VGAM Reference Card

by T. W. Yee, t.yee@auckland.ac.nz, 2018-03-27
See `http://www.stat.auckland.ac.nz/~yee` for the VGAM package and my book's homepage. This document is current for version VGAM-1.0-6.

A book on the entire statistical framework, including the theory[1] and applications and the software, is Yee (2015). The book is the most comprehensive description of everything within a single document, and the order of the functions listed here roughly follows the book order.
An alternative is to look at quite a few journal articles: Yee and Wild (1996), Yee (1998), Yee and Hastie (2003), Yee (2004*a*), Yee (2004*b*), Yee (2006), Yee and Stephenson (2007), Yee (2008), Yee and Dirnböck (2009), Yee (2010*a*), Yee (2010*b*), Yee (2014*b*), Yee et al. (2015), Yee and Hadi (2014), Yee (2016), Yee (2018).

Regardless of book or journal articles, please cite the appropriate references if you use the software!

The website
`www.stat.auckland.ac.nz/~yee/VGAM/prerelease` has the latest version of VGAM (source and Windows only). A companion package is VGAMdata, which is also available at `http://www.stat.auckland.ac.nz/~yee`.

DISCLAIMER: VGAM is under continual development, meaning that as well as new features being added and bugs found on a regular basis, changes of all sorts are occurring all the time. For example, function and argument names may change at any time, as well as default values of arguments; see the `NEWS` and `ChangeLog` files. The VGAM package is available on a use-at-your-own-risk basis: the Author assumes no liability for loss or damage of any kind resulting from its use.

Note that, in VGAM 1.0-0, the first and second derivatives of link functions have changed with respect to the argument `inverse`. Models fitted prior to VGAM 1.0-0 may need to be re-run.

Note that, in VGAM 0.9-5, numerous argument names, their order, and function names are changed. This was due to inconsistencies I detected while writing my book, and it was deemed necessary to standardize things. This particularly relates to family functions implementing discrete and continuous distributions. I have tried to summarize all changes in the `NEWS` file, however, a few changes might have gone undocumented. Users of previous versions of VGAM are cautioned to check their code. My apologies for this inconvenience.

Another R package VGAMextra, by Víctor Miranda, contains additional VGAM family and link functions, etc.

[1] A small resume is given below.

Modelling functions

The following functions form the heart of the VGAM package and use `formula` and `family` arguments, e.g., **`vglm(y ~ 1, family = maxwell, data = mdata)`**.

`vglm()` Vector generalized linear models.

`vgam()` Vector generalized additive models. See below for some more details.

`rrvglm()` Reduced-rank VGLMs (same as constrained linear ordination, or CLO).

`cqo()` Constrained quadratic ordination (QRR-VGLM).

`cao()` Constrained additive ordination (RR-VGAM).

`rcim(table or matrix)` Row–column interaction models.

The following functions do *not* use the **`formula, family =`** arguments.

`grc(table or matrix)` Goodman's RC model.

`vsmooth.spline(x, y, w, ...)` Vector smoothing spline.

Smoothing

`vgam()` can be used with three smoothers. They are as follows.

`s()` Generation-1 VGAMs. This is Yee and Wild (1996). Uses backfitting. Cannot handle constraint matrices whose columns are not orthogonal (try **`is.buggy()`**).

`sm.os()` and **`sm.ps()`**. O-splines and P-splines as penalized VGAMs. Generation-2 VGAMs. This is Yee et al. (2018). Does not use backfitting. Automatic smoothing parameter selection. Calls mgcv (Wood, 2004, 2017).

Of course, an alternative is to use regression splines, i.e., **`bs()`** and **`ns()`** and can be used with **`vglm()`**. The component functions can be plotted, e.g., **`plot(as(fit, "vgam"))`**.

BTW, **`vsmooth.spline()`** fits a vector (cubic) smoothing spline to scatter plot data.

Useful controlling options for the modelling functions

`trace = TRUE` Print a running log of the estimation.

`criterion = "coef"` Criterion on which to test convergence.

`maxit = 40` Maximum number of iterations allowed.

`coefstart`, **`etastart`**, **`mustart`** Starting values for β, η_i ($n \times M$ matrix), μ_i respectively. For example, **`vglm(y ~ x2, maxwell, data = mdata, etastart = predict(simplerModel))`**.

The following arguments are standard to formula-based modelling functions.

`data =` Data frame with the formula variables.

`subset =` Vector of logicals.

`na.action =` What to do with missing values. `"na.fail"` causes an error, `"na.omit"` deletes rows. Can be assigned a user-defined function.

Extractor functions

`AIC()` The Akaike information criterion or AIC.

`AICc()` Corrected AIC.

`BIC()` The Bayesian (Schwarz's) information criterion or BIC.

`class()` The object's class.

`coef()` Regression coefficients (the β_k^* in (1) but enumerated in a different order).

`Coef()` Regression coefficients, especially if the formula comprises of intercept only, i.e., **`~ 1`**.

`constraints()` Constraint matrices, $\mathbf{H}_k$.

`depvar()` Dependent variable (response), $\mathbf{Y}$.

`deviance()` Deviance, D.

