Package ‘logistf’

July 22, 2025

Version 1.26.1

Date 2025-04-16

Title Firth's Bias-Reduced Logistic Regression

Depends R (>= 3.0.0)

Imports mice, mgcv, formula.tools, Matrix

Suggests emmeans (>= 1.4), estimability

Description Fit a logistic regression model using Firth's bias reduction method, equivalent to penalization of the log-likelihood by the Jeffreys prior. Confidence intervals for regression coefficients can be computed by penalized profile likelihood. Firth's method was proposed as ideal solution to the problem of separation in logistic regression, see Heinze and Schemper (2002) <doi:10.1002/sim.1047>. If needed, the bias reduction can be turned off such that ordinary maximum likelihood logistic regression is obtained. Two new modifications of Firth's method, FLIC and FLAC, lead to unbiased predictions and are now available in the package as well, see Puhr et al (2017) <doi:10.1002/sim.7273>.

License GPL

URL `https://github.com/georgheinze/logistf`

LazyLoad yes

NeedsCompilation yes

RoxygenNote 7.3.2

LazyData true

BugReports `https://github.com/georgheinze/logistf/issues/`

Author Georg Heinze [aut, cre],
Meinhard Ploner [aut],
Daniela Dunkler [ctb],
Harry Southworth [ctb],
Lena Jiricka [aut],
Gregor Steiner [aut]

Maintainer Georg Heinze <`georg.heinze@meduniwien.ac.at`>

Repository CRAN

Date/Publication 2025-04-16 15:50:02 UTC

Contents

`logistf-package` *Firth's Bias-Reduced Logistic Regression*

Description

Fits a binary logistic regression model using Firth's bias reduction method, and its modifications FLIC and FLAC, which both ensure that the sum of the predicted probabilities equals the number of events. If needed, the bias reduction can be turned off such that ordinary maximum likelihood logistic regression is obtained.

Details

The package logistf provides a comprehensive tool to facilitate the application of Firth's correction for logistic regression analysis, including its modifications FLIC and FLAC.

The call of the main function of the library follows the structure of the standard functions as lm or glm, requiring a data.frame and a formula for the model specification. The resulting object belongs to the new class logistf, which includes penalized maximum likelihood (`Firth-Logistic'- or` FL'-type) logistic regression parameters, standard errors, confidence limits, p-values, the value of the maximized penalized log likelihood, the linear predictors, the number of iterations needed to arrive at the maximum and much more. Furthermore, specific methods for the resulting object are supplied. Additionally, a function to plot profiles of the penalized likelihood function and a function to perform penalized likelihood ratio tests have been included.

In explaining the details of the estimation process we follow mainly the description in Heinze & Ploner (2003). In general, maximum likelihood estimates are often prone to small sample bias. To reduce this bias, Firth (1993) suggested to maximize the penalized log likelihood $\log L(\beta)^* = \log L(\beta) + 1/2 \log |I(\beta)|$, where $I(\beta)$ is the Fisher information matrix, i. e. minus the second derivative of the log likelihood. Applying this idea to logistic regression, the score function $U(\beta)$ is replaced by the modified score function $U(\beta)^* = U(\beta) + a$, where a has rth entry $a_r = 0.5 tr I(\beta)^{-1}[dI(\beta)/d\beta_r], r = 1, ..., k$. Heinze and Schemper (2002) give the explicit formulae for $I(\beta)$ and $I(\beta)/d\beta_r$.

In our programs estimation of β can be based on a Newton-Raphson algorithm or on iteratively reweighted least squares. Parameter values are initialized usually with 0, but in general the user can specify arbitrary starting values.

With a starting value of $\beta^{(0)}$, the penalized maximum likelihood estimate β is obtained iteratively via Newton-Raphson:

$$\beta^{(s+1)} = \beta^{(s)} + I(\beta^{(s)})^{-1} U(\beta^{(s)})^*$$

If the penalized log likelihood evaluated at $\beta^{(s+1)}$ is less than that evaluated at $\beta^{(s)}$, then ($\beta^{(s+1)}$ is recomputed by step-halving. For each entry r of β with $r = 1, ..., k$ the absolute step size $|\beta_r^{(s+1)} - \beta_r^s|$ is restricted to a maximal allowed value `maxstep`. These two means should avoid numerical problems during estimation. The iterative process is continued until the parameter estimates converge, i. e., until three criteria are met: the change in log likelihood is less than `lconv`, the maximum absolute element of the score vector is less than `gconv`, the maximum absolute change in beta is less than `xconv`. `lconv`, `gconv`, `xconv` can be controlled by `control=logistf.control(lconv=..., gconv=..., xconv=...)`.

