Package 'glmc'

December 12, 2018

Version 0.3-1

Date 2018-12-11

Title Fitting Generalized Linear Models Subject to Constraints

Author Sanjay Chaudhuri [aut, cre],
Mark S. Handcock [aut],
Michael S. Rendall [ctb]

Maintainer Sanjay Chaudhuri <sanjay@stat.nus.edu.sg>

Description Fits generalized linear models where the parameters are subject to linear constraints. The model is specified by giving a symbolic description of the linear predictor, a description of the error distribution, and a matrix of constraints on the parameters.

Imports emplik,stats

License GPL (>= 2)

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-12-12 15:00:03 UTC

R topics documented:

glmc-package *Fitting Generalized Linear Models Subject to Constraints*

Description

glmc is a collection of functions to fit generalized linear models where the parameters are subject to linear constraints. The model is specified by giving a symbolic description of the linear predictor, a description of the error distribution, and a matrix of constraints on the parameters.

For a complete list of the functions, use `library(help="glmc")` or read the rest of the manual. For a simple demonstration, use `demo(packages="glmc")`.

When publishing results obtained using this package the original authors are to be cited as:

Mark S. Handcock, Sanjay Chaudhuri, and Michael S. Rendall. 2004 *glmc: An R package for generalized linear models subject to constraints*.
All programs derived from this package must cite it. For complete citation information, use `citation(package="glmc")`.

Details

In many situations information from a sample of individuals can be supplemented by population level information on the relationship between a dependent variable and explanatory variables. Inclusion of the population level information can reduce bias and increase the efficiency of the parameter estimates.

Population level information can be incorporated via constraints on functions of the model parameters. In general the constraints are nonlinear making the task of maximum likelihood estimation harder. In this package we provide an alternative approach exploiting the notion of an empirical likelihood. Within the framework of generalised linear models, the population level information corresponds to linear constraints, which are comparatively easy to handle. We provide a two-step algorithm that produces parameter estimates using only unconstrained estimation. We also provide computable expressions for the standard errors.

Author(s)

Mark S. Handcock <handcock@stat.ucla.edu>,
Sanjay Chaudhuri <sanjay@stat.nus.edu.sg>, and
Michael S. Rendall <mrendall@umd.edu>

Maintainer: Mark S. Handcock <handcock@stat.washington.edu>

References

Sanjay Chaudhuri, Mark S. Handcock, and Michael S. Rendall. 2004 Generalised Linear Models Incorporating Population Level Information: An Empirical Likelihood Based Approach, Working Paper, Center for Statistics and the Social Sciences, University of Washington.

glmc *Fitting Generalised Linear Models Subject to Population Constraints*

Description

glmc is used to fit generalised linear models where the parameters are subject to population constraints. The model is specified by giving a symbolic description of the linear predictor, a description of the error distribution, and a matrix of constraints on the parameters.

Usage

```
glmc(formula,family = gaussian, data, na.action,
     start = NULL,etastart, mustart, offset,control = glmc.control(...),
     model = TRUE,glm.method= "glm.fit",optim.method="Nelder-Mead",
     emplik.method="Owen",optim.hessian=FALSE,x = FALSE, y = TRUE,
     Amat=NULL, confn=NULL,...)
```

Arguments

formula	a symbolic description of the model to be fit. The details of model specification are given below.
family	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.)
data	an optional data frame containing the variables in the model. By default the variables are taken from environment(formula), typically the environment from which glmc is called. All variables named constraints is automatically assumed to be the value of the population constraints.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The "factory-fresh" default is na.omit.

start	starting values for the parameters in the linear predictor.
etastart	starting values for the linear predictor.
mustart	starting values for the vector of means.
offset	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting.
control	a list of parameters for controlling the fitting process. See the documentation for glmc.control for details.
model	a logical value indicating whether *model frame* should be included as a component of the returned value.
glm.method	the method to be used in fitting the model. The default method "glm.fit" uses iteratively re-weighted least squares (IWLS). The only current alternative is "model.frame" which returns the model frame and does no fitting.
emplik.method	the method used to maximise the empirical likelihood to compute the weights. The default is "Owen", due to Art Owen. Current alternative is "CSW", due to Chen, Sitter and Wu.
optim.method	the method used to maximise over the parameters. See optim for more details.
optim.hessian	Logical. If True returns a numerically calculated Hessian Matrix from the optim step.
x, y	For glmc: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For glmc.fit: x is a design matrix of dimension n * p, and y is a vector of observations of length n.
Amat	a matrix of population constraints on the parameters.
confn	a function returning the value of the population constraints on the parameter. Allows parameter dependent population constraints.
...	further arguments passed to or from other methods.

