Package: logbin (via r-universe)

August 24, 2024

Title Relative Risk Regression Using the Log-Binomial Model

Description Methods for fitting log-link GLMs and GAMs to binomial data, including EM-type algorithms with more stable convergence properties than standard methods.

Version 2.0.5.9000

Depends R (>= 3.0.1)

Imports splines, glm2, turboEM (>= 2021.1), Matrix, itertools2, iterators

License GPL (>= 2)

URL `https://github.com/mdonoghoe/logbin`

Repository https://mdonoghoe.r-universe.dev

RemoteUrl https://github.com/mdonoghoe/logbin

RemoteRef HEAD

RemoteSha a41d58a5efcca409bfaa85d30c72d70da0b8c2ab

Contents

`logbin-package` *Relative Risk Regression Using the Log-Binomial Model*

Description

Methods for fitting log-link GLMs and GAMs to binomial data, including EM-type algorithms with more stable convergence properties than standard methods.

Details

Package:	logbin
Type:	Package
Version:	2.0.5.9000
License:	GPL (>= 2)

This package provides methods to fit generalised linear models (GLMs) and generalised additive models (GAMs) with log link functions to binomial data, which can be used to estimate adjusted relative risks. It has two primary functions: `logbin` and `logbin.smooth`, together with various supporting functions.

Standard GLM routines such as base R's `glm` typically use a modified Fisher scoring algorithm, but this can experience numerical problems and fail to converge to the maximum likelihood estimate (MLE). The **glm2** package improves on this but can still have difficulties, particularly when the MLE is on or near the boundary of the parameter space (Marschner, 2015).

Alternative methods for finding the MLE are provided in this package. For both GLMs and GAMs, two approaches based on the EM algorithm can be used: a combinatorial EM (CEM) algorithm (Marschner, 2014) or an expanded EM algorithm. These accomodate the parameter constraints and are more stable than iteratively reweighted least squares.

In a CEM algorithm, a collection of restricted parameter spaces is defined which covers the full parameter space, and an EM algorithm is applied within each restricted parameter space in order to find a collection of restricted maxima of the log-likelihood function, from which can be obtained the global maximum over the full parameter space. The methodology implemented for this algorithm is presented in Marschner and Gillett (2012) and Donoghoe and Marschner (2015).

In the expanded EM approach, additional parameters are added to the model, and an EM algorithm finds the MLE of this overparameterised model by imposing constraints on each individual parameter. This requires a single application of the EM algorithm.

In each case, the EM algorithm may be accelerated by using the capabilities of the **turboEM** package.

For GLMs, an adaptive barrier approach, which uses a constrained optimisation algorithm, is also provided.

Author(s)

Mark W. Donoghoe <`markdonoghoe@gmail.com`>

Maintainer: Mark W. Donoghoe <`markdonoghoe@gmail.com`>

References

Donoghoe, M. W. and I. C. Marschner (2015). Flexible regression models for rate differences, risk differences and relative risks. *International Journal of Biostatistics* 11(1): 91–108.

Donoghoe, M. W. and I. C. Marschner (2018). logbin: An R package for relative risk regression using the log-binomial model. *Journal of Statistical Software* 86(9): 1–22.

Marschner, I. C. (2014). Combinatorial EM algorithms. *Statistics and Computing* 24(6): 921–940.

Marschner, I. C. (2015). Relative risk regression for binary outcomes: Methods and recommendations. *Australian & New Zealand Journal of Statistics*. In press.

Marschner, I. C. and A. C. Gillett (2012). Relative risk regression: Reliable and flexible methods for log-binomial models. *Biostatistics* 13(1): 179–192.

See Also

`glm`, **glm2**, **turboEM**

Examples

```
## For examples, see example(logbin) and example(logbin.smooth)
```

`anova.logbin` *Analysis of Deviance for logbin Fits*

Description

Compute an analysis of deviance table for more than one GLM fitted using `logbin`.

Usage

```
## S3 method for class 'logbin'
anova(object, ..., test = NULL)
```

Arguments

`object, ...`	objects of class `"logbin"`, typically the result of a call to `logbin`, or a `list` of objects for the `"logbinlist"` method.
`test`	a character string, (partially) matching one of `"Chisq"`, `"LRT"`, `"Rao"`, `"F"` or `"Cp"`. See `stat.anova`.

Details

Unlike `anova.glm`, specifying a single object is not allowed.

The table has a row for the residual degrees of freedom and deviance for each model. For all but the first model, the change in degrees of freedom and deviance is also given. (This only makes statistical sense if the models are nested.) It is conventional to list the models from smallest to largest, but this is up to the user.

Models where the MLE lies on the boundary of the parameter space will be automatically removed from the list (with a warning), because asymptotic results to not apply to such models.

The table will optionally contain test statistics (and p-values) comparing the reduction in deviance for the row to the residuals. Mallows' *Cp* statistic is the residual deviance plus twice the estimate of σ^2 times the residual degrees of freedom, which is closely related to AIC. You can also choose `"LRT"` and `"Rao"` for likelihood ratio tests and Rao's efficient score test. The former is synonymous with `"Chisq"` (although both have an asymptotic chi-square distribution).