Computation of profile penalized likelihood confidence intervals for parameters (`logistpl`) follows the algorithm of Venzon and Moolgavkar (1988). For testing the hypothesis of $\gamma = \gamma_0$, let the likelihood ratio statistic

$$LR = 2[\log L(\gamma, \delta) - \log L(\gamma_0, \delta_{\gamma_0})^*]$$

where (γ, δ) is the joint penalized maximum likelihood estimate of $\beta = (\gamma, \delta)$, and δ_{γ_0} is the penalized maximum likelihood estimate of δ when $\gamma = \gamma_0$. The profile penalized likelihood confidence interval is the continuous set of values γ_0 for which LR does not exceed the $(1-\alpha)100$th percentile of the χ^2_1-distribution. The confidence limits can therefore be found iteratively by approximating the penalized log likelihood function in a neighborhood of β by the quadratic function

$$l(\beta + \delta) = l(\beta) + \delta' U^* - 0.5 \delta' I \delta$$

where $U^* = U(\beta)^*$ and $-I = -I(\beta)$.

In some situations computation of profile penalized likelihood confidence intervals may be time consuming since the iterative procedure outlined above has to be repeated for the lower and for the upper confidence limits of each of the k parameters. In other problems one may not be interested in interval estimation, anyway. In such cases, the user can request computation of Wald confidence intervals and P-values, which are based on the normal approximation of the parameter estimates and do not need any iterative estimation process. Note that from version 1.24.1 on, the variance-covariance matrix is based on the second derivative of the likelihood of the augmented data rather than the original data, which proved to be a better approximation if the user chooses to set a higher value for τ, the penalty strength.

The adequacy of Wald confidence intervals for parameter estimates can be verified by plotting the profile penalized log likelihood (PPL) function. A symmetric shape of the PPL function allows use of Wald intervals, while an asymmetric shape demands profile penalized likelihood intervals (*Heinze & Schemper (2002)*). Further documentation can be found in *Heinze & Ploner (2004)*.

The package includes functions to work with multiply imputed data sets, such as generated by the `mice` package. Results on individual fits can be pooled to obtain point and interval estimates, as well as profile likelihood confidence intervals and likelihood profiles in general (Heinze, Ploner and Beyea, 2013).

Moreover, in the package the modifications FLIC and FLAC have been implemented, which were described in Puhr et al (2017) as solutions to obtain accurate predicted probabilities.

Author(s)

Georg Heinze georg.heinze@meduniwien.ac.at, Meinhard Ploner and Lena Jiricka.

References

Firth D (1993). Bias reduction of maximum likelihood estimates. *Biometrika* 80, 27–38.

Heinze G, Schemper M (2002). A solution to the problem of separation in logistic regression. *Statistics in Medicine* 21: 2409-2419.

Heinze G, Ploner M (2003). Fixing the nonconvergence bug in logistic regression with SPLUS and SAS. *Computer Methods and Programs in Biomedicine* 71: 181-187.

Heinze G, Ploner M (2004). Technical Report 2/2004: A SAS-macro, S-PLUS library and R package to perform logistic regression without convergence problems. Section of Clinical Biometrics, Department of Medical Computer Sciences, Medical University of Vienna, Vienna, Austria. `https://github.com/georgheinze/flicflac/blob/master/Ressources/tr2_2004.pdf`

Heinze G (2006). A comparative investigation of methods for logistic regression with separated or nearly separated data. *Statistics in Medicine* 25: 4216-4226.

Heinze G, Ploner M, Beyea J (2013). Confidence intervals after multiple imputation: combining profile likelihood information from logistic regressions. *Statistics in Medicine* 32:5062-5076.

Puhr R, Heinze G, Nold M, Lusa L, Geroldinger A (2017). Firth's logistic regression with rare events: accurate effect estimates and predictions? *Statistics in Medicine* 36: 2302-2317.

Venzon DJ, Moolgavkar AH (1988). A method for computing profile-likelihood based confidence intervals. *Applied Statistics* 37:87-94.

See Also

Useful links:

- `https://github.com/georgheinze/logistf`
- Report bugs at `https://github.com/georgheinze/logistf/issues/`

`add1.logistf` *Add or Drop All Possible Single Terms to/from a* `logistf` *Model*

Description

Compute all the single terms in the scope argument that can be added to or dropped from the model, fit those models and compute a table of the changes in fit.

