Package ‘addreg’

May 12, 2015

Title Additive Regression for Discrete Data
Description Methods for fitting identity-link GLMs and GAMs to discrete data, using EM-type algorithms with more stable convergence properties than standard methods.
Version 2.0
Date 2015-05-12
Author Mark W. Donoghoe <mark.donoghoe@mq.edu.au>
Maintainer Mark W. Donoghoe <mark.donoghoe@mq.edu.au>
Depends R (>= 3.0.1)
Imports splines, combinat, glm2
License GPL (>= 2)
LazyData true
NeedsCompilation no
Repository CRAN
Date/Publication 2015-05-12 09:16:18

R topics documented:

addreg-package .................................................. 2
addbin .......................................................... 3
addreg .......................................................... 4
addreg.allref .................................................. 8
addreg.control .................................................. 10
addreg.design .................................................. 11
addreg.smooth ................................................. 12
addreg.smooth.allref ......................................... 14
addreg.smooth.design .......................................... 16
anova.addreg .................................................. 17
B.Iso ............................................................ 18
confint.addreg ................................................ 20
contr.isotonic .................................................. 21
conv.test ...................................................... 22
interpret.addreg.smooth ...................................... 23
Description

Methods for fitting identity-link GLMs and GAMs to discrete data, using EM-type algorithms with more stable convergence properties than standard methods.

Details

Package: addreg
Type: Package
Version: 2.0
Date: 2015-05-12
License: GPL (>= 2)

This package provides methods to fit generalised linear models (GLMs) and generalised additive models (GAMs) with identity link functions to discrete data using binomial, Poisson and negative binomial models. It is planned that future versions will incorporate other types of discrete data models, such as multinomial regression.

The package has two primary functions: `addreg` and `addreg.smooth`, together with various supporting functions. It is useful in two main situations. The first is when a standard GLM routine, such as `glm`, fails to converge with such a model. The second is when a flexible semi-parametric component is desired in these models. One of the main purposes of this package is to provide parametric and semi-parametric adjustment of risk differences and rate differences.

The computational method is a combinatorial EM algorithm (Marschner, 2014), which accommodates the parameter constraints and is more stable than iteratively reweighted least squares. A collection of restricted parameter spaces is defined which covers the full parameter space, and the 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.

Author(s)

Mark W. Donoghoe <Mark.Donoghoe@mq.edu.au>
addbin

Maintainer: Mark W. Donoghoe <Mark.Donoghoe@mq.edu.au>

References


See Also

`glm`

Examples

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

addbin(y, x, start = NULL, control = list(), allref)

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 design matrix. Must have an intercept column.
- `start`: starting values for the parameters in the linear predictor.
- `control`: list of parameters for controlling the fitting process, passed to `addreg.control`.
- `allref`: a list of all parameterisations for this model, obtained from `addreg.allref`.
Details

An additive binomial fit can be converted into an additive Poisson fit via the multinomial–Poisson
transformation (Baker, 1994). This function transforms the data as described by Donoghoe and
Marschner (2014) and passes it to addreg with a Poisson family to get the maximum likelihood
estimate. The coefficients (and other values) from the Poisson model are transformed back to relate
to the additive binomial model.

This is a workhorse function for addreg when a binomial family is specified. It would not usually
be called directly.

Value

A list of (most of) the components needed for an object of class "addreg"; see addreg for details.

Author(s)

Mark W. Donoghoe <Mark.Donoghoe@mq.edu.au>

References

models with application to adjusted risk differences. Computational Statistics and Data Analysis
80: 184–196.

See Also

addreg

addreg

Additive Regression for Discrete Data

Description

addreg fits additive (identity-link) Poisson, negative binomial and binomial regression models using
a stable combinatorial EM algorithm.

Usage

addreg(formula, mono = NULL, family, data, standard, subset, na.action,
start = NULL, offset, control = list(...), model = TRUE, warn = TRUE,
...)

addreg

Additive Regression for Discrete Data
**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 — 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.

- **family**: a description of the error distribution 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), but here it is restricted to be poisson, negbin1 or binomial family with identity link.

- **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 addreg is called.

- **standard**: a numeric vector of length equal to the number of cases, where each element is a positive constant that (multiplicatively) standardises the fitted value of the corresponding element of the response vector. Ignored for binomial family (two-column specification of response should be used instead).

- **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, also with the starting value for the scale as the last element when family = negbin1.

- **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-negative* 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. Ignored for binomial family.

- **control**: list of parameters for controlling the fitting process, passed to addreg.control.

- **model**: a logical value indicating whether the model frame (and, for binomial models, the equivalent Poisson model) should be included as a component of the returned value.

- **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

`addreg` fits a generalised linear model (GLM) with a Poisson or binomial error distribution and identity link function, as well as additive NegBin I models (which are not GLMs). 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 `addreg.smooth`.

