Package ‘brglm’

October 12, 2020

Type Package
Title Bias Reduction in Binomial-Response Generalized Linear Models
Version 0.7.1
URL https://github.com/ikosmidis/brglm
BugReports https://github.com/ikosmidis/brglm/issues
Description Fit generalized linear models with binomial responses using either an adjusted-score approach to bias reduction or maximum penalized likelihood where penalization is by Jeffreys invariant prior. These procedures return estimates with improved frequentist properties (bias, mean squared error) that are always finite even in cases where the maximum likelihood estimates are infinite (data separation). Fitting takes place by fitting generalized linear models on iteratively updated pseudo-data. The interface is essentially the same as ‘glm’. More flexibility is provided by the fact that custom pseudo-data representations can be specified and used for model fitting. Functions are provided for the construction of confidence intervals for the reduced-bias estimates.

License GPL (>= 2)
Depends R (>= 2.6.0), profileModel
Suggests MASS
NeedsCompilation yes
Author Ioannis Kosmidis [aut, cre] (<https://orcid.org/0000-0003-1556-0302>)
Maintainer Ioannis Kosmidis <ioannis.kosmidis@warwick.ac.uk>
Repository CRAN
Date/Publication 2020-10-12 04:40:09 UTC

R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>brglm</td>
<td>2</td>
</tr>
<tr>
<td>brglm.control</td>
<td>7</td>
</tr>
<tr>
<td>confint.brglm</td>
<td>8</td>
</tr>
<tr>
<td>gethats</td>
<td>11</td>
</tr>
<tr>
<td>glm.control1</td>
<td>12</td>
</tr>
<tr>
<td>lizards</td>
<td>13</td>
</tr>
</tbody>
</table>
Description

Fits binomial-response GLMs using the bias-reduction method developed in Firth (1993) for the removal of the leading \(O(n^{-1})\) term from the asymptotic expansion of the bias of the maximum likelihood estimator. Fitting is performed using pseudo-data representations, as described in Kosmidis (2007, Chapter 5). For estimation in binomial-response GLMs, the bias-reduction method is an improvement over traditional maximum likelihood because:

- the bias-reduced estimator is second-order unbiased and has smaller variance than the maximum likelihood estimator and
- the resulting estimates and their corresponding standard errors are always finite while the maximum likelihood estimates can be infinite (in situations where complete or quasi separation occurs).

Usage

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

```r
brglm.fit(x, y, weights = rep(1, nobs), start = NULL, etastart = NULL,
          mustart = NULL, offset = rep(0, nobs), family = binomial(),
          control = glm.control(), control.brglm = brglm.control(),
          intercept = TRUE, pl = FALSE)
```

Arguments

- `formula` as in `glm`.
- `family` as in `glm`. `brglm` currently supports only the "binomial" family with links "logit", "probit", "cloglog", "cauchit".
- `data` as in `glm`.
- `weights` as in `glm`.
- `subset` as in `glm`.
- `na.action` as in `glm`.
**brglm**

- **start**
  - as in `glm`.
- **etastart**
  - as in `glm`.
- **mustart**
  - as in `glm`.
- **offset**
  - as in `glm`.
- **control glm**
  - `control.glm` replaces the `control` argument in `glm` but essentially does the same job. It is a list of parameters to control `glm.fit`. See the documentation of `glm.control` for details.
- **control**
  - same as in `glm`. Only available to `brglm.fit`.
- **intercept**
  - as in `glm`.
- **model**
  - as in `glm`.
- **method**
  - the method to be used for fitting the model. The default method is "brglm.fit", which uses either the modified-scores approach to estimation or maximum penalized likelihood (see the pl argument below). The standard `glm` methods "glm.fit" for maximum likelihood and "model.frame" for returning the model frame without any fitting, are also accepted.
- **pl**
  - a logical value indicating whether the model should be fitted using maximum penalized likelihood, where the penalization is done using Jeffreys invariant prior, or using the bias-reducing modified scores. It is only used when `method = "brglm.fit"`. The default value is `FALSE` (see also the Details section).
- **x**
  - as in `glm`.
- **y**
  - as in `glm`.
- **contrasts**
  - as in `glm`.
- **control.brglm**
  - a list of parameters for controlling the fitting process when `method = "brglm.fit"`. See documentation of `brglm.control` for details.
- **...**
  - further arguments passed to or from other methods.

**Details**

`brglm.fit` is the workhorse function for fitting the model using either the bias-reduction method or maximum penalized likelihood. If `method = "glm.fit"`, usual maximum likelihood is used via `glm.fit`.

The main iteration of `brglm.fit` consists of the following steps:

1. Calculate the diagonal components of the hat matrix (see `gethats` and `hatvalues`).
2. Obtain the pseudo-data representation at the current value of the parameters (see `modifications` for more information).
3. Fit a local GLM, using `glm.fit` on the pseudo data.
4. Adjust the quadratic weights to agree with the original binomial totals.