Details

A typical predictor has the form response ~ terms where response is the (numeric) response vector and terms is a series of terms which specifies a linear predictor for response.

A terms specification of the form first + second indicates all the terms in first together with all the terms in second with duplicates removed. A specification of the form first:second indicates the the set of terms obtained by taking the interactions of all terms in first with all terms in second. The specification first*second indicates the *cross* of first and second. This is the same as first + second + first:second.

glmc uses closely follows glm, uses glm.fit, optim and el.test form the emplik library.

If more than one of etastart, start and mustart is specified, the first in the list will be used.

Value

glmc returns an object of class inheriting from "glmc" which inherits from the class "lm". See later in this section.

The generic accessor functions coefficients, effects, fitted.values and residuals can be used to extract various useful features of the value returned by glmc.

An object of class "glmc" is a list containing at least the following components:

coefficients	a named vector of coefficients
residuals	the *working* residuals, that is the residuals in the final iteration of the IWLS fit.
fitted.values	the fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.
rank	the numeric rank of the fitted linear model.
family	the family object used.
linear.predictors	the linear fit on link scale.
deviance	up to a constant, minus twice the maximised log-likelihood. Where sensible, the constant is chosen so that a saturated model has deviance zero.
aic	Akaike's *An Information Criterion*, minus twice the maximised log-likelihood plus twice the number of coefficients (so assuming that the dispersion is known.)
null.deviance	The deviance for the null model, comparable with deviance. The null model will include the offset, and an intercept if there is one in the model
iter	the number of iterations of IWLS used.
weights	the *working* weights, that is the weights in the final iteration of the IWLS fit from the glm step (if there is one).
final.weights	the weights maximising the empirical likelihood
df.residual	the residual degrees of freedom.
df.null	the residual degrees of freedom for the null model.
y	the y vector used. (It is a vector even for a binomial model.)
converged	logical. Was the IWLS algorithm judged to have converged?
boundary	logical. Is the fitted value on the boundary of the attainable values?
call	the matched call.
formula	the formula supplied.
terms	the terms object used.
data	the data argument.
offset	the offset vector used.
control	the value of the control argument used.
glm.method	the name of the fitter function used in the final glm call, in R always "glm.fit".
emplik.method	the name of the method used to maximise the empirical likelihood.
optim.method	the name of the method supplied to the optim function for the outer maximisation over the parameters.

.

`xlevels` (where relevant) a record of the levels of the factors used in fitting.

In addition, non-empty fits will have components `qr`, `R` and `effects` relating to the final weighted linear fit.

Objects of class `"glmc"` are normally of class `c("glmc","glm", "lm")`, that is inherit from class `"lm"`, and well-designed methods for class `"lm"` will be applied to the weighted linear model at the final iteration of IWLS. However, care is needed, as extractor functions for class `"glmc"` such as `residuals` and `weights` do **not** just pick out the component of the fit with the same name. In-fact no `"type"` specification in `weights` retirns the weights which maximises the empirical likelihood. A call of wieghts with `type="prior"` returns a vector of 1 of length equaling the number of odsevations.

Author(s)

The R implementation of `glmc` is written by Sanjay Chaudhuri and Mark S Handcock at National University of Singapore and University of Washington, Seattle respectively.

References

Owen, A. B. (2001) *Empirical Likelihood.* Boca Raton, Fla : Chapman \& Hall/CRC.

Dobson, A. J. (1990) *An Introduction to Generalized Linear Models.* London: Chapman and Hall.

Hastie, T. J. and Pregibon, D. (1992) *Generalized linear models.* Chapter 6 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth \& Brooks/Cole.

McCullagh P. and Nelder, J. A. (1989) *Generalized Linear Models.* London: Chapman and Hall.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* New York: Springer.

