glm {stats}   R Documentation

Fitting Generalized Linear Models

Description

glm is used to fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution.

Usage

```
glm(formula, family = gaussian, data, weights, subset,
    na.action, start = NULL, etastart, mustart, offset,
    control = list(...), model = TRUE, method = "glm.fit",
    x = FALSE, y = TRUE, contrasts = NULL, ...)

glm.fit(x, y, weights = rep(1, nobs),
        start = NULL, etastart = NULL, mustart = NULL,
        offset = rep(0, nobs), family = gaussian(),
        control = list(), intercept = TRUE)

## S3 method for class 'glm'
weights(object, type = c("prior", "working"), ...)
```

Arguments

formula
>   an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.

family
>   a description of the error distribution and link function to be used in the model. For glm this can be a character string naming a family function, a family function or the result of a call to a family function. For glm.fit only the third option is supported. (See family for details of family functions.)

data
>   an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which glm is called.

weights
>   an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.

subset

an optional vector specifying a subset of observations to be used in the fitting process.

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`. Another possible value is NULL, no action. Value `na.exclude` can be useful.

`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. This should be NULL or a numeric vector of length equal to the number of cases. One or more `offset` terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See `model.offset`.

control

a list of parameters for controlling the fitting process. For `glm.fit` this is passed to `glm.control`.

model

a logical value indicating whether *model frame* should be included as a component of the returned value.

method

the method to be used in fitting the model. The default method `"glm.fit"` uses iteratively reweighted least squares (IWLS): the alternative `"model.frame"` returns the model frame and does no fitting.

User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as `glm.fit`. If specified as a character string it is looked up from within the **stats** namespace.

x, y

> For `glm`: 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 `glm.fit`: x is a design matrix of dimension `n` * `p`, and `y` is a vector of observations of length `n`.

contrasts

> an optional list. See the `contrasts.arg` of `model.matrix.default`.

intercept

> logical. Should an intercept be included in the *null* model?

object

> an object inheriting from class `"glm"`.

type

> character, partial matching allowed. Type of weights to extract from the fitted model object. Can be abbreviated.

...

> For `glm`: arguments to be used to form the default `control` argument if it is not supplied directly.
>
> For `weights`: 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`. For `binomial` and `quasibinomial` families the response can also be specified as a `factor` (when the first level denotes failure and all others success) or as a two-column matrix with the columns giving the numbers of successes and failures. A terms specification of the form `first + second` indicates all the terms in `first` together with all the terms in `second` with any duplicates removed.

A specification of the form `first:second` indicates 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`.

The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on: to avoid this pass a `terms` object as the formula.

Non-`NULL` `weights` can be used to indicate that different observations have different dispersions (with the values in `weights` being inversely proportional to the dispersions); or equivalently, when the elements of `weights` are positive integers w_i, that each response y_i is the mean of w_i unit-weight observations. For a binomial GLM prior weights are used to give the number of trials when the response is the proportion of successes: they would rarely be used for a Poisson GLM.

`glm.fit` is the workhorse function: it is not normally called directly but can be more efficient where the response vector, design matrix and family have already been calculated.

If more than one of `etastart`, `start` and `mustart` is specified, the first in the list will be used. It is often advisable to supply starting values for a <u>quasi</u> family, and also for families with unusual links such as `gaussian("log")`.

All of `weights`, `subset`, `offset`, `etastart` and `mustart` are evaluated in the same way as variables in `formula`, that is first in `data` and then in the environment of `formula`.

For the background to warning messages about 'fitted probabilities numerically 0 or 1 occurred' for binomial GLMs, see Venables & Ripley (2002, pp. 197–8).

Value

`glm` returns an object of class inheriting from `"glm"` which inherits from the class `"lm"`. See later in this section. If a non-standard `method` is used, the object will also inherit from the class (if any) returned by that function.

The function `summary` (i.e., `summary.glm`) can be used to obtain or print a summary of the results and the function `anova` (i.e., `anova.glm`) to produce an analysis of variance table.

The generic accessor functions <u>coefficients</u>, `effects`, `fitted.values` and `residuals` can be used to extract various useful features of the value returned by `glm`.

`weights` extracts a vector of weights, one for each case in the fit (after subsetting and `na.action`).