`df.residual()` Residual degrees of freedom.

`familyname()` Name of the family function.

`fitted()` Fitted values, usually $\widehat{\mu}_i$.

`has.intercept()` Does the model's formula have an intercept term?

`hatvalues()` Hat (or projection) matrix, $\mathbf{H}$.

`is.bell()` Are the response curve of a CQO bell-shaped?

`is.parallel()` Are the $\mathbf{H}_k = \mathbf{1}_M$?

`is.zero()` Are the η_j intercept-only?

`logLik()` Log-likelihood, ℓ.

`lrtest(model1, model2)` Likelihood ratio test (LRT).

`lrt.stat()` Likelihood ratio test statistics for testing $H_0 : \beta_{(j)k}^* = \theta_{(j)k}^0$ versus $H_1 : \beta_{(j)k}^* \neq \theta_{(j)k}^0$.

`linkfun()` Parameter link functions.

`model.matrix()` The big model matrix $\mathbf{X}_{\text{VLM}}$. This is never smaller than the **`lm`**-type model matrix.

`nobs()` Number of observations, n_{VLM} or n_{LM}.

`npred()` Number of linear/additive predictors η_j, M.

`QR.Q()` The $\mathbf{Q}$ matrix in the QR decomposition of the relevant model matrix used in the IRLS algorithm.

`QR.R()` The $\mathbf{R}$ matrix in the QR decomposition of the relevant model matrix used in the IRLS algorithm.

`predict()` $n \times M$ matrix of $\eta_j(x_i)$; also prediction of μ.

`print()` Print the model.

`Rank()` Rank of a reduced-rank (ordination) model, R.

`resid()` Residuals (working, Pearson, deviance, response).

`responseName()` The name of the response, as a character string.

`score.stat()` Rao's score test (Lagrange multiplier) statistics for testing $H_0 : \beta_{(j)k}^* = \theta_{(j)k}^0$ versus $H_1 : \beta_{(j)k}^* \neq \theta_{(j)k}^0$.

`show()` Show.

`summary()` Summary.

`terms()` Terms.

`term.names()` The model's formula: each term is converted to a character and they are all placed in a vector.

`TIC()` Takeuchi's information criterion or TIC.

`vcov()` Variance-covariance matrix, $\widehat{\text{Var}}(\widehat{\beta})$.

`wald.stat()` Wald test statistics for testing $H_0 : \beta_{(j)k}^* = \theta_{(j)k}^0$ versus $H_1 : \beta_{(j)k}^* \neq \theta_{(j)k}^0$. Then $\text{Var}(\widehat{\beta}_{(j)k}^*$ can be estimated at $\theta_{(j)k}^0$. Does not suffer from the HDE.

`weights()` Prior weights w_i, and working weights $(w_i \mathbf{W}_i)$, in matrix-band format.

`which.etas()` For variable x_k which η_js are modelled using that covariate?

`which.xij()` For each variable x_k does it represent an `xij` term?

Plotting functions

Not all of the following apply to a given fitted model.

`biplot()` Biplot for RR-VGLMs.

`deplot()` Density plot, e.g., for quantile regression.

`lvplot()` Latent variable plot (ordination diagram).

`persp()` Perspective (3-D; sometimes 2-D) plot.

`plot()` General plotting function.

`guplot()` Gumbel plot, e.g., for extreme values regression.

`meplot()` Mean excess plot, e.g., for extreme values regression.

`qtplot()` Quantile plot, e.g., for quantile regression.

`rlplot()` Return level plot, e.g., for extreme values regression.

`trplot()` Trajectory plot, e.g., for constrained ordination.

Link functions

Most parameters θ_j are transformed into a linear/additive predictor $\eta_j = \beta_j^T x$ or $\eta_j = \sum_{k=1}^{p} f_{(j)k}(x_k)$. All logarithms are to base e unless specified otherwise.

Usage: use full link function name (in quotes is ok), e.g., **`vglm(y ~ 1, family = maxwell(link = "identitylink"))`**.

Note: one day these functions might be renamed so that they end in "`link`", e.g., `loglink()`.

`loge()` Log, $\log(\theta)$, $\theta > 0$.

`logoff()` Log with an offset, $\log(\theta + A)$, $\theta + A > 0$.

`loglog()` Log-log, $\log\log(\theta)$, $\theta > 1$.

`cloglog` Complementary log-log, $\log(-\log(1-\theta))$, $0 < \theta < 1$.

`golf()` Gamma-ordinal link function, $0 < \theta < 1$.

`polf()` Poisson-ordinal link function, $0 < \theta < 1$.

`powerlink()` Power link function, $0 < \theta < \infty$.

`nbolf()` Negative binomial-ordinal link function, $0 < \theta < 1$.

`logc()` Complementary log, $\log(1-\theta)$, $\theta < 1$.

`logit()` Logit, $\log \frac{\theta}{1-\theta}$, $0 < \theta < 1$.

`extlogit()` Extended logit, $\log \frac{\theta - A}{B - \theta}$, $A < \theta < B$.

`foldsqrt()` Folded square root link function, $A < \theta < B$.