See Also

`glmc` methods, and the generic functions `anova`, `summary`, `effects`, `fitted.values`, and `residuals`. `optim`, `el.test`, the fitting procedure used by `glmc`. Further, `lm` for non-generalised *linear* models.

Examples

```
library(glmc)
#Specify the data.

n <- rbind(c(5903,230),c(5157,350))
mat <- matrix(0,nrow=sum(n),ncol=2)
mat <- rbind(matrix(1,nrow=n[1,1],ncol=1)%*%c(0,0),
             matrix(1,nrow=n[1,2],ncol=1)%*%c(0,1),
             matrix(1,nrow=n[2,1],ncol=1)%*%c(1,0),
             matrix(1,nrow=n[2,2],ncol=1)%*%c(1,1))

#Specifying the population constraints.

gfr <- .06179*matrix(1,nrow=nrow(mat),ncol=1)
g <- matrix(1,nrow=nrow(mat),ncol=1)
```

```
amat <- matrix(mat[,2]*g-gfr,ncol=1)

# Method 1. Defining constraints in the data frame.

hrh <- data.frame(birth=mat[,2], child=mat[,1], constraints=amat)

gfit <- glmc(birth~child, data=hrh, family="binomial",emplik.method="Owen",
             control=glmc.control(maxit.glm=10,maxit.weights=200,
             itertrace.weights=TRUE,gradtol.weights=10^(-6)))

summary.glmc(gfit)

# Method 2. Defining constraints through a matrix.

gfit<- glmc(mat[,2]~mat[,1],family=binomial(link=logit),
            emplik.method="Owen",control=glmc.control(maxit.glm=10,
            maxit.weights=200,itertrace.weights=TRUE,gradtol.weights=10^(-10)),
            Amat=amat,confn=NULL)

summary.glmc(gfit)

## Not run:
# Method 3. Defining constraints through a function.

fn <- function(t,Y,X){
grf <- .06179*matrix(1,nrow=nrow(as.matrix(X)),ncol=1)
g <- matrix(1,nrow=nrow(X),ncol=1)
amat <- matrix(Y*g-grf,ncol=1)
return(amat)
}

gfit <- glmc(birth~child,data=hrh,family=binomial(link=logit),
             optim.method="BFGS",emplik.method="Owen",
             control=glmc.control(maxit.glm=10,maxit.optim=10^(8),
             reltol.optim=10^(-10),trace.optim=9,REPORT.optim=1,
             maxit.weights=200,gradtol.weights=10^(-6),itertrace.weights=FALSE),
             optim.hessian=TRUE,Amat=NULL,confn=fn)

summary.glmc(gfit)

## End(Not run)
```

`glmc.control` *Auxiliary for Controlling GLM Fitting with population level constraints.*

Description

Auxiliary function as user interface for `glmc` fitting. Typically only used when calling `glmc`.

Usage

```
glmc.control(epsilon.glm = 1e-8, maxit.glm= 100, trace.glm= FALSE,
             trace.optim= 0, fnscale.optim=-1, parscale.optim = rep.int(1,1),
             ndeps.optim = rep.int(0.001,1), maxit.optim = 100,
             abstol.optim = -Inf, reltol.optim= sqrt(.Machine$double.eps),
             alpha.optim = 1, beta.optim = 0.5, gamma.optim = 2,
             REPORT.optim= 10, type.optim = 1, lmm.optim = 5,
             factr.optim= 1e+07, pgtol.optim = 0, tmax.optim = 10,
             temp.optim =10, maxit.weights = 25, gradtol.weights = 1e-07,
             svdtol.weights = 1e-09, itertrace.weights = FALSE)
```