Iteration is repeated until either the iteration limit has been reached or the sum of the absolute values of the modified scores is less than some specified positive constant (see the `br.maxit` and `br.epsilon` arguments in `brglm.control`).

The default value (FALSE) of pl, when `method = "brglm.fit"`, results in estimates that are free of any $O(n^{-1})$ terms in the asymptotic expansion of their bias. When `pl = TRUE` bias-reduction
is again achieved but generally not at such order of magnitude. In the case of logistic regression
the value of \( pl \) is irrelevant since maximum penalized likelihood and the modified-scores approach
coincide for natural exponential families (see Firth, 1993).

For other language related details see the details section in \texttt{glm}.

Value

\texttt{brglm} returns an object of class "brglm". A "brglm" object inherits first from "glm" and then from
"lm" and is a list containing the following components:

coefficients as in \texttt{glm}.
residuals as in \texttt{glm}.
fitted.values as in \texttt{glm}.
effects as in \texttt{glm}.
R as in \texttt{glm}.
rank as in \texttt{glm}.
qr as in \texttt{glm}.
family as in \texttt{glm}.
linear.predictors as in \texttt{glm}.
deviance as in \texttt{glm}.
aic as in \texttt{glm} (see Details).
null.deviance as in \texttt{glm}.
iter as in \texttt{glm}.
weights as in \texttt{glm}.
prior.weights as in \texttt{glm}.
df.residual as in \texttt{glm}.
df.null as in \texttt{glm}.
y as in \texttt{glm}.
converged as in \texttt{glm}.
boundary as in \texttt{glm}.
ModifiedScores the vector of the modified scores for the parameters at the final iteration. If \( pl = TRUE \) they are the derivatives of the penalized likelihood at the final iteration.
FisherInfo the Fisher information matrix evaluated at the resulting estimates. Only available when method = "brglm.fit".
hats the diagonal elements of the hat matrix. Only available when method = "brglm.fit".
nIter the number of iterations that were required until convergence. Only available when method = "brglm.fit".
cur.model a list with components ar and at which contains the values of the additive modifications to the responses (y) and to the binomial totals (prior.weights) at the resulting estimates (see \texttt{modifications} for more information). Only available when method = "brglm.fit".
model as in \texttt{glm}.
call as in \texttt{glm}.
formula as in \texttt{glm}.
terms as in \texttt{glm}.
data as in \texttt{glm}.
offset as in \texttt{glm}.
control.glm as control in the result of \texttt{glm}.
control.brglm the control.brglm argument that was passed to \texttt{brglm}. Only available when \texttt{method = "brglm.fit"}.
method the method used for fitting the model.
contrasts as in \texttt{glm}.
xlevels as in \texttt{glm}.
pl logical having the same value with the \texttt{pl} argument passed to \texttt{brglm}. Only available when \texttt{method = "brglm.fit"}.

Warnings

1. It is not advised to use methods associated with model comparison (\texttt{add1}, \texttt{drop1}, \texttt{anova}, etc.) on objects of class \texttt{"brglm"}. Model comparison when estimation is performed using the modified scores or the penalized likelihood is an on-going research topic and will be implemented as soon as it is concluded.

2. The use of Akaike’s information criterion (AIC) for model selection when \texttt{method = "brglm.fit"} is asymptotically valid, because the log-likelihood derivatives dominate the modification (in terms of asymptotic order).

Note

1. Supported methods for objects of class \texttt{"brglm"} are:
   - \texttt{print} through \texttt{print.brglm}.
   - \texttt{summary} through \texttt{summary.brglm}.
   - \texttt{coefficients} inherited from the \texttt{"glm"} class.
   - \texttt{vcov} inherited from the \texttt{"glm"} class.
   - \texttt{predict} inherited from the \texttt{"glm"} class.
   - \texttt{residuals} inherited from the \texttt{"glm"} class.
   - and other methods that apply to objects of class \texttt{"glm"}

2. A similar implementation of the bias-reduction method could be done for every GLM, following Kosmidis (2007) (see also Kosmidis and Firth, 2009). The full set of families and links will be available in a future version. However, bias-reduction is not generally beneficial as it is in the binomial family and it could cause inflation of the variance (see Firth, 1993).

3. Basically, the differences between maximum likelihood, maximum penalized likelihood and the modified scores approach are more apparent in small sample sizes, in sparse data sets and in cases
where complete or quasi-complete separation occurs. Asymptotically (as \( n \) goes to infinity), the three different approaches are equivalent to first order.

4. When an offset is not present in the model, the modified-scores based estimates are usually smaller in magnitude than the corresponding maximum likelihood estimates, shrinking towards the origin of the scale imposed by the link function. Thus, the corresponding estimated asymptotic standard errors are also smaller.

