rrmlm1)` is -151.55. The standard errors of each parameter can be obtained by `summary(bp.rrmlm1)`. The negative elements of $\widehat{\mathbf{C}}$ imply the latent variable $\widehat{\nu}$ decreases in value with increasing `sx1`, `sx2` and `sx3`. The elements of $\widehat{\mathbf{A}}$ tend to decrease so it suggests patients get worse as ν increases, i.e., get better as `sx1`, `sx2` and `sx3` increase.

A rank-2 model fitted *with a different normalization*

```
R> bp.rrmlm2 <- rrvglm(pain ~ sx1 + sx2 + sx3, multinomial,
+    backPain, Rank = 2, Corner = FALSE, Uncor = TRUE)
```

produces uncorrelated $\widehat{\boldsymbol{\nu}}_i = \widehat{\mathbf{C}}^\top \boldsymbol{x}_{2i}$. In fact `var(lv(bp.rrmlm2))` equals $\mathbf{I}_2$ so that the latent variables are also scaled to have unit variance. The fit was biplotted (rows of $\widehat{\mathbf{C}}$ plotted as arrow; rows of $\widehat{\mathbf{A}}$ plotted as labels) using

```
R> biplot(bp.rrmlm2, Acol = "blue", Ccol = "darkgreen", scores = TRUE,
+    xlim = c(-4.5, 2.2), ylim = c(-2.2, 2.2), chull = TRUE,
+    clty = 2, ccol = "blue")
```

to give Figure 5. It is interpreted via inner products due to (9). The different normalization means that the interpretation of ν_1 and ν_2 has changed, e.g., increasing `sx1`, `sx2` and `sx3` results in increasing $\widehat{\nu}_1$ and patients improve more. Many of the latent variable points $\widehat{\boldsymbol{\nu}}_i$ are coincidental due to discrete nature of the $\boldsymbol{x}_i$. The rows of $\widehat{\mathbf{A}}$ are centered on the blue labels (rather cluttered unfortunately) and do not seem to vary much as a function of ν_2. In fact this is confirmed by `?` who showed a rank-1 model is to be preferred.

This example demonstrates the ability to obtain a low dimensional view of higher dimensional data. The package's website has additional documentation including more detailed Goodman's RC and stereotype examples.

7. Some implementation details

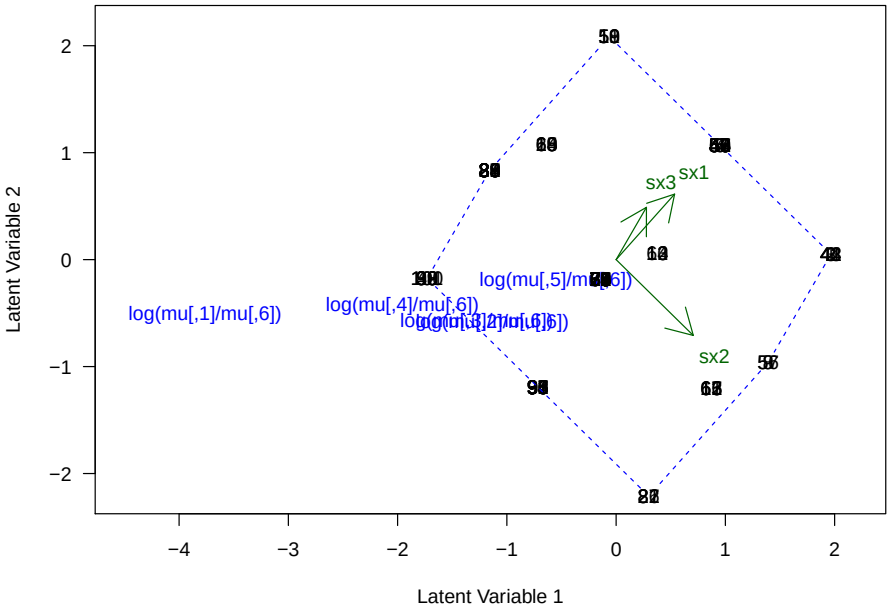

Figure 5: Biplot of a rank-2 reduced-rank multinomial logit (stereotype) model fitted to the back pain data. A convex hull surrounds the latent variable scores $\widehat{\boldsymbol{\nu}}_i$ (whose observation numbers are obscured because of their discrete nature). The position of the jth row of $\widehat{\mathbf{A}}$ is the center of the label "`log(mu[,j])/mu[,6]`".

