Package ‘brglm2’

November 22, 2025

Title Bias Reduction in Generalized Linear Models

Version 1.0.1

Description

Estimation and inference from generalized linear models based on various methods for bias reduction and maximum penalized likelihood with powers of the Jeffreys prior as penalty. The 'brglmFit()' fitting method can achieve reduction of estimation bias by solving either the mean bias-reducing adjusted score equations in Firth (1993) <doi:10.1093/biomet/80.1.27> and Kosmidis and Firth (2009) <doi:10.1093/biomet/asp055>, or the median bias-reducing adjusted score equations in Kenne et al. (2017) <doi:10.1093/biomet/asx046>, or through the direct subtraction of an estimate of the bias of the maximum likelihood estimator from the maximum likelihood estimates as in Cordeiro and McCullagh (1991) <`https://www.jstor.org/stable/2345592`>. See Kosmidis et al (2020) <doi:10.1007/s11222-019-09860-6> for more details. Estimation in all cases takes place via a quasi Fisher scoring algorithm, and S3 methods for the construction of of confidence intervals for the reduced-bias estimates are provided. In the special case of generalized linear models for binomial and multinomial responses (both ordinal and nominal), the adjusted score approaches to mean and media bias reduction have been found to return estimates with improved frequentist properties, that are also always finite, even in cases where the maximum likelihood estimates are infinite (e.g. complete and quasi-complete separation; see Kosmidis and Firth, 2020 <doi:10.1093/biomet/asaa052>, for a proof for mean bias reduction in logistic regression). The 'mdyplFit()' fitting method fits logistic regression models using maximum Diaconis-Ylvisaker prior penalized likelihood, which also guarantees finite estimates. High-dimensionality corrections under proportional asymptotics can be applied to the resulting objects; see Sterzinger and Kosmidis (2024) <doi:10.48550/arXiv.2311.07419> for details.

URL `https://github.com/ikosmidis/brglm2`

BugReports `https://github.com/ikosmidis/brglm2/issues`

Depends R (>= 3.5)

Imports MASS, stats, Matrix, graphics, nnet, enrichwith, numDeriv, statmod, nleqslv

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Suggests detectseparation, knitr, rmarkdown, covr, tinytest, VGAM, brglm, mbrglm

VignetteBuilder knitr

NeedsCompilation yes

Author Ioannis Kosmidis [aut, cre] (ORCID: <https://orcid.org/0000-0003-1556-0302>), Euloge Clovis Kenne Pagui [aut] (ORCID: <https://orcid.org/0000-0002-8998-9251>), Federico Boiocchi [ctb], Philipp Sterzinger [ctb] (ORCID: <https://orcid.org/0009-0007-7348-5810>), Kjell Konis [ctb], Nicola Sartori [ctb]

Maintainer Ioannis Kosmidis <ioannis.kosmidis@warwick.ac.uk>

Repository CRAN

Date/Publication 2025-11-22 06:10:17 UTC

Contents

`aids` *Effects of AZT in slowing the development of AIDS symptoms*

Description

The data is from a 3-year study on the effects of AZT in slowing the development of AIDS symptoms. 338 veterans whose immune systems were beginning to falter after infection with the AIDS virus were randomly assigned either to receive AZT immediately or to wait until their T cells showed severe immune weakness.

Usage

```
aids
```

Format

A data frame with 4 rows and 4 variables:

- `symptomatic`: counts of veterans showing AIDS symptoms during the 3-year study
- `asymptomatic`: counts of veterans not showing AIDS symptoms during the 3-year study
- `race`: the race of the veterans with levels `"White"` and `"Black"`
- `AZT_use`: whether the veterans received AZT immediately (`"Yes"`) or waited until their T cells showed severe immune weakness (`"No"`)

Source

The data set is analyzed in Agresti (2002, Subsection 5.4.2). Its original source is New York Times, Feb. 15, 1991.

References

Agresti A (2002). *Categorical Data Analysis*. Wiley Series in Probability and Statistics. Wiley.

See Also

`brmultinom()`

`alligators` *Alligator food choice data*

Description

Alligator food choice data

Usage

```
alligators
```

Format

A data frame with 80 rows and 5 variables:

- `foodchoice`: primary food type, in volume, found in an alligator's stomach, with levels `fish`, `invertebrate`,`reptile`, `bird`, `other`
- `lake`: lake of capture with levels `Hancock`, `Oklawaha`, `Trafford`, `George`.
- `gender`: gender of the alligator with levels `Male` and `Female`
- `size`: size of the alligator with levels `<=2.3` meters long and `>2.3` meters long
- `freq`: number of alligators for each foodchoice, lake, gender and size combination

Source

The alligators data set is analyzed in Agresti (2002, Subsection 7.1.2).

References

Agresti A (2002). *Categorical Data Analysis*. Wiley Series in Probability and Statistics. Wiley.

See Also

`brmultinom()`

bracl	*Bias reduction for adjacent category logit models for ordinal responses using the Poisson trick.*

Description

bracl() is a wrapper of brglmFit() that fits adjacent category logit models with or without proportional odds using implicit and explicit bias reduction methods. See Kosmidis & Firth (2011) for details.

Usage

```
bracl(
  formula,
  data,
  weights,
  subset,
  na.action,
  parallel = FALSE,
  contrasts = NULL,
  model = TRUE,
  x = TRUE,
  control = list(...),
  ...
)
```

Arguments

formula	a formula expression as for regression models, of the form `response ~ predictors`. The response should be a factor (preferably an ordered factor), which will be interpreted as an ordinal response, with levels ordered as in the factor. The model must have an intercept: attempts to remove one will lead to a warning and be ignored. An offset may be used. See the documentation of formula for other details.
data	an optional data frame, list or environment in which to interpret the variables occurring in `formula`.
weights	optional case weights in fitting. Default to 1.
subset	expression saying which subset of the rows of the data should be used in the fit. All observations are included by default.
na.action	a function to filter missing data.
parallel	if `FALSE` (default), then a non-proportional odds adjacent category model is fit, assuming different effects per category; if `TRUE` then a proportional odds adjacent category model is fit. See Details.
contrasts	a list of contrasts to be used for some or all of the factors appearing as variables in the model formula.

`model`	logical for whether the model matrix should be returned.
`x`	should the model matrix be included with in the result (default is `TRUE`).
`control`	a list of parameters for controlling the fitting process. See `brglmControl()` for details.
`...`	arguments to be used to form the default `control` argument if it is not supplied directly.

Details

The `bracl()` function fits adjacent category models, which assume multinomial observations with probabilities with proportional odds of the form

$$\log \frac{\pi_{ij}}{\pi_{ij+1}} = \alpha_j + \beta^T x_i$$

or with non-proportional odds of the form

$$\log \frac{\pi_{ij}}{\pi_{ij+1}} = \alpha_j + \beta_j^T x_i$$

where x_i is a vector of covariates and π_{ij} is the probability that category j is observed at the covariate setting i.

Author(s)

Ioannis Kosmidis `[aut, cre]` `<ioannis.kosmidis@warwick.ac.uk>`

References

Kosmidis I, Kenne Pagui E C, Sartori N (2020). Mean and median bias reduction in generalized linear models. *Statistics and Computing*, **30**, 43-59. doi:10.1007/s11222019098606.

Agresti, A (2010). *Analysis of Ordinal Categorical Data* (2nd edition). Wiley Series in Probability and Statistics. Wiley.

Albert A, Anderson J A (1984). On the Existence of Maximum Likelihood Estimates in Logistic Regression Models. *Biometrika*, **71**, 1-10. doi:10.2307/2336390.

Kosmidis I, Firth D (2011). Multinomial logit bias reduction via the Poisson log-linear model. *Biometrika*, **98**, 755-759. doi:10.1093/biomet/asr026.

Palmgren J (1981). The Fisher Information Matrix for Log Linear Models Arguing Conditionally on Observed Explanatory Variables. *Biometrika*, **68**, 563-566. doi:10.1093/biomet/68.2.563.

See Also

`nnet::multinom()`, `brmultinom()`

Examples

```
data("stemcell", package = "brglm2")

# Adjacent category logit (non-proportional odds)
fit_bracl <- bracl(research ~ as.numeric(religion) + gender, weights = frequency,
                   data = stemcell, type = "ML")
# Adjacent category logit (proportional odds)
fit_bracl_p <- bracl(research ~ as.numeric(religion) + gender, weights = frequency,
                    data = stemcell, type = "ML", parallel = TRUE)
```

---

brglm2 — *brglm2: Bias Reduction in Generalized Linear Models*

Description

Estimation and inference from generalized linear models using implicit and explicit bias reduction methods (Kosmidis, 2014), and other penalized maximum likelihood methods. Currently supported methods include the mean bias-reducing adjusted scores approach in Firth (1993) and Kosmidis & Firth (2009), the median bias-reduction adjusted scores approach in Kenne Pagui et al. (2017), the correction of the asymptotic bias in Cordeiro & McCullagh (1991), the mixed bias-reduction adjusted scores approach in Kosmidis et al (2020), maximum penalized likelihood with powers of the Jeffreys prior as penalty, and maximum likelihood.

Details

In the special case of generalized linear models for binomial, Poisson and multinomial responses (both nominal and ordinal), mean and median bias reduction and maximum penalized likelihood return estimates with improved frequentist properties, that are also always finite, even in cases where the maximum likelihood estimates are infinite (e.g. complete and quasi-complete separation in multinomial regression). Estimation in all cases takes place via a modified Fisher scoring algorithm, and S3 methods for the construction of confidence intervals for the reduced-bias estimates are provided.

The core model fitters are implemented by the functions `brglm_fit()` (univariate generalized linear models), `brmultinom()` (baseline category logit models for nominal multinomial responses), `bracl()` (adjacent category logit models for ordinal multinomial responses), and `brnb()` for negative binomial regression.

The similarly named **brglm** R package can only handle generalized linear models with binomial responses. Special care has been taken when developing **brglm2** in order not to have conflicts when the user loads **brglm2** and **brglm** simultaneously. The development and maintenance of the two packages will continue in parallel, until **brglm2** incorporates all **brglm** functionality and provides an appropriate wrapper to the `brglm::brglm()` function.

Author(s)

Ioannis Kosmidis `[aut, cre]` <ioannis.kosmidis@warwick.ac.uk>

References

Kosmidis I, Firth D (2021). Jeffreys-prior penalty, finiteness and shrinkage in binomial-response generalized linear models. *Biometrika*, **108**, 71-82. doi:10.1093/biomet/asaa052.

Cordeiro G M, McCullagh P (1991). Bias correction in generalized linear models. *Journal of the Royal Statistical Society. Series B (Methodological)*, **53**, 629-643. doi:10.1111/j.25176161.1991.tb01852.x.

Firth D (1993). Bias reduction of maximum likelihood estimates, Biometrika, **80**, 27-38. doi:10.2307/2336755.

Kenne Pagui E C, Salvan A, Sartori N (2017). Median bias reduction of maximum likelihood estimates. *Biometrika*, **104**, 923–938. doi:10.1093/biomet/asx046.

Kosmidis I, Kenne Pagui E C, Sartori N (2020). Mean and median bias reduction in generalized linear models. *Statistics and Computing*, **30**, 43-59. doi:10.1007/s11222019098606.

Kosmidis I, Firth D (2009). Bias reduction in exponential family nonlinear models. *Biometrika*, **96**, 793-804. doi:10.1093/biomet/asp055.

Kosmidis I, Firth D (2010). A generic algorithm for reducing bias in parametric estimation. *Electronic Journal of Statistics*, **4**, 1097-1112. doi:10.1214/10EJS579.

Kosmidis I (2014). Bias in parametric estimation: reduction and useful side-effects. *WIRE Computational Statistics*, **6**, 185-196. doi:10.1002/wics.1296.

See Also

`brglm_fit()`, `brmultinom()`, `bracl()`

`brglm2-defunct` *Defunct Functions in package* **brglm2**

Description

The functions or variables listed here are no longer part of **brglm2**.

Usage

```
check_infinite_estimates(...)

detect_separation(...)
```

Arguments

`...`	arguments to be passed to functions and methods.

Details

- `detect_separation()`: This function is defunct from **brglm2** since version 0.8.0. A new version of `detect_separation()` is now maintained in the **detectseparation** R package.
- `check_infinite_estimates()` is defunct from **brglm2** since version 0.8.0. An new version of `check_infinite_estimates()` is now maintained in the **detectseparation** R package.

brglmControl *Auxiliary function for glm() fitting using the brglmFit() method.*

Description

Typically only used internally by brglmFit(), but may be used to construct a control argument.