The same is true for the maximum penalized likelihood estimates when for example, the logit (where the maximum penalized likelihood and modified-scores approaches coincide) or the probit links are used. However, generally the maximum penalized likelihood estimates do not shrink towards the origin. In terms of mean-value parameterization, in the case of maximum penalized likelihood the fitted probabilities would shrink towards the point where the Jeffreys prior is maximized or equivalently where the quadratic weights are simultaneously maximized (see Kosmidis, 2007).

5. Implementations of the bias-reduction method for logistic regressions can also be found in the \texttt{logistf} package. In addition to the obvious advantage of \texttt{brglm} in the range of link functions that can be used ("logit", "probit", "cloglog" and "cauchit"), \texttt{brglm} is also more efficient computationally. Furthermore, for any user-specified link function (see the Example section of \texttt{family}), the user can specify the corresponding pseudo-data representation to be used within \texttt{brglm} (see \texttt{modifications} for details).

\section*{Author(s)}
Ioannis Kosmidis, <ioannis.kosmidis@warwick.ac.uk>

\section*{References}


\section*{See Also}
\texttt{glm}, \texttt{glm.fit}
Examples
## Begin Example
data(lizards)
# Fit the GLM using maximum likelihood
lizards.glm <- brglm(cbind(grahami, opalinus) ~ height + diameter +
                     light + time, family = binomial(logit), data=lizards,
                     method = "glm.fit")
# Now the bias-reduced fit:
lizards.brglm <- brglm(cbind(grahami, opalinus) ~ height + diameter +
                       light + time, family = binomial(logit), data=lizards,
                       method = "brglm.fit")
lizards.glm
lizards.brglm
# Other links
update(lizards.brglm, family = binomial(probit))
update(lizards.brglm, family = binomial(cloglog))
update(lizards.brglm, family = binomial(cauchit))
# Using penalized maximum likelihood
update(lizards.brglm, family = binomial(probit), pl = TRUE)
update(lizards.brglm, family = binomial(cloglog), pl = TRUE)
update(lizards.brglm, family = binomial(cauchit), pl = TRUE)

---

**brglm.control**

*Auxiliary for Controlling BRGLM Fitting*

**Description**

Auxiliary function as user interface for `brglm` fitting. Typically only used when calling `brglm` or `brglm.fit`.

**Usage**

```r
brglm.control(br.epsilon = 1e-08, br.maxit = 100, br.trace=FALSE,
              brconsts = NULL, ...)
```

**Arguments**

- `br.epsilon` positive convergence tolerance for the iteration described in `brglm.fit`
- `br.maxit` integer giving the maximum number of iterations for the iteration in `brglm.fit`
- `br.trace` logical indicating if output should be produced for each iteration.
- `brconsts` a (small) positive constant or a vector of such.
- `...` further arguments passed to or from other methods.
Details

If \( \text{br.trace} = \text{TRUE} \) then for each iteration the iteration number and the current value of the modified scores is \text{cat}ed. If \( \text{br.consts} \) is specified then \( \text{br.consts} \) is added to the original binomial counts and \( 2 \times \text{br.consts} \). Then the model is fitted to the adjusted data to provide starting values for the iteration in \text{brglm.fit}. If \( \text{br.consts} = \text{NULL} \) (default) then \text{brglm.fit} adjusts the responses and totals by "number of parameters"/"number of observations" and twice that, respectively.

Value

A list with the arguments as components.

Author(s)

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

References


See Also

\texttt{brglm.fit}, the fitting procedure used by \texttt{brglm}.

### Usage

\begin{verbatim}
## S3 method for class 'brglm'
confint(object, parm = 1:length(coef(object)), level = 0.95,
    verbose = TRUE, endpoint.tolerance = 0.001,
    max.zoom = 100, zero.bound = 1e-08, stepsize = 0.5,
    stdn = 5, gridsize = 10, scale = FALSE, method = "smooth",
    ci.method = "union", n.interpolations = 100, ...)

## S3 method for class 'profile.brglm'
confint(object, parm, level = 0.95, method = "smooth",
    ci.method = "union", n.interpolations = 100, ...)
\end{verbatim}
ci.method = "union", endpoint.tolerance = 0.001, 
max.zoom = 100, n.interpolations = 100, verbose = TRUE, 
...

Arguments

object an object of class "brglm" or "profile.brglm".

parm either a numeric vector of indices or a character vector of names, specifying the parameters for which confidence intervals are to be estimated. The default is all parameters in the fitted model. When object is of class "profile.brglm", parm is not used and confidence intervals are returned for all the parameters for which profiling took place.

level the confidence level required. The default is 0.95. When object is of class "profile.brglm", level is not used and the level attribute of object is used instead.

verbose logical. If TRUE (default) progress indicators are printed during the progress of calculating the confidence intervals.

endpoint.tolerance

max.zoom

zero.bound

stepsize

stdn

gridsize

scale

method

ci.method The method to be used for the construction of confidence intervals. It can take values "union" (default) and "mean" (see Details).

n.interpolations

... further arguments to or from other methods.