`probit()` Probit, $\Phi^{-1}(\theta)$, $0 < \theta < 1$.

`cauchit()` Cauchit, $\tan(\pi(\theta - \frac{1}{2}))$, $0 < \theta < 1$.

`identitylink()` Identity, θ.

`multilogit()` Multinomial logit link, $\log \frac{\theta_j}{\theta_{M+1}}$, $0 < \theta_j < 1$, $j = 1, \ldots, M$.

`nbcanlink()` Negative binomial canonical link, $\log(\theta/(\theta + k))$, where k is known.

`negidentity()` Negative-identity, $-\theta$.

`reciprocal()` Reciprocal, $1/\theta$, $\theta \neq 0$.

`negreciprocal()` Negative-reciprocal, $-1/\theta$, $\theta \neq 0$.

`fisherz()` Fisher's Z-transformation, $\frac{1}{2} \log \frac{1+\theta}{1-\theta}$, $-1 < \theta < 1$.

`rhobit()` Twice Fisher's Z-transformation, $\log \frac{1+\theta}{1-\theta}$.

Utility functions

`anova.vglm()` Analysis of deviance table for one or more VGLM fits. Allows the hypothesis testing of terms and between models.

`dtheta.deta()` $d\theta_j/d\eta_j$.

`d2theta.deta2()` $d^2\theta_j/d\eta_j^2$.

`d3theta.deta3()` $d^3\theta_j/d\eta_j^3$.

`eta2theta()` $\theta_j = g^{-1}(\eta_j)$.

`hdeff()` Detection function for the Hauck and Donner (1977) effect (HDE).

`margeff()` Marginal effects $\partial p_j(x_i)/\partial x_{ik}$, is a $p \times (M+1) \times n$ array. Available for most categorical data analysis families only.

`m2adefault()` Conversion from weight matrices (matrix-band) format to array format.

`ordsup()` Ordinal superiority measures for the LM and cumulative link models.

`R2latvar()` R^2 for latent variable models—a goodness of fit measure. Available for some categorical data analysis families only, e.g., parallel cumulative logit model.

`Select()` Selects columns from a data frame, can output formulas. A little similar to **`subset()`**.

`simulate()` Simulates new observations coming from the fitted model (for selected families only).

`theta2eta()` $\eta_j = g(\theta_j)$.

Common arguments in family functions

`zero` allows for $\eta_j = \beta_{(j)1}$, i.e., intercept-only. Can be assigned a vector with values from the set $\{1, 2, \ldots, M\}$. Negative values allowed for multiple responses. A `NULL` means *all* the η_j are modelled as functions of the covariates. Argument `zero` can also be assigned negative values, which means a form of recycling over multiple responses, e.g., **`vglm(cbind(y1, y2) ~ x2, family = uninormal(zero = -2))`** means that both responses' σ parameters are intercept-only. As of VGAM 1.0-1 and later, argument `zero` can also be assigned a vector of character strings, e.g., `zero = c("scale", "shape")` which means that all scale and shape parameters will be modelled as intercept-only.

`exchangeable` For some s and t, $\eta_s - \eta_t = 0$ or some constant. That is, η_s and η_t differ at most by an intercept.

`parallel` If `TRUE`, for all k, $\beta_{(s)k} = \beta_{(t)k}$ for some s and t; i.e., the slope for X_k of some of the linear/additive predictors are parallel or differ by a constant.

`parallel = FALSE ~ X1 + X2 - 1` means all terms, except for X_1 and X_2, have parallel slopes.

Next is a listing of VGAM family functions. These are assigned to the **family =** argument to the modelling functions, e.g., **vglm(y ~ x2 + x3, family = multinomial)**.

For distributions, the characters "dpqr" are placed within the parentheses to denote whether dpqr-type functions are available (these stand for density functions, cumulative distribution functions and quantile functions, and random variates generation, respectively). For example, **binomialff([dpqr])**, **sinmad(dpqr)**, **tobit(dpqr)**. Those dpqr-type functions from base R are wrapped in brackets [].

Generalized linear models

binomialff([dpqr]) Binomial. For multivariate responses use **binomialff(mv = TRUE)**.

quasibinomialff() Binomial with dispersion parameter to be estimated.

gaussianff([dpqr]) Gaussian or normal.

inverse.gaussianff([dpr]) Inverse Gaussian.

poissonff([dpqr]) Poisson.

quasipoissonff() Poisson with dispersion parameter to be estimated.

quasiff() Quasi- family (not working yet).

LM and binomial variants

normal.vcm() Linear model with varying-coefficients.

SURff() Seemingly unrelated regressions.

rrar() Reduced-rank autoregressive model for multiple time series.

binom2.or(dr) Bivariate logistic/probit/...odds ratio model for two binary responses.

binom2.rho(dr) Bivariate probit model for two binary responses. Based on a standard N_2 with correlation parameter ρ.

loglinb2()/loglinb3() Loglinear model for two/three binary responses.