This section describes some implementation details of **VGAM** which will be more of interest to the developer than to the casual user.

7.1. Common code

It is good programming practice to write reusable code where possible. All the **VGAM** family functions in Table 1 process the response in the same way because the same segment of code is executed. This offers a degree of uniformity in terms of how input is handled, and also for software maintenance (**?** enumerates good programming techniques and references). As well, the default initial values are computed in the same manner based on sample proportions of each level of Y.

7.2. Matrix-band format of `wz`

The working weight matrices $\mathbf{W}_i$ may become large for categorical regression models. In general, we have to evaluate the $\mathbf{W}_i$ for $i = 1, \ldots, n$, and naively, this could be held in an `array` of dimension `c(M, M, n)`. However, since the $\mathbf{W}_i$ are symmetric positive-definite it suffices to only store the upper or lower half of the matrix.

The variable `wz` in `vglm.fit()` stores the working weight matrices $\mathbf{W}_i$ in a special format called the *matrix-band* format. This format comprises a $n \times M^*$ matrix where

$$M^* \;=\; \sum_{i=1}^{hbw} (M - i + 1) \;=\; \frac{1}{2}\, hbw\, (2\,M - hbw + 1)$$

is the number of columns. Here, *hbw* refers to the *half-bandwidth* of the matrix, which is an integer between 1 and M inclusive. A diagonal matrix has unit half-bandwidth, a tridiagonal matrix has half-bandwidth 2, etc.

Suppose $M = 4$. Then `wz` will have up to $M^* = 10$ columns enumerating the unique elements of $\mathbf{W}_i$ as follows:

$$\mathbf{W}_i = \begin{pmatrix} 1 & 5 & 8 & 10 \\ & 2 & 6 & 9 \\ & & 3 & 7 \\ & & & 4 \end{pmatrix}. \tag{16}$$

That is, the order is firstly the diagonal, then the band above that, followed by the second band above the diagonal etc. Why is such a format adopted? For this example, if $\mathbf{W}_i$ is diagonal then only the first 4 columns of `wz` are needed. If $\mathbf{W}_i$ is tridiagonal then only the first 7 columns of `wz` are needed. If $\mathbf{W}_i$ *is* banded then `wz` needs not have all $\frac{1}{2}M(M+1)$ columns; only M^* columns suffice, and the rest of the elements of $\mathbf{W}_i$ are implicitly zero. As well as reducing the size of `wz` itself in most cases, the matrix-band format often makes the computation of `wz` very simple and efficient. Furthermore, a Cholesky decomposition of a banded matrix will be banded. A final reason is that sometimes we want to input $\mathbf{W}_i$ into **VGAM**: if `wz` is $M \times M \times n$ then `vglm(..., weights = wz)` will result in an error whereas it will work if `wz` is an $n \times M^*$ matrix.

To facilitate the use of the matrix-band format, a few auxiliary functions have been written. In particular, there is `iam()` which gives the indices for an array-to-matrix. In the 4×4 example above,

```
R> iam(NA, NA, M = 4, both = TRUE, diag = TRUE)


$row.index
 [1] 1 2 3 4 1 2 3 1 2 1

$col.index
 [1] 1 2 3 4 2 3 4 3 4 4
```

returns the indices for the respective array coordinates for successive columns of matrix-band format (see Equation 16). If `diag = FALSE` then the first 4 elements in each vector are omitted. Note that the first two arguments of `iam()` are not used here and have been assigned `NA`s for simplicity. For its use on the multinomial logit model, where $(\mathbf{W}_i)_{jj} = w_i\,\mu_{ij}(1-\mu_{ij})$, $j = 1, \ldots, M$, and $(\mathbf{W}_i)_{jk} = -w_i\,\mu_{ij}\mu_{ik}$, $j \neq k$, this can be programmed succinctly like

```
wz <- mu[, 1:M] * (1 - mu[, 1:M])
if (M > 1) {
    index <- iam(NA, NA, M = M, both = TRUE, diag = FALSE)
    wz <- cbind(wz, -mu[, index$row] * mu[, index$col])
}
wz <- w * wz
```

(the actual code is slightly more complicated). In general, **VGAM** family functions can be remarkably compact, e.g., `acat()`, `cratio()` and `multinomial()` are all less than 120 lines of code each.