Details

In the case of logistic regression Heinze & Schemper (2002) and Bull et. al. (2007) suggest the use of confidence intervals based on the profiles of the penalized likelihood, when estimation is performed using maximum penalized likelihood.

Kosmidis (2007) illustrated that because of the shape of the penalized likelihood, confidence intervals based on the penalized likelihood could exhibit low or even zero coverage for hypothesis testing on large parameter values and also misbehave illustrating severe oscillation (see Brown et. al., 2001). Kosmidis (2007) suggested an alternative confidence interval that is based on the union of the confidence intervals resulted by profiling the ordinary deviance for the maximum likelihood fit and by profiling the penalized deviance for the maximum penalized fit. Such confidence intervals, despite of being slightly conservative, illustrate less oscillation and avoid the loss of coverage.
Another possibility is to use the mean of the corresponding endpoints instead of “union”. Yet unpublished simulation studies suggest that such confidence intervals are not as conservative as the “union” based intervals but illustrate more oscillation, which however is not as severe as in the case of the penalized likelihood based ones.

The properties of the “union” and “mean” confidence intervals extend to all the links that are supported by `brglm`, when estimation is performed using maximum penalized likelihood.

In the case of estimation using modified scores and for models other than logistic, where there is not an objective that is maximized, the profiles of the penalized likelihood for the construction of the “union” and “mean” confidence intervals can be replaced by the profiles of modified score statistic (see `profileObjectives`).

The `confint` method for `brglm` and `profile.brglm` objects implements the “union” and “mean” confidence intervals. The method is chosen through the `ci.method` argument.

Value

A matrix with columns the endpoints of the confidence intervals for the specified (or profiled) parameters.

Author(s)

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

References


See Also

`confintModel`, `profileModel`, `profile.brglm`.

Examples

```r
## Begin Example 1
## Not run:
library(MASS)
data(bacteria)
contrasts(bacteria$trt) <- structure(contr.sdif(3),
                                 dimnames = list(NULL, c("drug", "encourage")))
# fixed effects analyses
```
m.glm.logit <- brglm(y ~ trt * week, family = binomial, 
data = bacteria, method = "glm.fit")
m.brglm.logit <- brglm(y ~ trt * week, family = binomial, 
data = bacteria, method = "brglm.fit")
p.glm.logit <- profile(m.glm.logit)
p.brglm.logit <- profile(m.brglm.logit)

# confidence intervals for the glm fit based on the profiles of the 
# ordinary deviance
confint(p.glm.logit)
# confidence intervals for the brglm fit
confint(p.brglm.logit, ci.method = "union")
confint(p.brglm.logit, ci.method = "mean")

# A cloglog link
m.brglm.cloglog <- update(m.brglm.logit, family = binomial(cloglog))
p.brglm.cloglog <- profile(m.brglm.cloglog)
plot(p.brglm.cloglog)
confint(m.brglm.cloglog, ci.method = "union")
confint(m.brglm.cloglog, ci.method = "mean")

## End example

## Not run:
## Begin Example 2
y <- c(1, 1, 0, 0)
totals <- c(2, 2, 2, 2)
x1 <- c(1, 0, 1, 0)
x2 <- c(1, 1, 0, 0)
m1 <- brglm(y/totals ~ x1 + x2, weights = totals, 
            family = binomial(cloglog))
p.m1 <- profile(m1)
confint(p.m1, method="zoom")

## End(Not run)

---

gethats

### Description

Calculates the Leverages for a GLM through a C Routine

### Usage

gethats(nobs, nvars, x.t, XWXinv, ww)
glm.control1

Arguments

Arguments

- `nobs` The number of observations, i.e. `dim(X)[1]`.
- `nvars` The number of parameters, i.e. `dim(X)[1]`, where `X` is the model matrix, excluding the columns that correspond to aliased parameters.
- `x.t` `t(X)`.
- `XWXinv` The inverse of the Fisher information.
- `ww` The ‘working’ weights.

Value

A vector containing the diagonal elements of the hat matrix.

Author(s)

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

See Also

- `hatvalues`, `brglm.fit`

Description

Auxiliary function as user interface for `brglm` fitting. Typically only used when calling `brglm` or `brglm.fit`.

Usage

```r
glm.control1(epsilon = 1e-08, maxit = 25, trace = FALSE, ...)
```

Arguments

Arguments

- `epsilon` as in `glm.control`.
- `maxit` as in `glm.control`.
- `trace` as in `glm.control`.
- `...` further arguments passed to or from other methods.

Details

The only difference with `glm.control` is that `glm.control1` supports further arguments to be passed from other methods. However, this additional arguments have no effect on the resulting list.