Usage

```
brglmControl(
  epsilon = 1e-06,
  maxit = 100,
  check_aliasing = TRUE,
  trace = FALSE,
 type = c("AS_mixed", "AS_mean", "AS_median", "correction", "MPL_Jeffreys", "ML"),
  transformation = "identity",
  slowit = 1,
  response_adjustment = NULL,
  max_step_factor = 12,
  a = 1/2,
  ...
)

brglm_control(
  epsilon = 1e-06,
  maxit = 100,
  check_aliasing = TRUE,
  trace = FALSE,
 type = c("AS_mixed", "AS_mean", "AS_median", "correction", "MPL_Jeffreys", "ML"),
  transformation = "identity",
  slowit = 1,
  response_adjustment = NULL,
  max_step_factor = 12,
  a = 1/2,
  ...
)
```

Arguments

epsilon	positive convergence tolerance epsilon. Default is 1e-06.
maxit	integer giving the maximal number of iterations allowed. Default is 100.
check_aliasing	logical indicating where a QR decomposition of the model matrix should be used to check for aliasing. Default is TRUE. See Details.
trace	logical indicating if output should be produced for each iteration. Default is FALSE.

type	the type of fitting method to be used. The options are "AS_mean" (mean-bias reducing adjusted scores), "AS_median" (median-bias reducing adjusted scores), "AS_mixed" (bias reduction using mixed score adjustments; default), "correction" (asymptotic bias correction), "MPL_Jeffreys" (maximum penalized likelihood with powers of the Jeffreys prior as penalty) and "ML" (maximum likelihood).
transformation	the transformation of the dispersion to be estimated. Default is "identity". See Details.
slowit	a positive real used as a multiplier for the stepsize. The smaller it is the smaller the steps are. Default is 1.
response_adjustment	a (small) positive constant or a vector of such. Default is NULL. See Details.
max_step_factor	the maximum number of step halving steps to consider. Default is 12.
a	power of the Jeffreys prior penalty. See Details.
...	further arguments passed to brglmControl(). Currently ignored in the output.

Details

brglmControl() provides default values and sanity checking for the various constants that control the iteration and generally the behaviour of brglmFit().

When trace = TRUE, calls to cat() produce the output for each iteration. Hence, options(digits = *) can be used to increase the precision.

When check_aliasing = TRUE (default), a QR decomposition of the model matrix is computed to check for aliasing. If the model matrix is known to be of full rank, then check_aliasing = FALSE avoids the extra computational overhead of an additional QR decomposition, which can be substantial for large model matrices. However, setting check_aliasing = FALSE tells brglmFit() that the model matrix is full rank, and hard to trace back errors will result if it is rank deficient.

transformation sets the transformation of the dispersion parameter for which the bias reduced estimates are computed. Can be one of "identity", "sqrt", "inverse", "log" and "inverseSqrt". Custom transformations are accommodated by supplying a list of two expressions (transformation and inverse transformation). See the examples for more details.

The value of response_adjustment is only relevant if brglmFit() is called with start = NULL, and family is binomial() or poisson(). For those models, an initial maximum likelihood fit is obtained on adjusted data to provide starting values for the iteration in brglmFit(). The value of response_adjustment governs how the data is adjusted. Specifically, if family is binomial(), then the responses and totals are adjusted by response_adjustment and 2 * response_adjustment, respectively; if family is poisson(), then the responses are adjusted by and response_adjustment. response_adjustment = NULL (default) is equivalent to setting it to "number of parameters" / "number of observations".

When type = "AS_mixed" (default), mean bias reduction is used for the regression parameters, and median bias reduction for the dispersion parameter, if that is not fixed. This adjustment has been developed based on equivariance arguments (see, Kosmidis et al, 2020, Section 4) in order to produce regression parameter estimates that are invariant to arbitrary contrasts, and estimates for the dispersion parameter that are invariant to arbitrary non-linear transformations. type = "AS_mixed" and type = "AS_mean" return the same results if brglmFit() is called with family binomial() or poisson() (i.e. families with fixed dispersion).

When type = "MPL_Jeffreys", brglmFit() will maximize the penalized log-likelihood

$$l(\beta, \phi) + a \log \det i(\beta, \phi)$$

where $i(\beta, \phi)$ is the expected information matrix about the regression parameters β and the dispersion parameter ϕ. See, vignette("iteration", "brglm2") for more information. The argument a controls the amount of penalization and its default value is a = 1/2, corresponding to maximum penalized likelihood using a Jeffreys-prior penalty. See, Kosmidis & Firth (2021) for proofs and discussion about the finiteness and shrinkage properties of the maximum penalized likelihood estimators for binomial-response generalized linear models.

The estimates from type = "AS_mean" and type = "MPL_Jeffreys" with a = 1/2 (default) are identical for Poisson log-linear models and logistic regression models, i.e. for binomial and Poisson regression models with canonical links. See, Firth (1993) for details.

brglm_control() is an alias to brglmControl().

Value

a list with components named as the arguments, including symbolic expressions for the dispersion transformation (Trans) and its inverse (inverseTrans)

Author(s)

Ioannis Kosmidis [aut, cre] <ioannis.kosmidis@warwick.ac.uk>

References

Kosmidis I, Firth D (2021). Jeffreys-prior penalty, finiteness and shrinkage in binomial-response generalized linear models. *Biometrika*, **108**, 71-82. doi:10.1093/biomet/asaa052.

Kosmidis I, Kenne Pagui E C, Sartori N (2020). Mean and median bias reduction in generalized linear models. *Statistics and Computing*, **30**, 43-59. doi:10.1007/s11222019098606.

Firth D (1993). Bias reduction of maximum likelihood estimates. Biometrika, **80**, 27-38. doi:10.2307/2336755.

See Also

brglm_fit() and glm.fit()

Examples

```
data("coalition", package = "brglm2")
## The maximum likelihood fit with log link
coalitionML <- glm(duration ~ fract + numst2, family = Gamma, data = coalition)

## Bias reduced estimation of the dispersion parameter
coalitionBRi <- glm(duration ~ fract + numst2, family = Gamma, data = coalition,
                    method = "brglmFit")
coef(coalitionBRi, model = "dispersion")

## Bias reduced estimation of log(dispersion)
coalitionBRl <- glm(duration ~ fract + numst2, family = Gamma, data = coalition,
```

```
                  method = "brglmFit", transformation = "log")
coef(coalitionBRl, model = "dispersion")

## Just for illustration: Bias reduced estimation of dispersion^0.25
my_transformation <- list(expression(dispersion^0.25), expression(transformed_dispersion^4))
coalitionBRc <- update(coalitionBRi, transformation = my_transformation)
coef(coalitionBRc, model = "dispersion")
```

---

brglmFit *Fitting function for* glm() *for reduced-bias estimation and inference*

Description

brglmFit() is a fitting method for glm() that fits generalized linear models using implicit and explicit bias reduction methods (Kosmidis, 2014), and other penalized maximum likelihood methods. Currently supported methods include the mean bias-reducing adjusted scores approach in Firth (1993) and Kosmidis & Firth (2009), the median bias-reduction adjusted scores approach in Kenne Pagui et al. (2017), the correction of the asymptotic bias in Cordeiro & McCullagh (1991), the mixed bias-reduction adjusted scores approach in Kosmidis et al (2020), maximum penalized likelihood with powers of the Jeffreys prior as penalty, and maximum likelihood. Estimation is performed using a quasi Fisher scoring iteration (see vignette("iteration", "brglm2"), which, in the case of mean-bias reduction, resembles an iterative correction of the asymptotic bias of the Fisher scoring iterates.

Usage

```
brglmFit(
  x,
  y,
  weights = rep(1, nobs),
  start = NULL,
  etastart = NULL,
  mustart = NULL,
  offset = rep(0, nobs),
  family = gaussian(),
  control = list(),
  intercept = TRUE,
  fixed_totals = NULL,
  singular.ok = TRUE
)

brglm_fit(
  x,
  y,
  weights = rep(1, nobs),
  start = NULL,
  etastart = NULL,
```

```
    mustart = NULL,
    offset = rep(0, nobs),
    family = gaussian(),
    control = list(),
    intercept = TRUE,
    fixed_totals = NULL,
    singular.ok = TRUE
)
```

Arguments

`x`	a design matrix of dimension `n * p`.
`y`	a vector of observations of length `n`.
`weights`	an optional vector of 'prior weights' to be used in the fitting process. Should be `NULL` or a numeric vector.
`start`	starting values for the parameters in the linear predictor. If `NULL` (default) then the maximum likelihood estimates are calculated and used as starting values.
`etastart`	applied only when start is not `NULL`. Starting values for the linear predictor to be passed to `glm.fit()` when computing starting values using maximum likelihood.
`mustart`	applied only when start is not `NULL`. Starting values for the vector of means to be passed to `glm.fit()` when computing starting values using maximum likelihood.
`offset`	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases. One or more `offset` terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See `model.offset`.
`family`	a description of the error distribution and link function to be used in the model. For `glm` this can be a character string naming a family function, a family function or the result of a call to a family function. For `glm.fit` only the third option is supported. (See `family` for details of family functions.)
`control`	a list of parameters controlling the fitting process. See `brglmControl()` for details.
`intercept`	logical. Should an intercept be included in the *null* model?
`fixed_totals`	effective only when `family` is `poisson()`. Either `NULL` (no effect) or a vector that indicates which counts must be treated as a group. See Details for more information and `brmultinom()`.
`singular.ok`	logical. If `FALSE`, a singular model is an error.

Details

A detailed description of the supported adjustments and the quasi Fisher scoring iteration is given in the iteration vignette (see, `vignette("iteration", "brglm2")` or Kosmidis et al, 2020). A shorter description of the quasi Fisher scoring iteration is also given in one of the vignettes of the *enrichwith* R package (see, https://cran.r-project.org/package=enrichwith/vignettes/

bias.html). Kosmidis and Firth (2010) describe a parallel quasi Newton-Raphson iteration with the same stationary point.

In the special case of generalized linear models for binomial, Poisson and multinomial responses, the adjusted score equation approaches for `type = "AS_mixed"`, `type = "AS_mean"`, and `type = "AS_median"` (see below for what methods each `type` corresponds) return estimates with improved frequentist properties, that are also always finite, even in cases where the maximum likelihood estimates are infinite (e.g. complete and quasi-complete separation in multinomial regression). See, Kosmidis and Firth (2021) for a proof for binomial-response GLMs with Jeffreys-prior penalties to the log-likelihood, which is equivalent to mean bias reduction for logistic regression. See, also, `detectseparation::detect_separation()` and `detectseparation::check_infinite_estimates()` for pre-fit and post-fit methods for the detection of infinite estimates in binomial response generalized linear models.

The type of score adjustment to be used is specified through the `type` argument (see `brglmControl()` for details). The available options are

- `type = "AS_mixed"`: the mixed bias-reducing score adjustments in Kosmidis et al (2020) that result in mean bias reduction for the regression parameters and median bias reduction for the dispersion parameter, if any; default.
- `type = "AS_mean"`: the mean bias-reducing score adjustments in Firth, 1993 and Kosmidis & Firth, 2009. `type = "AS_mixed"` and `type = "AS_mean"` will return the same results when `family` is `binomial()` or `poisson()`, i.e. when the dispersion is fixed
- `type = "AS_median"`: the median bias-reducing score adjustments in Kenne Pagui et al. (2017)
- `type = "MPL_Jeffreys"`: maximum penalized likelihood with powers of the Jeffreys prior as penalty.
- `type = "ML"`: maximum likelihood.
- `type = "correction"`: asymptotic bias correction, as in Cordeiro & McCullagh (1991).

The null deviance is evaluated based on the fitted values using the method specified by the `type` argument (see `brglmControl()`).

The `family` argument of the current version of `brglmFit()` can accept any combination of `"family"` objects and link functions, including families with user-specified link functions, `mis()` links, and `power()` links, but excluding `quasi()`, `quasipoisson()` and `quasibinomial()` families.

The description of `method` argument and the `Fitting functions` section in `glm()` gives information on supplying fitting methods to `glm()`.

`fixed_totals` specifies groups of observations for which the sum of the means of a Poisson model will be held fixed to the observed count for each group. This argument is used internally in `brmultinom()` and `bracl()` for baseline-category logit models and adjacent category logit models, respectively.

`brglm_fit()` is an alias to `brglmFit()`.

Value

An object inheriting from `"brglmFit"` object, which is a list having the same elements to the list that `stats::glm.fit()` returns, with a few extra arguments.

Author(s)

Ioannis Kosmidis `[aut, cre]` `<ioannis.kosmidis@warwick.ac.uk>`, Euloge Clovis Kenne Pagui `[ctb]` `<kenne@stat.unipd.it>`

References

Kosmidis I, Firth D (2021). Jeffreys-prior penalty, finiteness and shrinkage in binomial-response generalized linear models. *Biometrika*, **108**, 71-82. doi:10.1093/biomet/asaa052.

Kosmidis I, Kenne Pagui E C, Sartori N (2020). Mean and median bias reduction in generalized linear models. *Statistics and Computing*, **30**, 43-59. doi:10.1007/s11222019098606.

Cordeiro G M, McCullagh P (1991). Bias correction in generalized linear models. *Journal of the Royal Statistical Society. Series B (Methodological)*, **53**, 629-643. doi:10.1111/j.25176161.1991.tb01852.x.

Firth D (1993). Bias reduction of maximum likelihood estimates. *Biometrika*. **80**, 27-38. doi:10.2307/2336755.

Kenne Pagui E C, Salvan A, Sartori N (2017). Median bias reduction of maximum likelihood estimates. *Biometrika*, **104**, 923–938. doi:10.1093/biomet/asx046.