As well as allowing monotonicity constraints, the function is useful when a standard GLM routine, such as `glm`, fails to converge with an identity-link Poisson or binomial model. If `glm` does achieve successful convergence, and `addreg` converges to an interior point, then the two results will be identical. However, `glm` may still experience convergence problems even when `addreg` converges to an interior point. Note that if `addreg` 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. `addreg` 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 `addreg` converges to a boundary point.

The computational method is a combinatorial EM algorithm (Marschner, 2014), which accommodates the parameter constraints in the model and is more stable than iteratively reweighted least squares. A collection of restricted parameter spaces is defined which covers the full parameter space, and the 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 (2010) and Donoghoe and Marschner (2014) for further details.

Value

`addreg` returns an object of class "addreg", which inherits from classes "glm" and "lm". The function `summary.addreg` 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 `addreg`. Note that `effects` will not work.

An object of class "addreg" is a list containing the same components as an object of class "glm" (see the "Value" section of `glm`), but without contrasts, qr, R or effects components. 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 `addreg.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, for Poisson and negative binomial models:
nn.coefficients

estimated coefficients associated with the non-negative parameterisation corresponding to the MLE.

nn.x

non-negative model matrix associated with nn.coefficients.

standard

the standard argument.

Or, for binomial models:

model.addpois

if requested, the addreg object for the associated identity-link Poisson model.

The scale component of the result is fixed at 1 for Poisson and binomial models, and is the constant overdispersion parameter for negative binomial models (that is, scale = 1 + φ) where \( \text{Var}(\mu) = (1 + \phi)\mu \).

Note

Due to the way the covariate space is defined in the CEM algorithm, specifying interactions in the formula is not currently supported by addreg. 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.

Author(s)

Mark W. Donoghoe <Mark.Donoghoe@mq.edu.au>

References


Examples

```r
require(glm2)
data(crabs)

# identity-link Poisson model with periodic non-convergence when glm is used

#---------------------------------------------------------------
crabs.boot <- crabs[crabs$Rep1,-c(5:6)]

#---------------------------------------------------------------
crabs.boot$width.shifted <- crabs.boot$Width - min(crabs$Width)

#---------------------------------------------------------------
fit.glm <- glm(Satellites ~ width.shifted + factor(Dark) + factor(GoodSpine),
                  family = poisson)
```

family = poisson(identity), data = crabs.boot, start = rep(1,4),
control = glm.control(trace = TRUE))

fit.addreg <- addreg(formula(fit.glm), family = poisson, data = crabs.boot,
trace = 1)
summary(fit.addreg)
vcov(fit.addreg)
confint(fit.addreg)
summary(predict(fit.addreg), type = "response")

fit.addreg2 <- addreg(update(formula(fit.glm), ~ . - factor(GoodSpine)),
family = poisson, data = crabs.boot, trace = 1)
anova(fit.addreg2, fit.addreg, test = "LRT")

# Account for overdispersion (use start to speed it up a little)
fit.addreg.od <- addreg(Satellites ~ factor(Dark) + factor(GoodSpine),
family = negbin1, data = crabs.boot, trace = 1,
start = c(4.3423675,-2.4059273,-0.4531984,5.969648))
summary(fit.addreg.od)

---

addreg.allref  

Parameterisation for CEM Algorithm

Description

A workhorse function for addreg, addreg.allref takes the formula and data for an identity-link GLM and produces a list of all parameterisations needed for the associated CEM algorithm.

Usage

addreg.allref(object, data = environment(object), mono, family, start = NULL)

Arguments

object  a model formula or a terms object for the addreg model.
data  a data frame created with model.frame. If another sort of object, model.frame is called first.
mono  a vector indicating which terms should be restricted to have a monotonically non-decreasing relationship with the outcome.
family  the result of a call to a family function. Its component $family must be one of "poisson", "negbin1" or "binomial".
start  starting values for the parameters in the linear predictor.
Details

In the CEM algorithms employed by `addreg`, the parameter space is partitioned into a collection of restricted parameter spaces (see Marschner, 2014). `addreg.allref` finds the list of possible parameterisations of each term in the model.

If a term $x$ has a TRUE value for `is.factor(x)`, `is.character(x)` or `is.logical(x)`, it is considered to be a categorical covariate. For Poisson and negative binomial models, this has a parameterisation for each level of the factor, and for binomial models, every permutation of the levels must be considered (see Donoghoe and Marschner, 2014).

Otherwise the covariate is considered to be continuous, in which case it has two possible parameterisations, relating to the minimum and maximum observed values.

If a covariate is restricted to be monotonic via the `mono` argument, it has only one parameterisation.

The `addreg` function considers all possible combinations of the parameterisations of each covariate, and uses `addreg.design` to create the appropriate non-negative design matrix to be used in the EM algorithm.