Author(s)

Ioannis Kosmidis, <ioannis.kosmidis@warwick.ac.uk>
lizards

Description

The lizards data frame has 23 rows and 6 columns. 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.

Usage

data(lizards)

Format

This data frame contains the following 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"

Source


Originally from


Examples

data(lizards)
glm(cbind(grahami, opalinus) ~ height + diameter + light + time,
   family = binomial, data=lizards)
brglm(cbind(grahami, opalinus) ~ height + diameter + light + time,
   family = binomial, data=lizards)
**modifications**

*Additive Modifications to the Binomial Responses and Totals for Use within 'brglm.fit'*

### Description

Get, test and set the functions that calculate the additive modifications to the responses and totals in binomial-response GLMs, for the application of bias-reduction either via modified scores or via maximum penalized likelihood (where penalization is by Jeffreys invariant prior).

### Usage

```r
modifications(family, pl = FALSE)
```

### Arguments

- **family**: a family object of the form `binomial(link = "link")`, where "link" can be one of "logit", "probit", "cloglog" and "cauchit". The usual ways of giving the family name are supported (see `family`).
- **pl**: logical determining whether the function returned corresponds to modifications for the penalized maximum likelihood approach or for the modified-scores approach to bias-reduction. Default value is `FALSE`.

### Details

The function returned from `modifications` accepts the argument `p` which are the binomial probabilities and returns a list with components `ar` and `at`, which are the link-dependent parts of the additive modifications to the actual responses and totals, respectively.

Since the resulting function is used in `brglm.fit`, for efficiency reasons no check is made for \( p \geq 0 \mid p \leq 1 \), for `length(at) == length(p)` and for `length(ap) == length(p)`.

### Construction of custom pseudo-data representations

If \( y^* \) are the pseudo-responses (pseudo-counts) and \( m^* \) are the pseudo-totals then we call the pair \((y^*, m^*)\) a pseudo-data representation. Both the modified-scores approach and the maximum penalized likelihood have a common property:

there exists \((y^*, m^*)\) such that if we replace the actual data \((y, m)\) with \((y^*, m^*)\) in the expression for the ordinary scores (first derivatives of the likelihood) of a binomial-response GLM, then we end-up either with the modified-scores or with the derivatives of the penalized likelihood (see Kosmidis, 2007, Chapter 5).

Let \( \mu \) be the mean of the binomial response \( y \) (i.e. \( \mu = mp \), where \( p \) is the binomial probability corresponding to the count \( y \)). Also, let \( d \) and \( d' \) denote the first and the second derivatives, respectively, of \( \mu \) with respect to the linear predictor \( \eta \) of the model. All the above are viewed as functions of \( p \). The pseudo-data representations have the generic form

- **pseudo-response**: \( y^* = y + ha_r(p) \)
- **pseudo-totals**: \( m^* = m + ha_t(p) \),
where \( h \) is the leverage corresponding to \( y \). The general expressions for \( a_r(p) \) ("r" for "response") and \( a_t(p) \) ("t" for "totals") are:

**modified-scores approach**

\[
\begin{align*}
a_r(p) &= d'(p)/(2w(p)) \\
a_t(p) &= 0,
\end{align*}
\]

**maximum penalized likelihood approach**

\[
\begin{align*}
a_r(p) &= d'(p)/w(p) + p - 0.5 \\
a_t(p) &= 0.
\end{align*}
\]

For supplying \((y^*, m^*)\) in `glm.fit` (as is done by `brglm.fit`), an essential requirement for the pseudo-data representation is that it should mimic the behaviour of the original responses and totals, i.e. \( 0 \leq y^* \leq m^* \). Since \( h \in [0, 1] \), the requirement translates to \( 0 \leq a_r(p) \leq a_t(p) \) for every \( p \in (0, 1) \). However, the above definitions of \( a_r(p) \) and \( a_t(p) \) do not necessarily respect this requirement.

On the other hand, the pair \((a_r(p), a_t(p))\) is not unique in the sense that for a given link function and once the link-specific structure of the pair has been extrapolated, there is a class of equivalent pairs that can be resulted following only the following two rules:

- add and subtract the same quantity from either \( a_r(p) \) or \( a_t(p) \).
- if a quantity is to be moved from \( a_r(p) \) to \( a_t(p) \) it first has to be divided by \(-p\).

For example, in the case of penalized maximum likelihood, the pairs \((\frac{d'(p)}{w(p)} + p - 0.5, 0)\) and \((\frac{d'(p)}{w(p)} + p, 0.5/p)\) are equivalent, in the sense that if the corresponding pseudo-data representations are substituted in the ordinary scores both return the same expression.

So, in order to construct a pseudo-data representation that corresponds to a user-specified link function and has the property \( 0 \leq a_r(p) \leq a_t(p) \) for every \( p \in (0, 1) \), one merely has to pursue a simple algebraic calculation on the initial pair \((a_r(p), a_t(p))\) using only the two aforementioned rules until an appropriate pair is resulted. There is always a pair!