Value

An object of class `"anova"` inheriting from class `"data.frame"`.

Author(s)

Mark W. Donoghoe <`markdonoghoe@gmail.com`>

See Also

`logbin`, `anova.glm`, `anova`

Examples

```
## For an example, see example(logbin)
```

B.Iso *Defining Smooths in logbin.smooth Formulae*

Description

Function used in the definition of smooth terms within `logbin.smooth` model formulae. The function does not evaluate a smooth — it exists purely to help set up a model using smooths.

Usage

```
B(..., knots = NULL, knot.range = 0:5)

Iso(...)
```

Arguments

`...` variable that this smooth is a function of. Note that unlike `gam`, smooths that are functions of more than one variable are not supported.

`knots` *unique* positions of *interior* knots of a B-spline basis. Boundary knots are created automatically.

`knot.range` if `knots` is not specified, a vector containing a series of non-negative integers denoting the number of *interior* knots for which the model will be fit. These are placed at evenly-spaced quantiles of the observed covariate values.

At least one of `knots` or `knot.range` must be non-missing.

Details

The function does not evaluate the variable arguments; the output from this function is used when producing the model matrix, at which point the actual basis functions are constructed.

`B` is used to specify an order-3 B-spline basis (which can be restricted to be monotonically non-decreasing via the `mono` argument in `logbin.smooth`). If `length(knot.range) > 1`, models with each of the specified number of interior knots will be fit, and the model with the best (smallest) `aic.c` will be returned.

`Iso` is used to specify an isotonic basis, designed such that the resulting function has non-negative increments at each observed covariate value. When `Iso` is used, the resulting function will always be monotonically non-decreasing, regardless of the value of `mono`.

Value

An object of class `"B.smooth"` (for `B`) or `"Iso.smooth"` (for `Iso`), which is a list with the following elements:

`term` name of the term provided in the `...` argument.

`termlabel` label for the term in the model; e.g. for term `"x"` it will be `"B(x)"` or `"Iso(x)"`.

`knots` vector of interior knots (if specified). `NA` for `Iso`.

`knot.range` vector of number of interior knots. `NA` for `Iso`.

Author(s)

Mark W. Donoghoe <markdonoghoe@gmail.com>

See Also

`logbin.smooth`

`s` performs a similar function in the `mgcv` package.

Examples

```
## See example(logbin.smooth) for an example of specifying smooths in model
##  formulae.
```

confint.logbin *Confidence Intervals for logbin Model Parameters*

Description

Computes confidence intervals for one or more parameters in a fitted `logbin` model.

Usage

```
## S3 method for class 'logbin'
confint(object, parm, level = 0.95, ...)
```

Arguments

`object` a fitted model object, resulting from a call to `logbin`.

`parm` a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.

`level` the confidence level required.

`...` additional argument(s) passed to `confint.default`.

Details

Calculates confidence intervals for model parameters assuming asymptotic normality and using the result from `vcov.logbin(object)`. As such, if the MLE is on the boundary of the parameter space, (as per `object$boundary`) the normality assumption is invalid and `NA` is returned.

Value

A matrix (or vector) with columns giving lower and upper confidence limits for each parameter. These will be labelled as (1-level)/2 and 1-(1-level)/2 in % (by default 2.5% and 97.5%).

Author(s)

Mark W. Donoghoe <`markdonoghoe@gmail.com`>

See Also

`confint.default`, `vcov.logbin`

Examples

```
## For an example, see example(logbin)
```

contr.isotonic.rev *Contrast Matrix for Reversed Isotonic Covariate*

Description

Return something similar to a contrast matrix for a categorical covariate that we wish to be monotonically non-decreasing in a specified order.

Usage

```
contr.isotonic.rev(n, perm, contrasts = TRUE, sparse = FALSE)
```

Arguments

`n`	a vector of levels for a factor, or the number of levels.
`perm`	a permutation of the levels of `n` (or of the numbers `1:n`), which define the order in which the coefficients must be monotonically non-decreasing.
`contrasts`	a logical indicating whether constrasts should be computed.
`sparse`	included for compatibility reasons. Has no effect.

Details

This function is used in creating the design matrix for categorical covariates with a specified order under a particular parameterisation. This is required if a categorical covariate is defined as monotonic.

In the order specified by `perm`, the coefficient associated with each level is the sum of increments between the following levels. That is, if there are a total of k levels, the first level is defined as $d_2 + d_3 + d_4 + \cdots + d_k$, the second as $d_3 + d_4 + \cdots + d_k$, the third as $d_4 + \cdots + d_k$, and so on. In fitting the model, these increments are constrained to be non-positive.

Note that these are not 'contrasts' as defined in the theory for linear models, rather this is used to define the `contrasts` attribute of each variable so that `model.matrix` produces the desired design matrix.

