An overview of the VGAM package

© T. W. Yee

University of Auckland

12–13 Nov 2011 @ Shanghai

t.yee@auckland.ac.nz
http://www.stat.auckland.ac.nz/~yee

Outline of this talk

Introduction to VGLMs and VGAMs I

The VGAM package implements several large classes of regression models of which *vector generalized linear and additive models (VGLMs/VGAMs)* are most commonly used.

The primary key words are

- *iteratively reweighted least squares (IRLS),*
- *maximum likelihood estimation,*
- *Fisher scoring,*
- *additive models.*

Other concepts are

- *reduced-rank regression,*
- *constrained ordination,*
- *vector smoothing.*

Introduction to VGLMs and VGAMs II

Basically ...

VGLMs model each parameter, transformed if necessary, as a linear combination of the explanatory variables. That is,

$$g_j(\theta_j) \;=\; \eta_j \;=\; \boldsymbol{\beta}_j^T \mathbf{x} \;=\; \beta_{(j)1}\, x_1 + \cdots + \beta_{(j)p}\, x_p \tag{1}$$

where g_j is a parameter link function $(-\infty < \eta_j < \infty)$.

VGAMs extend (1) to

$$g_j(\theta_j) \;=\; \eta_j \;=\; f_{(j)1}(x_1) + \cdots + f_{(j)p}(x_p), \tag{2}$$

i.e., an *additive model* for each parameter. Estimated by smoothers, this is a data-driven approach.

Introduction to VGLMs and VGAMs III

Example: negative binomial

Y has a probability function that can be written as

$$P(Y = y; \mu, k) \;=\; \binom{y+k-1}{y} \left(\frac{\mu}{\mu+k}\right)^{y} \left(\frac{k}{k+\mu}\right)^{k}$$

where $y = 0, 1, 2, \ldots$. Parameters $\mu \; (= E(Y)) > 0$ and $k > 0$.

The MASS implementation is restricted to a intercept-only estimate of k, e.g., cannot fit $\log\, k = \beta_{(2)1} + \beta_{(2)2} x_2$. In contrast, VGAM can fit

$$\begin{aligned} \log\, \mu &= \eta_1 = \boldsymbol{\beta}_1^T \mathbf{x}, \\ \log\, k &= \eta_2 = \boldsymbol{\beta}_2^T \mathbf{x}. \end{aligned}$$

```
vglm(y ~ x2 + ··· + xp, family = negbinomial(zero = NULL))
```

Introduction to VGLMs and VGAMs IV

The framework extends GLMs and GAMs in three main ways:

(i) $\mathbf{y}$ not restricted to the exponential family,

(ii) multivariate responses $\mathbf{y}$ and/or *linear/additive predictors* $\boldsymbol{\eta}$ are handled,

(iii) η_j need not be a function of a mean μ: $\eta_j = g_j(\theta_j)$ for any parameter θ_j.

This formulation is deliberately general so that it encompasses as many distributions and models as possible. *We wish to be limited only by the assumption that the regression coefficients enter through a set of linear or additive predictors*.

Given the covariates, the conditional distribution of the response is intended to be completely general. *More general* $\Longrightarrow$ *more useful.*

Introduction to VGLMs and VGAMs V

The scope of VGAM is very broad; it potentially covers

- univariate and multivariate distributions,
- categorical data analysis,
- quantile and expectile regression,
- time series,
- survival analysis,
- mixture models,
- extreme value analysis,
- nonlinear regression,
- reduced-rank regression,
- ordination, ...

It conveys GLM/GAM-type modelling to a much broader range of models.

Introduction to VGLMs and VGAMs VI

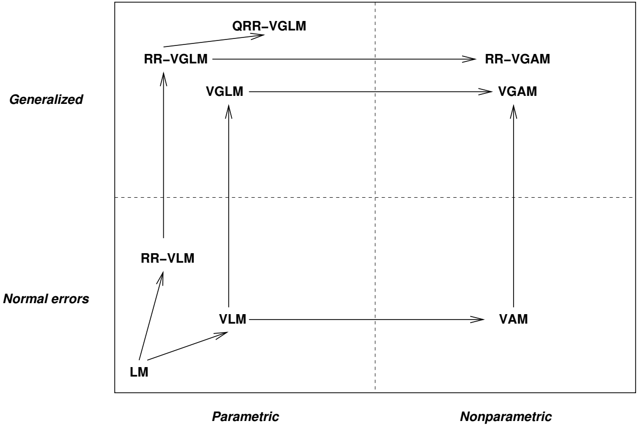


Figure: Flowchart for different classes of models. Legend: **LM** = linear model, **V** = vector, **G** = generalized, **A** = additive, **RR** = reduced-rank, **Q** = quadratic.

Introduction to VGLMs and VGAMs VII

t

$\boldsymbol{\eta}$	Model	S function	Reference
$\mathbf{B}_1^T \mathbf{x}_1 + \mathbf{B}_2^T \mathbf{x}_2 \; (= \mathbf{B}^T \mathbf{x})$	VGLM	`vglm()`	Yee & Hastie (2003)
$\mathbf{B}_1^T \mathbf{x}_1 + \sum_{k=p_1+1}^{p_1+p_2} \mathbf{H}_k \mathbf{f}_k^*(x_k)$	VGAM	`vgam()`	Yee & Wild (1996)
$\mathbf{B}_1^T \mathbf{x}_1 + \mathbf{A}\,\boldsymbol{\nu}$	RR-VGLM	`rrvglm()`	Yee & Hastie (2003)
$\mathbf{B}_1^T \mathbf{x}_1 + \mathbf{A}\,\boldsymbol{\nu} + \begin{pmatrix} \boldsymbol{\nu}^T \mathbf{D}_1 \boldsymbol{\nu} \\ \vdots \\ \boldsymbol{\nu}^T \mathbf{D}_M \boldsymbol{\nu} \end{pmatrix}$	QRR-VGLM	`cqo()`	Yee (2004)
$\mathbf{B}_1^T \mathbf{x}_1 + \sum_{r=1}^{R} \mathbf{f}_r(\nu_r)$	RR-VGAM	`cao()`	Yee (2006)

Table: A summary of VGAM and its framework. The latent variables $\boldsymbol{\nu} = \mathbf{C}^T \mathbf{x}_2$, or $\nu = \mathbf{c}^T \mathbf{x}_2$ if rank $R = 1$. Here, $\mathbf{x}^T = (\mathbf{x}_1^T, \mathbf{x}_2^T)$. Abbreviations: A = additive, C = constrained, L = linear, O = ordination, Q = quadratic, RR = reduced-rank, VGLM = vector generalized linear model.

Vector generalized linear models I

Data $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$ on n independent “individuals”.

Definition Conditional distribution of $\mathbf{y}$ given $\mathbf{x}$ is

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

where for $j = 1, \ldots, M$,

$$\begin{aligned} \eta_j &= \eta_j(\mathbf{x}) = \boldsymbol{\beta}_j^T \mathbf{x}, \\ \boldsymbol{\beta}_j &= (\beta_{(j)1}, \ldots, \beta_{(j)p})^T, \\ \boldsymbol{\beta} &= (\boldsymbol{\beta}_1^T, \ldots, \boldsymbol{\beta}_M^T)^T, \\ \boldsymbol{\phi} &= \text{a vector of scale factors.} \end{aligned} \tag{3}$$

Often $g_j(\theta_j) = \eta_j$ for parameters θ_j and link functions g_j.

VGLM examples I

1. GLMs $\quad M = 1$

t

Distribution	Density/probability function $f(y)$	Range of y	VGAM family function
Gaussian	$\left(2\pi\sigma^2\right)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(y-\mu)^2/\sigma^2\right\}$	$(-\infty, \infty)$	`gaussianff()`
Binomial	$\binom{A}{Ay} p^{Ay}(1-p)^{A(1-y)}$	$0(1/A)1$	`binomialff()`
Poisson	$\frac{\exp\{-\lambda\}\lambda^y}{y!}$	$0(1)\infty$	`poissonff()`
Gamma	$\frac{(k/\mu)^k\, y^{k-1}\, \exp\{-ky/\mu\}}{\Gamma(k)}$	$(0, \infty)$	`gammaff()`
Inverse Gaussian	$\left(\frac{\lambda}{2\pi\, y^3}\right)^{\frac{1}{2}} \exp\left\{-\frac{\lambda}{2\,\mu^2}\frac{(y-\mu)^2}{y}\right\}$	$(0, \infty)$	`inverse.gaussianff()`

Table: Summary of GLMs supported by VGAM. The known prior weight is A. These are incompatible with `glm()`.

VGLM examples II

2. Negative binomial distribution

For $y = 0, 1, 2, \ldots,$

$$f(y; \mu, k) \;=\; \binom{y+k-1}{y} \left(\frac{\mu}{\mu+k}\right)^{y} \left(\frac{k}{k+\mu}\right)^{k}, \quad \mu, \quad k > 0.$$

Good choice:

$$\begin{aligned} \eta_1 &= \log \mu, \\ \eta_2 &= \log k. \end{aligned}$$

```
vglm(y ~ x2 + x3 + ···, family = negbinomial(zero = NULL))
```

VGLM examples III

3. Bivariate logistic odds-ratio model

Data: (Y_1, Y_2) where $Y_j = 0$ or 1.

Examples

- $Y_1 = 1$ if left eye is blind, $Y_2 = 1$ if right eye is blind.
- $Y_j = 1/0$ if Species j is present/absent.
- $Y_1 = 1/0$ if person has/hasn't cancer,
  $Y_2 = 1/0$ if person has/hasn't diabetes.

VGLM examples IV

Table: The coalminers data set from UK collieries. Note: B = Breathlessness, W = Wheeze, 1 = presence, 0 = absence.

t

Age Group	$(B=1, W=1)$	$(B=1, W=0)$	$(B=0, W=1)$	$(B=0, W=0)$
20–24	9	7	95	1841
25–29	23	9	105	1654
30–34	54	19	177	1863
35–39	121	48	257	2357
40–44	169	54	273	1778
45–49	269	88	324	1712
50–54	404	117	245	1324
55–59	406	152	225	967
60–64	372	106	132	526

$$
\begin{aligned}
p_j &= P(Y_j = 1), \quad \text{marginal probabilities} \\
p_{rs} &= P(Y_1 = r, Y_2 = s), \quad r, s = 0, 1, \quad \text{joint probabilities} \\
\psi &= \frac{p_{00}\, p_{11}}{p_{01}\, p_{10}} \quad \text{(Odds ratio).}
\end{aligned}
$$

VGLM examples V

Model:

$$\begin{aligned} \mathrm{logit}\, p_j(\mathbf{x}) &= \eta_j(\mathbf{x}), \qquad j = 1, 2\,, \\ \log\ \psi(\mathbf{x}) &= \eta_3(\mathbf{x}). \end{aligned}$$

Recover p_{rs}'s from p_1, p_2 and ψ.