8. Extensions and utilities

This section describes some useful utilities/extensions of the above.

8.1. Marginal effects

Models such as the multinomial logit and cumulative link models model the posterior probability $p_j = P(Y = j|\boldsymbol{x})$ directly. In some applications, knowing the derivative of p_j with respect to some of the x_k is useful; in fact, often just knowing the sign is important. The function `margeff()` computes the derivatives and returns them as a $p \times (M+1) \times n$ array. For the multinomial logit model it is easy to show

$$\frac{\partial\, p_j(\boldsymbol{x}_i)}{\partial\, \boldsymbol{x}_i} \;=\; p_j(\boldsymbol{x}_i) \left\{ \boldsymbol{\beta}_j - \sum_{s=1}^{M+1} p_s(\boldsymbol{x}_i)\, \boldsymbol{\beta}_s \right\}, \tag{17}$$

while for `cumulative(reverse = FALSE)` we have $p_j = \gamma_j - \gamma_{j-1} = h(\eta_j) - h(\eta_{j-1})$ where $h = g^{-1}$ is the inverse of the link function (cf. Table 1) so that

$$\frac{\partial\, p_j(\boldsymbol{x})}{\partial\, \boldsymbol{x}} \;=\; h'(\eta_j)\, \boldsymbol{\beta}_j - h'(\eta_{j-1})\, \boldsymbol{\beta}_{j-1}. \tag{18}$$

The function `margeff()` returns an array with these derivatives and should handle any value of `reverse` and `parallel`.

8.2. The `xij` argument

There are many models, including those for categorical data, where the value of an explanatory variable x_k differs depending on which linear/additive predictor η_j. Here is a well-known example from consumer choice modeling. Suppose an econometrician is interested in peoples' choice of transport for travelling to work and that there are four choices: $Y = 1$ for "bus", $Y = 2$ "train", $Y = 3$ "car" and $Y = 4$ means "walking". Assume that people only choose one means to go to work. Suppose there are three covariates: $X_2 =$ cost, $X_3 =$ journey time, and $X_4 =$ distance. Of the covariates only X_4 (and the intercept X_1) is the same for all transport choices; the cost and journey time differ according to the means chosen. Suppose a random sample of n people is collected from some population, and that each person has access to all these transport modes. For such data, a natural regression model would be a multinomial logit model with $M = 3$: for $j = 1, \ldots, M$, we have $\eta_j =$

$$\log \frac{P(Y = j)}{P(Y = M+1)} \;=\; \beta^*_{(j)1} + \beta^*_{(1)2}\,(x_{i2j} - x_{i24}) + \beta^*_{(1)3}\,(x_{i3j} - x_{i34}) + \beta^*_{(1)4}\, x_{i4}, \tag{19}$$

where, for the ith person, x_{i2j} is the cost for the jth transport means, and x_{i3j} is the journey time of the jth transport means. The distance to get to work is x_{i4}; it has the same value regardless of the transport means.

Equation 19 implies $\mathbf{H}_1 = \mathbf{I}_3$ and $\mathbf{H}_2 = \mathbf{H}_3 = \mathbf{H}_4 = \mathbf{1}_3$. Note also that if the last response category is used as the baseline or reference group (the default of `multinomial()`) then $x_{ik,M+1}$ can be subtracted from x_{ikj} for $j = 1, \ldots, M$—this is the natural way $x_{ik,M+1}$ enters into the model.