Univariate discrete distributions

[dpqr]benf() 0-parameter Benford distribution.

betabinomial(dpr) Beta-binomial distribution.

betabinomialff(dpr) Beta-binomial distribution.

betageometric(dpr) Beta-geometric distribution.

borel.tanner(dr) Borel-Tanner distribution.

diffzeta(dpqr) Differenced zeta distribution.

double.expbinomial() Double exponential binomial distribution.

genpoisson(d) Generalized Poisson distribution.

geometric([dpqr]) Geometric distribution.

hzeta(dpqr) Haight's zeta function.

logff(dpqr) Logarithmic distribution.

inv.binomial() Inverse binomial distribution.

negbinomial([dpqr]) Negative binomial distribution with parameters μ and k.

negbinomial.size([dpqr]) Negative binomial distribution with parameter μ and *known* k.

-polono(dpr) Poisson-lognormal distribution (no family function available yet.)

polya([dpqr]) Pólya (negative binomial) distribution with parameters p and k.

seq2binomial() 2-stage sequential binomial distribution.

truncgeometric() Truncated geometric distribution.

yulesimon(dpr) Yule-Simon distribution.

zetaff(dpqr) Zeta distribution.

zipf(dpqr) Zipf distribution.

Univariate continuous distributions

alaplace[123](dpqr) Asymmetric Laplace distribution.

benini(dpqr) 1-parameter Benini distribution.

betaR([dpqr]) 2-parameter beta distribution (shape parameterization).

betaff([dpqr]) 2-parameter beta distribution (mean and precision parameterization).

betaprime() 2-parameter beta-prime distribution.

betaII() 3-parameter beta II distribution.

bisa(dpqr) Birnbaum-Saunders distribution.

cardioid(dpqr) Cardioid distribution.

cauchy([dpqr]) 2-parameter Cauchy distribution.

cauchy1([dpqr]) 1-parameter Cauchy distribution.

chisq([dpqr]) Chi-squared distribution.

cens.gumbel() Censored Gumbel distribution.

cens.normal() Censored univariate normal distribution (see also **tobit(dpqr)**).

cens.rayleigh() Censored Rayleigh distribution.

dagum(dpqr) 3-parameter Dagum distribution.

double.cens.normal() Double censored 2-parameter univariate normal distribution.

erlang() Erlang distribution.

expexp() 2-parameter Exponentiated exponential distribution.

expexp1() 2-parameter Exponentiated exponential distribution (using a profile (concentrated) likelihood).

explogff(dpqr) Exponential logarithmic distribution.

exponential([dpqr]) Exponential distribution.

fff([dpqr]) F-distribution.

fisk(dpqr) 2-parameter Fisk distribution.

foldnormal(dpqr) Folded normal distribution (univariate and generalized).

frechet(dpqr) 2-parameter Fréchet distribution.

gamma1([dpqr]) 1-parameter gamma distribution.

gamma2([dpqr]), **gammaR[dpqr]()** 2-parameter gamma distribution.

genbetaII(d) 4-parameter generalized beta II distribution.

gengamma.stacy(dpqr) Generalized gamma distribution.

gompertz(dpqr) 2-parameter Gompertz distribution.

hypersecant(), **hypersecant01()** Hyperbolic secant distribution.

inv.gaussianff(dpr) 2-parameter inverse Gaussian distribution.

inv.lomax(dpqr) 2-parameter inverse Lomax distribution.

inv.paralogistic(dpqr) 2-parameter inverse paralogistic distribution.

kumar(dpqr) Kumaraswamy distribution.

laplace(dpqr) Laplace distribution.

leipnik() Leipnik distribution.

levy(dpqr) Lévy distribution.

lgamma1(dpqr) 1-parameter log-gamma distribution.

lgamma3(dpqr) 3-parameter log-gamma distribution.

lindley(dpr) Lindley distribution.

lino(dpqr) 3-parameter generalized beta distribution (Libby and Novick, 1982).

logistic1([dpqr]) 1-parameter logistic distribution.

logistic([dpqr]) 2-parameter logistic distribution.

lognormal([dpqr]) 2-parameter lognormal distribution.

lognormal3([dpqr]) 3-parameter lognormal distribution.

lomax(dpqr) Lomax distribution.

makeham(dpqr) Makeham distribution.

maxwell(dpqr) Maxwell distribution.

mccullagh89() McCullagh (1989) distribution.

nakagami(dpqr) Nakagami distribution.

paralogistic(dpqr) 2-parameter paralogistic distribution.

paretoff(dpqr) Pareto distribution (Pareto(I)).

paretoIV(dpqr) Pareto(IV) distribution.

paretoIII(dpqr) Pareto(III) distribution.

paretoII(dpqr) Pareto(II) distribution.

perks(dpqr) Perks' distribution.

poisson.points(d) Distances to a fixed point, in a Poisson plane or volume.

rayleigh(dpqr) Rayleigh distribution.

riceff(dpqr) Rice distribution.

rigff() Reciprocal inverse Gaussian distribution.

sc.studentt2(dpqr) Scaled Student's t_2 distribution.

simplex(dr) 2-parameter simplex distribution.

sinmad(dpqr) 3-parameter Singh-Maddala distribution.

skewnormal(dr) 1-parameter univariate skew-normal distribution.

studentt([dpqr]), **studentt2([dpqr])**, **studentt3([dpqr])** Student t distribution.

tikuv(dpqr) Short-tailed symmetric distribution of Tiku and Vaughan (1999).

tobit(dpqr) Tobit model (censored normal; see also **cens.normal()**).

topple(dpqr) Topp-Leone distribution.

triangle(dpqr) Triangle distribution.

truncpareto(dpqr) Truncated (upper) Pareto distribution (Pareto(I)).

truncweibull() Truncated Weibull distribution.

uninormal([dpqr]) 2-parameter univariate normal distribution.

vonmises() von Mises distribution.

waldff() Standard Wald distribution.

weibullR() 2-parameter Weibull distribution.

weibull.mean() 2-parameter Weibull distribution (parameterized in terms of its mean).