Q: Why not allow a probit or complementary log-log link?

Exchangeable data $\Longrightarrow$ constrain $\eta_1 = \eta_2$ (e.g., ears and eyes), i.e.,

$$\begin{aligned} \mathtt{cloglog}\, p_j(\mathbf{x}) &= \eta_1(\mathbf{x}), \qquad j = 1, 2\,, \\ \log\ \psi(\mathbf{x}) &= \eta_3(\mathbf{x}). \end{aligned}$$

VGLM examples VI

Note:

$$\begin{pmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \end{pmatrix} = \sum_{k=1}^{p} \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \beta^*_{(1)k} \\ \beta^*_{(2)k} \end{pmatrix} x_k = \sum_{k=1}^{p} \begin{pmatrix} \beta^*_{(1)k} \\ \beta^*_{(1)k} \\ \beta^*_{(2)k} \end{pmatrix} x_k. \quad (4)$$

```
> vglm(..., family = binom2.or("cloglog", exchangeable = TRUE))
```

VGLM examples VII

4. Models for a categorical response i.e., $Y \in \{1, 2, \ldots, M + 1\}$.

   Y may be unordered (*nominal*) or ordered (*ordinal*).

   Table: Period of exposure (years) and severity of pneumoconiosis amongst a group of coalminers.

t

Exposure Time	Normal	Mild	Severe
5.8	98	0	0
15.0	51	2	1
21.5	34	6	3
27.5	35	5	8
33.5	32	10	9
39.5	23	7	8
46.0	12	6	10
51.5	4	2	5

VGLM examples VIII

(i) Multinomial logit model (nominal Y)

$$P(Y = j|\mathbf{x}) \;=\; \frac{\exp\{\eta_j(\mathbf{x})\}}{\sum_{t=1}^{M+1} \exp\{\eta_t(\mathbf{x})\}}, \quad j = 1, \ldots, M+1.$$

For identifiability: $\eta_{M+1} \equiv 0$.

Equivalently,

$$\log\left(\frac{P(Y = j|\mathbf{x})}{P(Y = M+1|\mathbf{x})}\right) \;=\; \eta_j(\mathbf{x}), \quad j = 1, \ldots, M+1.$$

```
vglm(ymatrix ~ x2 + x3 + ···, family = multinomial)
```

VGLM examples IX

(ii) Nonproportional odds model (ordinal Y)

$$\operatorname{logit} P(Y \leq j|\mathbf{x}) \;=\; \eta_j(\mathbf{x}), \qquad j = 1, \ldots, M.$$

Proportional odds model: constrain

$$\eta_j(\mathbf{x}) = \alpha_j + \eta(\mathbf{x})$$

(aka the *parallelism* assumption, which stops the probabilities from becoming negative or greater than 1).

```
vglm(ymatrix ∼ x2 + ···, family = cumulative(parallel = TRUE))
```

VGLM examples X

(iii) Continuation ratio model (ordinal Y)

$$\text{logit}\, P(Y > j | Y \geq j, \mathbf{x}) \;=\; \eta_j(\mathbf{x}), \qquad j = 1, \ldots, M.$$

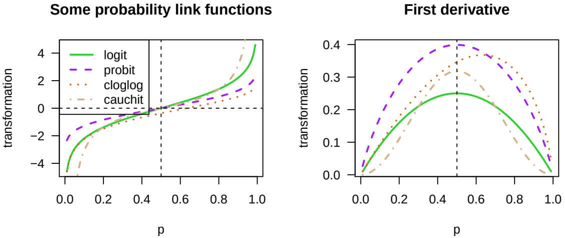
Good for sequential-type data, e.g., the effect of $\mathbf{x}$ on a couple who decide to have one child, two children, three children, ...

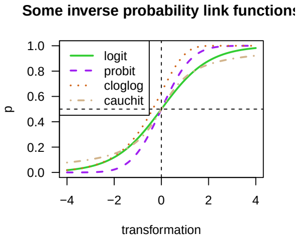
Other links (for $0 < p < 1$):

probit	$\Phi^{-1}(p)$,
complementary log-log	$\log\{-\log(1-p)\}$,
cauchit	$\tan(\pi(p - \frac{1}{2}))$.

```
vglm(..., family = cratio(link = probit))
```

VGLM examples XI





VGLM algorithm† I

Models with log-likelihood

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \ell_i\{\eta_1(\mathbf{x}_i), \ldots, \eta_M(\mathbf{x}_i)\},$$

where $\eta_j = \boldsymbol{\beta}_j^T \mathbf{x}_i$. Then

$$\begin{aligned} \frac{\partial \ell}{\partial \boldsymbol{\beta}_j} &= \sum_{i=1}^{n} \frac{\partial \ell_i}{\partial \eta_j} \mathbf{x}_i \\ \text{and} \qquad \frac{\partial^2 \ell}{\partial \boldsymbol{\beta}_j \, \partial \boldsymbol{\beta}_k^T} &= \sum_{i=1}^{n} \frac{\partial^2 \ell_i}{\partial \eta_j \, \partial \eta_k} \mathbf{x}_i \, \mathbf{x}_i^T . \end{aligned}$$

Newton-Raphson algorithm

$$\boldsymbol{\beta}^{(a+1)} = \boldsymbol{\beta}^{(a)} + \mathcal{J}\left(\boldsymbol{\beta}^{(a)}\right)^{-1} \mathbf{U}\left(\boldsymbol{\beta}^{(a)}\right)$$

VGLM algorithm† II

written in iterately reweighted least squares (IRLS) form is

$$\begin{aligned}\beta^{(a+1)} &= \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{W}\mathbf{X}\,\beta^{(a)} + \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{W}\,\mathbf{W}^{-1}\mathbf{u} \\ &= \left(\mathbf{X}_{\text{VLM}}^T\,\mathbf{W}^{(a)}\,\mathbf{X}_{\text{VLM}}\right)^{-1}\mathbf{X}_{\text{VLM}}^T\,\mathbf{W}^{(a)}\,\mathbf{z}^{(a)}\,.\end{aligned}$$

Let $\mathbf{z} = (\mathbf{z}_1^T, \ldots, \mathbf{z}_n^T)^T$ and $\mathbf{u} = (\mathbf{u}_1^T, \ldots, \mathbf{u}_n^T)^T$, where $\mathbf{u}_i$ has jth element

$$(\mathbf{u}_i)_j = \frac{\partial \ell_i}{\partial \eta_j},$$

and $\mathbf{z}_i = \boldsymbol{\eta}(\mathbf{x}_i) + \mathbf{W}_i^{-1}\mathbf{u}_i$ (*adjusted dependent vector* or *pseudo-response*).

Also, $\mathbf{W} = \text{Diag}(\mathbf{W}_1, \ldots, \mathbf{W}_n)$, $\quad (\mathbf{W}_i)_{jk} = -\dfrac{\partial^2 \ell_i}{\partial \eta_j\,\partial \eta_k}$,

$\mathbf{X}_{\text{VLM}} = (\mathbf{X}_1^T, \ldots, \mathbf{X}_n^T)^T, \quad \mathbf{X}_i = \text{Diag}(\mathbf{x}_i^T, \ldots, \mathbf{x}_i^T) = \mathbf{I}_M \otimes \mathbf{x}_i^T.$

VGLM algorithm† III

$\boldsymbol{\beta}^{(a+1)}$ is the solution to

$$\mathbf{z}^{(a)} \;=\; \mathbf{X}_{\text{VLM}}\,\boldsymbol{\beta}^{(a+1)} + \boldsymbol{\varepsilon}^{(a)}, \qquad \mathrm{Var}(\boldsymbol{\varepsilon}^{(a)}) \;=\; \phi\,{\mathbf{W}^{(a)}}^{-1}.$$

Fisher scoring:

$$(\mathbf{W}_i)_{jk} \;=\; -\,E\left[\frac{\partial^2 \ell_i}{\partial \eta_j\, \partial \eta_k}\right]$$

usually results in slower convergence but is preferable because the *working weight matrices* are positive-definite over a larger parameter space.

VGLM algorithm† IV

Some Notes

1. `wz` computed in `@weight` is usually

$$(\mathbf{W}_i)_{jk} \;=\; -E\left(\frac{\partial^2 \ell_i}{\partial\eta_j\,\partial\eta_k}\right), \quad \text{sometimes} \quad -\frac{\partial^2 \ell_i}{\partial\eta_j\,\partial\eta_k}.$$

2. The following formulae are useful.

$$\begin{aligned}
\frac{\partial \ell}{\partial \eta_j} &= \frac{\partial \ell}{\partial \theta_j}\,\frac{\partial \theta_j}{\partial \eta_j}, \\
\frac{\partial^2 \ell}{\partial \eta_j^2} &= \frac{\partial \ell}{\partial \theta_j}\,\frac{\partial^2 \theta_j}{\partial \eta_j^2} + \left(\frac{\partial \theta_j}{\partial \eta_j}\right)^2 \frac{\partial^2 \ell}{\partial \theta_j^2}, \\
\frac{\partial^2 \ell}{\partial \eta_j\,\partial \eta_k} &= \left\{\frac{\partial^2 \ell}{\partial \theta_j\,\partial \theta_k} - \frac{\partial \ell}{\partial \theta_k}\,\frac{\partial \theta_k}{\partial \eta_k}\,\frac{\partial^2 \eta_k}{\partial \theta_j\,\partial \theta_k}\right\} \frac{\partial \theta_j}{\partial \eta_j}\,\frac{\partial \theta_k}{\partial \eta_k}, \quad j \neq k,
\end{aligned}$$

VLMs† I

The *vector linear model (VLM)* is the central model behind VGLMs and VGAMs. Its crux is to minimize

$$\sum_{i=1}^{n} \left(\mathbf{z}_i - \sum_{k=1}^{p} \mathbf{H}_k \, \boldsymbol{\beta}_k^* \, x_{ik} \right)^T \mathbf{W}_i \left(\mathbf{z}_i - \sum_{k=1}^{p} \mathbf{H}_k \, \boldsymbol{\beta}_k^* \, x_{ik} \right),$$

where the $\mathbf{H}_k$ are known *constraint matrices* of full column rank.