Once the pair has been found the following steps should be followed.

1. For a user-specified link function the user has to write a modification function with name "br.custom.family" or "pml.custom.family" for \( pl=FALSE \) or \( pl=TRUE \), respectively. The function should take as argument the probabilities \( p \) and return a list of two vectors with same length as \( p \) and with names c("ar","at"). The result corresponds to the pair \((a_r(p), a_t(p))\).

2. Check if the custom-made modifications function is appropriate. This can be done via the function `checkModifications` which has arguments `fun` (the function to be tested) and `Length` with default value `Length=100`. `Length` is to be used when the user-specified link function takes as argument a vector of values (e.g. the `logexp` link in `?family`). Then the value of `Length` should be the length of that vector.

3. Put the function in the search patch so that `modifications` can find it.

4. `brglm` can now be used with the custom family as `glm` would be used.
Note

The user could also deviate from modified-scores and maximum penalized likelihood and experiment with implemented (or not) links, e.g. probit, constructing his own pseudo-data representations of the aforementioned general form. This could be done by changing the link name, e.g. by

```r
probitt <- make.link(probit); probitt$name <- "probitt"
```

and then setting a custom `br.custom.family` that does not necessarily depend on the probit link. Then, `brglm` could be used with `pl=FALSE`.

A further generalization would be to completely remove the hat value $h$ in the generic expression of the pseudo-data representation and have general additive modifications that depend on $p$. To do this divide both $ar$ and $at$ by $\max(get("hats",parent.frame()),.Machine$double.eps)$ within the custom modification function (see also Examples).

Author(s)

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

References


See Also

`brglm`, `brglm.fit`

Examples

```r
## Begin Example 1
## logistic exposure model, following the Example in ?family. See,
## # Definition of the link function
logexp <- function(days = 1) {
  linkfun <- function(mu) qlogis(mu^(1/days))
  linkinv <- function(eta) plogis(eta)^(days
  mu.eta <- function(eta) days * plogis(eta)^(days-1) * 
    binomial()$mu.eta(eta)
  valideta <- function(eta) TRUE
  link <- paste("logexp(", days, ", "", sep="")
  structure(list(linkfun = linkfun, linkinv = linkinv,
    mu.eta = mu.eta, valideta = valideta, name = link),
    class = "link-glm")
}
# Here d(p) = days * p * ( 1 - p^(1/days) )
# d'(p) = (days - (days+1) * p^(1/days)) * d(p)
# w(p) = days^2 * p * (1-p^(1/days))^2 / (1-p)
# Initial modifications, as given from the general expressions above:
```
br.custom.family <- function(p) {
  etas <- binomial(logexp(.days))$linkfun(p)
  # the link function argument ".days" will be detected by lexical
  # scoping. So, make sure that the link-function inputted arguments
  # have unusual names, like '.days' and that
  # the link function enters 'brglm' as
  # 'family=binomial(logexp(.days))'.
  list(ar = 0.5*(1-p)-0.5*(1-p)*exp(etas)/.days,
       at = @p/p) # so that to fix the length of at
}

.days <- 3
# ".days" could be a vector as well but then it should have the same
# length as the number of observations ('length(.days)' should be
# equal to 'length(p)'). In this case, 'checkModifications' should
# have argument 'Length=length(.days)'.
#
# Check:
## Not run: checkModifications(br.custom.family)
# OOOPS error message... the condition is not satisfied
#
# After some trivial algebra using the two allowed operations, we
# get new modifications:
br.custom.family <- function(p) {
  etas <- binomial(logexp(.days))$linkfun(p)
  list(ar = 0.5*p/p, # so that to fix the length of ar
       at = 0.5+exp(etas)*(1-p)/(2*p*.days))
}