Kosmidis I, Firth D (2009). Bias reduction in exponential family nonlinear models. *Biometrika*, **96**, 793-804. doi:10.1093/biomet/asp055.

Kosmidis I, Firth D (2010). A generic algorithm for reducing bias in parametric estimation. *Electronic Journal of Statistics*, **4**, 1097-1112. doi:10.1214/10EJS579.

Kosmidis I (2014). Bias in parametric estimation: reduction and useful side-effects. *WIRE Computational Statistics*, **6**, 185-196. doi:10.1002/wics.1296.

See Also

`brglmControl()`, `glm.fit()`, `glm()`

Examples

```
## The lizards example from ?brglm::brglm
data("lizards", package = "brglm2")
# Fit the model using maximum likelihood
lizardsML <- glm(cbind(grahami, opalinus) ~ height + diameter +
                 light + time, family = binomial(logit), data = lizards,
                 method = "glm.fit")
# Mean bias-reduced fit:
lizardsBR_mean <- glm(cbind(grahami, opalinus) ~ height + diameter +
                      light + time, family = binomial(logit), data = lizards,
                      method = "brglmFit")
# Median bias-reduced fit:
lizardsBR_median <- glm(cbind(grahami, opalinus) ~ height + diameter +
                        light + time, family = binomial(logit), data = lizards,
                        method = "brglmFit", type = "AS_median")
summary(lizardsML)
summary(lizardsBR_median)
summary(lizardsBR_mean)

# Maximum penalized likelihood with Jeffreys prior penatly
```

```
lizards_Jeffreys <- glm(cbind(grahami, opalinus) ~ height + diameter +
                        light + time, family = binomial(logit), data = lizards,
                        method = "brglmFit", type = "MPL_Jeffreys")
# lizards_Jeffreys is the same fit as lizardsBR_mean (see Firth, 1993)
all.equal(coef(lizardsBR_mean), coef(lizards_Jeffreys))

# Maximum penalized likelihood with powers of the Jeffreys prior as
# penalty. See Kosmidis & Firth (2021) for the finiteness and
# shrinkage properties of the maximum penalized likelihood
# estimators in binomial response models

a <- seq(0, 20, 0.5)
coefs <- sapply(a, function(a) {
      out <- glm(cbind(grahami, opalinus) ~ height + diameter +
             light + time, family = binomial(logit), data = lizards,
             method = "brglmFit", type = "MPL_Jeffreys", a = a)
      coef(out)
})
# Illustration of shrinkage as a grows
matplot(a, t(coefs), type = "l", col = 1, lty = 1)
abline(0, 0, col = "grey")



## Another example from
## King, Gary, James E. Alt, Nancy Elizabeth Burns and Michael Laver
## (1990).  "A Unified Model of Cabinet Dissolution in Parliamentary
## Democracies", _American Journal of Political Science_, **34**, 846-870

data("coalition", package = "brglm2")
# The maximum likelihood fit with log link
coalitionML <- glm(duration ~ fract + numst2, family = Gamma, data = coalition)
# The mean bias-reduced fit
coalitionBR_mean <- update(coalitionML, method = "brglmFit")
# The bias-corrected fit
coalitionBC <- update(coalitionML, method = "brglmFit", type = "correction")
# The median bias-corrected fit
coalitionBR_median <- update(coalitionML, method = "brglmFit", type = "AS_median")



## An example with offsets from Venables & Ripley (2002, p.189)
data("anorexia", package = "MASS")

anorexML <- glm(Postwt ~ Prewt + Treat + offset(Prewt),
                family = gaussian, data = anorexia)
anorexBC <- update(anorexML, method = "brglmFit", type = "correction")
anorexBR_mean <- update(anorexML, method = "brglmFit")
anorexBR_median <- update(anorexML, method = "brglmFit", type = "AS_median")

# All methods return the same estimates for the regression
# parameters because the maximum likelihood estimator is normally
# distributed around the `true` value under the model (hence, both
```

```
# mean and component-wise median unbiased). The Wald tests for
# anorexBC and anorexBR_mean differ from anorexML because the
# bias-reduced estimator of the dispersion is the unbiased, by
# degree of freedom adjustment (divide by n - p), estimator of the
# residual variance. The Wald tests from anorexBR_median are based
# on the median bias-reduced estimator of the dispersion that
# results from a different adjustment of the degrees of freedom
# (divide by n - p - 2/3)
summary(anorexML)
summary(anorexBC)
summary(anorexBR_mean)
summary(anorexBR_median)


## endometrial data from Heinze & Schemper (2002) (see ?endometrial)
data("endometrial", package = "brglm2")
endometrialML <- glm(HG ~ NV + PI + EH, data = endometrial,
                     family = binomial("probit"))
endometrialBR_mean <- update(endometrialML, method = "brglmFit",
                             type = "AS_mean")
endometrialBC <- update(endometrialML, method = "brglmFit",
                        type = "correction")
endometrialBR_median <- update(endometrialML, method = "brglmFit",
                               type = "AS_median")
summary(endometrialML)
summary(endometrialBC)
summary(endometrialBR_mean)
summary(endometrialBR_median)
```

`brmultinom` *Bias reduction for multinomial response models using the Poisson trick.*

Description

`brmultinom()` is a wrapper of `brglmFit()` that fits multinomial regression models using implicit and explicit bias reduction methods. See Kosmidis & Firth (2011) for details.

Usage

```
brmultinom(
  formula,
  data,
  weights,
  subset,
  na.action,
  contrasts = NULL,
  ref = 1,
```

```
    model = TRUE,
    x = TRUE,
    control = list(...),
    ...
)
```

Arguments

formula	a formula expression as for regression models, of the form response ~ predictors. The response should be a factor or a matrix with K columns, which will be interpreted as counts for each of K classes. A log-linear model is fitted, with coefficients zero for the first class. An offset can be included: it should be a numeric matrix with K columns if the response is either a matrix with K columns or a factor with K >= 2 classes, or a numeric vector for a response factor with 2 levels. See the documentation of formula() for other details.
data	an optional data frame in which to interpret the variables occurring in formula.
weights	optional case weights in fitting.
subset	expression saying which subset of the rows of the data should be used in the fit. All observations are included by default.
na.action	a function to filter missing data.
contrasts	a list of contrasts to be used for some or all of the factors appearing as variables in the model formula.
ref	the reference category to use for multinomial regression. Either an integer, in which case levels(response)[ref] is used as a baseline, or a character string. Default is 1.
model	logical. If true, the model frame is saved as component model of the returned object.
x	should the model matrix be included with in the result (default is TRUE).
control	a list of parameters for controlling the fitting process. See brglmControl() for details.
...	arguments to be used to form the default control argument if it is not supplied directly.

Details

The models brmultinom() handles are also known as baseline-category logit models (see, Agresti, 2002, Section 7.1), because they model the log-odds of every category against a baseline category. The user can control which baseline (or reference) category is used via the ref. By default brmultinom() uses the first category as reference.

The maximum likelihood estimates for the parameters of baseline-category logit models have infinite components with positive probability, which can result in problems in their estimation and the use of inferential procedures (e.g. Wad tests). Albert and Andreson (1984) have categorized the possible data patterns for such models into the exclusive and exhaustive categories of complete separation, quasi-complete separation and overlap, and showed that infinite maximum likelihood estimates result when complete or quasi-complete separation occurs.

The adjusted score approaches to bias reduction that `brmultinom()` implements for `type = "AS_mean"` and `type = "AS_median"` are alternatives to maximum likelihood that result in estimates with smaller asymptotic mean and median bias, respectively, that are also *always* finite, even in cases of complete or quasi-complete separation.

`brmultinom()` is a wrapper of `brglmFit()` that fits multinomial logit regression models through the 'Poisson trick' (see, for example, Palmgren, 1981; Kosmidis & Firth, 2011).

The implementation relies on the construction of an extended model matrix for the log-linear model and constraints on the sums of the Poisson means. Specifically, a log-linear model is fitted on a Kronecker product of the original model matrix `X` implied by the formula, augmented by `nrow(X)` dummy variables.

The extended model matrix is sparse, and the **Matrix** package is used for its effective storage.

While `brmultinom()` can be used for analyses using multinomial regression models, the current implementation is more of a proof of concept and is not expected to scale well with either of `nrow(X)`, `ncol(X)` or the number of levels in the categorical response.

Author(s)

Ioannis Kosmidis `[aut, cre] <ioannis.kosmidis@warwick.ac.uk>`

References

Kosmidis I, Kenne Pagui E C, Sartori N (2020). Mean and median bias reduction in generalized linear models. *Statistics and Computing*, **30**, 43-59. doi:10.1007/s11222019098606.

Agresti A (2002). *Categorical data analysis* (2nd edition). Wiley Series in Probability and Statistics. Wiley.

Albert A, Anderson J A (1984). On the Existence of Maximum Likelihood Estimates in Logistic Regression Models. *Biometrika*, **71** 1–10. doi:10.2307/2336390.

Kosmidis I, Firth D (2011). Multinomial logit bias reduction via the Poisson log-linear model. *Biometrika*, **98**, 755-759. doi:10.1093/biomet/asr026.

Palmgren, J (1981). The Fisher Information Matrix for Log Linear Models Arguing Conditionally on Observed Explanatory Variables. *Biometrika*, **68**, 563-566. doi:10.1093/biomet/68.2.563.

See Also

`nnet::multinom()`, `bracl()` for adjacent category logit models for ordinal responses

Examples

```
## The housing data analysis from ?MASS::housing

data("housing", package = "MASS")

# Maximum likelihood using nnet::multinom
houseML1nnet <- nnet::multinom(Sat ~ Infl + Type + Cont, weights = Freq,
                               data = housing)
# Maximum likelihood using brmultinom with baseline category 'Low'
houseML1 <- brmultinom(Sat ~ Infl + Type + Cont, weights = Freq,
                       data = housing, type = "ML", ref = 1)
```

```
# The estimates are numerically the same as houseML0
all.equal(coef(houseML1nnet), coef(houseML1), tolerance = 1e-04)

# Maximum likelihood using brmultinom with 'High' as baseline
houseML3 <- brmultinom(Sat ~ Infl + Type + Cont, weights = Freq,
                       data = housing, type = "ML", ref = 3)
# The fitted values are the same as houseML1
all.equal(fitted(houseML3), fitted(houseML1), tolerance = 1e-10)

# Bias reduction
houseBR3 <- update(houseML3, type = "AS_mean")
# Bias correction
houseBC3 <- update(houseML3, type = "correction")

## Reproducing Bull et al. (2002, Table 3)
data("hepatitis", package = "brglm2")

# Construct a variable with the multinomial categories according to
# the HCV and nonABC columns
hepat <- hepatitis
hepat$type <- with(hepat, factor(1 - HCV * nonABC + HCV + 2 * nonABC))
hepat$type <- factor(hepat$type, labels = c("noDisease", "C", "nonABC"))
contrasts(hepat$type) <- contr.treatment(3, base = 1)

# Maximum likelihood estimation fails to converge because some estimates are infinite
hepML <- brmultinom(type ~ group * time, data = hepat, weights = counts, type = "ML", slowit = 0.1)

# Mean bias reduction returns finite estimates
hep_meanBR <- brmultinom(type ~ group * time, data = hepat, weights = counts, type = "AS_mean")
# The estimates in Bull et al. (2002, Table 3, DOI: 10.1016/S0167-9473(01)00048-2)
coef(hep_meanBR)

# Median bias reduction also returns finite estimates, which are a bit larger in absolute value
hep_medianBR <- brmultinom(type ~ group * time, data = hepat, weights = counts, type = "AS_median")
coef(hep_medianBR)
```

---

brnb *Bias reduction for negative binomial regression models*

Description

brnb() is a function that fits negative binomial regression models using implicit and explicit bias reduction methods.

Usage

```
brnb(
  formula,
  data,
```

```
    subset,
    weights = NULL,
    offset = NULL,
    link = "log",
    start = NULL,
    etastart = NULL,
    mustart = NULL,
    control = list(...),
    na.action,
    model = TRUE,
    x = FALSE,
    y = TRUE,
    contrasts = NULL,
    intercept = TRUE,
    singular.ok = TRUE,
    ...
  )
```

Arguments

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

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

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

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

`offset` this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.

`link` The link function. Currently must be one of "log", "sqrt" or "identity".

`start` starting values for the parameters in the linear predictor.

`etastart` starting values for the linear predictor.

`mustart` starting values for the vector of means.

`control` a list of parameters for controlling the fitting process. See brglmControl() for details.

`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.

`model`	a logical value indicating whether *model frame* should be included as a component of the returned value.
`x, y`	For `glm`: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For `glm.fit`: `x` is a design matrix of dimension `n * p`, and `y` is a vector of observations of length `n`.
`contrasts`	an optional list. See the `contrasts.arg` of `model.matrix.default`.
`intercept`	logical. Should an intercept be included in the *null* model?
`singular.ok`	logical; if `FALSE` a singular fit is an error.
`...`	For `glm`: arguments to be used to form the default `control` argument if it is not supplied directly. For `weights`: further arguments passed to or from other methods.