Usage

```
## S3 method for class 'logistf'
add1(object, scope, data, test = "PLR", ...)
```

Arguments

`object`	A fitted `logistf`, `flic` or `flac` object
`scope`	The scope of variables considered for adding or dropping. Should be a vector of variable names. Can be left missing; the method will then use all variables in the object's data slot which are not identified as the response variable.
`data`	The data frame used to fit the object.
`test`	The type of test statistic. Currently, only the PLR test (penalized likelihood ratio test) is allowed for logistf fits.
`...`	Further arguments passed to or from other methods.

Details

`drop1` and `add1` generate a table where for each variable the penalized likelihood ratio chi-squared, the degrees of freedom, and the p-value for dropping/adding this variable are given.

Value

A matrix with `nvar` rows and 3 columns (Chisquared, degrees of freedom, p-value).

Examples

```
data(sex2)
fit<-logistf(data=sex2, case~1, pl=FALSE)
add1(fit, scope=c("dia", "age"), data=sex2)

fit2<-logistf(data=sex2, case~age+oc+dia+vic+vicl+vis)
drop1(fit2, data=sex2)
```

`anova.logistf` *Analysis of Penalized Deviance for* `logistf` *Models*

Description

This method compares hierarchical and non-hierarchical logistf models using penalized likelhood ratio tests. It replaces the function logistftest of former versions of logistf.

Usage

```
## S3 method for class 'logistf'
anova(object, fit2, formula, method = "nested", ...)
```

Arguments

`object`	A fitted `logistf` model object
`fit2`	Another fitted `logistf` model object, to be compared with `object`
`formula`	Alternatively to `fit2`, a formula which specifies terms to omit from the object model fit.
`method`	One of c("nested","PLR"). nested is the default for hierarchically nested models, and will compare the penalized likelihood ratio statistics (minus twice the difference between maximized penalized log likelihood and null penalized log likelihood), where the null penalized log likelihood is computed from the same, hierarchically superior model. Note that unlike in maximum likelihood analysis, the null penalized likelihood depends on the penalty (Jeffreys prior) which itself depends on the scope of variables of the hierarchically superior model. PLR compares the difference in penalized likelihood ratio between the two models, where for each model the null penalized likelihood is computed within the scope of variables in that model. For PLR, the models need not be hierarchically nested.
`...`	Further arguments passed to the method.

Details

Comparing models fitted by penalized methods, one must consider that the penalized likelihoods are not directly comparable, since a penalty is involved. Or in other words, inserting zero for some regression coefficients will not lead to the same penalized likelihood as if the corresponding variables are simply "unknown" to a model. The anova method takes care that the same penalty is used for two hierarchically nested models, and if the models are not hierarchically nested, it will first relate each penalized likelihood to its null penalized likelihood, and only compare the resulting penalized likelihod ratio statistics. The chi-squared approximation for this latter method (PLR) is considered less accurate than that of the nested method. Nevertheless, it is the only way to go for comparison of non-nested models.

Value

An object of class `anova.logistf` with items

`chisq`	the chisquared statistic for the model comparison
`df`	The degrees of freedom
`pval`	The p-value
`call`	The function call
`method`	The method of comparison (input)
`model1`	The first model
`model2`	The second model which was compared to the first model
`PLR1`	The PLR statistic of the first model
`PLR2`	the PLR statistic of the second model; for the nested method, this will be the drop in chi-squared due to setting the coefficients to zero

Examples

```
data(sex2)
fit<-logistf(data=sex2, case~age+oc+dia+vic+vicl+vis)

#simultaneous test of variables vic, vicl, vis:
anova(fit, formula=~vic+vicl+vis)

#test versus a simpler model
fit2<-logistf(data=sex2, case~age+oc+dia)
# or: fit2<-update(fit, case~age+oc+dia)
anova(fit,fit2)

# comparison of non-nested models (with different df):
fit3<-logistf(data=sex2, case~age+vic+vicl+vis)
anova(fit2,fit3, method="PLR")
```

`backward` *Backward Elimination/Forward Selection of Model Terms in logistf Models*

Description

These functions provide simple backward elimination/forward selection procedures for logistf models.