Value

A matrix with `n` rows and k columns, with $k = n - 1$ if `contrasts` is `TRUE` and $k = n$ if `contrasts` is `FALSE`.

Author(s)

Mark W. Donoghoe <markdonoghoe@gmail.com>

See Also

`model.matrix`, which uses `contr.isotonic.rev` to create the design matrix.

`contr.treatment`, `contrasts` for their usual use in regression models.

Examples

```
contr.isotonic.rev(4,1:4)
contr.isotonic.rev(4,c(1,3,2,4))

# Show how contr.isotonic.rev applies within model.matrix
x <- factor(round(runif(20,0,2)))
mf <- model.frame(~x)
contrasts(x) <- contr.isotonic.rev(levels(x), levels(x))
model.matrix(mf)
```

---

interpret.logbin.smooth

Interpret a logbin.smooth Formula

Description

This is an internal function of package logbin. It is a service routine for logbin.smooth which interprets the smooth parts of the model formula and returns modified formulas to be used in the fitting functions.

Not normally called directly.

Usage

```
interpret.logbin.smooth(formula)
```

Arguments

formula — A formula as supplied to logbin.smooth, which includes at least one B or Iso term.

Value

A list with components:

full.formula — a formula object which is the same as the formula supplied, but with additional arguments removed from the smooth terms. E.g. B(x, knot.range = 0:2) would appear as B(x) in this formula.

fake.formula — a formula object which is the same as the formula supplied, but with smooth terms replaced by their covariates alone. E.g. B(x, knot.range = 0:2) would appear as x in this formula. Used to construct the model matrix.

smooth.spec — a named list containing the results of evaluating the smooth terms. See B and Iso for details.

smooth.ind — a vector containing the indices of the smooth components in the formula.

terms — the result of running terms.formula(formula, specials = c("B", "Iso")).

Author(s)

Mark W. Donoghoe <markdonoghoe@gmail.com>

See Also

logbin.smooth

Examples

```
# Specify a smooth model with knot.range
res <- interpret.logbin.smooth(y ~ B(x, knot.range = 0:2) + x2)
# The knot.range is removed from the full.formula...
print(res$full.formula)
# ...but is stored in the $smooth.spec component of the result:
print(res$smooth.spec$x$knot.range)
```

logbin *Log-Binomial Regression*

Description

logbin fits relative risk (log-link) binomial regression models.

Usage

```
logbin(formula, mono = NULL, data, subset, na.action, start = NULL,
       offset, control = list(...), model = TRUE,
       method = c("cem", "em", "glm", "glm2", "ab"),
       accelerate = c("em", "squarem", "pem", "qn"),
       control.method = list(), warn = TRUE, ...)
```

Arguments

formula — an object of class "formula" (or one that can be coerced into that class): a symbolic description of the model to be fitted. The details of model specification are given under "Details". Note that the model must contain an intercept, and 2nd-order terms (such as interactions) or above are currently not supported by the "cem" and "em" methods — see "Note".

mono — a vector indicating which terms in formula should be restricted to have a monotonically non-decreasing relationship with the outcome. May be specified as names or indices of the terms.

method = "glm" and "glm2" cannot impose monotonicity constraints, and they are not currently supported for method = "ab".

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 logbin is called.

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 be 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.

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 *non-positive* 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, passed to logbin.control.

With method = "cem", epsilon should be smaller than bound.tol.

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

method — a character string that determines which algorithm to use to find the MLE. The main purpose of logbin is the implementation of stable EM-type algorithms: "cem" for the combinatorial EM algorithm, which cycles through a sequence of constrained parameter spaces, or "em" for a single EM algorithm based on an overparameterised model.

"ab" implements an adaptive barrier method, using the constrOptim function.

"glm" or "glm2" may be used to compare the results from the usual IWLS algorithms on the same model.

accelerate — for the "cem" and "em" methods, a character string that determines the acceleration algorithm to be used, (partially) matching one of "em" (no acceleration — the default), "squarem", "pem" or "qn". See turboem for further details. Note that "decme" is not permitted.

control.method — a list of control parameters for the fitting algorithm.

This is passed to the control.method argument of turboem if method = "cem" or "em".

If method = "ab", this is passed to the control argument of constrOptim (and hence to optim — see this documentation for full details). Note that the trace and maxit elements are ignored and the equivalent items from the supplied logbin.control argument are used instead. May also contain element method (default "BFGS"), which is passed to the method argument of constrOptim.

If any items are not specified, the defaults are used.

warn — a logical indicating whether or not warnings should be provided for non-convergence or boundary values.

... — arguments to be used to form the default control argument if it is not supplied directly.

Details

logbin fits a generalised linear model (GLM) with a binomial error distribution and log link function. Predictors are assumed to be continuous, unless they are of class factor, or are character or logical (in which case they are converted to factors). Specifying a predictor as monotonic using the mono argument means that for continuous terms, the associated coefficient will be restricted to be non-negative, and for categorical terms, the coefficients will be non-decreasing in the order of the factor levels. This allows semi-parametric monotonic regression functions, in the form of unsmoothed step-functions. For smooth regression functions see logbin.smooth.