Recall from (2) that we had

$$\eta_j(\boldsymbol{x}_i) \;=\; \boldsymbol{\beta}_j^\top \boldsymbol{x}_i \;=\; \sum_{k=1}^{p} x_{ik}\, \beta_{(j)k}. \tag{20}$$

Importantly, this can be generalized to

$$\eta_j(\boldsymbol{x}_{ij}) \;=\; \boldsymbol{\beta}_j^\top \boldsymbol{x}_{ij} \;=\; \sum_{k=1}^{p} x_{ikj}\, \beta_{(j)k}, \tag{21}$$

or writing this another way (as a mixture or hybrid),

$$\eta_j(\boldsymbol{x}_i^*, \boldsymbol{x}_{ij}^*) \;=\; \boldsymbol{\beta}_j^{*T} \boldsymbol{x}_i^* + \boldsymbol{\beta}_j^{**T} \boldsymbol{x}_{ij}^*. \tag{22}$$

Often $\boldsymbol{\beta}_j^{**} = \boldsymbol{\beta}^{**}$, say. In (22) the variables in $\boldsymbol{x}_i^*$ are common to all η_j, and the variables in $\boldsymbol{x}_{ij}^*$ have different values for differing η_j. This allows for covariate values that are specific to each η_j, a facility which is very important in many applications.

The use of the `xij` argument with the **VGAM** family function `multinomial()` has very important applications in economics. In that field the term "multinomial logit model" includes a variety of models such as the "generalized logit model" where (20) holds, the "conditional logit model" where (21) holds, and the "mixed logit model," which is a combination of the two, where (22) holds. The generalized logit model focusses on the individual as the unit of analysis, and uses individual characteristics as explanatory variables, e.g., age of the person in the transport example. The conditional logit model assumes different values for each alternative and the impact of a unit of x_k is assumed to be constant across alternatives, e.g., journey time in the choice of transport mode. Unfortunately, there is confusion in the literature for the terminology of the models. Some authors call `multinomial()` with (20) the "generalized logit model". Others call the mixed logit model the "multinomial logit model" and view the generalized logit and conditional logit models as special cases. In **VGAM** terminology there is no need to give different names to all these slightly differing special cases. They are all still called multinomial logit models, although it may be added that there are some covariate-specific linear/additive predictors. The important thing is that the framework accommodates $\boldsymbol{x}_{ij}$, so one tries to avoid making life unnecessarily complicated. And `xij` can apply in theory to any VGLM and not just to the multinomial logit model. **?** present another perspective on the $\boldsymbol{x}_{ij}$ problem with illustrations from **Zelig** (**?**).

Using the `xij` argument

VGAM handles variables whose values depend on η_j, (22), using the `xij` argument. It is assigned an S formula or a list of `S` formulas. Each formula, which must have M *different* terms, forms a matrix that premultiplies a constraint matrix. In detail, (20) can be written in vector form as

$$\boldsymbol{\eta}(\boldsymbol{x}_i) \;=\; \mathbf{B}^\top \boldsymbol{x}_i \;=\; \sum_{k=1}^{p} \mathbf{H}_k\, \boldsymbol{\beta}_k^*\, x_{ik}, \tag{23}$$

where $\boldsymbol{\beta}_k^* = \left(\beta_{(1)k}^*, \ldots, \beta_{(r_k)k}^*\right)^\top$ is to be estimated. This may be written

$$\boldsymbol{\eta}(\boldsymbol{x}_i) \quad = \quad \sum_{k=1}^{p} \operatorname{diag}(x_{ik}, \ldots, x_{ik})\, \mathbf{H}_k\, \boldsymbol{\beta}_k^*. \tag{24}$$

To handle (21)–(22) we can generalize (24) to

$$\boldsymbol{\eta}_i \quad = \quad \sum_{k=1}^{p} \operatorname{diag}(x_{ik1}, \ldots, x_{ikM})\, \mathbf{H}_k\, \boldsymbol{\beta}_k^* \qquad \left(= \sum_{k=1}^{p} \mathbf{X}_{(ik)}^*\, \mathbf{H}_k\, \boldsymbol{\beta}_k^*, \text{ say}\right). \tag{25}$$

Each component of the list `xij` is a formula having M terms (ignoring the intercept) which specifies the successive diagonal elements of the matrix $\mathbf{X}_{(ik)}^*$. Thus each row of the constraint matrix may be multiplied by a different vector of values. The constraint matrices themselves are not affected by the `xij` argument.