With no constraints ($\mathbf{H}_k = \mathbf{I}_M$), this is equivalent to fitting

$$\mathbf{z}_i = \begin{pmatrix} \mathbf{x}_i^T \boldsymbol{\beta}_1 \\ \vdots \\ \mathbf{x}_i^T \boldsymbol{\beta}_M \end{pmatrix} + \boldsymbol{\varepsilon}_i, \quad \boldsymbol{\varepsilon}_i \sim (\mathbf{0}, \mathbf{W}_i^{-1}), \quad i = 1, \ldots, n.$$

VLMs† II

Cf. *Multivariate linear model* (aka *multivarate regression)*

$$(\mathbf{Y}_{(1)} \cdots \mathbf{Y}_{(M)}) \;=\; \mathbf{X}\mathbf{B} + \mathbf{U}, \qquad \mathbf{u}_i \;\sim\; (\mathbf{0}, \boldsymbol{\Sigma}) \tag{5}$$

where $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_n)^T$.

The VGAM package for R I

Written in S, its central core are the functions `vglm()`, `vgam()` and `rrvglm()`.

Generic functions include `coef()`, `fitted()`, `plot()`, `predict()`, `print()`, `resid()`, `summary()`. Others are `lvplot()`, `Coef()`, `df.residual()`, `logLik()`, `vcov()`.

Plus lots and lots of VGAM family functions.

Modular construction, flexible, easy to use and useful.
Runs S4 (Chambers, 1998) in R.
Can install the VGAM package in R by typing

```
> install.packages("VGAM")
```

The VGAM package for R II

The central functions of VGAM

`vglm()`	Vector generalized linear models.
`vgam()`	Vector generalized additive models.
`rrvglm()`	Reduced-rank vector generalized linear models.
`cqo()`	Constrained quadratic (Gaussian) ordination (QRR-VGLM).
`cao()`	Constrained additive ordination (RR-VGAM).

Others:

`vlm()`	Vector linear models.
`grc()`	Goodman's RC(r) model.
`rcam()`	Row-column association models (not complete).

The VGAM package for R III

```
Package: VGAM
Version: 0.8-4
Date: 2011-11-03
Title: Vector Generalized Linear and Additive Models
Author: Thomas W. Yee <t.yee@auckland.ac.nz>
Maintainer: Thomas Yee <t.yee@auckland.ac.nz>
Depends: R (>= 2.11.1), splines, methods, stats, stats4
Description: Vector generalized linear and additive models,
    and associated models (Reduced-Rank VGLMs, Quadratic
    RR-VGLMs, Reduced-Rank VGAMs). This package fits many
    models and distribution by maximum likelihood estimation
    (MLE) or penalized MLE. Also fits constrained ordination
    models in ecology.
License: GPL-2
URL: http://www.stat.auckland.ac.nz/~yee/VGAM
```

See the `INDEX` file.

The VGAM package for R IV

Table: VGAM generic functions applied to a model called `fit`.

Function	Value
`coef(fit)`	$\widehat{\boldsymbol{\beta}}^*$
`coef(fit, matrix = TRUE)`	$\widehat{\mathbf{B}}$
`constraints(fit, type = "lm")`	$\mathbf{H}_k, \ k = 1, \ldots, p$
`deviance(fit)`	Deviance $D = \sum_{i=1}^{n} d_i$
`fitted(fit)`	$\widehat{\mu}_{ij}$ usually
`logLik(fit)`	Log-likelihood $\sum_{i=1}^{n} w_i \, \ell_i$
`model.matrix(fit, type = "lm")`	LM model matrix ($n \times p$)
`model.matrix(fit, type = "vlm")`	VLM model matrix $\mathbf{X}_{\text{VLM}}$
`predict(fit)`	$\widehat{\eta}_{ij}$

The VGAM package for R V

Table: VGAM generic functions applied to a model called `fit`.

Function	Value
`predict(fit, type = "response")`	$\widehat{\mu}_{ij}$ usually
`resid(fit, type = "response")`	$y_{ij} - \widehat{\mu}_{ij}$
`resid(fit, type = "deviance")`	$\mathrm{sign}(y_i - \hat{\mu}_i)\sqrt{d_i}$
`resid(fit, type = "pearson")`	$\mathbf{W}_i^{-\frac{1}{2}}\,\mathbf{u}_i$
`resid(fit, type = "working")`	$\mathbf{z}_i - \boldsymbol{\eta}_i = \mathbf{W}_i^{-1}\,\mathbf{u}_i$
`vcov(fit)`	$\widehat{\mathrm{Var}}(\widehat{\boldsymbol{\beta}})$
`weights(fit, type = "prior")`	w_i (`weights` argument)
`weights(fit, type = "working")`	$w_i\mathbf{W}_i$ (in matrix-band format)

The VGAM package for R VI

The VGAM package employs several feature to make the software more robust, e.g.,

- Parameter link functions, e.g.,
  - $\log\theta$ for $\theta > 0$,
  - $\mathrm{logit}\ \theta$ for $0 < \theta < 1$.
  - $\log(\theta - 1)$ for $\theta > 1$.
- *Half-step sizing*.
- Good initial values, e.g., self-starting VGAM family functions.
- Numerical linear algebra based on orthogonal methods, e.g., QR method in LINPACK. Yet to do: use LAPACK.
- B-splines, not the Reinsch algorithm.

Some computational and implementational details† I

- Is S4 object-orientated and very modular—simply have to write a VGAM "family function".
- VGAM minimizes `@deviance`, or maximizes `@loglikelihood` or iterates till change in $\widehat{\beta}^{(a+1)}$ is sufficiently small. User may specify some other objective function, e.g., `@rss`.
  Convergence criteria: for a scalar quantity A

$$\frac{\left|A^{(a+1)} - A^{(a)}\right|}{\texttt{epsilon} + \left|A^{(a)}\right|} < \texttt{epsilon}; \tag{6}$$

for the regression coefficients the maximum over $j = 1, \ldots, p$ of

$$\frac{\left|\beta_j^{(a+1)} - \beta_j^{(a)}\right|}{\texttt{epsilon} + \left|\beta_j^{(a)}\right|} < \texttt{epsilon}.$$

for all $j = 1, \ldots, p$.

Some computational and implementational details† II

```
> args(vglm)

function (formula, family, data = list(), weights = NULL, subset = NULL,
    na.action = na.fail, etastart = NULL, mustart = NULL, coefstart = NULL,
    control = vglm.control(...), offset = NULL, method = "vglm.fit",
    model = FALSE, x.arg = TRUE, y.arg = TRUE, contrasts = NULL,
    constraints = NULL, extra = list(), form2 = NULL, qr.arg = FALSE,
    smart = TRUE, ...)
NULL
```

```
> args(vglm.control)

function (checkwz = TRUE, criterion = names(.min.criterion.VGAM),
    epsilon = 1e-07, half.stepsizing = TRUE, maxit = 30, stepsize = 1,
    save.weight = FALSE, trace = FALSE, wzepsilon = .Machine$double.eps^0.75,
    xij = NULL, ...)
NULL
```

- Implements *"smart" prediction*. [*Safe prediction* not as good as *smart prediction*, e.g., `bs(scale(x))`, `I(bs(x))`, `poly(scale(x), 2)`].

VGAMs I

VGAMs allow additive-model extensions to all η_j in a VGLM, i.e., from

$$\eta_j(\mathbf{x}) \ = \ \beta_{(j)1}\, x_1 + \cdots + \beta_{(j)p}\, x_p \ = \ \boldsymbol{\beta}_j^T \mathbf{x}$$

to M additive predictors:

$$\eta_j(\mathbf{x}) \ = \ f_{(j)1}(x_1) + \cdots + f_{(j)p}(x_p),$$

a sum of arbitary smooth functions. Equivalently,

$$\begin{aligned} \boldsymbol{\eta}(\mathbf{x}) &= \mathbf{f}_1(x_1) + \cdots + \mathbf{f}_p(x_p) \\ &= \mathbf{H}_1\, \mathbf{f}_1^*(x_1) + \cdots + \mathbf{H}_p\, \mathbf{f}_p^*(x_p) \end{aligned} \tag{7}$$

for some *constraint matrices* $\mathbf{H}_k$ (default: $\mathbf{H}_k = \mathbf{I}_M$).

- $\mathbf{H}_1, \ldots, \mathbf{H}_p$ are known and of full-column rank,

VGAMs II

- $\mathbf{f}_k^*$ is a vector containing a possibly reduced set of component functions,

Starred quantities in (7) are unknown and to be estimated.

The $\mathbf{f}_k^*$ are centered for identifiability.

Examples of constraints

1. **Exchangeable bivariate logistic model** All

$$\mathbf{H}_k = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

2. **The proportional odds model**
   $\mathbf{H}_1 = \mathbf{I}_M, \qquad \mathbf{H}_2 = \cdots = \mathbf{H}_p = \mathbf{1}_M.$

VGAM facilitates the implementation and use of constraints, e.g.,

```
> binom2.or(exchangeable = TRUE)
> cumulative(parallel = FALSE ~ x5 - 1)
```

Constraints I

General formula:

$$\boldsymbol{\eta}(\mathbf{x}) \;=\; \mathbf{a} + \sum_{k=1}^{p} \mathbf{H}_k \, \mathbf{f}_k^{*}(x_k). \tag{8}$$

Some types of constraints can be more easily handled by arguments such as `exchangeable`, `parallel`, `zero`.

Examples

1. `> vglm(y ~ x, cumulative(parallel = FALSE))`

   stops the parallelism constraint from being applied to any of the explanatory variables (default). The result is a non-proportional odds model.

2. `> vglm(y ~ x, multinomial(zero=2))`

   will force η_2 to be an intercept-only, i.e., $\eta_2 = \beta_{(2)1}$.