Value

A list with components:

- `allref` a named list, with one component for each term in the model. Each component is itself a list, whose components are each of the parameterisations for that term. If `start` was specified, the first component for each term will correspond to the parameterisation specified by `start`.
- `terms` the `terms` component of `object`.
- `data` the object passed into the `data` argument, or the result of calling `model.frame` with `data`.
- `monotonic` a named logical vector indicating which components of `terms` are restricted to be monotonically non-decreasing.
- `start.new` a reparameterised version of `start`, corresponding to the first parameterisation in `allref`. NULL if `start` was not supplied.

Author(s)

Mark W. Donoghoe <mark.donoghoe@mq.edu.au>

References


See Also

`addreg`
addreg.control  

**Auxiliary for Controlling addreg Fitting**

**Description**

Auxiliary function for `addreg` fitting. Typically only used internally by `nnpois`, `nnnegbin` and `addbin`, but may be used to construct a control argument to these functions.

**Usage**

```r
addreg.control(bound.tol = 1e-06, epsilon = 1e-10, maxit = 10000, trace = 0)
```

**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 $\epsilon$; the estimates are considered to have converged when $\sqrt{\sum(\theta_{old} - \theta_{new})^2} / \sqrt{\sum \theta_{old}^2} < \epsilon$, where $\theta$ is the vector of parameter estimates. See `conv.test`.

- `maxit`  
  integer giving the maximum number of EM algorithm iterations for a given parameterisation.

- `trace`  
  number indicating level of output that should be produced. $\geq 1$ gives output for each parameterisation, $\geq 2$ gives output at each iteration.

**Details**

This is used similarly to `glm.control`. The control argument of `addreg` is by default passed to the control argument of `nnpois`, `nnnegbin` or `addbin`.

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 <Mark.Donoghoe@mq.edu.au>

**See Also**

- `glm.control`, the equivalent function for `glm` fitting.
- `nnpois`, `nnnegbin` and `addbin`, the functions used to fit `addreg` models.
## addreg.design

### Description

`addreg.design` constructs the design matrix for an `addreg` model, given a particular parameterisation. This is a workhorse function — it would not normally be called directly.

### Usage

```r
addreg.design(terms, data, allref, design.ref)
```

### Arguments

- **terms**: terms component of object returned from a call to `addreg.allref` for the desired model.
- **data**: data component of object returned from a call to `addreg.allref` for the desired model.
- **allref**: allref component of object returned from a call to `addreg.allref` for the desired model.
- **design.ref**: vector of indices for a particular parameterisation in allref. That is, each element corresponds to a term \( x \) in the model, and the value of the element indicates which item in the list `allref[[x]]` is the reference level in this parameterisation.

### Details

In the CEM algorithm employed by `addreg`, we must consider the Cartesian product of all possible parameterisations. The list of these for each term in the model is constructed by a call to `addreg.allref`, and a list of all possible combinations created using `expand.grid`. For a particular combination of reference levels, `addreg.design` constructs the associated design matrix by transforming data. Continuous covariates are transformed such that their minimum or maximum observed value corresponds to a transformed value of zero; categorical covariates are transformed by using either `contr.treatment` with a specified reference level or `contr.isotonic` so that the levels are increasing in the specified order.

### Examples

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

counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
oo <- options(digits = 12)
addreg.D93X <- addreg(counts ~ outcome + treatment, family = poisson,
  trace = 2, epsilon = 1e-2)
options(o0)
coef(addreg.D93X)
```
Value

A strictly non-negative design matrix to be passed to the relevant fitting function.

Author(s)

Mark W. Donoghoe <Mark.Donoghoe@mq.edu.au>

See Also

`addreg.allref, model.matrix, contr.treatment, contr.isotonic`

---

**addreg.smooth** *Smooth Additive Regression for Discrete Data*

**Description**

`addreg.smooth` fits additive (identity-link) Poisson, negative binomial and binomial regression models using a stable EM algorithm. It provides additional flexibility over `addreg` by allowing for semi-parametric terms.

**Usage**