Details

A detailed description of the fitting procedure is given in the iteration vignette (see, `vignette("iteration", "brglm2")` and Kosmidis et al, 2020). The number of iterations when estimating parameters are controlled by the `maxit` argument of `brglmControl()`.

The type of score adjustment to be used is specified through the `type` argument (see `brglmControl()` for details).

The available options are:

- `type = "AS_mixed"`: the mixed bias-reducing score adjustments in Kosmidis et al (2020) that result in mean bias reduction for the regression parameters and median bias reduction for the dispersion parameter, if any; default.
- `type = "AS_mean"`: the mean bias-reducing score adjustments in Firth (1993) and Kosmidis & Firth (2009).
- `type = "AS_median"`: the median bias-reducing score adjustments in Kenne Pagui et al. (2017)
- `type = "MPL_Jeffreys"`: maximum penalized likelihood with powers of the Jeffreys prior as penalty.
- `type = "ML"`: maximum likelihood.
- `type = "correction"`: asymptotic bias correction, as in Cordeiro & McCullagh (1991).

The choice of the parameterization for the dispersion is controlled by the `transformation` argument (see `brglmControl()` for details). The default is `"identity"`. Using `transformation = "inverse"` uses the dispersion parameterization that `MASS::glm.nb()` uses.

Value

A fitted model object of class `"brnb"` inheriting from `"negbin"` and `"brglmFit"`. The object is similar to the output of `brglmFit()` but contains four additional components: `theta` for the estimate of the dispersion parameter, `vcov.mean` for the estimated variance-covariance matrix of the regression coefficients, `vcov.dispersion` for the estimated variance of the dispersion parameter in the chosen parameterization (using the expected information), and `twologlik` for twice the log-likelihood function.

Author(s)

Euloge Clovis Kenne Pagui [aut] <kenne@stat.unipd.it>, Ioannis Kosmidis [aut, cre] <ioannis.kosmidis@warwick.

References

Cordeiro G M, McCullagh P (1991). Bias correction in generalized linear models. *Journal of the Royal Statistical Society. Series B (Methodological)*, **53**, 629-643. doi:10.1111/j.25176161.1991.tb01852.x.

Firth D (1993). Bias reduction of maximum likelihood estimates. *Biometrika*. **80**, 27-38. doi:10.2307/2336755.

Kenne Pagui E C, Salvan A, Sartori N (2017). Median bias reduction of maximum likelihood estimates. *Biometrika*, **104**, 923–938. doi:10.1093/biomet/asx046.

Kosmidis I, Kenne Pagui E C, Sartori N (2020). Mean and median bias reduction in generalized linear models. *Statistics and Computing*, **30**, 43-59. doi:10.1007/s11222019098606.

Kosmidis I, Firth D (2009). Bias reduction in exponential family nonlinear models. *Biometrika*, **96**, 793-804. doi:10.1093/biomet/asp055.

Examples

```
## Example in Saha, K., & Paul, S. (2005). Bias-corrected maximum
## likelihood estimator of the negative binomial dispersion
## parameter.  Biometrics, 61, 179--185.
#
# Number of revertant colonies of salmonella data
salmonella <- data.frame(freq = c(15, 16, 16, 27, 33, 20,
                                  21, 18, 26, 41, 38, 27,
                                  29, 21, 33, 60, 41, 42),
                         dose = rep(c(0, 10, 33, 100, 333, 1000), 3),
                         observation = rep(1:3, each = 6))

# Maximum likelihood fit with glm.nb of MASS
salmonella_fm <- freq ~ dose + log(dose + 10)
fitML_glmnb <- MASS::glm.nb(salmonella_fm, data = salmonella)

# Maximum likelihood fit with brnb
fitML <- brnb(salmonella_fm, data = salmonella,
              link = "log", transformation = "inverse", type = "ML")

# Mean bias-reduced fit
fitBR_mean <- update(fitML, type = "AS_mean")

# Median bias-reduced fit
fitBR_median <- update(fitML, type = "AS_median")

# Mixed bias-reduced fit
fitBR_mixed <- update(fitML, type = "AS_mixed")

# Mean bias-corrected fit
fitBC_mean <- update(fitML, type = "correction")

# Penalized likelihood with Jeffreys-prior penalty
```

```
fit_Jeffreys <- update(fitML, type = "MPL_Jeffreys")

# The parameter estimates from glm.nb and brnb with type = "ML" are
# numerically the same
all.equal(c(coef(fitML_glmnb), fitML_glmnb$theta),
            coef(fitML, model = "full"), check.attributes = FALSE)

# Because of the invariance properties of the maximum likelihood,
# median reduced-bias, and mixed reduced-bias estimators the
# estimate of a monotone function of the dispersion should be
# (numerically) the same as the function of the estimate of the
# dispersion:

# ML
coef(fitML, model = "dispersion")
1 / coef(update(fitML, transformation = "identity"), model = "dispersion")

# Median BR
coef(fitBR_median, model = "dispersion")
1 / coef(update(fitBR_median, transformation = "identity"), model = "dispersion")

# Mixed BR
coef(fitBR_mixed, model = "dispersion")
1 / coef(update(fitBR_mixed, transformation = "identity"), model = "dispersion")

## The same is not true for mean BR
coef(fitBR_mean, model = "dispersion")
1 / coef(update(fitBR_mean, transformation = "identity"), model = "dispersion")


## An example  from Venables & Ripley (2002, p.169).
data("quine", package = "MASS")
quineML <- brnb(Days ~ Sex/(Age + Eth*Lrn), link = "sqrt", transformation="inverse",
                data = quine, type="ML")
quineBR_mean <- update(quineML, type = "AS_mean")
quineBR_median <- update(quineML, type = "AS_median")
quineBR_mixed <- update(quineML, type = "AS_mixed")
quine_Jeffreys <- update(quineML, type = "MPL_Jeffreys")

fits <- list(ML = quineML,
             AS_mean = quineBR_mean,
             AS_median = quineBR_median,
             AS_mixed = quineBR_mixed,
             MPL_Jeffreys = quine_Jeffreys)
sapply(fits, coef, model = "full")
```

coalition *Coalition data*

Description

This data set contains survival data on government coalitions in parliamentary democracies (Belgium, Canada, Denmark, Finland, France, Iceland, Ireland, Israel, Italy, Netherlands, Norway, Portugal, Spain, Sweden, and the United Kingdom) for the period 1945-1987. For parsimony, country indicator variables are omitted in the sample data.

Usage

```
coalition
```

Format

A data frame with 314 rows and the 7 variables "duration", "ciep12", "invest", "fract", "polar", "numst2", and "crisis". For variable descriptions, please refer to King et al (1990).

Note

Data is as it is provided by the **Zeilig** R package.

References

King G, Alt J E, Burns N E, Laver M. (1990). A Unified Model of Cabinet Dissolution in Parliamentary Democracies. *American Journal of Political Science*, **34**, 846-870. doi:10.2307/2111401.

King G, Alt J E, Burns N E, Laver M. ICPSR Publication Related Archive, 1115.

See Also

`brglm_fit()`

`coef.brglmFit` *Extract model coefficients from* `"brglmFit"` *objects*

Description

Extract model coefficients from `"brglmFit"` objects

Usage

```
## S3 method for class 'brglmFit'
coef(object, model = c("mean", "full", "dispersion"), ...)
```

Arguments

`object`	an object for which the extraction of model coefficients is meaningful.
`model`	one of `"mean"` (default), `"dispersion"`, ‘"full", to return the estimates of the parameters in the linear prediction only, the estimate of the dispersion parameter only, or both, respectively.
`...`	other arguments.

Details

See coef() for more details.

See Also

coef()

---

coef.brglmFit_expo *Extract estimates from "brglmFit_expo" objects*

Description

Extract estimates from "brglmFit_expo" objects

Usage

```
## S3 method for class 'brglmFit_expo'
coef(object, ...)
```

Arguments

object	an object for which the extraction of model coefficients is meaningful.
...	other arguments.

---

coef.brnb *Extract model coefficients from "brnb" objects*

Description

Extract model coefficients from "brnb" objects

Usage

```
## S3 method for class 'brnb'
coef(object, model = c("mean", "full", "dispersion"), ...)
```

Arguments

object	an object for which the extraction of model coefficients is meaningful.
model	one of "mean" (default), "full", "dispersion", to return the estimates of the parameters in the linear prediction only, or both, the estimate of the dispersion parameter only, respectively.
...	other arguments.

Details

See coef() for more details.

confint.brglmFit	*Method for computing confidence intervals for one or more regression parameters in a* "brglmFit" *object*

Description

Method for computing confidence intervals for one or more regression parameters in a "brglmFit" object

Usage

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

Arguments

object	a fitted model object.
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) for methods.

confint.brmultinom	*Method for computing confidence intervals for one or more regression parameters in a* "brmultinom" *object*

Description

Method for computing confidence intervals for one or more regression parameters in a "brmultinom" object

Usage

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

Arguments

object	a fitted model object.
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) for methods.

`confint.brnb` *Method for computing Wald confidence intervals for one or more regression parameters in a* `"brnb"` *object*

Description

Method for computing Wald confidence intervals for one or more regression parameters in a `"brnb"` object

Usage

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

Arguments

`object`	a fitted model object.
`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) for methods.

`confint.mdyplFit` *Method for computing confidence intervals for one or more regression parameters in a* `"mdyplFit"` *object*

Description

Method for computing confidence intervals for one or more regression parameters in a `"mdyplFit"` object

Usage

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

Arguments

`object`	a fitted model object.
`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.

hd_correction — if FALSE (default), then the corresponding quantities are computed according to standard asymptotics. If TRUE then the high-dimensionality corrections in Sterzinger & Kosmidis (2024) are employed to updates estimates, estimated standard errors, z-statistics, etc. See Details.

... — additional argument(s) for methods.

Author(s)

Ioannis Kosmidis [aut, cre] <ioannis.kosmidis@warwick.ac.uk>

See Also

mdyplFit(), summary.mdyplFit()

Examples

```
set.seed(123)
n <- 2000
p <- 800
set.seed(123)
betas <- c(rnorm(p / 4, mean = 7, sd = 1), rep(0, 3 * p / 4))
X <- matrix(rnorm(n * p, 0, 1/sqrt(n)), nrow = n, ncol = p)
probs <- plogis(drop(X %*% betas))
y <- rbinom(n, 1, probs)
fit_mdypl <- glm(y ~ -1 + X, family = binomial(), method = "mdyplFit")

wald_ci <- confint(fit_mdypl)
adj_wald_ci <- confint(fit_mdypl, hd_correction = TRUE)
ag_coverage <- function(cis, beta) mean((cis[, 1] < beta) & (cis[, 2] > beta))
ag_coverage(wald_ci, betas)
ag_coverage(adj_wald_ci, betas)
```

---

endometrial — *Histology grade and risk factors for 79 cases of endometrial cancer*

Description

Histology grade and risk factors for 79 cases of endometrial cancer

Usage

```
endometrial
```

Format

A data frame with 79 rows and 4 variables:

- `NV`: neovasculization with coding 0 for absent and 1 for present
- `PI`: pulsality index of arteria uterina
- `EH`: endometrium height
- `HG` histology grade with coding 0 for low grade and 1 for high grade

Source

The packaged data set was downloaded in `.dat` format from `https://users.stat.ufl.edu/~aa/glm/data/`. The latter link provides the data sets used in Agresti (2015).

```
The endometrial data set was first analyzed in Heinze and
Schemper (2002), and was originally provided by Dr
E. Asseryanis from the Medical University of Vienna.
```

References

Agresti A (2015). *Foundations of Linear and Generalized Linear Models*. Wiley Series in Probability and Statistics. Wiley.

Heinze G, Schemper M (2002). A Solution to the Problem of Separation in Logistic Regression. *Statistics in Medicine*, **21**, 2409–2419. doi:10.1002/sim.1047.

Kosmidis I, Firth D (2021). Jeffreys-prior penalty, finiteness and shrinkage in binomial-response generalized linear models. *Biometrika*, **108**, 71-82. doi:10.1093/biomet/asaa052.

See Also

`brglm_fit()`

`enzymes` *Liver enzyme data*

Description

Liver enzyme data collected from 218 patients with liver disease (Plomteux, 1980). The laboratory profile consists of enzymatic activity measured for four liver enzymes: aspartate aminotransferase (`AST`), alanine aminotransferase (`ALT`), glutamate dehydrogenase (`GLDH`) and ornithine carbamyltransferase (`OCT`).

Usage

```
enzymes
```

Format

A data frame with 218 rows and the following 6 columns:

- `Patient`: Patient ID
- `Group`: Four diagnostic groups were considered: acute viral hepatitis (1), persistent chronic hepatitis (2), aggressive chronic hepatitis (3) and post-necrotic cirrhosis (4).
- `AST`: Aspartate aminotransferase (in U/L)
- `ALT`: Alanine aminotransferase (in U/L)
- `GLDH`: Glutamate dehydrogenase (in U/L)
- `OCT`: Ornithine carbamyltransferase (in U/L)

Source

Data from Albert and Harris (1984, Chapter 5, Appendix I), and is also provided by the **pmlr** R package.