Constraints II

3. Both

```
> vgam(y~x2+s(x3,3)+s(x4)+x5, binom2.or(exch= TRUE ~ s(x3,3)+x5, zero=
> vgam(y~x2+s(x3,3)+s(x4)+x5, binom2.or(exch= FALSE ~ x2+s(x4)-1, zero
```

makes the effect of $s(X_3)$ and X_5 to be the same for both marginal probabilities. Explicitly, the model they fit is

$$\begin{aligned} \operatorname{logit} p_1 &= \beta^*_{(1)1} + \beta^*_{(1)2}\, x_2 + f^*_{(1)3}(x_3) + f^*_{(1)4}(x_4) + \beta^*_{(1)5}\, x_5, \\ \operatorname{logit} p_2 &= \beta^*_{(1)1} + \beta^*_{(2)2}\, x_2 + f^*_{(1)3}(x_3) + f^*_{(2)4}(x_4) + \beta^*_{(1)5}\, x_5, \\ \log\ \psi(\mathbf{x}) &= \beta^*_{(2)1} + \beta^*_{(3)2}\, x_2 + f^*_{(2)3}(x_3) + f^*_{(3)4}(x_4) + \beta^*_{(2)5}\, x_5. \end{aligned}$$

Constraints III

4. `> vgam(y ~ x2 + s(x3), binom2.or(zero=3))`

   constrains $\eta_3 = \beta_{(3)1}$. So the odds ratio is simply a point estimate and not a function of the covariates. In general, `zero` may be assigned a vector of integers in the range 1 to M. Negative values allowed for multiple responses. Note that the exchangeability constraint applies to the intercepts, whereas the parallelism constraint doesn't.

5. Set `zero = NULL` if the `constraints` argument is used.

More examples I

1. ($\eta_1 = \eta_2$; e.g., exchangeability in the bivariate logit model)

$$\begin{aligned}
\eta_1 &= \alpha_1^* + f_{(1)2}^*(x_2) + f_{(1)3}^*(x_3) \\
\eta_2 &= \alpha_1^* + f_{(1)2}^*(x_2) + f_{(1)3}^*(x_3) \\
\eta_3 &= \alpha_2^* + f_{(2)2}^*(x_2) + f_{(2)3}^*(x_3)
\end{aligned}$$

Then

$$\mathbf{H}_1 = \mathbf{H}_2 = \mathbf{H}_3 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

If the $f_{(j)k}^*$ are linear then

```
> cmat = matrix(c(1,1,0, 0,0,1), 3, 2)
> clist = list("(Intercept)"=cmat, x2=cmat, x3=cmat)
> vglm(y ~ 1 + x2 + x3, constraints = clist, ...)
```

More examples II

2. $(\eta_j(\mathbf{x}) = \alpha_j + \eta(\mathbf{x})$; e.g., parallelism in the proportional odds model)

$$\begin{aligned}
\eta_1 &= \alpha_1^* + f_{(1)2}^*(x_2) + f_{(1)3}^*(x_3) \\
\eta_2 &= \alpha_2^* + f_{(1)2}^*(x_2) + f_{(1)3}^*(x_3) \\
\eta_3 &= \alpha_3^* + f_{(1)2}^*(x_2) + f_{(1)3}^*(x_3)
\end{aligned}$$

Then

$$\mathbf{H}_1 = \mathbf{I}_3, \qquad \mathbf{H}_2 = \mathbf{H}_3 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

If the $f_{(j)k}$ are linear then

```
> cm = matrix(1, 3, 1)
> clist = list("(Intercept)"=diag(3), x2=cm, x3=cm)
> vglm(y ~ x2 + x3, constraints=clist, ...)
```

More examples III

VGAM allows the input of the constraint matrices $\mathbf{H}_1, \ldots, \mathbf{H}_p$ through the `constraints` argument. (The default value of `NULL` signifies no constraints at all).

`constraints` must be assigned a `list` containing (constraint) matrices or functions that create them. Each of these must be named with the variable name, or term in the case of `s()`, `bs()` etc.

If any explanatory variable is a factor then only the name of the factor (not any of its levels) needs to appear in `constraints`.

As it can be seen, `constraints` is powerful and flexible, although a little cumbersome. Users are encouraged to use `zero`/`parallel`/`exchangeable` where possible.

The constraint matrices of a VGAM object can be seen by typing, e.g., `constraints(fit)`.

Estimation† I

Estimate the $\mathbf{f}_k$ by *penalized likelihood*. Let

$$
\begin{aligned}
\mathbf{f}_k &= (f_{(1)k}(x_{1k}), \ldots, f_{(M)k}(x_{1k}), \ldots, f_{(1)k}(x_{nk}), \ldots, f_{(M)k}(x_{nk}))^T \quad \text{and} \\
\boldsymbol{\eta} &= (\boldsymbol{\eta}_1^T, \ldots, \boldsymbol{\eta}_n^T)^T \;=\; \sum_{k=1}^{p} \mathbf{f}_k, \quad \text{where} \\
\boldsymbol{\eta}_i^T &= (\eta_1(\mathbf{x}_i), \ldots, \eta_M(\mathbf{x}_i)).
\end{aligned}
$$

The penalized log-likelihood can be written as

$$
\begin{aligned}
j(\mathbf{f}_1, \ldots, \mathbf{f}_p) &= \ell(\boldsymbol{\eta}; \mathbf{y}) - \frac{1}{2} \sum_{k=1}^{p} \sum_{j=1}^{M} \lambda_{(j)k} \int_{a_k}^{b_k} \left\{ f''_{(j)k}(x_k) \right\}^2 dx_k \\
&= \ell(\boldsymbol{\eta}; \mathbf{y}) - \frac{1}{2} \sum_{k=1}^{p} \mathbf{f}_k^T \mathbf{K}_k \mathbf{f}_k \qquad (9)
\end{aligned}
$$

Estimation† II

where $\mathbf{K}_k$ is the penalty matrix corresponding to smoothing with respect to the kth covariate and depends upon λ_k.

Maximizing $j(\mathbf{f}_1, \ldots, \mathbf{f}_p)$ with respect to $\mathbf{f}_1, \ldots, \mathbf{f}_p$ using Newton-Raphson gives

$$\begin{pmatrix} \mathbf{W}+\mathbf{K}_1 & \mathbf{W} & \cdots & \mathbf{W} \\ \mathbf{W} & \mathbf{W}+\mathbf{K}_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{W} \\ \mathbf{W} & \cdots & \mathbf{W} & \mathbf{W}+\mathbf{K}_p \end{pmatrix} \begin{pmatrix} \mathbf{f}_1^1 - \mathbf{f}_1^0 \\ \mathbf{f}_2^1 - \mathbf{f}_2^0 \\ \vdots \\ \mathbf{f}_p^1 - \mathbf{f}_p^0 \end{pmatrix} = \begin{pmatrix} \mathbf{u} - \mathbf{K}_1\,\mathbf{f}_1^0 \\ \mathbf{u} - \mathbf{K}_2\,\mathbf{f}_2^0 \\ \vdots \\ \mathbf{u} - \mathbf{K}_p\,\mathbf{f}_p^0 \end{pmatrix}.$$

See Hastie and Tibshirani (1990, sec. 6.5.2) and Green and Silverman (1994).

With $p = 1$ covariates, this leads to *vector smoothing* with vector splines.

Result: A VGAM using a vector spline maximizes the penalized likelihood (9).

Vector smoothing†

One has a *vector response* $\mathbf{y}_i$ $(M \times 1)$ at each scalar x_i:

$$\mathbf{y}_i \;=\; \mathbf{f}(x_i) + \boldsymbol{\varepsilon}_i, \qquad \boldsymbol{\varepsilon}_i \sim (\mathbf{0}, \boldsymbol{\Sigma}_i) \tag{10}$$

independently. Here,

$$\mathbf{f}(x) \;=\; \begin{pmatrix} f_1(x) \\ \vdots \\ f_M(x) \end{pmatrix}$$

is a vector of M arbitary smooth functions and $i = 1, \ldots, n$.

Applications of vector smoothing†

Actually, the vector smoothing problem has few applications in practice because $\boldsymbol{\Sigma}_i$ are unknown. Its primary use is in fitting the VGAM class. There are two good ways of vector smoothing:

- **Vector splines** Originally proposed by Fessler (1991), his solution was based on the Reinsch algorithm. Unfortunately, this may sometimes be numerically unstable. Better solution: use B-splines.
- **Local regression for vector responses** Joint work with Alan Welsh has resulted in a quasi-likelihood estimator, and a derivation of its asymptotic properties (for the simplest case of $M = 2$ and local linear fitting).

Vector splines†

Vector splines minimize

$$\sum_{i=1}^{n} \{\mathbf{y}_i - \mathbf{f}(x_i)\}^T \boldsymbol{\Sigma}_i^{-1} \{\mathbf{y}_i - \mathbf{f}(x_i)\} + \sum_{j=1}^{M} \lambda_j \int_a^b \{f_j''(x)\}^2 \, dx,$$

over a Sobolev space of order 2. Here, $a < x_1 < \cdots < x_n < b$ for some a and b, and $\lambda \geq 0$.

Write

$$\begin{aligned} \mathbf{y} &= (\mathbf{y}_1^T, \ldots, \mathbf{y}_n^T)^T \\ \boldsymbol{\Sigma} &= \mathrm{diag}(\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_n) \\ \mathbf{f} &= (f_1(x_1), \ldots, f_M(x_1), \ldots, f_1(x_n), \ldots, f_M(x_n))^T. \end{aligned}$$

Some notes† I

- Special case of $M = 1$ simplifies to a cubic smoothing spline.
- Fits into the penalty function framework of Green and Silverman (1994).
- Penalized least squares Minimize:

$$(\mathbf{y} - \mathbf{f})^T \boldsymbol{\Sigma}^{-1} (\mathbf{y} - \mathbf{f}) + \mathbf{f}^T \mathbf{K} \mathbf{f}.$$

Solution:

$$\widehat{\mathbf{f}} = \mathbf{A}(\boldsymbol{\lambda}) \, \mathbf{y}$$

where $\mathbf{A}(\boldsymbol{\lambda}) = (\mathbf{I}_{nM} + \boldsymbol{\Sigma}\mathbf{K})^{-1}$ is the influence or smoother matrix.