Usage

```
backward(object, ...)

## S3 method for class 'logistf'
backward(
  object,
  scope,
  data,
  steps = 1000,
  slstay = 0.05,
  trace = TRUE,
  printwork = FALSE,
  full.penalty = FALSE,
  ...
)

## S3 method for class 'flic'
backward(
  object,
  scope,
  steps = 1000,
  slstay = 0.05,
  trace = TRUE,
  printwork = FALSE,
  full.penalty = FALSE,
  ...
)

forward(object, ...)

## S3 method for class 'logistf'
forward(
  object,
  scope,
  data,
  steps = 1000,
  slentry = 0.05,
  trace = TRUE,
  printwork = FALSE,
  pl = TRUE,
  ...
)
```

Arguments

`object`	A fitted logistf model object. To start with an empty model, create a model fit with a formula= y~1, pl=FALSE. (Replace y by your response variable.)
`...`	Further arguments to be passed to methods.

`scope`	The scope of variables to add/drop from the model. Can be missing for backward, backward will use the terms of the object fit. Alternatively, an arbitrary vector of variable names can be given, to allow that only some of the variables will be competitively selected or dropped. Has to be provided for forward.
`data`	The data frame used to fit the object.
`steps`	The number of forward selection/backward elimination steps.
`slstay`	For `backward`, the significance level to stay in the model.
`trace`	If `TRUE`, protocols selection steps.
`printwork`	If `TRUE`, prints each working model that is visited by the selection procedure.
`full.penalty`	If `TRUE` penalty is not taken from current model but from start model.
`slentry`	For `forward`, the significance level to enter the model.
`pl`	For forward, computes profile likelihood confidence intervals for the final model if `TRUE`.

Details

The variable selection is simply performed by repeatedly calling add1 or drop1 methods for logistf, and is based on penalized likelihood ratio test.

Note that selecting among factor variables is not supported. One way to use forward or backward with factor variables is to first convert them into numeric variables (0/1 coded dummy variables, choosing a sensible reference category). Forward and backward will then perform selection on the dummy variables, meaning that it will collapse levels of a factor variable with similar outcomes.

Value

An updated `logistf`, `flic` or `flac` fit with the finally selected model.

Functions

- `forward()`: Forward Selection

Examples

```
data(sex2)
fit<-logistf(data=sex2, case~1, pl=FALSE)
fitf<-forward(fit, scope=c("dia", "age"), data=sex2)

fit2<-logistf(data=sex2, case~age+oc+vic+vicl+vis+dia)
fitb<-backward(fit2, data=sex2)
```

`CLIP.confint` *Confidence Intervals after Multiple Imputation: Combination of Likelihood Profiles*

Description

This function implements the new combination of likelihood profiles (CLIP) method described in Heinze, Ploner and Beyea (2013). This method is useful for computing confidence intervals for parameters after multiple imputation of data sets, if the normality assumption on parameter estimates and consequently the validity of applying Rubin's rules (pooling of variances) is in doubt. It consists of combining the profile likelihoods into a posterior. The function CLIP.confint searches for those values of a regression coefficient, at which the cumulative distribution function of the posterior is equal to the values specified in the argument ci.level (usually 0.025 and 0.975). The search is performed using R's optimize function.

Usage

```
CLIP.confint(
  obj = NULL,
  variable = NULL,
  data,
  firth = TRUE,
  weightvar = NULL,
  control = logistf.control(),
  ci.level = c(0.025, 0.975),
  pvalue = TRUE,
  offset = NULL,
  bound.lo = NULL,
  bound.up = NULL,
  legacy = FALSE
)
```

Arguments

`obj`	Either a list of logistf fits (on multiple imputed data sets), or the result of analysis of a `mice` (multiply imputed) object using `with.mids`
`variable`	The variable of interest, for which confidence intervals should be computed. If missing, confidence intervals for all variables will be computed.
`data`	A list of data set corresponding to the model fits. Can be left blank if obj was obtained with the `dataout=TRUE` option or if obj was obtained by mice
`firth`	If `TRUE`, applies the Firth correction. Should correspond to the entry in obj.
`weightvar`	An optional weighting variable for each observation.
`control`	Control parameters for `logistf`, usually obtained by `logistf.control()`
`ci.level`	The two confidence levels for each tail of the posterior distribution.
`pvalue`	If `TRUE`, will also compute a P-value from the posterior.

`offset`	An optional offset variable
`bound.lo`	Bounds (vector of length 2) for the lower limit. Can be left blank. Use only if problems are encountered.
`bound.up`	Bounds (vector of length 2) for the upper limit. Can be left blank. Use only if problems are encountered.
`legacy`	If `TRUE`, will use pure R code for all model fitting. Can be slow. Not recommended.