```r
addreg.smooth(formula, mono = NULL, family, data, standard, subset, 
               na.action, offset, control = list(...), model = TRUE, 
               model.addreg = FALSE, ...)
```

**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 `addreg`).

- **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.

- **family**: a description of the error distribution 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), but here it is restricted to be `poisson`, `negbin1` or `binomial` family with identity link.

- **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 `addreg.smooth` is called.
standard

a numeric vector of length equal to the number of cases, where each element is a positive constant that (multiplicatively) standardises the fitted value of the corresponding element of the response vector. Ignored for binomial family (the two-column specification of response should be used instead).

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.

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-negative* 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`.

Ignored for binomial family.

control

list of parameters for controlling the fitting process, passed to `addreg.control`.

model

a logical value indicating whether the `model.frame` (and, for binomial models, the equivalent Poisson model) should be included as a component of the returned value.

model.addreg

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

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

Details

`addreg.smooth` performs the same fitting process as `addreg`, providing a stable maximum likelihood estimation procedure for identity-link Poisson, negative binomial or binomial models, with the added flexibility of allowing semi-parametric `b` and `iso` terms (note that `addreg.smooth` will stop with an error if no semi-parametric terms are specified in the right-hand side of the formula; `addreg` 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 `addreg` 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 "`addreg.smooth`", which contains the same objects as class "`addreg`" (the same as "`glm`" objects, without contrasts, qr, R or effects components), as well as:

- `model.addreg` if `model.addreg` is `TRUE`; the `addreg` 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.
addreg.smooth.allref

full.formula  the component from \texttt{interpret.addreg.smooth(formula)} that contains the formula term with any additional arguments to the \texttt{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 <Mark.Donoghoe@mq.edu.au>

References


See Also
\texttt{addreg}

Examples

```r
## Simple example
dat <- data.frame(x1 = c(3.2, 3.3, 3.4, 7.9, 3.8, 0.7, 2.0, 5.4, 8.4, 3.0, 1.8, 5.6, 5.5, 9.0, 8.2),
                 x2 = c(1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0),
                 n = c(6, 7, 5, 9, 10, 7, 9, 6, 6, 7, 7, 8, 6, 8, 10),
                 y = c(2, 1, 2, 6, 3, 1, 2, 2, 4, 4, 1, 2, 5, 7, 7))
m1 <- addreg.smooth(cbind(y, n-y) ~ B(x1), knot.range = 1:3) + factor(x2), mono = 1,
data = dat, family = binomial, trace = 1)
plot(m1, at = data.frame(x2 = 0:1))
points(dat$x1, dat$y / dat$n)
```

addreg.smooth.allref \hspace{1cm} \textit{Parameterisation for CEM Algorithm with Smooth Terms}

Description
A workhorse function for \texttt{addreg.smooth}, \texttt{addreg.smooth.allref} takes the formula and data for an identity-link GLM with smooth terms and produces a list of all parameterisations needed for the CEM algorithm associated with the semi-parametric part of the model.

Usage
```
addreg.smooth.allref(object, data = environment(object), mono, family,
addreg.smooth.spec, num.knots)
```
Arguments

- **object**: a terms object for the "fake.formula" associated with an `addreg.smooth` model (see `interpret.addreg.smooth`).
- **data**: a data frame created with `get_all_vars` for the `fake.formula`.
- **mono**: a vector indicating which terms in `fake.formula` should be restricted to have a monotonically non-decreasing relationship with the outcome. May be specified as names or indices of the terms.
- **family**: the family object for the `addreg.smooth` model.
- **addreg.smooth.spec**: details of the smooth terms in the formula; must be a list in the format returned by `interpret.addreg.smooth`.
- **num.knots**: a vector containing the number of interior knots to be used for each smooth term in the model (NA for Iso terms).

Details

Semi-parametric models in `addreg.smooth` use an extended CEM algorithm by partitioning the parameter space associated with the smooth terms into a collection of restricted parameter spaces, each corresponding to a restricted fully parametric model that can be fitted using `addreg`. The workhorse function `addreg.smooth.allref` creates the list of possible parameterisations of each smooth term.

Isotonic terms and monotonic B-spline terms have only one parameterisation: where the minimum fitted value occurs at the minimum of the covariate range.

For Poisson and negative binomial models, general B-spline terms have $k+3$ parameterisations each (where $k$ is the number of internal knots), corresponding to the possible locations of the minimum of the smooth curve along the range of the covariate.

For binomial models, general B-spline terms have $(k+3)!$ parameterisations, corresponding to the permutations of the coefficients.

`addreg.smooth` considers all possible combinations of the number of knots for each smooth term, and all possible combinations of the associated parameterisations, and `addreg.smooth.design` creates the appropriate formula and design matrix to be used in the call to `addreg`.

Value

A list with components:

- **allref**: a named list, with one component for each smooth term in the model. Each component is itself a list, whose components are each of the parameterisations for that term.
- **terms**: the terms component of object.
- **data**: the object passed into the data argument.
- **monotonic**: a named logical vector indicating which components of terms are restricted to be monotonically non-decreasing.
addreg.smooth.design

Description

addreg.smooth.design constructs the design matrix and associated formula for an addreg.smooth model, given a particular parameterisation, to be passed into addreg for fitting.

This is a workhorse function — it would not normally be called directly.

Usage

addreg.smooth.design(interpret, allref, design.knots, design.param)

Arguments

- `interpret` the object returned by running `interpret.addreg.smooth` for the desired model, containing details of the smooth components.
- `allref` the object returned by running `addreg.smooth.allref` for the desired model.
- `design.knots` a vector containing the number of internal knots for each smooth term (NA for iso terms).
- `design.param` a vector of indices for a particular parameterisation in allref$allref. Each element corresponds to a smooth term in the model, and the value indicates which item in the associated list is the reference level for this parameterisation.

Details

For a particular combination of reference levels, addreg.smooth.design constructs the associated design matrix and formula. Specifically, for iso smooth components, it creates the matrix of indicator covariates for increments between levels. For B smooth components, it creates the basis functions using `splinedesign` and then either removing the column associated with the reference level, or transforming them into monotonic B-spline bases (see Donoghoe and Marschner, 2015).

The formula component is altered to include the terms in the design matrix, and allref$monotonic is altered such that all of the smooth coefficients are restricted to be non-negative, as required.
Value

A list with components:

- **formula**: an updated version of `interpret$full.formula`, with smooth terms removed and replaced by the names of their associated basis components.
- **data**: an updated version of `interpret$data`, with columns for the basis functions of the smooth terms added.
- **monotonic**: an updated version of `allref$monotonic`, such that the coefficients associated with the smooth terms for this parameterisation are constrained to be non-negative.
- **knots**: a list, with one component for each smooth term, containing the knot vector for that term (NA for `Iso` terms).

Author(s)

Mark W. Donoghoe <Mark.Donoghoe@mq.edu.au>

References


See Also

`addreg.smooth`, `interpret.addreg.smooth`, `addreg.smooth.allref`, `addreg.design`

Description

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

Usage

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

Arguments

- **object, ...**: objects of class "addreg", typically the result of a call to `addreg`, or a list of objects for the "addreglist" 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’ \( C_p \) statistic is the residual deviance plus twice the estimate of \( \sigma^2 \) 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".

Warning

The comparison between two or more models will only be valid if they are fitted to the same dataset. This may be a problem if there are missing values and R’s default of `na.action = na.omit` is used, and `anova` will detect this with an error.

Author(s)

Mark W. Donoghoe <Mark.Donoghoe@mq.edu.au>

See Also

`addreg`, `anova.glm`, `anova`

Examples