Bivariate distributions

bilogistic4(dpr) 4-parameter bivariate logistic distribution.

amh(dpr) Ali-Mikhail-Haq's bivariate distribution.

biclaytoncop(dr) Bivariate Clayton copula distribution.

bifrankcop(dpr) Bivariate Frank's copula distribution.

bistudentt(d) Bivariate Student-t distribution.

binormal(dpr) Bivariate normal distribution, $N_2(\mu_1,\mu_2,\sigma_{11},\sigma_{22},\rho)$.

binormalcop(dp) (Bivariate) Gaussian copula distribution, $F(y_1,y_2;\rho)=\Phi_2(\Phi^{-1}(y_1),\Phi^{-1}(y_2);\rho)$.

fgm(dpr) Farlie-Gumbel-Morgenstern's bivariate distribution

freund61() Freund's (1961) bivariate extension of the exponential distribution.

gammahyp() 1-parameter gamma hyperbola bivariate distribution.

bigamma.mckay() McKay's bivariate gamma distribution.

morgenstern() Morgenstern's bivariate distribution

bigumbelI() Gumbel's Type I bivariate distribution

plackett(dpr) Plackett's bivariate distribution.

Categorical data

In the following, g is a link function, η_j the linear/additive predictors, and $Y \in \{1,\ldots,M+1\}$ is a categorical response. See also **margeff()**.

acat() Adjacent categories model, $\eta_j = g(P[Y=j+1]/P[Y=j])$.

cumulative() Cumulative categories model, $\eta_j = g(P[Y \leq j])$. Includes the proportional odds model.

propodds() Proportional odds model $\eta_j = \text{logit}(P[Y>j])$.

cratio() Continuation ratio model, $\eta_j = g(P[Y>j|Y \geq j])$.

sratio() Stopping ratio model, $\eta_j = g(P[Y=j|Y \geq j])$.

multinomial() Multinomial logit model, $\eta_j = \log(P[Y=j]/P[Y=M+1])$.

brat() Bradley Terry model (without ties).

bratt() Bradley Terry model (with ties).

ordpoisson() Ordinal Poisson model.

The argument `reverse` reverses the direction of many of the above probabilities.

See documentation on the `xij` argument at the package's website: this handles covariates that have different values for differing linear/additive predictors η_j. This will handle consumer choice or discrete choice models.

Genetical data

AA.Aa.aa() AA-Aa-aa blood group system.

AB.Ab.aB.ab2() AB-Ab-aB-ab2 blood group system.

AB.Ab.aB.ab() AB-Ab-aB-ab blood group system.

ABO() ABO blood group system.

G1G2G3() G1G2G3 blood group system.

MNSs() MNSs blood group system.

Quantile and expectile regression

(I) LMS methods

lms.bcn() Box-Cox transformation to normality.

lms.bcg() Box-Cox transformation to the gamma distribution.

lms.yjn() Yeo-Johnson transformation to normality.

Special methods functions for these models are:

qtplot() Quantile plot.

deplot() Density plot.

cdf() Cumulative distribution function.

(II) Asymmetric Laplace distribution (ALD) methods

alaplace1(dpqr) 1-parameter ALD.

alaplace2(dpqr) 2-parameter ALD.

alaplace3(dpqr) 3-parameter ALD.

(III) Asymmetric Maximum Likelihood (AML) estimation methods

amlnormal() Asymmetric least squares (expectile) regression (Efron, 1991).

amlbinomial() Logistic (expectile) regression.

amlexponential() Exponential (expectile) regression.

amlpoisson() Poisson (expectile) regression (Efron, 1992).

Extreme value data

gev(dpqr) 3-parameter generalized extreme value distribution.

gevff(dpqr) 3-parameter generalized extreme value distribution (handles independent multiple responses, like usual).

gpd(dpqr) 2-parameter generalized Pareto distribution.

gumbelff(dpqr) 2-parameter Gumbel distribution (handles independent multiple responses, like usual).

gumbel(dpqr) 2-parameter Gumbel distribution for multivariate responses.

gumbelII(dpqr) Gumbel's Type II distribution.

recnormal() Records: univariate normal data.

recexp1() Records: univariate exponential data.

See also **guplot()**, **meplot()**, **rlplot()**.