Details

For each confidence limit, this function performs a binary search to evaluate the combined posterior, which is obtained by first transforming the imputed-data likelihood profiles into cumulative distribution functions (CDFs), and then averaging the CDFs to obtain the CDF of the posterior. Usually, the binary search manages to find the confidence intervals very quickly. The number of iterations (mean and maximum) will be supplied in the output object. Further details on the method can be found in Heinze, Ploner and Beyea (2013).

Value

An object of class `CLIP.confint`, with items:

`variable`	The variable(s) which were analyzed
`estimate`	The pooled estimate (average over imputations)
`ci`	The confidence interval(s)
`pvalue`	The p-value(s)
`imputations`	The number of imputed datasets
`ci.level`	The confidence level (input)
`bound.lo`	The bounds used for finding the lower confidence limit; usually not of interest. May be useful for error-tracing.
`bound.up`	The bounds used for finding the upper confidence limit
`iter`	The number of iterations (for each variable and each tail)
`call`	The call object

Author(s)

Georg Heinze and Meinhard Ploner

References

Heinze G, Ploner M, Beyea J (2013). Confidence intervals after multiple imputation: combining profile likelihood information from logistic regressions. Statistics in Medicine, to appear.

See Also

`logistf()` for Firth's bias-Reduced penalized-likelihood logistic regression.

Examples

```
#generate data set with NAs
freq=c(5,2,2,7,5,4)
y<-c(rep(1,freq[1]+freq[2]), rep(0,freq[3]+freq[4]), rep(1,freq[5]), rep(0,freq[6]))
x<-c(rep(1,freq[1]), rep(0,freq[2]), rep(1,freq[3]), rep(0,freq[4]),
rep(NA,freq[5]),rep(NA,freq[6]))
toy<-data.frame(x=x,y=y)

# impute data set 5 times
set.seed(169)
toymi<-list(0)
for(i in 1:5){
  toymi[[i]]<-toy
  y1<-toymi[[i]]$y==1 & is.na(toymi[[i]]$x)
  y0<-toymi[[i]]$y==0 & is.na(toymi[[i]]$x)
  xnew1<-rbinom(sum(y1),1,freq[1]/(freq[1]+freq[2]))
  xnew0<-rbinom(sum(y0),1,freq[3]/(freq[3]+freq[4]))
  toymi[[i]]$x[y1==TRUE]<-xnew1
  toymi[[i]]$x[y0==TRUE]<-xnew0
  }

 # logistf analyses of each imputed data set
 fit.list<-lapply(1:5, function(X) logistf(data=toymi[[X]], y~x, pl=TRUE))

 # CLIP confidence limits
 CLIP.confint(obj=fit.list, data = toymi)
```

`CLIP.profile` *Combine Profile Likelihoods from Imputed-Data Model Fits*

Description

This function uses CLIP (combination of likelihood profiles) to compute the pooled profile of the posterior after multiple imputation.

Usage

```
CLIP.profile(
  obj = NULL,
  variable,
  data,
  which,
  firth = TRUE,
  weightvar,
  control = logistf.control(),
  offset = NULL,
  from = NULL,
  to = NULL,
```

```
  steps = 101,
  legacy = FALSE,
  keep = FALSE
)
```

Arguments

`obj`	Either a list of logistf fits (on multiple imputed data sets), or the result of analysis of a `mice` (multiply imputed) object using `with.mids`.
`variable`	The variable of interest, for which confidence intervals should be computed. If missing, confidence intervals for all variables will be computed.
`data`	A list of data set corresponding to the model fits. Can be left blank if obj was obtained with the dataout=TRUE option or if obj was obtained by mice.
`which`	Alternatively to variable, the argument which allows to specify the variable to compute the profile for as righthand formula, e.g. which=~X.
`firth`	If `TRUE`, applies the Firth correction. Should correspond to the entry in obj.
`weightvar`	An optional weighting variable for each observation
`control`	control parameters for `logistf`, usually obtained by `logistf.control()`
`offset`	An optional offset variable
`from`	Lowest value for the sequence of values for the regression coefficients for which the profile will be computed. Can be left blank.
`to`	Highest value for the sequence of values for the regression coefficients for which the profile will be computed. Can be left blank
`steps`	Number of steps for the sequence of values for the regression coefficients for which the profile will be computed
`legacy`	If `TRUE`, only R code will be used. Should be avoided.
`keep`	If `TRUE`, keeps the profiles for each imputed data sets in the output object.