```r
## For an example, see example(addreg)
```
Usage

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

Iso(...)  

Arguments

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

\texttt{knots} unique positions of interior knots of a B-spline basis. Boundary knots are created automatically.

\texttt{knot.range} if \texttt{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 \texttt{knots} or \texttt{knot.range} must be non-missing.

Details

The function does not evaluate the variable arguments; the output from this function is passed as part of the arguments to \texttt{addreg.smooth.design}, which constructs the actual basis functions.

\texttt{B} is used to specify an order-3 B-spline basis (which can be restricted to be monotonically non-decreasing via the \texttt{mono} argument in \texttt{addreg.smooth}). If \texttt{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.

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

Value

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

\begin{itemize}
  \item \texttt{term} name of the term provided in the \ldots\ argument.
  \item \texttt{term.label} label for the term in the model; e.g. for term "x" it will be "B(x)" or "Iso(x)".
  \item \texttt{knots} vector of interior knots (if specified). NA for \texttt{Iso}.
  \item \texttt{knot.range} vector of number of interior knots. NA for \texttt{Iso}.
\end{itemize}

Author(s)

Mark W. Donoghoe <Mark.Donoghoe@mq.edu.au>

See Also

\texttt{addreg.smooth, addreg.smooth.design}s performs a similar function in the \texttt{mgcv} package.
Confidence Intervals for addreg Model Parameters

Description

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

Usage

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

Arguments

- **object**: a fitted model object, resulting from a call to addreg.
- **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, using vcov.addreg(object). As such, if the MLE is on the boundary of the parameter space, (i.e. object$boundary == TRUE) 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 <Mark.Donoghoe@mq.edu.au>

See Also

cconfint.default, vcov.addreg

Examples

## For an example, see example(addreg)
contr.isotonic

Contrast Matrix for Isotonic Covariate

Description

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

Usage

contr.isotonic(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 contrasts should be computed.
- `sparse`: included for compatibility reasons. Has no effect.

Details

This function is used within `addreg.design` for categorical covariates with a specified order under
a particular parameterisation. For Poisson and negative binomial models, this occurs if a categorical
covariate is defined as monotonic; for binomial models, each parameterisation defines a permutation
of the levels that must be monotonically increasing.

In the order specified by `perm`, the coefficient associated with each level is the sum of increments
between the preceding levels. That is, the first level is defined as 0, the second as 0 + d_2, the third
as 0 + d_2 + d_3, and so on. In fitting the model, these increments are constrained to be non-negative.
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 <Mark.Donoghoe@mq.edu.au>

See Also

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

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

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

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

---

### conv.test

**Convergence Test Based on L2 Norm**

**Description**

Performs a test of convergence based on the L2 norm of the change in the parameter estimates.

**Usage**