References

Albert A, Harris E K (1984). *Multivariate Interpretation of Clinical Laboratory Data*. Dekker: New York.

Plomteux G (1980). Multivariate analysis of an enzyme profile for the differential diagnosis of viral hepatitis. *Clinical Chemistry*, **26**, 1897-1899.

`expo.brglmFit` *Estimate the exponential of parameters of generalized linear models using various methods*

Description

The `expo()` method uses the supplied `"brglmFit"` or `"glm"` object to estimate the exponential of parameters of generalized linear models with maximum likelihood or various mean and median bias reduction methods. `expo()` is useful for computing (corrected) estimates of the multiplicative impact of a unit increase on a covariate on the mean of a Poisson log-linear model (`family = poisson("log")` in `glm()`) while adjusting for other covariates, the odds ratio associated with a unit increase on a covariate in a logistic regression model (`family = binomial("logit")` in `glm()`) while adjusting for other covariates, the relative risk associated with a unit increase on a covariate in a relative risk regression model (`family = binomial("log")` in `glm()`) while adjusting for other covariates, among others.

Usage

```
## S3 method for class 'brglmFit'
expo(
  object,
  type = c("correction*", "correction+", "Lylesetal2012", "AS_median", "ML"),
  level = 0.95
)

## S3 method for class 'glm'
expo(
  object,
  type = c("correction*", "correction+", "Lylesetal2012", "AS_median", "ML"),
  level = 0.95
)
```

Arguments

`object`	an object of class `"brglmFit"` or `"glm"`.
`type`	the type of correction to be used. The available options are `"correction*"` (explicit mean bias correction with a multiplicative adjustment), `"correction*"` (explicit mean bias correction with an additive adjustment), `"Lylesetal2012"` (explicit median bias correction using the multiplicative adjustment in Lyles et al., 2012), `"AS_median"` (median bias reduction), and `"ML"` (maximum likelihood). See Details.
`level`	the confidence level required. Default is `0.95`.

Details

The supported methods through the `type` argument are:

- `"ML"`: the estimates of the exponentiated parameters are $\exp(\hat{\theta}_j)$, where θ_j is the maximum likelihood estimates for the jth regression parameter.
- `"correction*"`: the estimates of the exponentiated parameters are $\exp(\hat{\theta}_j)/(1+\hat{v}_j/2)$, where $\hat{\theta}_j$ is the estimate of the jth regression parameter using `type = "AS_mixed"` in `brglmFit()`.
- `"correction+"`: the estimates of the exponentiated parameters are $\exp(\hat{\theta}_j)(1-\hat{v}_j/2)$, where $\hat{\theta}_j$ is the estimate of the jth regression parameter using `type = "AS_mixed"` in `brglmFit()`.
- `"Lylesetal2012"`: the estimates of the exponentiated parameters are $\exp(\hat{\theta}_j)exp(-\hat{v}_j/2)$, where $\hat{\theta}_j$ is the estimate of the jth regression parameter using `type = "AS_mixed"` in `brglmFit()`. This estimator has been proposed in Lyles et al. (2012).
- `"AS_median"`: the estimates of the exponentiated parameters are $\exp(\hat{\theta}_j)$, where $\hat{\theta}_j$ is the estimate of the jth regression parameter using `type = "AS_median"` in `brglmFit()`.

`"correction*"` and `"correction+"` are based on multiplicative and additive adjustments, respectively, of the exponential of a reduced-bias estimator (like the ones coming from `brglmFit()` with `type = "AS_mixed"`, `type = "AS_mean"`, and `type = "correction"`). The form of those adjustments results from the expression of the first-term in the mean bias expansion of the exponential of a reduced-bias estimator. See, for example, Di Caterina & Kosmidis (2019, expression 12) for the general form of the first-term of the mean bias of a smooth transformation of a reduced-bias estimator.

The estimators from "correction+", "correction*", "Lylesetal2012" have asymptotic mean bias of order smaller than than of the maximum likelihood estimator. The estimators from "AS_median" are asymptotically closed to being median unbiased than the maximum likelihood estimator is.

Estimated standard errors are computed using the delta method, where both the Jacobin and the information matrix are evaluated at the logarithm of the estimates of the exponentiated parameters.

Confidence intervals results by taking the exponential of the limits of standard Wald-type intervals computed at the logarithm of the estimates of the exponentiated parameters.

Value

a list inheriting from class "brglmFit_expo" with components coef (the estimates of the exponentiated regression parameters), se (the corresponding estimated standard errors for the exponentiated parameters), ci (confidence intervals of level level for the exponentiated parameters), and type for the type of correction that has been requested.

Author(s)

Ioannis Kosmidis [aut, cre] <ioannis.kosmidis@warwick.ac.uk>

References

Di Caterina C, Kosmidis I (2019). Location-Adjusted Wald Statistics for Scalar Parameters. *Computational Statistics & Data Analysis*, **138**, 126-142. doi:10.1016/j.csda.2019.04.004.

Kosmidis I, Kenne Pagui E C, Sartori N (2020). Mean and median bias reduction in generalized linear models. *Statistics and Computing*, **30**, 43-59. doi:10.1007/s11222019098606.

Cordeiro G M, McCullagh P (1991). Bias correction in generalized linear models. *Journal of the Royal Statistical Society. Series B (Methodological)*, **53**, 629-643. doi:10.1111/j.25176161.1991.tb01852.x.

Lyles R H, Guo Y, Greenland S (2012). Reducing bias and mean squared error associated with regression-based odds ratio estimators. *Journal of Statistical Planning and Inference*, **142** 3235–3241. doi:10.1016/j.jspi.2012.05.005.

See Also

brglm_fit() and and brglm_control()

Examples

```
## The lizards example from ?brglm::brglm
lizardsML <- glm(cbind(grahami, opalinus) ~ height + diameter +
                  light + time, family = binomial(logit), data = lizards,
                  method = "glm.fit")
# Get estimates, standard errors, and confidence intervals of odds
# ratios with various methods
expo(lizardsML, type = "ML")
expo(lizardsML, type = "correction*")
expo(lizardsML, type = "Lylesetal2012")
```

```
expo(lizardsML, type = "correction+")
expo(lizardsML, type = "AS_median")

## Example from ?glm
## Dobson (1990) Page 93: Randomized Controlled Trial :
df_D93 <- data.frame(counts = c(18,17,15,20,10,20,25,13,12),
                     outcome = gl(3,1,9),
                     treatment = gl(3,3))
glm.D93 <- glm(counts ~ outcome + treatment, family = poisson(), data = df_D93)
expo(glm.D93, type = "correction*")
```

---

hepatitis *Post-transfusion hepatitis: impact of non-A, non-B hepatitis surrogate tests*

Description

Data from a randomized double-blind trial to assess whether withholding donor blood positive for the non-A, non-B ("NANB") surrogate markers would reduce the frequency of post-transfusion hepatitis. The dataset contains 4588 subjects enrolled from 1988 to 1992 into two study groups that received allogenic blood from which units positive for NANB surrogate markers were withheld (n = 2311) or not withheld (n = 2277). Subjects were followed up for 6 months and assessed for the presence of post-transfusion hepatitis.

Usage

```
hepatitis
```

Format

A data frame with 28 rows and the following 6 columns:

- `city`: Subjects were recruited from 3 Canadian Red Cross Society Blood Centres and 13 university-affiliated hospitals in 3 cities: Toronto, Hamilton and Winnipeg.
- `group`: Eligible subjects were assigned to one of two allogenic blood recipient groups. One group received products that had only routine Canadian transfusion-transmissible disease marker screening (no-withhold). The other group received only products that were not positive for NANB surrogate markers (withhold).
- `time`: Hepatitis C (HCV) screening was introduced in Canada in May, 1990. Subjects were recruited into the study before (pre) and after (post) the introduction of anti-HCV testing.
- `HCV`: Post-transfusion HCV hepatitis present (1) or absent (0).
- `nonABC`: Post-transfusion non-A, non-B, non-C hepatitis present (1) or absent (0)
- `counts`: Number of subjects

Source

Data is from Blajchman et al. (1995), also analyzed in Bull et al. (2002), and is also provided by the **pmlr** R package.

References

Bull S B, Mak C, Greenwood C M T (2002). A modified score function estimator for multinomial logistic regression in small samples. *Computational Statistics & Data Analysis*, **39**, 57-74. doi:10.1016/S01679473(01)000482

Blajchman M A, Bull S B and Feinman S V (1995). Post-transfusion hepatitis: impact of non-A, non-B hepatitis surrogate tests. *The Lancet*, **345**, 21–25. doi:10.1016/S01406736(95)911537

`lizards` *Habitat preferences of lizards*

Description

Habitat preferences of lizards

Usage

```
lizards
```

Format

A data frame with 23 rows and 6 columns:

- `grahami`. count of grahami lizards
- `opalinus`. count of opalinus lizards
- `height`. a factor with levels `<5ft`, `>=5ft`
- `diameter`. a factor with levels `<=2in`, `>2in`
- `light`. a factor with levels `sunny`, `shady`
- `time`. a factor with levels `early`, `midday`, `late`

The variables `grahami` and `opalinus` are counts of two lizard species at two different perch heights, two different perch diameters, in sun and in shade, at three times of day.

Source

McCullagh P, Nelder J A (1989) *Generalized Linear Models* (2nd Edition). London: Chapman and Hall.

Originally from

```
Schoener T W (1970) Nonsynchronous spatial overlap of lizards
in patchy habitats.  _Ecology_ *51*, 408-418.
```

See Also

brglm_fit()

---

mdyplControl *Auxiliary function for glm() fitting using the brglmFit() method.*

Description

Typically only used internally by brglmFit(), but may be used to construct a control argument.

Usage

```
mdyplControl(alpha = NULL, epsilon = 1e-08, maxit = 25, trace = FALSE)

mdypl_control(alpha = NULL, epsilon = 1e-08, maxit = 25, trace = FALSE)
```

Arguments

alpha	the shrinkage parameter (in [0, 1]) in the Diaconis-Ylvisaker prior penalty. Default is NULL, which results in alpha = n / (n + p), where n is the sum of the binomial totals and p is the number of model parameters. Setting alpha = 1 corresponds to using maximum likelihood, i.e. no penalization. See Details.
epsilon	positive convergence tolerance epsilon. Default is 1e-08.
maxit	integer giving the maximal number of iterations allowed. Default is 25.
trace	logical indicating if output should be produced for each iteration. Default is FALSE.

Details

Internally, mdyplFit() uses stats::glm.fit() to fit a logistic regression model on responses alpha * y + (1 - alpha) / 2, where y are the original binomial responses scaled by the binomial totals. epsilon, maxit and trace control the stats::glm.fit() call; see stats::glm.control().

Value

A list with components named as the arguments.

Author(s)

Ioannis Kosmidis [aut, cre] <ioannis.kosmidis@warwick.ac.uk>

See Also

mdyplFit(), glm.control()

`mdyplFit` *Fitting function for `glm()` for maximum Diaconis-Ylvisaker prior penalized likelihood estimation of logistic regression models*

Description

`mdyplFit()` is a fitting method for `glm()` that fits logistic regression models using maximum Diaconis-Ylvisaker prior penalized likelihood estimation.

Usage

```
mdyplFit(
  x,
  y,
  weights = rep(1, nobs),
  start = NULL,
  etastart = NULL,
  mustart = NULL,
  offset = rep(0, nobs),
  family = binomial(),
  control = list(),
  intercept = TRUE,
  singular.ok = TRUE
)

mdypl_fit(
  x,
  y,
  weights = rep(1, nobs),
  start = NULL,
  etastart = NULL,
  mustart = NULL,
  offset = rep(0, nobs),
  family = binomial(),
  control = list(),
  intercept = TRUE,
  singular.ok = TRUE
)
```

Arguments

`x`	a design matrix of dimension `n * p`.
`y`	a vector of observations of length `n`.
`weights`	an optional vector of 'prior weights' to be used in the fitting process. Should be `NULL` or a numeric vector.
`start`	starting values for the parameters in the linear predictor.

etastart	starting values for the linear predictor.
mustart	starting values for the vector of means.
offset	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
family	a description of the error distribution and link function to be used in the model. For glm this can be a character string naming a family function, a family function or the result of a call to a family function. For glm.fit only the third option is supported. (See family for details of family functions.)
control	a list of parameters controlling the fitting process. See mdyplControl() for details.
intercept	logical. Should an intercept be included in the *null* model?
singular.ok	logical; if FALSE a singular fit is an error.

Details

mdyplFit() uses stats::glm.fit() to fit a logistic regression model on responses alpha * y + (1 - alpha) / 2, where y are the original binomial responses scaled by the binomial totals. This is equivalent to penalizing the likelihood by the Diaconis-Ylvisaker prior with shrinkage parameter α and regression parameters set to zero. See Rigon & Aliverti (2023) and Sterzinger & Kosmidis (2024).