As well as allowing monotonicity constraints, the function is useful when a standard GLM routine, such as glm, fails to converge with a log-link binomial model. For convenience in comparing convergence on the same model, logbin can be used as a wrapper function to glm and glm2 through the method argument.

If glm does achieve successful convergence, and logbin converges to an interior point, then the two results will be identical. However, as illustrated in one of the examples below, glm may still experience convergence problems even when logbin converges to an interior point. Note that if logbin converges to a boundary point, then it may differ slightly from glm even if glm successfully converges, because of differences in the definition of the parameter space. logbin produces valid fitted values for covariate values within the Cartesian product of the observed range of covariate values, whereas glm produces valid fitted values just for the observed covariate combinations (assuming it successfully converges). This issue is only relevant when logbin converges to a boundary point. The adaptive barrier approach defines the parameter space in the same way as glm, so the same comments apply when comparing its results to those from method = "cem" or "em".

The main computational method is an EM-type algorithm which accommodates the parameter constraints in the model and is more stable than iteratively reweighted least squares. This is done in one of two ways, depending on the choice of the method argument.

method = "cem" implements a CEM algorithm (Marschner, 2014), in which a collection of restricted parameter spaces is defined that covers the full parameter space, and an EM algorithm is applied within each restricted parameter space in order to find a collection of restricted maxima of the log-likelihood function, from which can be obtained the global maximum over the full parameter space. See Marschner and Gillett (2012) for further details.

method = "em" implements a single EM algorithm on an overparameterised model, and the MLE of this model is transformed back to the original parameter space.

Acceleration of the EM algorithm in either case can be achieved through the methods of the turboem package, specified through the accelerate argument. However, note that these methods do not have the guaranteed convergence of the standard EM algorithm, particularly when the MLE is on the boundary of its (possibly constrained) parameter space.

Alternatively, an adaptive barrier method can be used by specifying method = "ab", which maximises the likelihood subject to constraints on the fitted values.

Value

logbin returns an object of class "logbin", which inherits from classes "glm" and "lm". The function summary.logbin can be used to obtain or print a summary of the results.

The generic accessor functions coefficients, fitted.values and residuals can be used to extract various useful features of the value returned by logbin. Note that effects will not work.

An object of class "logbin" is a list containing the same components as an object of class "glm" (see the "Value" section of glm). It also includes:

loglik	the maximised log-likelihood.

`aic.c` a small-sample corrected version of Akaike's *An Information Criterion* (Hurvich, Simonoff and Tsai, 1998). This is used by `logbin.smooth` to choose the optimal number of knots for smooth terms.

`xminmax` the minimum and maximum observed values for each of the continuous covariates, to help define the covariate space of the model.

As well as:

`np.coefficients` estimated coefficients associated with the non-positive parameterisation corresponding to the MLE.

`nn.x` non-negative model matrix associated with `np.coefficients`.

`coefhist` (if `control$coeftrace = TRUE`), a matrix or list of matrices containing the coefficient estimates after each EM iteration.

Note

Due to the way in which the covariate space is defined in the CEM algorithm, models that include terms that are functionally dependent on one another — such as interactions and polynomials — may give unexpected results. Categorical covariates should always be entered directly as factors rather than dummy variables. 2-way interactions between factors can be included by calculating a new factor term that has levels corresponding to all possible combinations of the factor levels (see the Example). Non-linear relationships can be included by using `logbin.smooth`.

Author(s)

Mark W. Donoghoe <`markdonoghoe@gmail.com`>

References

Hurvich, C. M., J. S. Simonoff and C.-L. Tsai (1998). Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 60(2): 271–293.

Donoghoe, M. W. and I. C. Marschner (2018). logbin: An R package for relative risk regression using the log-binomial model. *Journal of Statistical Software* 86(9): 1–22.

Marschner, I. C. (2014). Combinatorial EM algorithms. *Statistics and Computing* 24(6): 921–940.

Marschner, I. C. and A. C. Gillett (2012). Relative risk regression: reliable and flexible methods for log-binomial models. *Biostatistics* 13(1): 179–192.

See Also

`logbin.smooth` for semi-parametric models

`turboem` for acceleration methods

`constrOptim` for the adaptive barrier approach.