How can one fit such models in **VGAM**? Let us fit (19). Suppose the journey cost and time variables have had the cost and time of walking subtracted from them. Then, using "`.trn`" to denote train,

```
fit2 <- vglm(cbind(bus, train, car, walk) ~ Cost + Time + Distance,
             fam = multinomial(parallel = TRUE ~ Cost + Time + Distance - 1),
             xij = list(Cost ~ Cost.bus + Cost.trn + Cost.car,
                        Time ~ Time.bus + Time.trn + Time.car),
             form2 = ~  Cost.bus + Cost.trn + Cost.car +
                        Time.bus + Time.trn + Time.car +
                        Cost + Time + Distance,
             data = gotowork)
```

should do the job. Here, the argument `form2` is assigned a second `S` formula which is used in some special circumstances or by certain types of **VGAM** family functions. The model has $\mathbf{H}_1 = \mathbf{I}_3$ and $\mathbf{H}_2 = \mathbf{H}_3 = \mathbf{H}_4 = \mathbf{1}_3$ because the lack of parallelism only applies to the intercept. However, unless `Cost` is the same as `Cost.bus` and `Time` is the same as `Time.bus`, this model should not be plotted with `plotvgam()`; see the author's homepage for further documentation.

By the way, suppose $\beta_{(1)4}^*$ in (19) is replaced by $\beta_{(j)4}^*$. Then the above code but with

```
fam = multinomial(parallel = FALSE ~ 1 + Distance),
```

should fit this model. Equivalently,

```
fam = multinomial(parallel = TRUE ~ Cost + Time - 1),
```

A more complicated example

The above example is straightforward because the variables were entered linearly. However, things become more tricky if data-dependent functions are used in any `xij` terms, e.g., `bs()`, `ns()` or `poly()`. In particular, regression splines such as `bs()` and `ns()` can be used to estimate a general smooth function $f(x_{ij})$, which is very useful for exploratory data analysis.

Suppose we wish to fit the variable `Cost` with a smoother. This is possible with regression splines and using a trick. Firstly note that

```
fit3 <- vglm(cbind(bus, train, car, walk) ~ ns(Cost) + Time + Distance,
             multinomial(parallel = TRUE ~ ns(Cost) + Time + Distance - 1),
             xij = list(ns(Cost) ~ ns(Cost.bus) + ns(Cost.trn) + ns(Cost.car),
                        Time ~ Time.bus + Time.trn + Time.car),
             form2 = ~  ns(Cost.bus) + ns(Cost.trn) + ns(Cost.car) +
                        Time.bus + Time.trn + Time.car +
                        ns(Cost) + Cost + Time + Distance,
             data = gotowork)
```

will *not* work because the basis functions for `ns(Cost.bus)`, `ns(Cost.trn)` and `ns(Cost.car)` are not identical since the knots differ. Consequently, they represent different functions despite having common regression coefficients.

Fortunately, it is possible to force the `ns()` terms to have identical basis functions by using a trick: combine the vectors temporarily. To do this, one can let

```
NS <- function(x, ..., df = 3)
      ns(c(x, ...), df = df)[1:length(x), , drop = FALSE]
```

This computes a natural cubic B-spline evaluated at `x` but it uses the other arguments as well to form an overall vector from which to obtain the (common) knots. Then the usage of `NS()` can be something like

```
fit4 <- vglm(cbind(bus, train, car, walk) ~ NS(Cost.bus, Cost.trn, Cost.car)
                                          + Time + Distance,
             multinomial(parallel = TRUE ~  NS(Cost.bus, Cost.trn, Cost.car)
                                          + Time + Distance - 1),
             xij = list(NS(Cost.bus, Cost.trn, Cost.car) ~
                        NS(Cost.bus, Cost.trn, Cost.car) +
                        NS(Cost.trn, Cost.car, Cost.bus) +
                        NS(Cost.car, Cost.bus, Cost.trn),
                        Time ~ Time.bus + Time.trn + Time.car),
             form2 = ~  NS(Cost.bus, Cost.trn, Cost.car) +
                        NS(Cost.trn, Cost.car, Cost.bus) +
                        NS(Cost.car, Cost.bus, Cost.trn) +
                        Time.bus + Time.trn + Time.car +
                        Cost.bus + Cost.trn + Cost.car +
                        Time + Distance,
             data = gotowork)
```