```r
conv.test(theta1, theta2, epsilon)
```

**Arguments**

- `theta1`: vector of parameter estimates at previous step.
- `theta2`: vector of parameter estimates at current step.
- `epsilon`: positive convergence tolerance.

**Details**

This is used as the convergence test in the `addreg` fitting functions, because the EM algorithm may converge slowly such that the test based on the deviance used in `glm.fit` (see `glm.control`) may report convergence at a point away from the actual optimum.

**Value**

A logical; TRUE if `sqrt(sum((theta1-theta2)**2))/sqrt(sum(theta1**2)) < epsilon`, FALSE otherwise.

**Author(s)**

Mark W. Donoghoe <Mark.Donoghoe@mq.edu.au>

**Examples**

```r
theta.old <- c(4,5,6)
theta.new <- c(4.05,5,6)

conv.test(theta.old, theta.new, 0.01)
conv.test(theta.old, theta.new, 0.005)
```
**interpret.addreg.smooth**

*Interpret an addreg.smooth Formula*

**Description**

This is an internal function of package `addreg`. It is a service routine for `addreg.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.addreg.smooth(formula)`

**Arguments**

- `formula`: A formula as supplied to `addreg.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 <Mark.Donoghoe@mq.edu.au>

**See Also**

`addreg.smooth`
Examples

# Specify a smooth model with knot.range
res <- interpret.addreg.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)

negbin1

Family Functions for Negative Binomial 1 Models

Description

Specifies the information required to fit a negative binomial 1 (NB1) model.

Usage

negbin1(link, phi = stop("'phi' must be given"))

Arguments

link included for compatibility with family. For addreg models, this will always be "identity".

phi the value of the scale parameter of the NB1 distribution (see "Details"). This can be set to NA for initialisation, but during estimation the family should be updated with the current estimate, and must be strictly positive.

Details

The NB1 distribution can be parameterised in terms of a mean $\mu$ and scale parameter $\phi$ (the phi argument of this function), such that if $Y \sim NB1(\mu, \phi)$, then $E(Y) = \mu$ and $Var(Y) = (1 + \phi)\mu$.

These can be related to the size and prob arguments of the NegBinomial functions by size = $\mu/\phi$ and prob = $1/(1 + \phi)$.

Value

An object of class "family": see family for full details. Note that when the estimate of phi is updated in a model, this family object must be reloaded using the new estimate.

Author(s)

Mark W. Donoghoe <Mark.Donoghoe@mq.edu.au>
**Description**

Finds the maximum likelihood estimate of an additive negative binomial (NB1) model using an ECME algorithm, where each of the mean coefficients is restricted to be non-negative.

**Usage**