Arguments

`epsilon.glm`	positive convergence tolerance $epsilon$; the iterations converge when $\|dev - devold\|/(\|dev\| + 0.1) < epsilon$.
`maxit.glm`	integer giving the maximal number of IWLS iterations.
`trace.glm`	logical indicating if output should be produced for each iteration.
`trace.optim`	Non-negative integer. If positive, tracing information on the progress of the optimization is produced. Higher values may produce more tracing information: for method "L-BFGS-B" there are six levels of tracing. (To understand exactly what these do see the source code: higher levels give more detail.)
`fnscale.optim`	A negative number determining the overall scaling to be applied to the value of fn and gr during optimization. In `glmc` optimization is performed on $fn(par)/(fnscale.optim)$ if fnscale.optim is negative and on $fn(par)/((-1) * fnscale.optim)$ if fnscale.optim is positive.
`parscale.optim`	A vector of scaling values for the parameters. Optimization is performed on par/parscale and these should be comparable in the sense that a unit change in any element produces about a unit change in the scaled value.
`ndeps.optim`	A vector of step sizes for the finite-difference approximation to the gradient, on par/parscale scale. Defaults to 1e-3.
`maxit.optim`	The maximum number of iterations. Defaults to 100 for the derivative-based methods, and 500 for "Nelder-Mead". For "SANN" maxit gives the total number of function evaluations. There is no other stopping criterion. Defaults to 10000.
`abstol.optim`	The absolute convergence tolerance. Only useful for non-negative functions, as a tolerance for reaching zero.
`reltol.optim`	Relative convergence tolerance. The algorithm stops if it is unable to reduce the value by a factor of reltol * (abs(val) + reltol) at a step. Defaults to sqrt(.Machine\$double.eps), typically about 1e-8.
`alpha.optim, beta.optim, gamma.optim`	Scaling parameters for the "Nelder-Mead" method. alpha is the reflection factor (default 1.0), beta the contraction factor (0.5) and gamma the expansion factor (2.0).
`REPORT.optim`	The frequency of reports for the "BFGS" and "L-BFGS-B" methods if control\$trace is positive. Defaults to every 10 iterations.

`type.optim` for the conjugate-gradients method. Takes value 1 for the Fletcher–Reeves update, 2 for Polak–Ribiere and 3 for Beale–Sorenson.

`lmm.optim` is an integer giving the number of BFGS updates retained in the "L-BFGS-B" method, It defaults to 5.

`factr.optim` controls the convergence of the "L-BFGS-B" method. Convergence occurs when the reduction in the objective is within this factor of the machine tolerance. Default is 1e7, that is a tolerance of about 1e-8.

`pgtol.optim` helps controls the convergence of the "L-BFGS-B" method. It is a tolerance on the projected gradient in the current search direction. This defaults to zero, when the check is suppressed.

`temp.optim` controls the "SANN" method. It is the starting temperature for the cooling schedule. Defaults to 10.

`tmax.optim` is the number of function evaluations at each temperature for the "SANN" method. Defaults to 10.

`maxit.weights` an optional integer to control iteration when solve constrained maximisation for the weights.

`gradtol.weights`
an optional real value for convergence test while calculating the weights.

`svdtol.weights` an optional real value to detect singularity while solve equations. This is used to compute the weights.

`itertrace.weights`
a logical value. If the iteration history when calculating the weights needs to be printed out.

Value

A list with components

`glmc.summaries` *Accessing objects in glmc Fits*

Description

All these functions are `methods` for class `"lm"` objects.

Usage

```
## S3 method for class 'glmc'
coef(object, ...)
## S3 method for class 'glmc'
deviance(object, ...)
## S3 method for class 'glmc'
effects(object, ...)
## S3 method for class 'glmc'
family(object, ...)
```

```
## S3 method for class 'glmc'
fitted(object, ...)


## S3 method for class 'glmc'
residuals(object,
          type = c("deviance", "pearson","working", "response",
                   "partial"),
          ...)
```

Arguments

object	an object inheriting from class lm, usually the result of a call to lm or aov.
...	further arguments passed to or from other methods.
type	the type of residuals which should be returned.

Details

The generic accessor functions coef, effects, fitted and residuals can be used to extract various useful features of the value returned by lm.

The working and response residuals are "observed - fitted". The deviance and pearson residuals are weighted residuals, scaled by the square root of the weights used in fitting. The partial residuals are a matrix with each column formed by omitting a term from the model. In all these, zero weight cases are never omitted (as opposed to the standardized rstudent residuals, and the weighted.residuals).

How residuals treats cases with missing values in the original fit is determined by the na.action argument of that fit. If na.action = na.omit omitted cases will not appear in the residuals, whereas if na.action = na.exclude they will appear, with residual value NA. See also naresid.

The "lm" method for generic labels returns the term labels for estimable terms, that is the names of the terms with an least one estimable coefficient.