# Check:
checkModifications(br.custom.family)
# It is OK.
# Now,
modifications(binomial(logexp(.days)))
# works.
# Notice that for ".days <- 1", 'logexp(.days)' is the 'logit' link
# model and 'a_r=0.5', 'a_t=1'.
# In action:
library(MASS)
example(birthwt)
m.glm <- glm(formula = low ~ ., family = binomial, data = bwt)
.days <- bwt$age
m.glm.logexp <- update(m.glm,family=binomial(logexp(.days))))
m.brglm.logexp <- brglm(formula = low ~ ., family = binomial(logexp(.days)), data = bwt)
# The fit for the 'logexp' link via maximum likelihood
m.glm.logexp
# and the fit for the 'logexp' link via modified scores
m.brglm.logexp
## End Example
## Begin Example 2
## Another possible use of brglm.fit:
## Deviating from bias reducing modified-scores:
## Add 1/2 to the response of a probit model.
y <- c(1,2,3,4)
totals <- c(5,5,5,5)
x1 <- c(1,0,1,0)
x2 <- c(1,1,0,0)
my.probit <- make.link("probit")
my.probit$name <- "my.probit"
br.custom.family <- function(p) {
  h <- pmax(get("hats",parent.frame()),.Machine$double.eps)
  list(ar=0.5/h,at=1/h)
}
m1 <- brglm(y/totals~x1+x2,weights=totals,family=binomial(my.probit))
m2 <- glm((y+0.5)/(totals+1)~x1+x2,weights=totals+1,family=binomial(probit))
# m1 and m2 should be the same.
# End example
# Begin example 3: Maximum penalized likelihood for logistic regression,
# with the penalty being a power of the Jeffrey's prior (".const" below)
# Setup a custom logit link
mylogit <- make.link("logit")
mylogit$name <- "mylogit"
## Set-up the custom family
br.custom.family <- function(p) {
  list(ar = .const * p/p, at = 2 * .const * p/p)
}
data("lizards")
## The reduced-bias fit is
.const <- 1/2
brglm(cbind(grahami, opalinus) ~ height + diameter +
      light + time, family = binomial(mylogit), data=lizards)
## which is the same as what brglm does by default for logistic regression
brglm(cbind(grahami, opalinus) ~ height + diameter +
      light + time, family = binomial(logit), data=lizards)
## Stronger penalization (e.g. 5/2) can be achieved by
.const <- 5/2
brglm(cbind(grahami, opalinus) ~ height + diameter +
      light + time, family = binomial(mylogit), data=lizards)
# End example

plot.profile.brglm

## S3 method for class 'profile.brglm'
plot(x, signed = FALSE, interpolate = TRUE, n.interpolations = 100, print.grid.points = FALSE, ...)
## S3 method for class 'profile.brglm'
pairs(x, colours = 2:3, ...)

Arguments

- **x**: a "profile.brglm" object.
- **signed**: as in `plot.profileModel`.
- **interpolate**: as in `plot.profileModel`.
- **n.interpolations**: as in `plot.profileModel`.
- **print.grid.points**: as in `plot.profileModel`.
- **colours**: as in `plot.profileModel`.
- **...**: further arguments passed to or from other methods.

Details

See Details in `plot.profileModel`.

Author(s)

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

See Also

`plot.profileModel`, `profile.brglm`.

Examples

# see example in 'confint.brglm'.

---

**profile.brglm**

Calculate profiles for objects of class 'brglm'.

Description

Creates "profile.brglm" objects to be used for the calculation of confidence intervals and for plotting.

Usage

## S3 method for class 'brglm'
profile(fitted, gridsize = 10, stdn = 5,
    stepsize = 0.5, level = 0.95,
    which = 1:length(coef(fitted)), verbose = TRUE,
    zero.bound = 1e-08, scale = FALSE, ...)
profile.brglm

Arguments

fitted       an object of class "brglm".
gridsize     as in profileModel.
stdn         as in profileModel.
stepsize     as in profileModel.
level        qchisq(level,1) indicates the range that the profiles must cover.
which        as in profileModel.
verbose      as in profileModel.
zero.bound   as in profileModel.
scale        as in profileModel.
...          further arguments passed to or from other methods.

Details

profile.brglm calculates the profiles of the appropriate objectives to be used for the construction of confidence intervals for the bias-reduced estimates (see confint.brglm for the objectives that are profiled).

Value

An object of class "profile.glm" with attribute "level" corresponding to the argument level. The object supports the methods print, plot, pairs and confint and it is a list of the components:

profilesML   a "profileModel" object containing the profiles of the ordinary deviance for the maximum likelihood fit corresponding to fitted.
profilesBR   NULL if method = "glm.fit" in brglm. If method = "brglm.fit" and pl = TRUE, profilesBR is a "profileModel" object containing the profiles of the penalized deviance for the parameters of fitted. If method = "brglm.fit" and pl = FALSE profilesBR is a "profileModel" object containing the profiles of the modified score statistic (see profileObjectives) for the parameters of fitted.

Note

Objects of class "profile.brglm" support the methods:

• print which prints the "level" attribute of the object, as well as the supported methods.
• confint see confint.brglm.
• plot see plot.profile.brglm.
• pairs see plot.profile.brglm.

Author(s)

Ioannis Kosmidis, <ioannis.kosmidis@warwick.ac.uk>
profileObjectives-brglm

See Also

profileModel, profile.brglm.

Examples

# see example in 'confint.brglm'.

profileObjectives-brglm

Objectives to be profiled

Description

Objectives that are used in profile.brglm

Usage

penalizedDeviance(fm, X, dispersion = 1)

modifiedScoreStatistic(fm, X, dispersion = 1)

Arguments

fm       the restricted fit.
X        the model matrix of the fit on all parameters.
dispersion the dispersion parameter.