```r
nnnegbin(y, x, standard, offset, start, control = list())
```

**Arguments**

- `y`: non-negative integer response vector.
- `x`: non-negative covariate matrix.
- `standard`: standardising vector, where each element is a positive constant that (multiplicatively) standardises the fitted value of the corresponding element of the response vector. The default is a vector of ones.
- `offset`: non-negative additive offset vector. The default is a vector of zeros.
- `start`: vector of starting values for the parameter estimates. The last element is the starting value of the scale, and must be > 1. The remaining elements are for the additive mean parameters, and must be greater than `control$bound.tol`.
- `control`: an `addreg.control` object, which controls the fitting process.

**Details**

This is a workhorse function for `addreg`, and runs the ECME algorithm to find the constrained non-negative MLE associated with an additive NB1 model.

**Value**

A list containing the following components:

- `coefficients`: the constrained non-negative maximum likelihood estimate of the mean parameters.
- `scale`: the maximum likelihood estimate of the scale parameter.
- `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 `negbin1(identity)`.
- `linear.predictors`: included for compatibility — same as `fitted.values` (as this is an identity-link model).
deviance

up to a constant, minus twice the maximised log-likelihood (with respect to a
saturated NB1 model with the same scale).

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* (Hur-
vich, 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

included for compatibility — a vector of ones.

standard

the standard vector passed to this function.

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 ECME algorithm converge (according to conv.test)?

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 <Mark.Donoghoe@mq.edu.au>.

References

parametric regression using an improved Akaike information criterion. *Journal of the Royal Statis-
tical Society: Series B (Statistical Methodology)* 60(2): 271–293.

---

**nnpois**

*EM Algorithm for Identity-link Poisson GLM*

**Description**

Finds the maximum likelihood estimate of an identity-link Poisson GLM using an EM algorithm,
where each of the coefficients is restricted to be non-negative.

**Usage**

`nnpois(y, x, standard, offset, start, control = list())`
Arguments

- **y**: non-negative integer response vector.
- **x**: non-negative covariate matrix.
- **standard**: standardising vector, where each element is a positive constant that (multiplicatively) standardises the fitted value of the corresponding element of the response vector. The default is a vector of ones.
- **offset**: non-negative additive offset vector. The default is a vector of zeros.
- **start**: starting values for the parameter estimates. Each element must be greater than `control$bound.tol`.
- **control**: an `addreg.control` object, which controls the fitting process.

Details

This is a workhorse function for `addreg`, and runs the EM algorithm to find the constrained non-negative MLE associated with an identity-link Poisson GLM. See Marschner (2010) for full details.

Value

A list containing the following components:

- **coefficients**: the constrained non-negative 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 `poisson(identity)`.
- **linear.predictors**: included for compatibility — same as `fitted.values` (as this is an identity-link model).
- **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**: included for compatibility — a vector of ones.
- **standard**: the standard vector passed to this function.
- **df.residual**: the residual degrees of freedom.
- **df.null**: the residual degrees of freedom for the null model.
plot.addreg.smooth

y

the y vector used.

converged

logical. Did the EM algorithm converge (according to conv.test)?

boundary

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

loglik

the maximised log-likelihood.

nn.design

the non-negative x matrix used.

Author(s)

Mark W. Donoghoe <mark.donoghoe@mq.edu.au>.

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

References

regression using an improved Akaike information criterion. Journal of the Royal Statistical
Society: Series B (Statistical Methodology) 60(2): 271–293.


Marschner, I. C., A. C. Gillett and R. L. O’Connell (2012). Stratified additive Poisson models:
Computational methods and applications in clinical epidemiology. Computational Statistics and
Data Analysis 56(5): 1115–1130.

plot.addreg.smooth

Default addreg.smooth Plotting

Description

Takes a fitted addreg.smooth object produced by addreg.smooth and plots
the component smooth functions that make it up, on the scale of the linear predictor,
for specified values of the other covariates.

Usage

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

---
Arguments

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

type  the type of prediction required. Note that, unlike `predict.addreg.smooth`, "terms" is not a valid option. Also, because `addreg.smooth` only applies identity-link models, "response" and "link" will have the same results — they are included for consistency.

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 (note: some will not work).

Details

For each smooth covariate in the model of `x`, `predict.addreg.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.addreg.smooth` to get predictions for the dataset of your choice, and do the plotting manually.

Author(s)

Mark W. Donoghoe <Mark.Donoghoe@mq.edu.au>

See Also

`addreg.smooth`, `predict.addreg.smooth`

Examples

```R
## For an example, see example(addreg.smooth)
```
predict.addreg

Predict Method for addreg Fits

Description

Obtains predictions from a fitted addreg object.

Usage

```r
# S3 method for class 'addreg'
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 "addreg".
- **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 specifying 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 <Mark.Donoghoe@mq.edu.au>

See Also

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

Examples

## For an example, see example(addreg)

predict.addreg.smooth  \textit{Predict Method for addreg.smooth Fits}

Description

Obtains predictions from a fitted \texttt{addreg.smooth} object.

Usage

\begin{verbatim}
## S3 method for class 'addreg.smooth'
predict(object, newdata = NULL, type = c("link", "response", "terms"),
    terms = NULL, na.action = na.pass, ...)
\end{verbatim}

Arguments

\begin{itemize}
  \item \textbf{object} \hspace{1cm} a fitted object of class inheriting from "addreg.smooth".
  \item \textbf{newdata} \hspace{1cm} optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
  \item \textbf{type} \hspace{1cm} 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.
  \item \textbf{terms} \hspace{1cm} with type = "terms" by default all terms are returned. A character vector specifies which terms are to be returned.
  \item \textbf{na.action} \hspace{1cm} function determining what should be done with missing values in newdata. The default is to predict NA.
  \item \ldots \hspace{1cm} further arguments passed to or from other methods.
\end{itemize}
Details

predict.addreg.smooth constructs the underlying basis functions for smooth variables in newdata and runs predict.addreg 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; 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 <Mark.Donoghoe@mq.edu.au>

See Also

addreg.smooth, predict.addreg
predict.glm for the equivalent method for models fit using glm.

Examples

## For an example, see example(addreg.smooth)

summary.addreg summary(object, correlation = FALSE, ...)

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

object an object of class "addreg", usually from a call to addreg or addreg.smooth.
x an object of class "summary.addreg", usually from a call to summary.addreg.
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 addreg models. print.summary.addreg additionally prints the small-sample corrected AIC (aic.c), the number of EM iterations for the parameterisation corresponding to the MLE, and for negative binomial models, the estimate of $\phi$ (scale-1) and its standard error.

The dispersion used in calculating standard errors is fixed as 1 for binomial and Poisson models, and is estimated via maximum likelihood for negative binomial models.