By default, alpha = n / (p + n) is used, where n is the sum of the binomial totals. Alternative values of alpha can be passed to the control argument; see mdyplControl() for setting up the list passed to control. If alpha = 1 then mdyplFit() will simply do maximum likelihood estimation.

Note that null.deviance, deviance and aic in the resulting object are computed at the adjusted responses. Hence, methods such as logLik() and AIC() use the penalized log-likelihood. With the default alpha, the inferential procedures based on penalized likelihood are asymptotically equivalent to the ones that use the unpenalized likelihood when p/n is vanishing asymptotically.

For high-dimensionality corrected estimates, standard errors and z statistics, use the summary method for "mdyplFit" objects with hd_correction = TRUE.

mdypl_fit() is an alias to mdyplFit().

Value

An object inheriting from "mdyplFit" object, which is a list having the same elements to the list that stats::glm.fit() returns, with a few extra arguments.

Author(s)

Ioannis Kosmidis [aut, cre] <ioannis.kosmidis@warwick.ac.uk>

References

Sterzinger P, Kosmidis I (2024). Diaconis-Ylvisaker prior penalized likelihood for $p/n \to \kappa \in (0, 1)$ logistic regression. *arXiv*:2311.07419v2, https://arxiv.org/abs/2311.07419.

Rigon T, Aliverti E (2023). Conjugate priors and bias reduction for logistic regression models. *Statistics & Probability Letters*, **202**, 109901. doi:10.1016/j.spl.2023.109901.

See Also

mdyplControl(), summary.mdyplFit(), plrtest.mdyplFit(), glm()

Examples

```
data("lizards", package = "brglm2")
liz_fm <- cbind(grahami, opalinus) ~ height + diameter + light + time
## ML fit = MDYPL fit with `alpha = 1`
liz_ml <- glm(liz_fm, family = binomial(), data = lizards,
              method = "mdyplFit", alpha = 1)
liz_ml0 <- glm(liz_fm, family = binomial(), data = lizards)

## liz_ml is the same fit as liz_ml0
summ_liz_ml <- summary(liz_ml)
summ_liz_ml0 <- summary(liz_ml0)
all.equal(coef(summ_liz_ml), coef(summ_liz_ml0))

## MDYPL fit with default `alpha` (see `?mdyplControl`)
liz_fm <- cbind(grahami, opalinus) ~ height + diameter + light + time
liz_mdypl <- glm(liz_ml, family = binomial(), data = lizards,
                 method = "mdyplFit")

## Comparing outputs from ML and MDYPL, with and without
## high-dimensionality corrections.
summary(liz_mdypl)
summary(liz_mdypl, hd_correction = TRUE)
summ_liz_ml
summary(liz_ml, hd_correction = TRUE)
## Not much difference in fits here as this is a low dimensional
## problem with dimensionality constant
(liz_ml$rank - 1) / sum(weights(liz_ml))



## The case study in Section 8 of Sterzinger and
## Kosmidis (2024)
data("MultipleFeatures", package = "brglm2")

## Center the fou.* and kar.* features
vars <- grep("fou|kar", names(MultipleFeatures), value = TRUE)
train_id <- which(MultipleFeatures$training)
MultipleFeatures[train_id, vars] <- scale(MultipleFeatures[train_id, vars], scale = FALSE)
## Compute the MDYPL fits
kappa <- length(vars) / sum(MultipleFeatures$training)
```

```
full_fm <- formula(paste("I(digit == 7) ~", paste(vars, collapse = " + ")))
nest_vars <- grep("fou", vars, value = TRUE)
nest_fm <- formula(paste("I(digit == 7) ~", paste(nest_vars, collapse = " + ")))
full_m <- glm(full_fm, data = MultipleFeatures, family = binomial(),
              method = mdyplFit, alpha = 1 / (1 + kappa), subset = training)
nest_m <- update(full_m, nest_fm)

## With a naive penalized likelihood ratio test we get no evidence
## against the hypothesis that the model with only `fou` features
## is an as good descrition of `7` as the model with both `fou` and
## `kar` features.
plrtest(nest_m, full_m)

## With a high-dimensionality correction theres is strong evidence
## against the model with only `fou` features
plrtest(nest_m, full_m, hd_correction = TRUE)



## A simulated data set as in Rigon & Aliverti (2023, Section 4.3)

set.seed(123)
n <- 1000
p <- 500
gamma <- sqrt(5)
X <- matrix(rnorm(n * p, 0, 1), nrow = n, ncol = p)
betas0 <- rep(c(-1, -1/2, 0, 2, 3), each = p / 5)
betas <- gamma * betas0 / sqrt(sum(betas0^2))
probs <- plogis(drop(X %*% betas))
y <- rbinom(n, 1, probs)
fit_mdypl <- glm(y ~ -1 + X, family = binomial(), method = "mdyplFit")

## The default value of `alpha` is `n / (n + p)` here
identical(n / (n + p), fit_mdypl$alpha)

## Aggregate bias of MDYPL and rescaled MDYPL estimators
ag_bias <- function(estimates, beta) mean(estimates - beta)
ag_bias(coef(summary(fit_mdypl))[, "Estimate"], betas)
ag_bias(coef(summary(fit_mdypl, hd_correction = TRUE))[, "Estimate"], betas)
```

`mis` *A `"link-glm"` object for misclassified responses in binomial regression models*

Description

`mis()` is a `"link-glm"` object that specifies the link function in Neuhaus (1999, expression (8)) for handling misclassified responses in binomial regression models using maximum likelihood. A prior specification of the sensitivity and specificity is required.

Usage

```
mis(link = "logit", sensitivity = 1, specificity = 1)
```

Arguments

`link`	the baseline link to be used.
`sensitivity`	the probability of observing a success given that a success actually took place given any covariate values.
`specificity`	the probability of observing a failure given that a failure actually took place given any covariate values.

Details

`sensitivity` + `specificity` should be greater or equal to 1, otherwise it is implied that the procedure producing the responses performs worse than chance in terms of misclassification.

References

Neuhaus J M (1999). Bias and efficiency loss due to misclassified responses in binary regression. Biometrika, **86**, 843-855. https://www.jstor.org/stable/2673589.

See Also

`glm()`, `brglm_fit()`

Examples

```
## Define a few links with some misclassification
logit_mis <- mis(link = "logit", sensitivity = 0.9, specificity = 0.9)

lizards_f <- cbind(grahami, opalinus) ~ height + diameter + light + time

lizardsML <- glm(lizards_f, family = binomial(logit), data = lizards)

lizardsML_mis <- update(lizardsML, family = binomial(logit_mis),
                        start = coef(lizardsML))

## A notable change is coefficients is noted here compared to when
## specificity and sensitity are 1
coef(lizardsML)
coef(lizardsML_mis)

## Bias reduction is also possible
update(lizardsML_mis, method = "brglmFit", type = "AS_mean",
       start = coef(lizardsML))

update(lizardsML_mis, method = "brglmFit", type = "AS_median",
       start = coef(lizardsML))
```

MultipleFeatures *Multiple features data*

Description

Digits (0-9) extracted from a collection of maps from a Dutch public utility. Two hundred `30 × 48` binary images per digit are available, which have then been used to extract feature sets; see Jain et al. (2000), for details, where that dataset is used for assessing the performance of various classifiers for digit recognition.

Usage

```
MultipleFeatures
```

Format

A data frame with 2000 rows and 382 columns:

- `digit`. The digits to which the feature sets `fou.*`, `kar.*` and `pix.*` correspond to.
- `fou.*`. 76 Fourier coefficients of the character shapes, which are computed to be rotation invariant.
- `kar.*`. 64 Karhunen-Lo\'eve coefficients of the character shapes.
- `pix.*`. 240 pixel averages in 2 x 3 windows of each character shape.
- `training`. `TRUE` if the digit is part of the training set and `FALSE` if the digit is allocated to the test set.

Source

The data provides the `fou`, `kar` and `pix` features of the Multiple Features data set from the UCI Machine Learning Repository (Duin, 1998).

References

Duin, R. (1998). Multiple Features Dataset. UCI Machine Learning Repository. doi:10.24432/C5HC70.

Jain A, Duin R, Mao J (2000). Statistical pattern recognition: a review. IEEE Transactions on Pattern Analysis and Machine Intelligence, **22**, 4–37. doi:10.1109/34.824819.

See Also

`mdypl_fit()`

Examples

```
data("MultipleFeatures", package = "brglm2")

par(mfrow = c(10, 20), mar = numeric(4) + 0.1)
for (c_digit in 0:9) {
   df <- subset(MultipleFeatures, digit == c_digit)
   df <- as.matrix(df[, paste("pix", 1:240, sep = ".")])
   for (inst in 1:20) {
       m <- matrix(df[inst, ], 15, 16)[, 16:1]
       image(m, col = grey.colors(7, 1, 0), xaxt = "n", yaxt = "n")
  }
}
```

---

ordinal_superiority.bracl

Ordinal superiority scores of Agresti and Kateri (2017)

Description

`ordinal_superiority()` is a method for the estimation and inference about model-based ordinal superiority scores introduced in Agresti and Kateri (2017, Section 5) from fitted objects. The mean bias of the estimates of the ordinal superiority scores can be corrected.

Usage

```
## S3 method for class 'bracl'
ordinal_superiority(
  object,
  formula,
  data,
  measure = c("gamma", "Delta"),
  level = 0.95,
  bc = FALSE
)
```

Arguments

`object`	a fitted object from an ordinal regression model. Currently only models from class `"bracl"` are supported.
`formula`	a RHS formula indicating the group variable to use.
`data`	an optional data frame in which to look for variables with which to compute ordinal superiority measures. If omitted, an attempt is made to use the data that produced `object`.

measure	either "gamma" (default) or "Delta", specifying the ordinal superiority measure to be returned.
level	the confidence level required when computing confidence intervals for the ordinal superiority measures.
bc	logical. If FALSE (default) then the ordinal superiority measures are computed using the estimates in object. If TRUE then the ordinal superiority measure estimates are corrected for mean bias.

References

Agresti, A., Kateri, M. (2017). Ordinal probability effect measures for group comparisons in multinomial cumulative link models. *Biometrics*, **73** 214-219. doi:10.1111/biom.12565.

Examples

```
data("stemcell", package = "brglm2")

# Adjacent category logit (proportional odds)
stem <- within(stemcell, {nreligion = as.numeric(religion)})
fit_bracl_p <- bracl(research ~ nreligion + gender, weights = frequency,
                     data = stem, type = "ML", parallel = TRUE)

# Estimates and 95% confidence intervals for the probabilities that the response
# category for gender "female" is higher than the response category for gender "male",
# while adjusting for religion.
ordinal_superiority(fit_bracl_p, ~ gender)


# And their (very-similar in value here) bias corrected versions
# with 99% CIs
ordinal_superiority(fit_bracl_p, ~ gender, bc = TRUE, level = 0.99)
# Note that the object is refitted with type = "AS_mean"
```

---

plrtest.mdyplFit — *Penalized likelihood ratio test for "mdyplFit" objects*

Description

Computes the Diaconis-Ylvisaker prior penalized likelihood ratio test statistic or its adjusted version using high-dimensionality correction under proportional asymptotics. Associated p-values are also computed using a chi squared distribution.

Usage

```
## S3 method for class 'mdyplFit'
plrtest(object1, object2, hd_correction = FALSE, ...)
```

Arguments

object1	a "mdyplFit" object
object2	a "mdyplFit" object
hd_correction	if FALSE (default), then the corresponding quantities are computed according to standard asymptotics. If TRUE then the high-dimensionality corrections in Sterzinger & Kosmidis (2024) are employed to updates estimates, estimated standard errors, z-statistics, etc. See Details.
...	further arguments to be passed to summary.mdyplFit().

Details

Both object1 and object2 should have been fitted using the mdyplFit() method for glm(), and the same shrinkage parameter alpha; see mdyplFit() and mdyplControl() for setting alpha.

If hd_correction = TRUE then the deviance and the associated p-value are adjusted using a high-dimensionality correction under proportional asymptotics as in Sterzinger & Kosmidis (2024); see summary.mdyplFit().

Author(s)

Ioannis Kosmidis [aut, cre] <ioannis.kosmidis@warwick.ac.uk>

References

Sterzinger P, Kosmidis I (2024). Diaconis-Ylvisaker prior penalized likelihood for $p/n \to \kappa \in (0, 1)$ logistic regression. *arXiv*:2311.07419v2, https://arxiv.org/abs/2311.07419.

See Also

mdyplFit(), summary.mdyplFit(), mdypl_control()

---

predict.bracl *Predict method for bracl fits*

Description

Obtain class and probability predictions from a fitted adjacent category logits model.

Usage

```
## S3 method for class 'bracl'
predict(object, newdata, type = c("class", "probs"), ...)
```

Arguments

- `object` a fitted object of class inheriting from "bracl".
- `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 "class", which produces predictions of the response category at the covariate values supplied in "newdata", selecting the category with the largest probability; the alternative "probs" returns all category probabilities at the covariate values supplied in newdata.
- `...` further arguments passed to or from other methods.

Details

If newdata is omitted the predictions are based on the data used for the fit.