An object of class `"glm"` 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. Since cases with zero weights are omitted, their

working residuals are NA.

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 maximized log-likelihood. Where sensible, the constant is chosen so that a saturated model has deviance zero.

aic

A version of Akaike's *An Information Criterion*, minus twice the maximized log-likelihood plus twice the number of parameters, computed by the aic component of the family. For binomial and Poison families the dispersion is fixed at one and the number of parameters is the number of coefficients. For gaussian, Gamma and inverse gaussian families the dispersion is estimated from the residual deviance, and the number of parameters is the number of coefficients plus one. For a gaussian family the MLE of the dispersion is used so this is a valid value of AIC, but for Gamma and inverse gaussian families it is not. For families fitted by quasi-likelihood the value is NA.

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. Note that this will be incorrect if the link function depends on the data other than through the fitted mean: specify a zero offset to force a correct calculation.

iter

the number of iterations of IWLS used.

weights

the *working* weights, that is the weights in the final iteration of the IWLS fit.

prior.weights

the weights initially supplied, a vector of 1s if none were.

df.residual

the residual degrees of freedom.

df.null

the residual degrees of freedom for the null model.

y

if requested (the default) the y vector used. (It is a vector even for a binomial model.)

x

if requested, the model matrix.

model

if requested (the default), the model frame.

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.

method
> the name of the fitter function used, currently always `"glm.fit"`.

contrasts
> (where relevant) the contrasts used.

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

na.action
> (where relevant) information returned by `model.frame` on the special handling of NAs.

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

Objects of class `"glm"` are normally of class `c("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 `"glm"` such as `residuals` and `weights` do **not** just pick out the component of the fit with the same name.

If a `binomial` `glm` model was specified by giving a two-column response, the weights returned by `prior.weights` are the total numbers of cases (factored by the supplied case weights) and the component `y` of the result is the proportion of successes.

Fitting functions

The argument `method` serves two purposes. One is to allow the model frame to be recreated with no fitting. The other is to allow the default fitting function `glm.fit` to be replaced by a function which takes the same arguments and uses a different fitting algorithm. If `glm.fit` is supplied as a character string it is used to search for a function of that name, starting in the **stats** namespace.

The class of the object return by the fitter (if any) will be prepended to the class returned by `glm`.

Author(s)

The original **R** implementation of `glm` was written by Simon Davies working for

Ross Ihaka at the University of Auckland, but has since been extensively re-written by members of the R Core team.

The design was inspired by the S function of the same name described in Hastie & Pregibon (1992).

References

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

`anova.glm`, `summary.glm`, etc. for `glm` methods, and the generic functions `anova`, `summary`, `effects`, `fitted.values`, and `residuals`.

`lm` for non-generalized *linear* models (which SAS calls GLMs, for 'general' linear models).

`loglin` and `loglm` (package **MASS**) for fitting log-linear models (which binomial and Poisson GLMs are) to contingency tables.

`bigglm` in package **biglm** for an alternative way to fit GLMs to large datasets (especially those with many cases).

`esoph`, `infert` and `predict.glm` have examples of fitting binomial glms.

Examples

```
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))
glm.D93 <- glm(counts ~ outcome + treatment, family = poisson())
anova(glm.D93)
summary(glm.D93)

## an example with offsets from Venables & Ripley (2002, p.189)
```

```
    utils::data(anorexia, package = "MASS")

    anorex.1 <- glm(Postwt ~ Prewt + Treat + offset(Prewt),
                    family = gaussian, data = anorexia)
    summary(anorex.1)


    # A Gamma example, from McCullagh & Nelder (1989, pp. 300-2)
    clotting <- data.frame(
        u = c(5,10,15,20,30,40,60,80,100),
        lot1 = c(118,58,42,35,27,25,21,19,18),
        lot2 = c(69,35,26,21,18,16,13,12,12))
    summary(glm(lot1 ~ log(u), data = clotting, family = Gamma))
    summary(glm(lot2 ~ log(u), data = clotting, family = Gamma))

    ## Not run:
    ## for an example of the use of a terms object as a formula
    demo(glm.vr)

    ## End(Not run)
```

[Package *stats* version 3.3.0 [Index]]