Examples

```
require(glm2)
data(heart)

#======================================================
#  Model with periodic non-convergence when glm is used
#======================================================

start.p <- sum(heart$Deaths) / sum(heart$Patients)

fit.glm <- glm(cbind(Deaths, Patients-Deaths) ~ factor(AgeGroup) + factor(Severity) +
  factor(Delay) + factor(Region), family = binomial(log),
  start = c(log(start.p), rep(c(0.2, 0.4), 4)), data = heart,
  trace = TRUE, maxit = 100)

fit.logbin <- logbin(formula(fit.glm), data = heart,
                     start = c(log(start.p), rep(c(0.2, 0.4), 4)),
                     trace = 1)
summary(fit.logbin)

# Speed up convergence by using single EM algorithm
fit.logbin.em <- update(fit.logbin, method = "em")

# Speed up convergence by using acceleration methods
fit.logbin.acc <- update(fit.logbin, accelerate = "squarem")
fit.logbin.em.acc <- update(fit.logbin.em, accelerate = "squarem")


#=============================
#  Model with interaction term
#=============================

heart$AgeSev <- 10 * heart$AgeGroup + heart$Severity

fit.logbin.int <- logbin(cbind(Deaths, Patients-Deaths) ~ factor(AgeSev) +
  factor(Delay) + factor(Region), data = heart, trace = 1, maxit = 100000)

summary(fit.logbin.int)
vcov(fit.logbin.int)
confint(fit.logbin.int)
summary(predict(fit.logbin.int, type = "response"))
anova(fit.logbin, fit.logbin.int, test = "Chisq")
```

`logbin.control` *Auxiliary for Controlling logbin Fitting*

Description

Auxiliary function for `logbin` fitting. Typically only used internally by `nplbin`, but may be used to construct a `control` argument to that function.

Usage

```
logbin.control(bound.tol = 1e-06, epsilon = 1e-08, maxit = 10000, trace = 0,
               coeftrace = FALSE)
```

Arguments

`bound.tol` positive tolerance specifying the interior of the parameter space. If the fitted model is more than `bound.tol` away from the boundary of the parameter space then it is assumed to be in the interior. This can allow the computational method to terminate early if an interior maximum is found. No early termination is attempted if `bound.tol = Inf`.

`epsilon` positive convergence tolerance ϵ; the estimates are considered to have converged when $\sqrt{\sum(\theta_{old} - \theta_{new})^2}/\sqrt{\sum \theta_{old}^2} < \epsilon$, where θ is the vector of parameter estimates.

This should be smaller than `bound.tol`.

`maxit` integer giving the maximum number of iterations (for a given parameterisation in the case of the CEM algorithm).

`trace` number indicating level of output that should be produced. >= 1 gives output for each parameterisation, >= 2 gives output at each iteration.

`coeftrace` logical indicating whether the coefficient history should be included as a component of the returned value (for `method = "em"` and `method = "cem"`).

Details

This is used similarly to `glm.control`. The `control` argument of `logbin` is by default passed to the `control` argument of `nplbin`.

When `trace` is greater than zero, calls to `cat` produce the output. Hence, `options(digits = *)` can be used to increase the precision.

Value

A list with components named as the arguments.

Author(s)

Mark W. Donoghoe <`markdonoghoe@gmail.com`>

See Also

`glm.control`, the equivalent function for `glm` fitting.

`nplbin`, the function used to fit `logbin` models.

Examples

```
## Variation on example(glm.control) :

evts <- c(18,17,15,20,10,20,25,13,12)
obs <- rep(30,9)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
oo <- options(digits = 12)
logbin.D93X <- logbin(cbind(evts,obs-evts) ~ outcome + treatment, trace = 2, epsilon = 1e-2)
options(oo)
coef(logbin.D93X)
```

---

logbin.smooth *Smooth Log-Binomial Regression*

Description

`logbin.smooth` fits log-link binomial regression models using a stable CEM algorithm. It provides additional flexibility over `logbin` by allowing for smooth semi-parametric terms.

Usage

```
logbin.smooth(formula, mono = NULL, data, subset, na.action, offset,
              control = list(...), model = TRUE, model.logbin = FALSE,
           method = c("cem", "em"), accelerate = c("em", "squarem", "pem", "qn"),
              control.accelerate = list(), ...)
```

Arguments

`formula` an object of class "`formula`" (or one that can be coerced into that class): a symbolic description of the model to be fitted. The details of model specification are given under "Details". The model must contain an intercept and at least one semi-parametric term, included by using the `B` or `Iso` functions. Note that 2nd-order terms (such as interactions) or above are not currently supported (see `logbin`).

`mono` a vector indicating which terms in `formula` should be restricted to have a monotonically non-decreasing relationship with the outcome. May be specified as names or indices of the terms.

`Iso()` terms are always monotonic.

`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 `logbin.smooth` is called.

`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 `NA`s. The default is set be 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.

`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 *non-positive* 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, passed to `logbin.control`.

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

`model.logbin` a logical value indicating whether the fitted `logbin` object should be included as a component of the returned value.

`method` a character string that determines which EM-type algorithm to use to find the MLE: `"cem"` for the combinatorial EM algorithm, which cycles through a sequence of constrained parameter spaces, or `"em"` for a single EM algorithm based on an overparameterised model.

Unlike `logbin`, methods `"glm"` and `"ab"` are not available because they do not support the necessary monotonicity constraints.

`accelerate` a character string that determines the acceleration algorithm to be used, (partially) matching one of `"em"` (no acceleration – the default), `"squarem"`, `"pem"` or `"qn"`. See `turboem` for further details. Note that `"decme"` is not permitted.

`control.accelerate`

a list of control parameters for the acceleration algorithm. See `turboem` for details of the parameters that apply to each algorithm. If not specified, the defaults are used.