Positive and zero-inflated or zero-altered distributions

Here, positive distributions have the zero probability that the response is 0, i.e., zero-truncated. Zero-altered models are also called hurdle models.

posbernoulli.[b,t,tb](dr) Positive Bernoulli distribution for closed-population capture–recapture experiments based on the conditional likelihood.

posbinomial(dpqr) Positive binomial distribution.

posgeom(dpqr) Positive geometric distribution (dpqr-only).

posnormal(dpqr) Positive (univariate) normal distribution.

posnegbinomial(dpqr) Positive negative binomial distribution.

pospoisson(dpqr) Positive Poisson distribution.

zabinomial[ff](dpqr) Zero-altered binomial distribution.

zageometric[ff](dpqr) Zero-altered geometric distribution.

zanegbinomial[ff](dpqr) Zero-altered negative binomial distribution.

zapoisson[ff](dpqr) Zero-altered Poisson distribution.

zibinomial[ff](dpqr) Zero-inflated binomial distribution.

zigeometric[ff](dpqr) Zero-inflated geometric distribution.

zinegbinomial[ff](dpqr) Zero-inflated negative binomial distribution.

zipoisson[ff](dpqr) Zero-inflated Poisson distribution.

yip88() Zero-inflated Poisson distribution.

zipebcom() Exchangeable zero-inflated Poisson distribution and **binom2.or()**.

One-inflated or one-altered distributions

oalog(dpqr) One-altered logarithmic distribution.

oilog(dpqr) One-inflated logarithmic distribution.

oiposbinomial(dpqr) One-inflated positive binomial distribution.

oipospoisson(dpqr) One-inflated positive Poisson distribution.

oizeta(dpqr) One-inflated zeta distribution.

oizipf(dpqr) One-inflated Zipf distribution.

otlog(dpqr) One-truncated logarithmic distribution.

otpospoisson(dpqr) One-truncated positive Poisson distribution.

otzeta(dpqr) One-truncated zeta distribution.

zoabetaR(dpqr) 3 or 4-parameter beta distribution (shape parameterization).

Finite mixture models

mix2exp() Two exponential distributions.

mix2normal() Two univariate normals.

mix2poisson() Two Poisson distributions.

Miscellaneous models and distributions

AR1() Time series AR(1) for autoregressive errors.

DeLury() De Lury's model for fish depletion analysis [in VGAMdata].

dirichlet() Dirichlet distribution.

dirmultinomial(), dirmul.old() Dirichlet-multinomial distribution.

huber(dpqr) Huber's robust regression method.

Nonlinear regression models

micmen() Michaelis-Menten model, $\mu_i = \theta_1 x_i/(\theta_2 + x_i)$.

Miscellaneous mathematical functions

erf() Error function, $\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)\, dt$.

erfc() Complementary error function, $1-\text{erf}(x) = \frac{2}{\sqrt{\pi}}\int_x^\infty \exp(-t^2)\,dt$.

expint() Exponential integral $\text{Ei}(x) = \int_0^x \frac{\exp(t)}{t}\,dt$, $x>0$.

expexpint() $\exp(-x)\,\text{Ei}(x)$, $x>0$.

expint.E1() $\text{E}_1(x) = \int_x^\infty \frac{\exp(-t)}{t}\,dt$, $x>0$.

lambertW() Lambert's W function for $W(z)\exp(W(z)) = z$.

lerch() Lerch's $\Phi(x,s,v)$ function.

mills.ratio(), **mills.ratio2()** Mills ratio, $\phi(x)/\Phi(x)$ and $\phi^2(x)/\Phi(x)$ respectively.

pgamma.deriv() First 2 derivatives of the incomplete gamma integral.

pgamma.deriv.unscaled() First 2 derivatives (wrt the shape parameter) of the unscaled incomplete gamma integral.

pbinorm() CDF of N_2, $= P(Y_1 \le y_1, Y_2 \le y_2; \mu, \Sigma)$.

zeta() Riemann's $\zeta(x)$ zeta function.

Quadratic and additive ordination

cqo() Canonical quadratic (Gaussian) ordination (QRR-VGLM).

cao() Constrained additive ordination (RR-VGAM). Not fully finished yet.

The fast algorithm currently works for families **poissonff()**, and **binomialff()** (`logit` and `cloglog` links available).

Special methods functions for these models are:

Coef() $\widehat{\mathbf{A}}$, $\widehat{\mathbf{B}}_1$, $\widehat{\mathbf{C}}$, $\widehat{\mathbf{D}}$, $\widehat{u}_s$, $\widehat{\mathbf{T}}_s$, $\widehat{\nu}_i$, etc.

concoef() Constrained (canonical) coefficients $\widehat{\mathbf{C}}$

latvar() Latent variables $\widehat{\nu}_i = \widehat{\mathbf{C}}^T x_{2i}$ (site scores)

Max() Maxima $E[Y_s|\widehat{u}_s] = g^{-1}(\widehat{\alpha}_s)$

Opt() Optima $\widehat{u}_s$ (species scores)

Tol() Tolerances $\widehat{\mathbf{T}}_s$

lvplot() Latent variable plot (ordination diagram; for rank $R=1$ or 2)

persp() Perspective (3-D; sometimes 2-D) plot

calibrate() Calibration: estimate ν from y

trplot() Trajectory plot (for $R=1$ only)

Miscellaneous

methods(class = class(fit)) lists all the methods to handle objects of class `fit`.

slotNames(fit) lists the slots of the object `fit`, but it is best to use extractor functions where possible.