Some notes† II

- **Degrees of freedom of smooth:**

$$
\begin{aligned}
df &= \mathrm{trace}(\mathbf{A}), \\
df^{\mathrm{var}} &= \mathrm{trace}(\mathbf{A}\,\mathbf{A}^T), \\
df^{\mathrm{err}} &= nM - \mathrm{trace}(2\mathbf{A} - \mathbf{A}^T\mathbf{A}) \\
df_{(m)} &= df \text{ of } f_m \;=\; \text{sum of diagonal elements corresponding to } f_m
\end{aligned}
$$

- **Standard Errors**

$$
\begin{aligned}
\mathrm{Cov}(\widehat{\mathbf{f}}) &= \mathbf{A}(\boldsymbol{\lambda})\,\boldsymbol{\Sigma}\,\mathbf{A}(\boldsymbol{\lambda})^T \text{ is used to form pointwise SE bands} \\
\mathrm{Cov}(\widehat{\mathbf{f}}) &= \mathbf{A}(\boldsymbol{\lambda})\,\boldsymbol{\Sigma} \text{ is a Bayesian alternative. Cheaper.}
\end{aligned}
$$

VGAM algorithm: vector backfitting†

Vector backfitting is the backfitting algorithm using vector smoothing.

$$E(\mathbf{Y}|\mathbf{x}) = \mathbf{f}_t(x_t) + \sum_{k=1,\ k\neq t}^{p} \mathbf{f}_k(x_k)$$

so

$$\mathbf{f}_t(x_t) = E\left(\mathbf{Y} - \sum_{k=1,\ k\neq t}^{p} \mathbf{f}_k(x_k) \middle| X_t \right).$$

Modified vector backfitting possible—and is implemented. It decomposes

$$\boldsymbol{\eta}(x) = \mathbf{X}\,\beta + \sum_{k=1}^{p} \mathbf{r}_k(x_k)$$

i.e., into a linear and nonlinear components.

Some examples I

The following are some simple VGAM examples.

Some examples II

t

Distribution	Density function $f(y;\boldsymbol{\theta})$	Range of y	VGAM family
Negative binomial	$\binom{y+k-1}{y} \left(\frac{\mu}{\mu+k}\right)^y \left(\frac{k}{k+\mu}\right)^k$	$\{0,1,\ldots\}$	`negbinomial`
Hyperbolic secant	$\frac{\exp\{\theta y + \log(\cos(\theta))\}}{2\cosh(\pi y/2)}$	$(-\infty,\infty)$	`hypersecant`
Hyperbolic secant	$\frac{\cos(\theta)}{\pi}\ u^{-\frac{1}{2}+\frac{\theta}{\pi}}\ (1-u)^{-\frac{1}{2}-\frac{\theta}{\pi}}$	$(0,1)$	`hypersecant.1`
Inverse binomial	$\frac{\lambda\ \Gamma(2y+\lambda)\ \{\rho(1-\rho)\}^y\ \rho^\lambda}{\Gamma(y+1)\ \Gamma(y+\lambda+1)}$	$\{0,1,\ldots\}$	`invbinomial`
Reciprocal inverse Gaussian	$\sqrt{\frac{\lambda}{2\pi y}}\ \exp\left\{-\frac{\lambda(y-\mu)^2}{2y}\right\}$	$(0,\infty)$	`rig`
Leipnik (transformed)	$\frac{\{y(1-y)\}^{-\frac{1}{2}}}{\mathrm{Beta}(\frac{\lambda+1}{2},\frac{1}{2})}\left[1+\frac{(y-\mu)^2}{y(1-y)}\right]^{-\frac{\lambda}{2}}$	$(0,1)$	`leipnik`
Generalized Poisson	$\frac{\theta(\theta+y\lambda)^{y-1}}{y!}\ \exp(-y\lambda-\theta)$	$\{0,1,\ldots\}$	`genpoisson`
Simplex	$\frac{\exp\left\{-\frac{1}{2\sigma^2}\frac{(y-\mu)^2}{y(1-y)\,\mu^2\,(1-\mu)^2}\right\}}{\sqrt{2\pi\sigma^2\{y(1-y)\}^3}}$	$(0,1)$	`simplex`

Table: Dispersion models implemented in VGAM (Jørgensen, 1997).

Gamma Distributions

t

Distribution	Density function $f(y;\boldsymbol{\theta})$	Range of y	Mean	VGAM family
gamma (2-parameter)	$\frac{\lambda\,(\lambda y/\mu)^{\lambda-1}\,\exp(-\lambda y/\mu)}{\mu\,\Gamma(\lambda)}$	$(0,\infty)$	μ	gamma2
gamma (generalized)	$\frac{d b^{-dk} y^{dk-1}\,\exp(-(y/b)^d)}{\Gamma(k)}$	$(0,\infty)$	$b\,k$	gengamma
log gamma (standard)	$\frac{\exp\{ky - \exp(y)\}}{\Gamma(k)}$	$(-\infty,\infty)$	$\psi(k)$	lgammaff
log gamma (3-parameter)	$\frac{\exp\left\{\frac{k(y-a)}{b} - \exp\left(\frac{y-a}{b}\right)\right\}}{b\,\Gamma(k)}$	$(-\infty,\infty)$	$a + b\,\psi(k)$	lgamma3ff
McCullagh (1989)	$\frac{\{1-y^2\}^{\nu-\frac{1}{2}}}{B(\nu+\frac{1}{2},\frac{1}{2})\,(1-2\theta y+\theta^2)^{\nu}}$	$(-1,1)$	$\frac{\nu\,\theta}{1+\nu}$	mccullagh89
Prentice (1974)	$\frac{\lvert q\rvert\,\exp(w/q^2 - e^w)}{b\,\Gamma(1/q^2)}$, $w = (y-a)q/b + \psi(1/q^2)$	$(-\infty,\infty)$	a	prentice74

Table: Some gamma-type distributions currently supported by VGAM.

Size Distributions

t

Distribution	Density function $f(y;\boldsymbol{\theta})$	Mean (subject to range restrictions)	VGAM family
Beta II	$\dfrac{y^{p-1}}{b^p B(p,q)\,\{1+y/b\}^{p+q}}$	$\dfrac{b\,\Gamma(p+1)\,\Gamma(q-1)}{\Gamma(p)\,\Gamma(q)}$	betaII
Dagum	$\dfrac{ap\,y^{ap-1}}{b^{ap}\,\{1+(y/b)^a\}^{p+1}}$	$\dfrac{b\,\Gamma(p+a^{-1})\,\Gamma(1-a^{-1})}{\Gamma(p)}$	dagum
Fisk	$\dfrac{a\,y^{a-1}}{b^a\,\{1+(y/b)^a\}^2}$	$b\,\Gamma(1+a^{-1})\,\Gamma(1-a^{-1})$	fisk
Generalized beta II	$\dfrac{a\,y^{ap-1}}{b^{ap}\,B(p,q)\,\{1+(y/b)^a\}^{p+q}}$	$\dfrac{b\,\Gamma(p+a^{-1})\,\Gamma(q-a^{-1})}{\Gamma(p)\,\Gamma(q)}$	genbetaII
Inverse Lomax	$\dfrac{p\,y^{p-1}}{b^p\,\{1+y/b\}^{p+1}}$	NA	invlomax
Inverse paralogistic	$\dfrac{a^2\,y^{a^2-1}}{b^{a^2}\,\{1+(y/b)^a\}^{a+1}}$	$\dfrac{b\,\Gamma(a+a^{-1})\,\Gamma(1-a^{-1})}{\Gamma(a)}$	invparalogistic
Lomax	$\dfrac{q}{b\,\{1+(y/b)^a\}^{1+q}}$	$\dfrac{b}{q-1}$	lomax
Paralogistic	$\dfrac{a^2\,y^{a-1}}{b^a\,\{1+(y/b)^a\}^{1+a}}$	$\dfrac{b\,\Gamma(1+a^{-1})\,\Gamma(a-a^{-1})}{\Gamma(a)}$	paralogistic
Singh-Maddala	$\dfrac{aq\,y^{a-1}}{b^a\,\{1+(y/b)^a\}^{1+q}}$	$\dfrac{b\,\Gamma(1+a^{-1})\,\Gamma(q-a^{-1})}{\Gamma(q)}$	sinmad

Table: Kleiber and Kotz (2003) models implemented in VGAM.

Bivariate Distributions

t

Distribution	Cumulative distribution function $F(y_1, y_2; \boldsymbol{\theta})$	VGAM family
Ali-Mikhail-Haq	$\dfrac{y_1\, y_2}{1 - \alpha(1 - y_1)(1 - y_2)}$	amh
Farlie-Gumbel-Morgenstern	$y_1 y_2 \left[1 + \alpha(1 - y_1)(1 - y_2)\right]$	fgm
Frank	$\log_\alpha \left[1 + \dfrac{(\alpha^{y_1} - 1)(\alpha^{y_2} - 1)}{\alpha - 1}\right]$	frank
Gamma hyperbola	$f(\mathbf{y}) = \exp\left\{-e^{-\theta} y_1/\theta - \theta y_2\right\}$	gammahyp
Gumbel's Type I	$e^{-y_1 - y_2 + \alpha y_1 y_2} + 1 - e^{-y_1} - e^{-y_2}$	gumbelIbiv
McKay's bivariate gamma	$f(\mathbf{y}) = \dfrac{y_1^{p-1}\,(y_2 - y_1)^{q-1}\, e^{-y_2/a}}{a^{p+q}\,\Gamma(p)\,\Gamma(q)}$	bivgamma.mckay
Morgenstern	$e^{-y_1 - y_2}\left(1 + \alpha[1 - e^{-y_1}][1 - e^{-y_2}]\right) + 1 - e^{-y_1} - e^{-y_2}$	morgenstern
Plackett	$f(\mathbf{y}) = \dfrac{\psi\,[1 + (\psi - 1)(y_1 + y_2 - 2 y_1 y_2)]}{\left([1 + (\psi - 1)(y_1 + y_2)]^2 - 4\psi(\psi - 1) y_1 y_2\right)^{3/2}}$	plackett

Table: Some bivariate distributions currently supported by VGAM.

Zero-inflated, Zero-Altered and Positive Models

t

Distribution	Random variates functions
Zero-altered negative binomial	`[dpqr]zanegbin()`
Zero-altered Poisson	`[dpqr]zapois()`
Zero-inflated binomial	`[dpqr]zibinom()`
Zero-inflated geometric	`[dpqr]zigeom()`
Zero-inflated negative binomial	`[dpqr]zinegbin()`
Zero-inflated Poisson	`[dpqr]zipois()`
Positive binomial	`[dpqr]posbinom()`
Positive negative binomial	`[dpqr]posnegbinom()`
Positive normal	`[dpqr]posnorm()`
Positive Poisson	`[dpqr]pospois()`

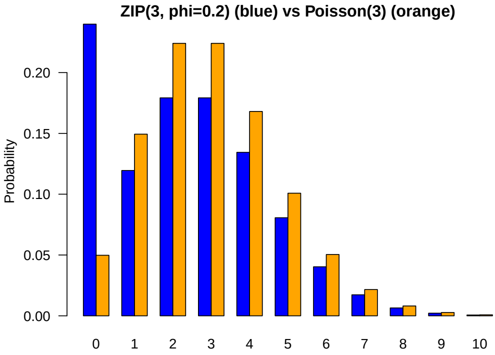
Table: Some of VGAM functions for generating random variates etc. The prefix "`d`" means the density, "`p`" means the distribution function, "`q`" means the quantile function and "`r`" means random deviates. Note: most functions have a `[dpqr]foo()` for density, distribution function, quantile function and random generation.