`...` arguments to be used to form the default `control` argument if it is not supplied directly.

Details

`logbin.smooth` performs the same fitting process as `logbin`, providing a stable maximum likelihood estimation procedure for log-link binomial GLMs, with the added flexibility of allowing semi-parametric `B` and `Iso` terms (note that `logbin.smooth` will stop with an error if no semi-parametric terms are specified in the right-hand side of the `formula`; `logbin` should be used instead).

The method partitions the parameter space associated with the semi-parametric part of the model into a sequence of constrained parameter spaces, and defines a fully parametric `logbin` model for each. The model with the highest log-likelihood is the MLE for the semi-parametric model (see Donoghoe and Marschner, 2015).

Value

An object of class `"logbin.smooth"`, which contains the same objects as class `"logbin"` (the same as `"glm"`), as well as:

`model.logbin` if `model.logbin` is `TRUE`; the `logbin` object for the fully parametric model corresponding to the fitted model.

`xminmax.smooth` the minimum and maximum observed values for each of the smooth terms in the model, to help define the covariate space.

`full.formula` the component from `interpret.logbin.smooth(formula)` that contains the `formula` term with any additional arguments to the `B` function removed.

`knots` a named list containing the knot vectors for each of the smooth terms in the model.

Author(s)

Mark W. Donoghoe <markdonoghoe@gmail.com>

References

Donoghoe, M. W. and I. C. Marschner (2015). Flexible regression models for rate differences, risk differences and relative risks. *International Journal of Biostatistics* 11(1): 91–108.

Donoghoe, M. W. and I. C. Marschner (2018). logbin: An R package for relative risk regression using the log-binomial model. *Journal of Statistical Software* 86(9): 1–22.

Marschner, I. C. (2014). Combinatorial EM algorithms. *Statistics and Computing* 24(6): 921–940.

See Also

`logbin`

Examples

```
## Simple example
x <- c(0.3, 0.2, 0.0, 0.1, 0.2, 0.1, 0.7, 0.2, 1.0, 0.9)
y <- c(5, 4, 6, 4, 7, 3, 6, 5, 9, 8)
system.time(m1 <- logbin.smooth(cbind(y, 10-y) ~ B(x, knot.range = 0:2), mono = 1, trace = 1))
## Compare with accelerated version
system.time(m1.acc <- update(m1, accelerate = "squarem"))
## Isotonic relationship
m2 <- logbin.smooth(cbind(y, 10-y) ~ Iso(x))

plot(m1)
plot(m2)

summary(predict(m1, type = "response"))
summary(predict(m2, type = "response"))
```

nplbin *Non-Positive Log-Binomial Regression*

Description

Finds the maximum likelihood estimate of a log-link binomial GLM using an EM algorithm, where each of the coefficients in the linear predictor is restricted to be non-positive.

Usage

```
nplbin(y, x, offset, start, Amat = diag(ncol(x)), control = logbin.control(),
       accelerate = c("em", "squarem", "pem", "qn"),
       control.accelerate = list(list()))
```

Arguments

`y` binomial response. May be a single column of 0/1 or two columns, giving the number of successes and failures.

`x` non-negative covariate matrix.

`offset` non-positive additive offset vector. The default is a vector of zeros.

`start` starting values for the parameter estimates. All elements must be less than or equal to `-control$bound.tol`.

`Amat` matrix that parameter estimates are left-multiplied by before testing for convergence (e.g. to check reduced version of expanded parameter vector).

`control` a `logbin.control` object, which controls the fitting process.

`accelerate` a character string that determines the acceleration algorithm to be used, (partially) matching one of `"em"` (no acceleration – the default), `"squarem"`, `"pem"` or `"qn"`. See `turboem` for further details. Note that `"decme"` is not permitted.

`control.accelerate` a list of control parameters for the acceleration algorithm. See `turboem` for details of the parameters that apply to each algorithm. If not specified, the defaults are used.

Details

This is a workhorse function for `logbin`, and runs the EM algorithm to find the constrained non-positive MLE associated with a log-link binomial GLM. See Marschner and Gillett (2012) for full details.

Value

A list containing the following components

`coefficients` the constrained non-positive maximum likelihood estimate of the parameters.

`residuals` the residuals at the MLE, that is `y - fitted.values` `fitted.values` the fitted mean values.

`rank` the number of parameters in the model (named `"rank"` for compatibility — we assume that models have full rank)

`family` included for compatibility — will always be `binomial(log)`.

`linear.predictors` the linear fit on link scale.

`deviance` up to a constant, minus twice the maximised log-likelihood.

`aic` a version of Akaike's *An Information Criterion*, minus twice the maximised log-likelihood plus twice the number of parameters.

`aic.c` a small-sample corrected version of Akaike's *An Information Criterion* (Hurvich, Simonoff and Tsai, 1998).

`null.deviance` the deviance for the null model, comparable with `deviance`. The null model will include the offset and an intercept.

`iter` the number of iterations of the EM algorithm used.

`weights` included for compatibility — a vector of ones.