References

Chambers, J. M. (1992) *Linear models.* Chapter 4 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth \& Brooks/Cole.

See Also

The model fitting function lm, anova.lm.

coef, deviance, df.residual, effects, fitted, glm for **generalized** linear models, influence (etc on that page) for regression diagnostics, weighted.residuals, residuals, residuals.glm, summary.glmc.

Examples

```
## Not run:
##-- Continuing the  glmc(.) example:
coef(gfit)# the bare coefficients

## The 2 basic regression diagnostic plots [plot.glmc(.) is preferred]
plot(resid(gfit), fitted(gfit))# Tukey-Anscombe's
abline(h=0, lty=2, col = 'gray')

qqnorm(residuals(gfit))

## End(Not run)
```

---

summary.glmc *Summarizing Generalized Linear Model Fits*

Description

These functions are all methods for class glmc or summary.glmc objects.

Usage

```
## S3 method for class 'glmc'
summary(object, dispersion = NULL, correlation = FALSE,
        symbolic.cor = FALSE, ...)

## S3 method for class 'summary.glmc'
print(x, digits = max(3, getOption("digits") - 3),
      symbolic.cor = x$symbolic.cor,
      signif.stars = getOption("show.signif.stars"), ...)
```

Arguments

object	an object of class "glmc", usually, a result of a call to glmc.
x	an object of class "summary.glmc", usually, a result of a call to summary.glmc.
dispersion	the dispersion parameter for the family used. Either a single numerical value or NULL (the default), when it is inferred from object (see Details).
correlation	logical; if TRUE, the correlation matrix of the estimated parameters is returned and printed.
digits	the number of significant digits to use when printing.
symbolic.cor	logical. If TRUE, print the correlations in a symbolic form (see symnum) rather than as numbers.
signif.stars	logical. If TRUE, "significance stars" are printed for each coefficient.
...	further arguments passed to or from other methods.

Details

print.summary.glmc tries to be smart about formatting the coefficients, standard errors, etc. and additionally gives "significance stars" if signif.stars is TRUE. The coefficients component of the result gives the estimated coefficients and their estimated standard errors, together with their ratio. This third column is labelled t ratio if the dispersion is estimated, and z ratio if the dispersion is known (or fixed by the family). A fourth column gives the two-tailed p-value corresponding to the t or z ratio based on a Student t or Normal reference distribution. (It is possible that the dispersion is not known and there are no residual degrees of freedom from which to estimate it. In that case the estimate is NaN.)

Aliased coefficients are omitted in the returned object but restored by the print method.

Correlations are printed to two decimal places (or symbolically): to see the actual correlations print summary(object)$correlation directly.

The dispersion of a GLM is not used in the fitting process, but it is needed to find standard errors. If dispersion is not supplied or NULL, the dispersion is taken as 1 for the binomial and Poisson families, and otherwise estimated by the residual Chisquared statistic (calculated from cases with non-zero weights) divided by the residual degrees of freedom.

summary can be used with Gaussian glmc fits to handle the case of a linear regression with known error variance, something not handled by summary.lm.

Value

summary.glmc returns an object of class "summary.glmc", a list with components

call	the component from object.
family	the component from object.
deviance	the component from object.
contrasts	the component from object.
df.residual	the component from object.
null.deviance	the component from object.
df.null	the component from object.
deviance.resid	the deviance residuals: see residuals.glmc.
coefficients	the matrix of coefficients, standard errors, z-values and p-values. Aliased coefficients are omitted.
aliased	named logical vector showing if the original coefficients are aliased.
dispersion	either the supplied argument or the inferred/estimated dispersion if the latter is NULL.
df	a 3-vector of the rank of the model and the number of residual degrees of freedom, plus number of non-aliased coefficients.
cov.unscaled	the unscaled (dispersion = 1) estimated covariance matrix of the estimated coefficients.
cov.scaled	ditto, scaled by dispersion.
correlation	(only if correlation is true.) The estimated correlations of the estimated coefficients.
symbolic.cor	(only if correlation is true.) The value of the argument symbolic.cor.

See Also

glmc, summary.

Examples

```
## --- Continuing the Example from  '?glmc':%\code{\link{glmc}}:

## Not run: summary(gfit)
```

Index