So `NS(Cost.bus, Cost.trn, Cost.car)` is the smooth term for `Cost.bus`, etc. Furthermore, `plotvgam()` may be applied to `fit4`, in which case the fitted regression spline is plotted against its first inner argument, viz. `Cost.bus`.

One of the reasons why it will predict correctly, too, is due to "smart prediction" (**?**).

Implementation details

The `xij` argument operates *after* the ordinary $\mathbf{X}_{\mathrm{VLM}}$ matrix is created. Then selected columns of $\mathbf{X}_{\mathrm{VLM}}$ are modified from the constraint matrices, `xij` and `form2` arguments. That is, from

`form2`'s model matrix $\mathbf{X}_{\mathrm{F2}}$, and the $\mathbf{H}_k$. This whole operation is possible because $\mathbf{X}_{\mathrm{VLM}}$ remains structurally the same. The crucial equation is (25).

Other `xij` examples are given in the online help of `fill()` and `vglm.control()`, as well as at the package's webpage.

9. Discussion

This article has sought to convey how VGLMs/VGAMs are well suited for fitting regression models for categorical data. Its primary strength is its simple and unified framework, and when reflected in software, makes practical CDA more understandable and efficient. Furthermore, there are natural extensions such as a reduced-rank variant and covariate-specific η_j. The **VGAM** package potentially offers a wide selection of models and utilities.

There is much future work to do. Some useful additions to the package include:

1. Bias-reduction (**?**) is a method for removing the $O(n^{-1})$ bias from a maximum likelihood estimate. For a substantial class of models including GLMs it can be formulated in terms of a minor adjustment of the score vector within an IRLS algorithm (**?**). One by-product, for logistic regression, is that while the maximum likelihood estimate (MLE) can be infinite, the adjustment leads to estimates that are always finite. At present the R package **brglm** (**?**) implements bias-reduction for a number of models. Bias-reduction might be implemented by adding an argument `bred = FALSE`, say, to some existing **VGAM** family functions.

2. Nested logit models were developed to overcome a fundamental shortcoming related to the multinomial logit model, viz. the independence of irrelevant alternatives (IIA) assumption. Roughly, the multinomial logit model assumes the ratio of the choice probabilities of two alternatives is not dependent on the presence or absence of other alternatives in the model. This presents problems that are often illustrated by the famed red bus-blue bus problem.

3. The generalized estimating equations (GEE) methodology is largely amenable to IRLS and this should be added to the package in the future (**?**).

4. For logistic regression SAS's `proc logistic` gives a warning if the data is completely separate or quasi-completely separate. Its effects are that some regression coefficients tend to $\pm\infty$. With such data, all (to my knowledge) R implementations give warnings that are vague, if any at all, and this is rather unacceptable (**?**). The **safeBinaryRegression** package (**?**) overloads `glm()` so that a check for the existence of the MLE is made before fitting a binary response GLM.

In closing, the **VGAM** package is continually being developed, therefore some future changes in the implementation details and usage may occur. These may include non-backward-compatible changes (see the `NEWS` file.) Further documentation and updates are available at the author's homepage whose URL is given in the `DESCRIPTION` file.

Acknowledgments

The author thanks Micah Altman, David Firth and Bill Venables for helpful conversations,

and Ioannis Kosmidis for a reprint. Thanks also to The Institute for Quantitative Social Science at Harvard University for their hospitality while this document was written during a sabbatical visit.

Affiliation:

Thomas W. Yee
Department of Statistics
University of Auckland, Private Bag 92019
Auckland Mail Centre
Auckland 1142, New Zealand
E-mail: t.yee@auckland.ac.nz
URL: http://www.stat.auckland.ac.nz/~yee/