`prior.weights` the number of trials associated with each binomial response.

`df.residual` the residual degrees of freedom.

`df.null` the residual degrees of freedom for the null model.

`y` the `y` vector used.

`converged` logical. Did the EM algorithm converge?

`boundary` logical. Is the MLE on the boundary of the parameter space — i.e. are any of the `coefficients` < `control$bound.tol`?

`loglik` the maximised log-likelihood.

`nn.design` the non-negative `x` matrix used.

Author(s)

Mark W. Donoghoe `<markdonoghoe@gmail.com>`.

This function is based on code from Marschner and Gillett (2012) written by Alexandra Gillett.

References

Hurvich, C. M., J. S. Simonoff and C.-L. Tsai (1998). Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 60(2): 271–293.

Marschner, I. C. and A. C. Gillett (2012). Relative risk regression: reliable and flexible methods for log-binomial models. *Biostatistics* 13(1): 179–192.

`plot.logbin.smooth` *Default logbin.smooth Plotting*

Description

The main use is to take a fitted `logbin.smooth` object produced by `logbin.smooth` and plot the component smooth functions that make it up, for specified values of the other covariates.

Alternatively, plots the model diagnostics usually provided by `plot.lm`.

Usage

```
## S3 method for class 'logbin.smooth'
plot(x, type = c("response", "link", "diagnostics"), at = data.frame(),
     knotlines = TRUE, nobs = 1000, ...)
```

Arguments

`x` a fitted `logbin.smooth` object as produced by `logbin.smooth`.

`type` for `"response"` and `"link"`, the type of prediction required. Note that, unlike `predict.logbin.smooth`, `"terms"` is not a valid option.

for `"diagnostics"`, `plot.lm` is called.

`at` a data frame containing the values at which the prediction should be evaluated. The columns must contain the covariates in the model, and several rows may be provided (in which case, multiple lines are drawn on the same plot). Cannot be missing or `NULL`.

`knotlines` logical; if vertical lines should be drawn on the plot to indicate the locations of the knots for B-spline terms.

`nobs` the number of points which should be used to create the curve. These are placed evenly along the range of the observed covariate values from the original model.

`...` other graphics parameters to pass on to plotting commands, in particular any arguments to `plot.lm` (e.g. `which`).

Details

For each smooth covariate in the model of `x`, `predict.logbin.smooth` is used to obtain predicted values for the range of that covariate, with the other covariates remaining fixed at their values given in `at`. Several rows may be provided in `at`, in which case, one curve is drawn for each, and they are coloured using `rainbow(nrow(at))`. If the model contains a single smooth covariate and no other covariates, `at` may be provided as an empty data frame, `data.frame()`.

Value

The function simply generates plots.

Note

If this function is too restrictive, it may be easier to use `predict.logbin.smooth` to get predictions for the dataset of your choice, and do the plotting manually.

Author(s)

Mark W. Donoghoe <`markdonoghoe@gmail.com`>

See Also

`logbin.smooth`, `predict.logbin.smooth`

Examples

```
## For an example, see example(logbin.smooth)
```

`predict.logbin` *Predict Method for logbin Fits*

Description

Obtains predictions from a fitted `logbin` object.

Usage

```
## S3 method for class 'logbin'
predict(object, newdata = NULL, type = c("link", "response", "terms"),
        terms = NULL, na.action = na.pass, checkminmax = TRUE, ...)
```

Arguments

`object` a fitted object of class inheriting from `"logbin"`.

`newdata` optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.

`type` the type of prediction required. The default is on the scale of the linear predictors; the alternative `"response"` is on the scale of the response variable. The `"terms"` option returns a matrix giving the fitted values of each term in the model formula on the linear predictor scale.

The value of this argument can be abbreviated.

`terms` with `type = "terms"` by default all terms are returned. A character vector specifies which terms are to be returned.

`na.action` function determining what should be done with missing values in `newdata`. The default is to predict `NA`.

`checkminmax` logical indicating whether or not values of continuous covariates in `newdata` should be checked to ensure they lie within the covariate space associated with the fitted model. Otherwise predicted values could lie outside the parameter space.

`...` further arguments passed to or from other methods.

Details

If newdata is omitted the predictions are based on the data used for the fit. In that case how cases with missing values in the original fit are treated is determined by the na.action argument of that fit. If na.action = na.omit, omitted cases will not appear in the residuals. If na.action = na.exclude they will appear, with residual value NA. See also napredict.

Value

A vector or matrix of predictions. For type = "terms", this is a matrix with a column per term, and may have an attribute "constant".

Note

Variables are first looked for in newdata and then searched for in the usual way (which will include the environment of the formula used in the fit). A warning will be given if the variables found are not of the same length as those in newdata if it was supplied.

Author(s)

Mark W. Donoghoe <markdonoghoe@gmail.com>

See Also

logbin

predict.glm for the equivalent method for models fit using glm.

Examples

```
## For an example, see example(logbin)
```

predict.logbin.smooth *Predict Method for logbin.smooth Fits*

Description

Obtains predictions from a fitted logbin.smooth object.