Zero-inflated Poisson model I

Loosely,

$$P(Y = y; \phi, \lambda) \quad = \quad \phi \, P(Y = 0) + (1 - \phi) \, \mathrm{Poisson}(\lambda).$$

where $0 < \phi < 1$ and $\lambda > 0$.



Zero-inflated Poisson model II

Example: $Y =$ the number of insects on a leaf of a particular plant (some leaves have no insects because they are unsuitable for feeding).

Let the overall proportion of such leaves be ϕ.
Actually,

$$\begin{aligned} P(Y=0) &= \phi + (1-\phi)\, e^{-\lambda}, \\ P(Y=y) &= (1-\phi)\, \frac{e^{-\lambda}\, \lambda^{y}}{y!}, \qquad y = 1, 2, \ldots, \end{aligned}$$

where $0 < \phi < 1$ and $\lambda > 0$. Then a good idea is

$$\boldsymbol{\eta} = \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix} = \begin{pmatrix} \mathrm{logit}\, \phi \\ \log \lambda \end{pmatrix}.$$

Zero-inflated Poisson model III

If $\boldsymbol{\theta} = (\phi, \lambda)^T$ then the expected information matrix (EIM) is given by

$$\begin{pmatrix} \dfrac{1 - e^{-\lambda}}{(1-\phi)(\phi + (1-\phi)\, e^{-\lambda})} & \dfrac{-e^{-\lambda}}{\phi + (1-\phi)\, e^{-\lambda}} \\ \dfrac{-e^{-\lambda}}{\phi + (1-\phi)\, e^{-\lambda}} & \dfrac{1-\phi}{\lambda} - \dfrac{\phi(1-\phi)\, e^{-\lambda}}{\phi + (1-\phi)\, e^{-\lambda}} \end{pmatrix}.$$

Here is some simulated data example.

Zero-inflated Poisson model IV

```
> set.seed(1111)
> N = 2000
> zdata = data.frame(x2 = runif(n = N))
> zdata = transform(zdata,
        phi = logit(-1 + 1*x2, inverse=TRUE),
        lambda = loge(2 - 2*x2, inverse=TRUE))
> zdata = transform(zdata,
        y = rzipois(N, lambda, phi))
> with(zdata, table(y))

y
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
868 229 218 190 122 116  91  65  45  20  19   5   7   2   1
 15  16
  1   1

> fit = vglm(y ~ x2, zipoisson, zdata, trace=TRUE)
```

Zero-inflated Poisson model V

```
VGLM    linear loop  1 :  loglikelihood = -3402.505
VGLM    linear loop  2 :  loglikelihood = -3338.938
VGLM    linear loop  3 :  loglikelihood = -3338.638
VGLM    linear loop  4 :  loglikelihood = -3338.638

> # fit = vglm(y ~ x2, zipoisson, zdata, crit = "coef", trace=TRUE)
> coef(fit, matrix=TRUE)  # These should agree with the above values

            logit(phi) log(lambda)
(Intercept) -0.9494578    2.034661
x2           0.7684691   -2.122872

> # fit2 = vglm(y ~ x2, zipoisson(shrinkage.init = 0.95), zdata, trace=TRUE
> # coef(fit2, matrix=TRUE)  # These should agree with the above values
```

Loglinear models for binary responses† I

For bivariate binary responses Y_1 and Y_2,

$$\log P(Y_1 = y_1, Y_2 = y_2 | \mathbf{x}) = u_0(\mathbf{x}) + u_1(\mathbf{x})\, y_1 + u_2(\mathbf{x})\, y_2 + u_{12}(\mathbf{x})\, y_1\, y_2 \tag{11}$$

where $y_j = 0$ or 1,

$$\begin{pmatrix} u_1(\mathbf{x}) \\ u_2(\mathbf{x}) \\ u_{12}(\mathbf{x}) \end{pmatrix} = \boldsymbol{\eta}(\mathbf{x}) = \begin{pmatrix} \eta_1(\mathbf{x}) \\ \eta_2(\mathbf{x}) \\ \eta_3(\mathbf{x}) \end{pmatrix}.$$

Loglinear models for binary responses† II

In general, suppose the data are $(y_{i1}, \ldots, y_{iS}, \mathbf{x}_i)$, $i = 1, \ldots, n$, where each y_{ij} is a binary response. Then only allow for pairwise associations:

$$\log P(Y_1 = y_1, \ldots, Y_S = y_S | \mathbf{x}) = u_0(\mathbf{x}) + \sum_{j=1}^{S} u_j(\mathbf{x})\, y_j + \sum_{j<k} u_{jk}(\mathbf{x})\, y_j\, y_k. \quad (12)$$

The normalizing parameter u_0 satisfies

$$e^{-u_0} = 1 + \sum_{j=1}^{S} e^{u_j} + \sum_{j<k} e^{u_j + u_k + u_{jk}} + \sum_{j<k<\ell} e^{u_j + u_k + u_\ell + u_{jk} + u_{j\ell} + u_{k\ell}} + \cdots + \exp\left(\sum_{j=1}^{S} u_j + \sum_{j<k} u_{jk} \right).$$

Loglinear models for binary responses† III

One has

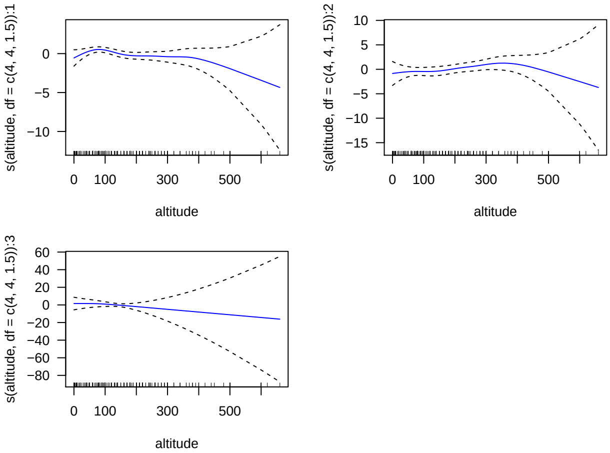
$$\boldsymbol{\eta} \;=\; (\eta_1, \ldots, \eta_M)^T \;=\; (u_1, \ldots, u_S, u_{12}, \ldots, u_{S-1,S})^T$$

where $M = S(S+1)/2$. (An identity link for each of the u's is chosen because the parameter space is unconstrained.)

With IRLS, Newton-Raphson $\equiv$ Fisher scoring.

```
> # data(hunua)
> fits = vgam(cbind(dacdac,metrob) ~ s(altitude, df=c(4,4,1.5)),
              loglinb2, hunua)
> par(mfrow=c(2,2), las=1, mar=c(5,5,1,1))
> plot(fits, se=TRUE, cex=0.9, lcol="blue")
```

Loglinear models for binary responses† IV



More complicated constraints† I

The xij facility allows for quite a lot of flexibility among all the regression coefficients, e.g., mixed logit models and nested logit models in discrete choice models. Further examples follow soon.

The crucial equation is (13):

$$\boldsymbol{\eta}_i \;=\; \sum_{k=1}^{p} \mathrm{diag}(x_{ik1}, \ldots, x_{ikM})\, \mathbf{H}_k\, \boldsymbol{\beta}_k^{*} \quad \left(= \sum_{k=1}^{p} \mathbf{X}_{(ik)}^{*}\, \mathbf{H}_k\, \boldsymbol{\beta}_k^{*}, \text{ say.} \right) \tag{13}$$

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.

More complicated constraints† II

Example 1

We assume that the data frame `myframe` has variables `x1` and `x2`.

Q1: How can we fit

$$\eta_1 = \beta_{(1)1}\, x_1 + \beta_{(1)2}\, x_2 \qquad (14)$$

$$\eta_2 = \beta_{(2)1}\, x_1 + \beta_{(2)2}\, x_2 \qquad (15)$$

subject to $\beta_{(1)1} + \beta_{(2)2} = \beta_{(2)1} + \beta_{(1)2}$?

A1:

$$\begin{aligned} \eta_2 &= \beta_{(2)1}\, x_1 + \left(\beta_{(2)1} + \beta_{(1)2} - \beta_{(1)1}\right) x_2 \\ &= \beta_{(2)1}\,(x_1 + x_2) + \beta_{(1)2}\, x_2 - \beta_{(1)1} x_2. \end{aligned}$$

More complicated constraints† III

$$\begin{aligned}\boldsymbol{\eta} &= \operatorname{diag}(x_1, x_2)\begin{pmatrix} 1 \\ -1 \end{pmatrix}\beta_{(1)1} + x_2\begin{pmatrix} 1 \\ 1 \end{pmatrix}\beta_{(1)2} + x_3\begin{pmatrix} 0 \\ 1 \end{pmatrix}\beta_3 \\ &= \operatorname{diag}(x_1, x_2)\begin{pmatrix} 1 \\ -1 \end{pmatrix}\beta^*_{(1)1} + (x_2\,\mathbf{I}_2)\begin{pmatrix} 1 \\ 1 \end{pmatrix}\beta^*_{(1)2} + (x_3\,\mathbf{I}_2)\begin{pmatrix} 0 \\ 1 \end{pmatrix}\beta^*_{(3)1}\end{aligned}$$

where $x_3 = x_1 + x_2$ and $\beta_3 = \beta_{(2)1}$, say. That is, $\beta_{(1)1} = \beta^*_{(1)1}$, $\beta_{(1)2} = \beta^*_{(1)2}$, $\beta^*_{(3)1} = \beta_3$.

```
myframe = transform(myframe, X1 = x1, x3 = x1 + x2)
Hlist = list(X1 = rbind(1, -1),
             x2 = rbind(1,  1), x3 = rbind(0, 1))
fit = vglm(y ~ -1 + X1 + x2 + x3,
           VGAMfamilyFunction, myframe,
           constraints = Hlist,
           xij = list(X1 ~ -1 + x1 + x2),
           form2 = ~ -1 + x1 + x2 + x3 + X1)
```

More complicated constraints† IV

Example 2

Q2: How can we fit

$$\begin{aligned}
\eta_1 &= \beta_{(1)1}\,x_1 + \beta_{(1)2}\,x_2 && (16)\\
\eta_2 &= \beta_{(2)1}\,x_1 + \beta_{(2)2}\,x_2 && (17)\\
\eta_3 &= \beta_{(3)1}\,x_1 + \beta_{(3)2}\,x_2 && (18)
\end{aligned}$$

subject to $\beta_{(1)1} + \beta_{(2)1} + \beta_{(3)1} = \beta_{(1)2} + \beta_{(2)2} + \beta_{(3)2}$?