Details

These objectives follow the specifications for objectives in the profileModel package and are used from profile.brglm.

penalizedDeviance returns a deviance-like value corresponding to a likelihood function penalized by Jeffreys invariant prior. It has been used by Heinze & Schemper (2002) and by Bull et. al. (2002) for the construction of confidence intervals for the bias-reduced estimates in logistic regression. The X argument is the model matrix of the full (not the restricted) fit.

modifiedScoreStatistic mimics RaoScoreStatistic in profileModel, but with the ordinary scores replaced with the modified scores used for bias reduction. The argument X has the same interpretation as for penalizedDeviance.

Value

A scalar.

Author(s)

Ioannis Kosmidis, <ioannis.kosmidis@warwick.ac.uk>
References


See Also

profileModel, profile.brglm.

---

separation.detection  Separation Identification.

Description

Provides a tool for identifying whether or not separation has occurred.

Usage

separation.detection(fit, nsteps = 30)

Arguments

- **fit**: the result of a `glm` call.
- **nsteps**: `Starting from maxit = 1, the GLM is refitted for maxit = 2, maxit = 3, ..., maxit = nsteps. Default value is 30.`

Details

Identifies separated cases for binomial-response GLMs, by refitting the model. At each iteration the maximum number of allowed IWLS iterations is fixed starting from 1 to `nsteps` (by setting `control = glm.control(maxit = j)`, where `j` takes values 1, ..., `nsteps` in `glm`). For each value of `maxit`, the estimated asymptotic standard errors are divided to the corresponding ones resulted for `control = glm.control(maxit = 1)`. Based on the results in Lesaffre & Albert (1989), if the sequence of ratios in any column of the resulting matrix diverges, then separation occurs and the maximum likelihood estimate for the corresponding parameter has value minus or plus infinity.

Value

A matrix of dimension `nsteps` by `length(coef(fit))`, that contains the ratios of the estimated asymptotic standard errors.

Author(s)

Ioannis Kosmidis, <ioannis.kosmidis@warwick.ac.uk>
References


Examples

```r
## Begin Example
y <- c(1,1,0,0)
totals <- c(2,2,2,2)
x1 <- c(1,0,1,0)
x2 <- c(1,1,0,0)
m1 <- glm(y/totals ~ x1 + x2, weights = totals, family = binomial())
# No warning from glm...
m1

# However estimates for (Intercept) and x2 are unusually large in
# absolute value... Investigate further:
# separation.detection(m1, nsteps=30)
# Note that the values in the column for (Intercept) and x2 diverge,
# while for x1 converged. Hence, separation has occurred and the
# maximum likelihood estimate for (Intercept) is minus infinity and
# for x2 is plus infinity. The signs for infinity are taken from the
# signs of (Intercept) and x1 in coef(m1).
## End Example
```
Index

* datasets
  lizards, 13
* dplot
  plot.profile.brglm, 18
* hplot
  plot.profile.brglm, 18
* htest
  confint.brglm, 8
* iteration
  brglm, 2
  brglm.control, 7
  glm.control1, 12
* models
  brglm, 2
  confint.brglm, 8
  modifications, 14
  profile.brglm, 19
  profileObjectives-brglm, 21
  separation.detection, 22
* regression
  brglm, 2
  gethats, 11
  modifications, 14
* utilities
  separation.detection, 22
  add1, 5
  anova, 5
  brglm, 2, 4, 7, 8, 10, 12, 15, 16, 20
  brglm.control, 3, 7
  brglm.fit, 7, 8, 11, 12, 14–16
  cat, 8
  checkModifications, 15
  checkModifications (modifications), 14
  coefficients, 5
  confint, 20
  confint.brglm, 8, 20
  confint.profile.brglm (confint.brglm), 8
  confintModel, 9, 10
  drop1, 5
  family, 6, 14
  gethats, 3, 11
  glm, 2–6, 15, 22
  glm.control, 12
  glm.control1, 12
  glm.fit, 3, 6, 15
  hatvalues, 3, 12
  lizards, 13
  modifications, 3, 4, 6, 14
  modifiedScoreStatistic
    (profileObjectives-brglm), 21
  pairs, 20
  pairs.profile.brglm
    (plot.profile.brglm), 18
  penalizedDeviance
    (profileObjectives-brglm), 21
  plot, 20
  plot.profile.brglm, 18, 20
  plot.profileModel, 19
  predict, 5
  print, 5, 20
  print.brglm (brglm), 2
  print.profile.brglm (profile.brglm), 19
  print.summary.brglm (brglm), 2
  profile.brglm, 10, 19, 19, 21, 22
  profileModel, 10, 20–22
  profileObjectives, 8, 10, 20
  profileObjectives
    (profileObjectives-brglm), 21
    profileObjectives-brglm, 21
  RaoScoreStatistic, 21
INDEX

residuals, 5

separation.detection, 22
summary, 5
summary.brglm (brglm), 2
vcov, 5