Value

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

call the component from object.
family the component from object.
device 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.
For negative binomial models, the object also contains

\[ \phi \] the estimate of \( \phi \) (scale-1).
\[ \text{var}.\phi \] the estimated variance of \( \phi \).

**Note**

If `object$boundary == TRUE`, the standard errors of the coefficients are not valid, and a matrix of NaNs is returned by `vcov.addreg`.

**Author(s)**

Mark W. Donoghoe <Mark.Donoghoe@mq.edu.au>

**See Also**

`addreg`, `summary.glm`

**Examples**

```r
## For an example, see example(addreg)
```

```r
ccalculates variance-covariance matrix for a fitted addreg model ob-
ject
```

**Usage**

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

**Arguments**

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

**Details**

An equivalent method to `vcov`, to use with `addreg` 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$bboundary == TRUE, the standard errors of the coefficients are not valid, and a matrix of NaNs is returned.

Author(s)

Mark W. Donoghoe <Mark.Donoghoe@mq.edu.au>

See Also

summary.addreg, vcov.glm

Examples

## For an example, see example(addreg)
Index

*Topic **design**
  contr.isotonic, 21

*Topic **misc**
  addreg.allref, 8
  addreg.smooth.allref, 14
  conv.test, 22

*Topic **models**
  addbin, 3
  addreg, 4
  addreg.control, 10
  addreg.design, 11
  addreg.smooth.design, 16
  anova.addreg, 17
  confint.addreg, 20
  conv.test, 22
  interpret.addreg.smooth, 23
  negbin1, 24
  plot.addreg.smooth, 28
  predict.addreg, 30
  predict.addreg.smooth, 31
  summary.addreg, 32
  vcov.addreg, 34

*Topic **optimize**
  addreg.control, 10

*Topic **package**
  addreg-package, 2

*Topic **regression**
  addbin, 3
  addreg, 4
  addreg-package, 2
  addreg.smooth, 12
  anova.addreg, 17
  nnnegbin, 25
  nnpois, 26
  plot.addreg.smooth, 28
  predict.addreg, 30
  predict.addreg.smooth, 31
  summary.addreg, 32
  vcov.addreg, 34

*Topic **smooth**
  addreg.smooth, 12
  addreg.smooth.design, 16
  B.Iso, 18
  interpret.addreg.smooth, 23
  plot.addreg.smooth, 28
  predict.addreg.smooth, 31

addbin, 3, 10
addreg, 2–4, 4, 8–18, 20, 22–25, 27, 30, 31, 33, 34
addreg-package, 2
addreg.allref, 3, 8, 11, 12
addreg.control, 3, 5, 10, 13, 25, 27
addreg.design, 9, 11, 17, 21
addreg.smooth, 2, 6, 12, 14–19, 23, 28, 29, 31–34
addreg.smooth.allref, 14, 16, 17
addreg.smooth.design, 15, 16, 19
anova, 18
anova.addreg, 17
anova.addreglist (anova.addreg), 17
anova.glm, 18
as.data.frame, 5, 12

B, 12–14, 16, 23
B (B.Iso), 18
B.Iso, 18
binomial, 4, 5, 12

cat, 10
coeff, 34
coefficients, 6
confint.addreg, 20
confint.default, 20
contr.isotonic, 11, 12, 21
contr.treatment, 11, 12, 21
contrasts, 21
conv.test, 10, 22, 26, 28

effects, 6
INDEX

environment, 5, 12
eval, 23
expand.grid, 11

factor, 6
family, 5, 12, 15, 24
fitted.values, 6
formula, 5, 8, 12, 23

gam, 19
get_all_vars, 15
glm, 2, 3, 6, 10, 13, 31, 32
glm.control, 10, 22
glm.fit, 22

interpret.addreg.smooth, 14–17, 23
is.character, 9
is.factor, 9
is.logical, 9
Iso, 12, 13, 15–17, 23
Iso (B.Iso), 18

list, 17

methods, 32
mgcv, 19
model.frame, 8, 9
model.matrix, 12, 21
model.offset, 5, 13

na.exclude, 5, 13
na.fail, 5, 13
na.omit, 5, 13
napredict, 30, 32
negbin1, 5, 12, 24, 25
NegBinomial, 24
nnnegbin, 10, 25
nnpois, 10, 26

offset, 5, 13
options, 5, 10, 13

plot.addreg.smooth, 28
poisson, 5, 12, 27
predict.addreg, 30, 32
predict.addreg.smooth, 29, 31
predict.glm, 31, 32
print.summary.addreg(summary.addreg), 32
print.summary.glm, 33

rainbow, 29
residuals, 6
residuals.glm, 33

s, 19
splineDesign, 16
stat.anova, 17
summary.addreg, 6, 32, 35
summary.glm, 33, 34

terms, 8, 9, 15
terms.formula, 23

vcov, 34
vcov.addreg, 20, 34, 34
vcov.glm, 35