Value

If type = "class" a vector with the predicted response categories; if type = "probs" a matrix of probabilities for all response categories at newdata.

Examples

```
data("stemcell", package = "brglm2")

# Adjacent category logit (non-proportional odds)
fit_bracl <- bracl(research ~ as.numeric(religion) + gender, weights = frequency,
                   data = stemcell, type = "ML")
# Adjacent category logit (proportional odds)
fit_bracl_p <- bracl(research ~ as.numeric(religion) + gender, weights = frequency,
                    data = stemcell, type = "ML", parallel = TRUE)

# New data
newdata <- expand.grid(gender = c("male", "female"),
                       religion = c("liberal", "moderate", "fundamentalist"))

# Predictions
sapply(c("class", "probs"), function(what) predict(fit_bracl, newdata, what))
sapply(c("class", "probs"), function(what) predict(fit_bracl_p, newdata, what))
```

---

predict.brmultinom *Predict method for brmultinom fits*

Description

Obtain class and probability predictions from a fitted baseline category logits model.

Usage

```
## S3 method for class 'brmultinom'
predict(object, newdata, type = c("class", "probs"), ...)
```

Arguments

`object`	a fitted object of class inheriting from `"brmultinom"`.
`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 `"class"`, which produces predictions of the response category at the covariate values supplied in `"newdata"`, selecting the category with the largest probability; the alternative `"probs"` returns all category probabilities at the covariate values supplied in `newdata`.
`...`	further arguments passed to or from other methods.

Details

If `newdata` is omitted the predictions are based on the data used for the fit.

Value

If `type = "class"` a vector with the predicted response categories; if `type = "probs"` a matrix of probabilities for all response categories at `newdata`.

Examples

```
data("housing", package = "MASS")

# Maximum likelihood using brmultinom with baseline category 'Low'
houseML1 <- brmultinom(Sat ~ Infl + Type + Cont, weights = Freq,
                       data = housing, type = "ML", ref = 1)

# New data
newdata <- expand.grid(Infl = c("Low", "Medium"),
                       Type = c("Tower", "Atrium", "Terrace"),
                       Cont = c("Low", NA, "High"))

## Predictions
sapply(c("class", "probs"), function(what) predict(houseML1, newdata, what))
```

`residuals.brmultinom` *Residuals for multinomial logistic regression and adjacent category logit models*

Description

Residuals for multinomial logistic regression and adjacent category logit models

Usage

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

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

Arguments

`object`	the object coming out of `bracl()` and `brmultinom()`.
`type`	the type of residuals which should be returned. The options are: `"pearson"` (default), `"response"`, `"deviance"`, `"working"`. See Details.
`...`	Currently not used.

Details

The residuals computed are the residuals from the equivalent Poisson log-linear model fit, organized in a form that matches the output of `fitted(object, type = "probs")`. As a result, the output is residuals defined in terms of the object and expected multinomial counts.

See Also

brmultinom bracl

---

se0 *MDYPL state evolution functions with no intercept*

Description

MDYPL state evolution functions with no intercept

Usage

```
se0(mu, b, sigma, kappa, gamma, alpha, gh = NULL, prox_tol = 1e-10)
```

Arguments

`mu`	aggregate bias parameter.
`b`	parameter `b` in the state evolution functions.
`sigma`	square root of the aggregate variance of the MDYPL estimator.
`kappa`	asymptotic ratio of columns/rows of the design matrix. `kappa` should be in `(0, 1)`.
`gamma`	the square root of the limit of the variance of the linear predictor.
`alpha`	the shrinkage parameter of the MDYPL estimator. `alpha` should be in `(0, 1)`.

gh	A list with the Gauss-Hermite quadrature nodes and weights, as returned from `statmod::gauss.quad()` with `kind = "hermite"`. Default is `NULL`, in which case `gh` is set to `statmod::gauss.quad(200, kind = "hermite")` is used.
prox_tol	tolerance for the computation of the proximal operator; default is `1e-10`.

`se0_ridge` *Logistic ridge regression state evolution functions with no intercept*

Description

Logistic ridge regression state evolution functions with no intercept

Usage

```
se0_ridge(mu, b, sigma, kappa, gamma, lambda, gh = NULL, prox_tol = 1e-10)
```

Arguments

mu	aggregate bias parameter.
b	parameter `b` in the state evolution functions.
sigma	square root of the aggregate variance of the MDYPL estimator.
kappa	asymptotic ratio of columns/rows of the design matrix. `kappa` should be in `(0, 1)`.
gamma	the square root of the limit of the variance of the linear predictor.
lambda	the shrinkage parameter of the logistic regression penalty estimator. `lambda` should be in greater than zero.
gh	A list with the Gauss-Hermite quadrature nodes and weights, as returned from `statmod::gauss.quad()` with `kind = "hermite"`. Default is `NULL`, in which case `gh` is set to `statmod::gauss.quad(200, kind = "hermite")` is used.
prox_tol	tolerance for the computation of the proximal operator; default is `1e-10`.

Details

It is assumed that the ridge penalty to the logistic regression log-likelihood is `n * lambda * sum(beta^2) / (2 * length(beta))`, where `n` is the sum of the binomial totals.

se1 *MDYPL state evolution functions with intercept*

Description

MDYPL state evolution functions with intercept

Usage

```
se1(
  mu,
  b,
  sigma,
  iota,
  kappa,
  gamma,
  alpha,
  intercept,
  gh = NULL,
  prox_tol = 1e-10
)
```

Arguments

mu	aggregate bias parameter.
b	parameter b in the state evolution functions.
sigma	square root of the aggregate variance of the MDYPL estimator.
iota	limits of the MDYPL estimate for the intercept as the sample size goes to +Inf
kappa	asymptotic ratio of columns/rows of the design matrix. kappa should be in (0, 1).
gamma	the square root of the limit of the variance of the linear predictor.
alpha	the shrinkage parameter of the MDYPL estimator. alpha should be in (0, 1).
intercept	intercept of the logistic regression model
gh	A list with the Gauss-Hermite quadrature nodes and weights, as returned from statmod::gauss.quad() with kind = "hermite". Default is NULL, in which case gh is set to statmod::gauss.quad(200, kind = "hermite") is used.
prox_tol	tolerance for the computation of the proximal operator; default is 1e-10. fixed point problem solved via Newton-Raphson

simulate.brmultinom *Method for simulating a data set from "brmultinom" and "bracl" objects*

Description

Method for simulating a data set from "brmultinom" and "bracl" objects

Usage

```
## S3 method for class 'brmultinom'
simulate(object, ...)
```

Arguments

object	an object of class "brmultinom" or "bracl".
...	currently not used.

Value

A "data.frame" with object$ncat times the rows that model.frame(object) have and the same variables. If weights has been specified in the call that generated object, then the simulate frequencies will populate the weights variable. Otherwise, the resulting data.frame will have a ".weights" variable with the simulated multinomial counts.

Examples

```
## Multinomial logistic regression
data("housing", package = "MASS")
houseML1 <- brmultinom(Sat ~ Infl + Type + Cont, weights = Freq,
                       data = housing, type = "ML", ref = 1)
simulate(houseML1)

## Adjacent-category logits
data("stemcell", package = "brglm2")
stemML1 <- bracl(research ~ religion + gender, weights = frequency,
                 data = stemcell, type = "ML")

simulate(stemML1)
```

`simulate.brnb` *Simulate Responses*

Description

Simulate one or more responses from the distribution corresponding to a fitted model "brnb" object.

Usage

```
## S3 method for class 'brnb'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

object	an object representing a fitted model.
nsim	number of response vectors to simulate. Defaults to 1.
seed	an object specifying if and how the random number generator should be initialized; see set.seed() for details.
...	extra arguments to be passed to methods. Not currently used.

Examples

```
# Example in Saha, K., & Paul, S. (2005). Bias-corrected maximum
# likelihood estimator of the negative binomial dispersion
# parameter.  Biometrics, 61, 179--185.
#
# Frequency distribution of red mites on apple leaves.
nomites <- 0:8
noleaves <- c(70, 38, 17, 10, 9, 3, 2, 1, 0)
fit_glmnb <- MASS::glm.nb(nomites~1,link="identity",weights = noleaves)
fit_brnb <- brnb(nomites ~ 1, link = "identity", transformation = "inverse",
                 type = "ML", weights = noleaves)
## Let us simulate 10 response vectors
sim_glmnb <- simulate(fit_glmnb, nsim = 10, seed = 123)
sim_brnb <-  simulate(fit_brnb, nsim = 10, seed = 123)
# The results from glm.nb and brnb with type = "ML" are
# exactly the same
all.equal(sim_glmnb, sim_brnb, check.attributes = FALSE)
```

sloe — *Estimate the corrupted signal strength in a model with (sub-)Gaussian covariates*

Description

Estimate the corrupted signal strength in a model with (sub-)Gaussian covariates

Usage

```
sloe(object)
```

Arguments

`object`	an `"mdyplFit"` object.

Details

The Signal Strength Leave-One-Out Estimator (SLOE) is defined in Yadlowsky et al. (2021) when the model is estimated using maximum likelihood (i.e. when `object$alpha = 1`; see `mdyplControl()` for what `alpha` is). The SLOE adaptation when estimation is through maximum Diaconis-Ylvisaker prior penalized likelihood (`mdypl_fit()`) has been put forward in Sterzinger & Kosmidis (2025).

In particular, `sloe()` computes an estimate of the corrupted signal strength which is the limit

$$\nu^2$$

of $var(X\hat{\beta}(\alpha))$, where $\hat{\beta}(\alpha)$ is the maximum Diaconis-Ylvisaker prior penalized likelihood (MDYPL) estimator as computed by `mdyplFit()` with shrinkage parameter $alpha$.

Value

A scalar.

Author(s)

Ioannis Kosmidis `[aut, cre]` `<ioannis.kosmidis@warwick.ac.uk>`

References

Sterzinger P, Kosmidis I (2024). Diaconis-Ylvisaker prior penalized likelihood for $p/n \to \kappa \in (0, 1)$ logistic regression. *arXiv*:2311.07419v2, https://arxiv.org/abs/2311.07419.

Yadlowsky S, Yun T, McLean C Y, D' Amour A (2021). SLOE: A Faster Method for Statistical Inference in High-Dimensional Logistic Regression. In M Ranzato, A Beygelzimer, Y Dauphin, P Liang, J W Vaughan (eds.), *Advances in Neural Information Processing Systems*, **34**, 29517–29528. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2021/file/f6c2a0c4b566bc99d596e58638e342b0-Paper.pdf.

See Also

summary.mdyplFit()

---

solve_se *Solve the MDYPL state evolution equations with or without intercept, with signal strength or contaminated signal strength*

Description

Solve the MDYPL state evolution equations with or without intercept, with signal strength or contaminated signal strength

Usage