Some formulae

For most VGLMs the log-likelihood $\ell = \sum_{i=1}^n w_i \ell_i(\eta_1,\ldots,\eta_M)$ is maximized, where $\eta_j = \beta_j^T x$. The prior weights w_i are inputted using **vglm(..., weights =...)**.

For VGLMs (Yee and Hastie, 2003):

$$\eta(x) \;=\; \mathbf{H}_1\beta_1^* x_1 + \cdots + \mathbf{H}_p\beta_p^* x_p \;=\; \mathbf{B}^T x \qquad (1)$$

where $\mathbf{H}_1,\ldots,\mathbf{H}_p$ are *known* full-column rank *constraint matrices*, and β_k^* is a vector containing a possibly reduced set of unknown regression coefficients. With no constraints at all, $\mathbf{H}_k = \mathbf{I}_M$ for all k. Usually $x_1 = 1$ (intercept term). In general,

$$\mathbf{B}^T \;=\; \left(\mathbf{H}_1\beta_1^* \;\cdots\; \mathbf{H}_p\beta_p^*\right). \qquad (2)$$

Then **coef(fit, matrix = TRUE)** is the estimate of $\mathbf{B}$, and **constraints(fit)** are the $\mathbf{H}_k$. The $\mathbf{H}_k$ can be inputted with **vglm(..., constraints = list("(Intercept)" = ..., x2 = ...))**, or with arguments such as `parallel`, `exchangeable` and `zero` in the VGAM family function itself.

For the `xij` argument, one has the central formula

$$\begin{aligned}\eta_i &= o_i + \sum_{k=1}^p \text{diag}(x_{ik1},\ldots,x_{ikM})\,\mathbf{H}_k\,\beta_{(k)}^* \qquad (3)\\ &= o_i + \sum_{k=1}^p \mathbf{X}_{(ik)}^{\#}\,\mathbf{H}_k\,\beta_{(k)}^*, \text{ say,}\end{aligned}$$

with provision for offsets o_i. This is the central formula for the `xij` facility and the most general for VGLMs. Then the big model matrix has the block form

$$\mathbf{X}_{\text{VLM}} \;=\; \begin{pmatrix} \mathbf{X}_{(11)}^{\#}\mathbf{H}_1 & \cdots & \mathbf{X}_{(1p)}^{\#}\mathbf{H}_p \\ \vdots & & \vdots \\ \mathbf{X}_{(n1)}^{\#}\mathbf{H}_1 & \cdots & \mathbf{X}_{(np)}^{\#}\mathbf{H}_p \end{pmatrix}. \qquad (4)$$

For example, for an exchangeable bivariate odds ratio model,

```
vglm(formula = cbind(leye, reye) ~ iop,
     binom2.or(exchangeable = TRUE, zero = 3),
     data = eyesData,
     xij = list(iop ~ iop + riop + fill1(liop)),
     form2 =  ~ iop + liop + riop + fill1(liop))
```

where `iop` is the *intraocular ocular pressure* (which is different for left eye and right eye). The argument `form2` contains all the terms, and the RHS of each formula in the `xij` list are the successive (unique) elements/terms of $\mathbf{X}_{(ik)}^{\#}$.

For VGAMs (Yee and Wild, 1996): (1) extends to

$$\eta(x) \;=\; \mathbf{H}_1\beta_1^* x_1 + \mathbf{H}_2 f_2^*(x_2) + \cdots + \mathbf{H}_p f_p^*(x_p) \qquad (5)$$

where $f_k^*(x_k) = (f_{(1)k}(x_k),\ldots,f_{(r_k)k}(x_k))^T$ is a r_k-vector of smooth functions of x_k (estimated by a vector smoothing spline). With no constraints, $\eta_j = \sum_{k=1}^p f_{(j)k}(x_k)$.

For RR-VGLMs (Yee and Hastie, 2003; Yee, 2014*a*):

$$\eta(x) \;=\; \mathbf{B}_1^T x_1 + \mathbf{A}\nu \qquad (6)$$

where $x = (x_1^T, x_2^T)^T$, $\nu = \mathbf{C}^T x_2$ is a vector of latent variables, $\mathbf{A}$ is $M\times R$ and $\mathbf{C}$ is $p_2\times R$. Here, $\mathbf{A}$ and $\mathbf{C}$ are estimated, and $\mathbf{B} = (\mathbf{B}_1^T\;\mathbf{B}_2^T)^T$ with $\mathbf{B}_2 = \mathbf{C}\mathbf{A}^T$, a reduced-rank approximation of a subset of $\mathbf{B}$ (cf. (1)). The *rank* R is often 1 or 2, maybe 3 ....