A2: Now

$$\eta_3 = \beta_{(3)1}\,x_1 + \left(\beta_{(1)1} + \beta_{(2)1} + \beta_{(3)1} - \beta_{(1)2} - \beta_{(2)2}\right) x_2.$$

More complicated constraints† V

Usually (trivial constraints)

$$\boldsymbol{\eta} \;=\; x_1\,\mathbf{I}_3 \begin{pmatrix} \beta_{(1)1} \\ \beta_{(2)1} \\ \beta_{(3)1} \end{pmatrix} + x_2\,\mathbf{I}_3 \begin{pmatrix} \beta_{(1)2} \\ \beta_{(2)2} \\ \beta_{(3)2} \end{pmatrix}$$

but here,

$$\eta_3 \;=\; \beta_{(3)1}\,(x_1 + x_2) + \beta_{(1)1}\,x_2 + \beta_{(2)1}\,x_2 + \beta_{(1)2}(-x_2) + \beta_{(2)2}(-x_2).$$

So

$$\boldsymbol{\eta} \;=\; \mathbf{I}_3 \begin{pmatrix} x_1 & 0 & 0 \\ 0 & x_1 & 0 \\ x_2 & x_2 & x_1 + x_2 \end{pmatrix} \begin{pmatrix} \beta_{(1)1} \\ \beta_{(2)1} \\ \beta_{(3)1} \end{pmatrix} + \begin{pmatrix} x_2 & 0 \\ 0 & x_2 \\ -x_2 & -x_2 \end{pmatrix} \begin{pmatrix} \beta_{(1)2} \\ \beta_{(2)2} \end{pmatrix}.$$

More complicated constraints† VI

The second term is easy:

$$x_2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -1 & -1 \end{pmatrix} \begin{pmatrix} \beta_{(1)2} \\ \beta_{(2)2} \end{pmatrix}.$$

The first term is problematic if dealt with wholly. Instead, we need to break it up by columns:

$$\begin{pmatrix} x_1 \\ 0 \\ x_2 \end{pmatrix} \beta_{(1)1} + \begin{pmatrix} 0 \\ x_1 \\ x_2 \end{pmatrix} \beta_{(2)1} + \begin{pmatrix} 0 \\ 0 \\ x_1 + x_2 \end{pmatrix} \beta_{(3)1} \;=$$

$$\begin{pmatrix} x_1 & 0 & 0 \\ 0 & a_1 & 0 \\ 0 & 0 & x_2 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \beta_{(1)1} + \begin{pmatrix} a_2 & 0 & 0 \\ 0 & x_1 & 0 \\ 0 & 0 & x_2 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \beta_{(2)1} +$$

More complicated constraints† VII

$$\begin{pmatrix} a_3 & 0 & 0 \\ 0 & a_4 & 0 \\ 0 & 0 & x_3 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \beta_{(3)1}$$

where $x_3 = x_1 + x_2$ and a_j has any value. Thus $p = 4$.

More complicated constraints† VIII

Try something like

```
cmat = matrix(c(1, 0, -1,  0, 1, -1), 3, 2)
myframe = transform(myframe, X1 = x1, X2 = x1, X3 = x1,
                             x3 = x1 + x2,
                             a1 = 0 * x1, a2 = 0 * x1,
                             a3 = 0 * x1, a4 = 0 * x1)
fit = vglm(y ~ X1 + X2 + X3 + x2 - 1,
           VGAMfamilyFunction, myframe,
           constraints = list(X1 = rbind(1, 0, 1),
                              X2 = rbind(0, 1, 1),
                              X3 = rbind(0, 0, 1),
                              x2 = cmat),
           form2 = ~ X1 + X2 + X3 - 1 +
                     x1 + x2 + x3 +
                     a1 + a2 + a3 + a4,
           xij = list(X1 ~ x1 + a1 + x2 - 1,
                      X2 ~ a2 + x1 + x2 - 1,
                      X3 ~ a3 + a4 + x3 - 1))
```

More on VGAMs† I

Let's do two simultaneous logistic regressions: We will fit two species' presence/absence versus X_2 = `altitude`. Data is from 392 sites from the Hunua forest.

- `agaaus` is *Agathis australis*, better known as "Kauri".
- `kniexc` is *Knightia excelsa*, or "Rewarewa".

```
> # nrow(hunua)
> fit2 = vgam(cbind(agaaus, kniexc) ~ s(altitude, df = c(2, 3)),
             binomialff(mv = TRUE), hunua)
> round(coef(fit2, mat = TRUE), dig=6)   # Not really interpretable

                          logit(E[agaaus]) logit(E[kniexc])
(Intercept)                      -1.308694        -0.073857
s(altitude, df = c(2, 3))         0.000122         0.002744

> #coef(fit2, mat = TRUE)   # Not really interpretable
> plot(fit2, se = TRUE, overlay = TRUE, lcol = c("blue", "orange"),
      scol = c("blue", "orange"))
```

The output to `coef()` is not really interpretable; they are the coefficients to the linear part of the fit.

More on VGAMs† II

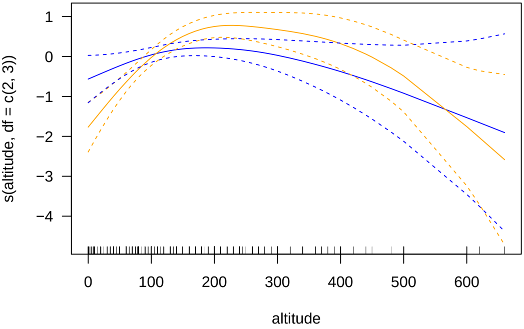


Figure: Two nonparametric logistic regressions fitted as a VGAM. Blue is Kauri, orange is Rewarewa.

More on VGAMs† III

Now notice the O-splines here

```
> fit2@Bspline

$`s(altitude, df = c(2, 3))`
An object of class "vsmooth.spline.fit"
Slot "Bcoefficients":
              [,1]        [,2]
 [1,] -0.545752005 -1.32505661
 [2,] -0.543526883 -1.31945612
 [3,] -0.532401279 -1.29145370
 [4,] -0.514600728 -1.24665158
 [5,] -0.483461950 -1.16830000
 [6,] -0.450130888 -1.08448848
 [7,] -0.401395611 -0.96221952
 [8,] -0.357345763 -0.85219250
 [9,] -0.302986231 -0.71733776
[10,] -0.260222187 -0.61218805
[11,] -0.197658830 -0.45944000
[12,] -0.147647842 -0.33773620
[13,] -0.094891851 -0.20881150
[14,] -0.067812019 -0.14250131
[15,] -0.033036408 -0.05701255
[16,]  0.003421857  0.03312613
[17,]  0.041013992  0.12713371
[18,]  0.075331053  0.21431114
[19,]  0.100125600  0.27843259
[20,]  0.133206057  0.36579467
[21,]  0.156166632  0.42873344
[22,]  0.185364953  0.51403679
```

More on VGAMs† IV

```
[23,]  0.202522534  0.57150973
[24,]  0.213323031  0.62056162
[25,]  0.215241541  0.63970159
[26,]  0.213283985  0.65107630
[27,]  0.206914270  0.64577245
[28,]  0.190703158  0.61480542
[29,]  0.171845576  0.57357295
[30,]  0.147476587  0.52270725
[31,]  0.119320416  0.46710009
[32,]  0.084921160  0.40655642
[33,]  0.024948684  0.31190939
[34,] -0.044699535  0.21367149
[35,] -0.149124456  0.06944959
[36,] -0.234927976 -0.05584417
[37,] -0.326203264 -0.19816957
[38,] -0.405866064 -0.32832379
[39,] -0.524187668 -0.53272684
[40,] -0.668164989 -0.79889328
[41,] -1.014853850 -1.49298616
[42,] -1.458548841 -2.62195642
[43,] -1.840213406 -3.62007860
[44,] -1.967465889 -3.95187756

Slot "knots":
 [1] 0.000000000 0.000000000 0.000000000 0.000000000
 [5] 0.001515152 0.007575758 0.012121212 0.022727273
 [9] 0.030303030 0.045454545 0.053030303 0.068181818
[13] 0.075757576 0.098484848 0.106060606 0.118181818
[17] 0.121212121 0.136363636 0.151515152 0.159090909
```

More on VGAMs† V

```
[21] 0.174242424 0.181818182 0.204545455 0.212121212
[25] 0.242424242 0.257575758 0.280303030 0.287878788
[29] 0.318181818 0.333333333 0.363636364 0.378787879
[33] 0.393939394 0.424242424 0.439393939 0.484848485
[37] 0.515151515 0.560606061 0.575757576 0.606060606
[41] 0.636363636 0.681818182 0.727272727 0.909090909
[45] 1.000000000 1.000000000 1.000000000 1.000000000

Slot "xmin":
s(altitude, df = c(2, 3))
                        0

Slot "xmax":
s(altitude, df = c(2, 3))
                      660
```

More on VGAMs† VI

```
>       ooo = with(hunua, order(altitude))
>       with(hunua, matplot(altitude[ooo], fitted(fit2)[ooo,], ylim = c(0, .8),
            xlab = "Altitude (m)", ylab = "Probability of presence", las = 1,
            col = c("blue", "orange"),
            main = "Two plant species' response curves", type = "l", lwd = 2))
>       with(hunua, rug(altitude))
```

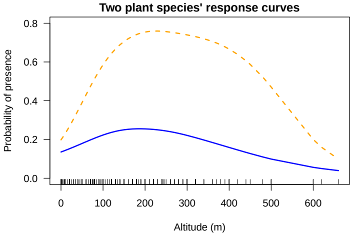


Figure: Two nonparametric logistic regressions fitted as a VGAM.

Working weights† I

Each $\mathbf{W}_i$ needs to be positive-definite.