Usage

```
## S3 method for class 'logbin.smooth'
predict(object, newdata = NULL, type = c("link", "response", "terms"),
        terms = NULL, na.action = na.pass, ...)
```

Arguments

`object` a fitted object of class inheriting from `"logbin.smooth"`.

`newdata` optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.

`type` the type of prediction required. The default is on the scale of the linear predictors; the alternative `"response"` is on the scale of the response variable. The `"terms"` option returns a matrix giving the fitted values of each term in the model formula on the linear predictor scale.

The value of this argument can be abbreviated.

`terms` with `type = "terms"` by default all terms are returned. A character vector specifies which terms are to be returned.

`na.action` function determining what should be done with missing values in `newdata`. The default is to predict `NA`.

`...` further arguments passed to or from other methods.

Details

`predict.logbin.smooth` constructs the underlying basis functions for smooth variables in `newdata` and runs `predict.logbin` to obtain predictions. Note that if values of smooth covariates in `newdata` are outside the covariate space of `object`, an error will be returned.

If `newdata` is omitted, the predictions are based on the data used for the fit. In that case how cases with missing values in the original fit are treated 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 `napredict`.

Value

A vector or matrix of predictions. For `type = "terms"`, this is a matrix with a column per term, and may have an attribute `"constant"`.

Note

Variables are first looked for in `newdata` and then searched for in the usual way (which will include the environment of the formula used in the fit). A warning will be given if the variables found are not of the same length as those in `newdata` if it was supplied.

Author(s)

Mark W. Donoghoe `<markdonoghoe@gmail.com>`

See Also

`logbin.smooth`, `predict.logbin`

`predict.glm` for the equivalent method for models fit using `glm`.

Examples

```
## For an example, see example(logbin.smooth)
```

summary.logbin *Summarising logbin Model Fits*

Description

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

Usage

```
## S3 method for class 'logbin'
summary(object, correlation = FALSE, ...)

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

Arguments

object	an object of class "logbin", usually from a call to logbin or logbin.smooth.
x	an object of class "summary.logbin", usually from a call to summary.logbin.
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.
signif.stars	logical; if TRUE, 'significance stars' are printed for each coefficient.
...	further arguments passed to or from other methods.

Details

These perform the same function as summary.glm and print.summary.glm, producing similar results for logbin models. print.summary.logbin additionally prints the small-sample corrected AIC (aic.c), and the number of EM iterations for the parameterisation corresponding to the MLE.

The dispersion used in calculating standard errors is fixed as 1.

Value

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

call	the component from object.
family	the component from object.
deviance	the component from object.
aic	the component from object.
aic.c	the component from object.
df.residual	the component from object.
null.deviance	the component from object.

`df.null` the component from `object`.

`iter` the component from `object`.

`deviance.resid` the deviance residuals: see `residuals.glm`.

`coefficients` the matrix of coefficients, standard errors, z-values and p-values.

`aliased` included for compatibility — always `FALSE`.

`dispersion` the inferred/estimated dispersion.

`df` included for compatibility — a 3-vector of the number of coefficients, the number of residual degrees of freedom, and the number of coefficients (again).

`cov.unscaled` the unscaled (`dispersion` = `1`) estimated covariance matrix of the estimated coefficients. `NaN` if `object$boundary` == `TRUE`.

`cov.scaled` ditto, scaled by `dispersion`.

`correlation` if `correlation` is `TRUE`, the estimated correlations of the estimated coefficients. `NaN` if `object$boundary` == `TRUE`.

Note

If `object$boundary` == `TRUE`, the standard errors of the coefficients are not valid, and a matrix of `NaN`s is returned by `vcov.logbin`. If the MLE is not on the boundary but the model contains parameters with monotonicity constraints, the standard errors do not take this into account and should be used with caution.

Author(s)

Mark W. Donoghoe <`markdonoghoe@gmail.com`>

See Also

`logbin`, `summary.glm`

Examples

```
## For examples see example(logbin)
```

`vcov.logbin` *Calculate Variance-Covariance Matrix for a Fitted logbin Model Object*

Description

Returns the variance-covariance matrix of the main parameters of a fitted `logbin` model object.

Usage

```
## S3 method for class 'logbin'
vcov(object, ...)
```

Arguments

`object`	an object of class `"logbin"`, usually from a call to `logbin` or `logbin.smooth`.
`...`	additional arguments for method functions.

Details

An equivalent method to `vcov`, to use with `logbin` models.

Value

A matrix of the estimated covariances between the parameter estimates in the linear or non-linear predictor of the model. This should have row and column names corresponding to the parameter names given by the `coef` method.

Note

If `object$boundary` == `TRUE`, the standard errors of the coefficients are not valid, and a matrix of `NaN`s is returned.

Author(s)

Mark W. Donoghoe <`markdonoghoe@gmail.com`>

See Also

`summary.logbin`, `vcov.glm`

Examples

```
## For an example see example(logbin)
```

Index