For QRR-VGLMs (Yee, 2004*a*):

$$\begin{aligned}\eta(x) &= \mathbf{B}_1^T x_1 + \mathbf{A}\nu + \sum_{j=1}^M (\nu^T\mathbf{D}_j\nu)e_j \qquad (7)\\ &= \mathbf{B}_1^T x_1 + \mathbf{A}\nu + \begin{pmatrix} \nu^T\mathbf{D}_1\nu \\ \vdots \\ \nu^T\mathbf{D}_M\nu \end{pmatrix},\end{aligned}$$

where e_i is a vector of zeros but with a one in the ith position, and $\mathbf{D}_j$ are $R\times R$ symmetric matrices. Then $\mathbf{T}_j = -\frac{1}{2}\mathbf{D}_j^{-1}$ are *tolerance matrices*.

For RCIMs (Yee and Hadi, 2014): these are RR-VGLMs applied to **Y** (no **X**!), with

$$g_1(\theta_1) \equiv \eta_{1ij} = \mu + \alpha_i + \gamma_j + \sum_{r=1}^{R} c_{ir} a_{jr}, \qquad (8)$$

where $R \leq \min(M, p_2)$. The other parameters $\theta_2, \ldots$ are usual intercept-only.

For a rank-1 CAO (Yee, 2006): these are RR-VGAMs with

$$g(\mu_{iq}) \equiv \eta_{iq} = f_q(\nu_i) \qquad (9)$$

where $q = 1, \ldots, Q$, $y_i = (y_{i1}, \ldots, y_{iQ})^T$, $\nu_i = c^T x_i$ is a latent variable or site score, and the f_q are estimated by a smoothing spline. Only Poisson and Bernoulli responses are handled currently, and for rank-1 only too.

References

Hauck, J. W. W. and Donner, A. (1977), 'Wald's test as applied to hypotheses in logit analysis', *J. Amer. Statist. Assoc.* **72**(360), 851–853.

Wood, S. N. (2004), 'Stable and efficient multiple smoothing parameter estimation for generalized additive models', *J. Amer. Statist. Assoc.* **99**(467), 673–686.

Wood, S. N. (2017), *Generalized Additive Models: An Introduction with R*, second edn, Chapman & Hall/CRC, London.

Yee, T. W. (1998), 'On an alternative solution to the vector spline problem', *J. Roy. Statist. Soc. Ser. B* **60**(1), 183–188.

Yee, T. W. (2004*a*), 'A new technique for maximum-likelihood canonical Gaussian ordination', *Ecol. Monogr.* **74**(4), 685–701.

Yee, T. W. (2004*b*), 'Quantile regression via vector generalized additive models', *Stat. Med.* **23**(14), 2295–2315.

Yee, T. W. (2006), 'Constrained additive ordination', *Ecology* **87**(1), 203–213.

Yee, T. W. (2008), 'The VGAM package', *R News* **8**(2), 28–39.
URL: *https://CRAN.R-project.org/doc/Rnews/*

Yee, T. W. (2010*a*), 'The VGAM package for categorical data analysis', *J. Statist. Soft.* **32**(10), 1–34.
URL: *http://www.jstatsoft.org/v32/i10/*

Yee, T. W. (2010*b*), 'VGLMs and VGAMs: an overview for applications in fisheries research', *Fish. Res.* **101**(1–2), 116–126.

Yee, T. W. (2014*a*), 'Reduced-rank vector generalized linear models with two linear predictors', *Computational Statistics & Data Analysis* **71**, 889–902.

Yee, T. W. (2014*b*), 'Scoring rules, and the role of chance: Analysis of the 2008 World Fly Fishing Championships', *Journal of Quantitative Analysis in Sports* **10**(4), 397–409.

Yee, T. W. (2015), *Vector Generalized Linear and Additive Models: With an Implementation in R*, Springer, New York, USA.

Yee, T. W. (2016), 'Comment on: "Smoothing parameter and model selection for general smooth models" by Wood, S. N. and Pya, N. and Säfken, N.', *Journal of the American Statistical Association* **111**(516), 1565–1568.

Yee, T. W. (2018), 'Detecting the Hauck-Donner effect in Wald tests', *In preparation* .

Yee, T. W. and Dirnböck, T. (2009), 'Models for analysing species' presence/absence data at two time points', *Journal of Theoretical Biology* **259**(4), 684–694.

Yee, T. W. and Hadi, A. F. (2014), 'Row-column interaction models, with an R implementation', *Computational Statistics* **29**(6), 1427–1445.

Yee, T. W. and Hastie, T. J. (2003), 'Reduced-rank vector generalized linear models', *Statistical Modelling* **3**(1), 15–41.

Yee, T. W., Somchit, C. and Wild, C. J. (2018), 'Penalized vector generalized additive models', *Manuscript in preparation* .

Yee, T. W. and Stephenson, A. G. (2007), 'Vector generalized linear and additive extreme value models', *Extremes* **10**(1–2), 1–19.

Yee, T. W., Stoklosa, J. and Huggins, R. M. (2015), 'The VGAM package for capture–recapture data using the conditional likelihood', *J. Statist. Soft.* **65**(5), 1–33.
URL: *http://www.jstatsoft.org/v65/i05/*

Yee, T. W. and Wild, C. J. (1996), 'Vector generalized additive models', *J. Roy. Statist. Soc. Ser. B* **58**(3), 481–493.