1. *Expected information matrix (EIM)* is often positive-definite over a larger parameter space than the *observed information matrix (OIM)*. But the EIM may be intractable.
2. Under mild regularity conditions,

$$\mathrm{Var}\left(\frac{\partial \ell_i}{\partial \boldsymbol{\theta}}\right) \quad = \quad -E\left(\frac{\partial^2 \ell_i}{\partial \boldsymbol{\theta}\, \partial \boldsymbol{\theta}^T}\right).$$

Often the score vector is easy. Use random variates to compute the sample variance of the score vector. The `nsimEIM` argument implements this.

Working weights† II

For example, the negative binomial has

$$\frac{\partial^2 \ell_i}{\partial k^2} \;=\; \psi'(y_i + k) - \psi'(k) \;=\; -\sum_{r=0}^{y_i - 1} (k + r)^{-2},$$

where $\psi'(z)$ is the trigamma function (the digamma function $\psi(z) = \Gamma'(z)/\Gamma(z)$). Its expected value involves an infinite series

$$I_{k,k} \;=\; k^4 \left(\sum_{j=0}^{\infty} (k + j)^{-2} \, P(Y_i \geq j) - \frac{\mu_i / k}{k + \mu_i} \right). \qquad (19)$$

The `weight` slot of `negbinomial()`:

Working weights† III

```
weight = eval(substitute(expression({
    wz = matrix(as.numeric(NA), n, M)  # wz is 'diagonal'
    run.varcov = matrix(0, n, NOS)
    ind1 = iam(NA, NA, M = M, both = TRUE, diag = TRUE)
    for(ii in 1:( .nsimEIM )) {
        ysim = rnbinom(n = n*NOS, mu = c(mu), size = c(kmat))
        dl.dk = digamma(ysim+kmat) - digamma(kmat) -
                (ysim+kmat)/(mu+kmat) + 1 + log(kmat/(kmat+mu))
        run.varcov = run.varcov + dl.dk^2
    }
    run.varcov = cbind(run.varcov / .nsimEIM)
# Can do even better if it is an intercept-only model
    wz[,2*(1:NOS)] = if(intercept.only)
        matrix(colMeans(run.varcov),
               n, ncol(run.varcov), byrow = TRUE) else run.varcov

    wz[,2*(1:NOS)] = wz[,2*(1:NOS)] * dk.deta^2
# The 1-1 element (known exactly):
    ed2l.dmu2 = 1/mu - 1/(mu+kmat)
    wz[,2*(1:NOS)-1] = dmu.deta^2 * ed2l.dmu2
    w * wz
}), list( .cutoff = cutoff, .Maxiter = Maxiter, .nsimEIM = nsimEIM )))
```

RR-VGLMs I

Recall $\boldsymbol{\eta} = \mathbf{B}^T \mathbf{x}$ for VGLMs.

Motivation: if M and p are large then $\mathbf{B}$ is "too big" (Mp elements).

Idea: approximate part of $\mathbf{B}$ by a lower rank matrix, i.e, the product of two 'thin' matrices $\mathbf{A}\,\mathbf{C}^T$.

Great for dimension reduction! Can do biplots etc.

A simple and important result

Solving by an *alternating algorithm* shows that RR-VGLMs are VGLMs where the constraint matrices are unknown and to be estimated.

RR-VGLMs II

Some special cases

1. A *reduced-rank multinomial logit model* is a *stereotype model* (Anderson, 1984).
2. A *reduced-rank negative binomial distribution* has variance function

$$\mathrm{Var}(Y) \;=\; \mu + \delta_1\, \mu^{\delta_2}$$

for parameters δ_1 and δ_2.

```
rrvglm(y ~ x2 + ··· + xp, negbinomial(zero = NULL))
```

3. A *reduced-rank zero-inflated Poisson distribution* is known as a constrained zero-inflated generalized additive model (Liu and Chan, 2010, COZIGAM). Examples: trawl survey studies, grasshopper species' abundances.

```
rrvglm(y ~ x2 + ··· + xp, zipoisson(zero = NULL))
```

Constrained Quadratic Ordination (CQO)† I

Example: Hunting spider data

Data consists of abundances (numbers trapped over a 60 week period) of $S = 12$ species of hunting spiders in a Dutch dune area.

Have $n = 28$ and $p_2 = 6$.

Table: Hunting spiders data. Log transformed environmental variables.

R variable	Description	
`WaterCon`	Water Content	Percentage of soil dry mass
`BareSand`	Bare Sand	Percentage cover of bare sand
`FallTwig`	Fallen Twigs	Percentage cover of fallen leaves and twigs
`CoveMoss`	Cover Moss	Percentage cover of the moss layer
`CoveHerb`	Cover Herbs	Percentage cover of the herb layer
`ReflLux`	Light Refl	Reflection of the soil surface with cloudless sky

Constrained Quadratic Ordination (CQO)† II

Table: Hunting spiders data. Species' abbreviations.

R variable	Species name
`Alopacce`	*Alopecosa accentuata*
`Alopcune`	*Alopecosa cuneata*
`Alopfabr`	*Alopecosa fabrilis*
`Arctlute`	*Arctosa lutetiana*
`Arctperi`	*Arctosa perita*
`Auloalbi`	*Aulonia albimana*
`Pardlugu`	*Pardosa lugubris*
`Pardmont`	*Pardosa monticola*
`Pardnigr`	*Pardosa nigriceps*
`Pardpull`	*Pardosa pullata*
`Trocterr`	*Trochosa terricola*
`Zoraspin`	*Zora spinimana*

Constrained Quadratic Ordination (CQO)† III

VGAM has the data in a data frame called `hspider`. A rank-1 Poisson CQO with equal tolerances can be obtained from

```
> data(hspider)
> set.seed(111)
> hspider[,1:6] = scale(hspider[,1:6]) # Standardize environ vars
> p1et = cqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute,
                   Arctperi, Auloalbi, Pardlugu, Pardmont,
                   Pardnigr, Pardpull, Trocterr, Zoraspin) ~
           WaterCon + BareSand + FallTwig +
           CoveMoss + CoveHerb + ReflLux,
        family = quasipoissonff, data = hspider,
        Crow1pos =FALSE,
 #      EqualTolerances=FALSE, ITolerances=FALSE,
        trace=FALSE)
> persp(p1et,
        main="Hunting spider data",
        col=1:ncol(p1et@y), llwd=2, las=1, llty=1)
```

Constrained Quadratic Ordination (CQO)† IV

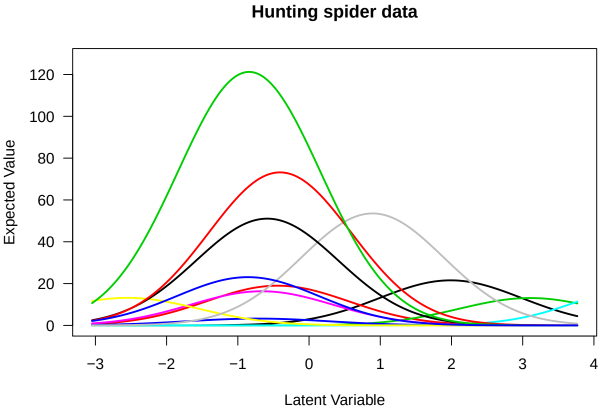


Figure: Rank-1 Poisson QRR-VGLM with equal tolerances.

Constrained Quadratic Ordination (CQO)† V

```
> summary(p1et)

Call:
cqo(formula = cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi,
    Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpull, Trocterr,
    Zoraspin) ~ WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb +
    ReflLux, family = quasipoissonff, data = hspider, Crow1pos = FALSE,
    trace = FALSE)

C matrix (constrained/canonical coefficients)
                 lv
WaterCon -0.3572911
BareSand  0.5534843
FallTwig -0.9204249
CoveMoss  0.3170025
CoveHerb -0.3055749
ReflLux   0.7022278

B1 and A matrices
```

Constrained Quadratic Ordination (CQO)† VI

```
                  (Intercept)          A
log(E[Alopacce])   1.0946607  1.9879733
log(E[Alopcune])   2.8465611 -0.4417111
log(E[Alopfabr])  -2.2453933  3.1040488
log(E[Arctlute])   0.9216726 -0.7216428
log(E[Arctperi])  -8.6858178  4.8349970
log(E[Auloalbi])   2.5729096 -0.6633768
log(E[Pardlugu])  -0.6767298 -2.5515725
log(E[Pardmont])   3.5830119  0.8916557
log(E[Pardnigr])   3.7606464 -0.5870566
log(E[Pardpull])   4.2098964 -0.4076585
log(E[Trocterr])   4.4439428 -0.8412503
log(E[Zoraspin])   2.7697050 -0.8593639

Optima and maxima
            Optimum   Maximum
Alopacce  1.9879733 21.556547
Alopcune -0.4417111 18.993855
Alopfabr  3.1040488 13.094160
```

Constrained Quadratic Ordination (CQO)† VII

```
Arctlute -0.7216428   3.261074
Arctperi  4.8349970  20.141453
Auloalbi -0.6633768  16.329021
Pardlugu -2.5515725  13.177771
Pardmont  0.8916557  53.545775
Pardnigr -0.5870566  51.058094
Pardpull -0.4076585  73.184902
Trocterr -0.8412503 121.242640
Zoraspin -0.8593639  23.079810

Tolerance
         lv
Alopacce  1
Alopcune  1
Alopfabr  1
Arctlute  1
Arctperi  1
Auloalbi  1
Pardlugu  1
```

Constrained Quadratic Ordination (CQO)† VIII

```
Pardmont  1
Pardnigr  1
Pardpull  1
Trocterr  1
Zoraspin  1

Standard deviation of the latent variables (site scores)
      lv
2.371051

Dispersion parameters:
    Alopacce     Alopcune     Alopfabr     Arctlute
3.486247e+00 7.725849e+00 4.306666e+00 1.953142e+00
    Arctperi     Auloalbi     Pardlugu     Pardmont
8.560377e-01 4.770231e+00 1.356310e+05 1.569950e+01
    Pardnigr     Pardpull     Trocterr     Zoraspin
1.235828e+01 6.590131e+00 4.100787e+01 3.089348e+00
```

Concluding remarks

1. VGLMs and VGAMs fit a very large class of models. VGLMs are model-driven while VGAMs are data-driven.
2. The framework is purposely general. *More general $\Longrightarrow$ more useful.*
3. VGAM is freely available on CRAN or at the author's web page.

Tēnā koutou katoa

谢谢！

Xiè xie nǐ

That's all folks!