```
solve_se(
  kappa,
  ss,
  alpha,
  intercept = NULL,
  start,
  corrupted = FALSE,
  gh = NULL,
  prox_tol = 1e-10,
  transform = TRUE,
  init_method = "Nelder-Mead",
  init_iter = 50,
  ...
)
```

Arguments

kappa	asymptotic ratio of columns/rows of the design matrix. kappa should be in (0, 1).
ss	square root of signal strength or of corrupted signal strength, depending on whether corrupted = TRUE or not. See Details.
alpha	the shrinkage parameter of the MDYPL estimator. alpha should be in (0, 1).
intercept	if NULL (default) then the MDYPL state evolution equations for the model with no intercept parameter are solved. If a real then the equations for the models with intercept parameter equal to intercept are solved. See Details.
start	a vector with starting values for mu, b,sigma (and iota if intercept is numeric).
corrupted	if FALSE (default) then ss is the square root of the signal strength and intercept, if numeric, is the oracle intercept value. If TRUE, then ss is the square root of the corrupted signal strength, and intercept, if numeric, is the limit of the estimator computed by mdyplFit() with shrinkage parameter alpha. See Details.

`gh`	A list with the Gauss-Hermite quadrature nodes and weights, as returned from `statmod::gauss.quad()` with `kind = "hermite"`. Default is `NULL`, in which case `gh` is set to `statmod::gauss.quad(200, kind = "hermite")`.
`prox_tol`	tolerance for the computation of the proximal operator; default is `1e-10`.
`transform`	if `TRUE` (default), the optimization is with respect to `log(mu)`, `log(b)`,`log(sigma)`, (and `iota` if `intercept` is numeric). If `FALSE`, then it is over `mu`, `b`, `sigma` (and `iota` if `intercept` is numeric). The solution is returned in terms of the latter set, regardless of how optimization took place.
`init_method`	The method to be passed to `optim()`. Default is `"Nelder-Mead"`.
`init_iter`	how many iterations of `optim()` should we make to get starting values for `nleqslv::nleqslv()`? Default is `50`, but can also be `0` in which case `start` is directly passed to `nleqslv:nleqslv()`. `init_iter = "only"` results in only `optim()` being used. See Details.
`...`	further arguments to be passed to `nleqslv::nleqslv()`, unless `init_iter = "only"`, in which case `...` is further arguments to be passed to `optim()`.

Details

`init_iter` iterations of `optim()` with `method = init_method` are used towards minimizing `sum(se)^2`, where `se` is a vector of the state evolution functions. The solution is then passed to `nleqslv::nleqslv()` for a more aggressive iteration. The state evolution equations are given in expressions (8) (model without intercept) and expression (15) (model with intercept) in Sterzinger & Kosmidis (2024).

If `corrupted = FALSE` (default), then `ss` is the square root of the signal strength, which is the limit γ^2 of $var(X\beta)$. If `corrupted = TRUE`, then `ss` is the square root of the corrupted signal strength which is the limit ν^2 of $var(X hat(beta)(\alpha))$, where $hat(\beta)(\alpha)$ is the maximum Diaconis-Ylvisaker prior penalized likelihood (MDYPL) estimator as computed by `mdyplFit()` with shrinkage parameter $alpha$.

If `intercept = NULL`, then the state evolution equations are solved for the model without intercept. If `intercept` is a real number, then the state evolution equations for the model with intercept are solved (i.e. with predictor $\eta_i = \theta + x_i^T\beta$). In that case, what `intercept` represents depends on the value of `corrupted`. If `corrupted = FALSE`, `intercept` represents the oracle value of θ, otherwise it represents the limit `iota` of the MDYPL estimator of θ as computed by `mdyplFit()` with shrinkage parameter `alpha`.

Note that `start` is always for `mu`, `b`,`sigma`, as is the result, regardless whether `transform = TRUE` or not. Transformations during optimization are done internally.

Value

If `intercept = NULL`, a vector with the values of `mu`, `b`,`sigma`. Otherwise, a vector with the values of `mu`, `b`,`sigma`, and `iota`, if `corrupted = FALSE`, or the value of the intercept otherwise. The vector has attributes the state evolution functions at the solution (`"funcs"`), the number of iterations used by the last optimization method (`"iter"`), any messages from the last optimization method (`"message"`), and information on the optimization methods used (`"optimization-chain"`).

Author(s)

Ioannis Kosmidis `[aut, cre]` `<ioannis.kosmidis@warwick.ac.uk>`, Federico Boiocchi `[ctb]` `<federico.boiocchi@gmail.com>`, Philipp Sterzinger `[ctb, earlier Julia code by]` `<P.Sterzinger@lse.ac.uk>`

References

Zhao Q, Sur P, Cand\‘es E J (2022). The asymptotic distribution of the MLE in high-dimensional logistic models: Arbitrary covariance. *Bernoulli*, **28**, 1835–1861. doi:10.3150/21BEJ1401.

Sterzinger P, Kosmidis I (2024). Diaconis-Ylvisaker prior penalized likelihood for $p/n \to \kappa \in (0, 1)$ logistic regression. *arXiv*:2311.07419v2, https://arxiv.org/abs/2311.07419.

Examples

```
## Reproducing Table 13 of Zhao et al. (2022, DOI: 10.3150/21-BEJ1401)


thetas <- c(0, 0.5, 1, 2, 2.5)
gamma0 <- 5
pars3 <- matrix(NA, length(thetas), 3)
pars4 <- matrix(NA, length(thetas), 4)
colnames(pars4) <- c("I_mu", "I_b", "I_sigma", "I_iota")
colnames(pars3) <- c("II_mu", "II_b", "II_sigma")
for (i in seq_along(thetas)) {
    start3 <- c(0.5, 1, 1)
    pars3[i, ] <- solve_se(kappa = 0.2, ss = sqrt(5 + thetas[i]^2),
                           alpha = 1, start = start3, init_iter = 0)
    start4 <- c(pars3[i, ], thetas[i])
    pars4[i, ] <- solve_se(kappa = 0.2, ss = sqrt(5), intercept = thetas[i],
                           alpha = 1, start = start4, init_iter = 0)
}

cbind(pars3, pars4)
```

---

stemcell *Opinion on stem cell research and religious fundamentalism*

Description

A data set from the 2006 General Social Survey that shows the relationship in the United States between opinion about funding stem cell research and the fundamentalism/liberalism of one’s religious beliefs, stratified by gender.

Usage

```
stemcell
```

Format

A data frame with 24 rows and 4 variables:

- `research`: opinion about funding stem cell research with levels `definitely`, `probably`, `probably not`, `definitely not`

- `gender`: the gender of the respondent with levels `female` and `male`
- `religion`: the fundamentalism/liberalism of one's religious beliefs with levels `fundamentalist`, `moderate`, `liberal` `frequency`: the number of times a respondent fell in each of the combinations of levels for `research`, `religion` and `gender`

Source

The `stemcell` data set is analyzed in Agresti (2010, Subsection 4.1.5).

References

Agresti A (2010). *Analysis of Ordinal Categorical Data* (2nd edition). Wiley Series in Probability and Statistics. Wiley.

See Also

`bracl()`

---

summary.brglmFit — `summary()` *method for* `"brglmFit"` *objects*

Description

`summary()` method for `"brglmFit"` objects

Usage

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

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

Arguments

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

Details

The interface of the summary method for `"brglmFit"` objects is identical to that of `"glm"` objects. The summary method for `"brglmFit"` objects computes the p-values of the individual Wald statistics based on the standard normal distribution, unless the family is Gaussian, in which case a t distribution with appropriate degrees of freedom is used.

See Also

`summary.glm()` and `glm()`

Examples

```
## For examples see `examples(brglmFit)`
```

---

summary.brnb — `summary()` *method for* `"brnb"` *objects*

Description

`summary()` method for `"brnb"` objects

Usage

```
## S3 method for class 'brnb'
summary(object, ...)

## S3 method for class 'summary.brnb'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`object`	an object of class `"brnb"`, typically, a result of a call to `brnb()`.
`...`	further arguments passed to or from other methods.
`x`	an object of class `"summary.brnb"`, usually, a result of a call to summary.brnb.
`digits`	the number of significant digits to use when printing.

Details

The interface of the summary method for `"brnb"` objects is similar to that of `"brglmFit"` objects with additional information.

```
p-values of the individual Wald statistics are based on the
standard normal distribution.
```

See Also

`summary.brglmFit()` and `glm()`

Examples

```
# For examples see examples(brnb)
```

---

summary.mdyplFit *Summary method for "mdyplFit" objects*

Description

Summary method for `"mdyplFit"` objects

Usage

```
## S3 method for class 'mdyplFit'
summary(object, hd_correction = FALSE, solve_se_control = list(), ...)

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

Arguments

`object`	an object of class "glm", usually, a result of a call to `glm`.
`hd_correction`	if `FALSE` (default), then the corresponding quantities are computed according to standard asymptotics. If `TRUE` then the high-dimensionality corrections in Sterzinger & Kosmidis (2024) are employed to updates estimates, estimated standard errors, z-statistics, etc. See Details.
`solve_se_control`	a list of further arguments to be passed to `solve_se()`. Even if explicitly specified, the arguments `kappa`, `ss`, `alpha`, and `intercept` are always set according to `object`, and `corrupted` is set to `TRUE`.
`...`	further arguments to be passed to `summary.glm()`.
`x`	an object of class "summary.glm", usually, a result of a call to `summary.glm`.
`digits`	the number of significant digits to use when printing.
`symbolic.cor`	logical. If `TRUE`, print the correlations in a symbolic form (see `symnum`) rather than as numbers.
`signif.stars`	logical. If `TRUE`, 'significance stars' are printed for each coefficient.

Details

If `hd_correction = TRUE`, the `sloe()` estimator of the square root of the corrupted signal strength is estimated from `object`, as are the conditional variances of each covariate given the others (excluding the intercept). The latter are estimated using residual sums of squares from the linear regression of each covariate on all the others, as proposed in Zhao et al (2021, Section 5.1). Then the appropriate state evolution equations are solved using `solve_se()` with `corrupted = TRUE`, and the obtained constants are used to rescale the estimates, and adjust estimated standard errors and z-statistics as in Sterzinger & Kosmidis (2024).

The key assumptions under which the rescaled estimates and corrected standard errors and z-statistics are asymptotically valid are that the covariates have sub-Gaussian distributions, and that the signal strength, which is the limit γ^2 of $var(X\beta)$ is finite as $p/n \to \kappa \in (0,1)$, with $\kappa \in (0,1)$. See Sterzinger & Kosmidis (2024).

If `hd_correction = TRUE`, and the model has an intercept, then the result provides only a corrected estimate of the intercept with no accompanying standard error, z-statistic, and p-value. Also, `vcov(summary(object, hd_correction = TRUE))` is always `NULL`. Populating those objects with appropriate estimates is the subject of current work.

Value

A list with objects as in the result of `stats::summary.glm()`, with extra component `se_parameters`, which is the vector of the solution to the state evolution equations with extra attributes (see `solve_se()`).

Author(s)

Ioannis Kosmidis `[aut, cre]` `<ioannis.kosmidis@warwick.ac.uk>`

References

Zhao Q, Sur P, Cand\`es E J (2022). The asymptotic distribution of the MLE in high-dimensional logistic models: Arbitrary covariance. *Bernoulli*, **28**, 1835–1861. doi:10.3150/21BEJ1401.

Sterzinger P, Kosmidis I (2024). Diaconis-Ylvisaker prior penalized likelihood for $p/n \to \kappa \in (0, 1)$ logistic regression. *arXiv*:2311.07419v2, `https://arxiv.org/abs/2311.07419`.

See Also

`mdyplFit()`, `solve_se()`

Examples

```
set.seed(123)
n <- 2000
p <- 400
set.seed(123)
betas <- c(rnorm(p / 2, mean = 7, sd = 1), rep(0, p / 2))
X <- matrix(rnorm(n * p, 0, 1/sqrt(n)), nrow = n, ncol = p)
probs <- plogis(drop(X %*% betas))
y <- rbinom(n, 1, probs)
fit_mdypl <- glm(y ~ -1 + X, family = binomial(), method = "mdyplFit")

st_summary <- summary(fit_mdypl)
hd_summary <- summary(fit_mdypl, hd_correction = TRUE)

cols <- hcl.colors(3, alpha = 0.2)
par(mfrow = c(1, 2))
plot(betas, type = "l", ylim = c(-3, 14),
     main = "MDYPL estimates",
     xlab = "Parameter index", ylab = NA)
points(coef(st_summary)[, "Estimate"], col = NA, bg = cols[1], pch = 21)

plot(betas, type = "l", ylim = c(-3, 14),
     main = "rescaled MDYPL estimates",
     xlab = "Parameter index", ylab = NA)
points(coef(hd_summary)[, "Estimate"], col = NA, bg = cols[2], pch = 21)

## z-statistics
z_mdypl <- coef(st_summary)[betas == 0, "z value"]
qqnorm(z_mdypl, col = NA, bg = cols[1], pch = 21, main = "z value")
abline(0, 1, lty = 2)
z_c_mdypl <- coef(hd_summary)[betas == 0, "z value"]
qqnorm(z_c_mdypl, col = NA, bg = cols[2], pch = 21, main = "corrected z value")
abline(0, 1, lty = 2)
```

vcov.brglmFit *Return the variance-covariance matrix for the regression parameters in a* brglmFit() *object*

Description

Return the variance-covariance matrix for the regression parameters in a brglmFit() object

Usage

```
## S3 method for class 'brglmFit'
vcov(object, model = c("mean", "full", "dispersion"), complete = TRUE, ...)
```

Arguments

object	a fitted model object, typically. Sometimes also a summary() object of such a fitted model.
model	character specifying for which component of the model coefficients should be extracted.
complete	for the aov, lm, glm, mlm, and where applicable summary.lm etc methods: logical indicating if the full variance-covariance matrix should be returned also in case of an over-determined system where some coefficients are undefined and coef(.) contains NAs correspondingly. When complete = TRUE, vcov() is compatible with coef() also in this singular case.
...	additional arguments for method functions. For the glm method this can be used to pass a dispersion parameter.

Details

The options for model are "mean" for mean regression parameters only (default), "dispersion" for the dispersion parameter (or the transformed dispersion; see brglm_control()), and "full" for both the mean regression and the (transformed) dispersion parameters.

vcov.brnb *Extract model variance-covariance matrix from* "brnb" *objects*

Description

Extract model variance-covariance matrix from "brnb" objects

Usage

```
## S3 method for class 'brnb'
vcov(object, model = c("mean", "full", "dispersion"), complete = TRUE, ...)
```

Arguments

object	an object of class "brnb", typically, a result of a call to brnb().
model	character specifying for which component of the model variance-covariance matrix should be extracted.
complete	for the aov, lm, glm, mlm, and where applicable summary.lm etc methods: logical indicating if the full variance-covariance matrix should be returned also in case of an over-determined system where some coefficients are undefined and coef(.) contains NAs correspondingly. When complete = TRUE, vcov() is compatible with coef() also in this singular case.
...	additional arguments for method functions. For the glm method this can be used to pass a dispersion parameter.

Details

The options for model are "mean" for mean regression only (default), "dispersion" for the dispersion parameter (in a chosen transformation; see brglmControl(), and "full" for both the mean regression and the (transformed) dispersion parameters. See vcov() for more details.

See Also

vcov()

Index