Details

While CLIP.confint iterates to find those values at which the CDF of the pooled posterior equals the confidence levels, CLIP.profile will evaluate the whole profile, which enables plotting and evaluating the skewness of the combined and the completed-data profiles. The combined and completed-data profiles are available as cumulative distribution function (CDF) or in the scaling of relative profile likelihood (minus twice the likelihood ratio statistic compared to the maximum). Using a plot method, the pooled posterior can also be displayed as a density.

Value

An object of class `CLIP.profile` with items:

`beta`	The values of the regression coefficient
`cdf`	The cumulative distribution function of the posterior
`profile`	The profile of the posterior

`cdf.matrix`	An imputations x steps matrix with the values of the completed-data CDFs for each beta
`profile.matrix`	An imputations x steps matrix with the values of the completed-data profiles for each beta
`call`	The function call

Author(s)

Georg Heinze und Meinhard Plonar

References

Heinze G, Ploner M, Beyea J (2013). Confidence intervals after multiple imputation: combining profile likelihood information from logistic regressions. Statistics in Medicine, to appear.

Examples

```
#generate data set with NAs
freq=c(5,2,2,7,5,4)
y<-c(rep(1,freq[1]+freq[2]), rep(0,freq[3]+freq[4]), rep(1,freq[5]), rep(0,freq[6]))
x<-c(rep(1,freq[1]), rep(0,freq[2]), rep(1,freq[3]), rep(0,freq[4]), rep(NA,freq[5]),
rep(NA,freq[6]))
toy<-data.frame(x=x,y=y)

# impute data set 5 times
set.seed(169)
toymi<-list(0)
for(i in 1:5){
  toymi[[i]]<-toy
  y1<-toymi[[i]]$y==1 & is.na(toymi[[i]]$x)
  y0<-toymi[[i]]$y==0 & is.na(toymi[[i]]$x)
  xnew1<-rbinom(sum(y1),1,freq[1]/(freq[1]+freq[2]))
  xnew0<-rbinom(sum(y0),1,freq[3]/(freq[3]+freq[4]))
  toymi[[i]]$x[y1==TRUE]<-xnew1
  toymi[[i]]$x[y0==TRUE]<-xnew0
}

# logistf analyses of each imputed data set
fit.list<-lapply(1:5, function(X) logistf(data=toymi[[X]], y~x, pl=TRUE))

# CLIP profile
xprof<-CLIP.profile(obj=fit.list, variable="x",data =toymi, keep=TRUE)
plot(xprof)

#plot as CDF
plot(xprof, "cdf")

#plot as density
plot(xprof, "density")
```

`emmeans-logistf` *Emmeans support for logistf*

Description

Support for the `emmeans` package is available. See below for an example of using `emmeans::emmeans()` with a `logistf` object.

Examples

```
data(sex2)
fit<-logistf(case ~ age+oc+vic+vicl+vis+dia, data=sex2)

emmeans::emmeans(fit, ~age+dia)
```

`flac` *FLAC - Firth's logistic regression with added covariate*

Description

`flac` implements Firth's bias-reduced penalized-likelihood logistic regression with added covariate.

Usage

```
flac(...)

## Default S3 method:
flac(
  formula,
  data,
  model = TRUE,
  control,
  modcontrol,
  weights,
  offset,
  na.action,
  pl = TRUE,
  plconf = NULL,
  ...
)

## S3 method for class 'logistf'
flac(lfobject, data, model = TRUE, ...)
```

Arguments

`...`	Further arguments passed to the method or `logistf`-call.
`formula`	A formula object, with the response on the left of the operator, and the model terms on the right. The response must be a vector with 0 and 1 or `FALSE` and `TRUE` for the outcome, where the higher value (1 or `TRUE`) is modeled.
`data`	A data frame containing the variables in the model.
`model`	If TRUE the corresponding components of the fit are returned.
`control`	Controls iteration parameter. Taken from `logistf`-object when specified. Otherwise default is `control= logistf.control()`.
`modcontrol`	Controls additional parameter for fitting. Taken from `logistf`-object when specified. Otherwise default is `logistf.mod.control()`.
`weights`	specifies case weights. Each line of the input data set is multiplied by the corresponding element of weights
`offset`	a priori known component to be included in the linear predictor
`na.action`	a function which indicates what should happen when the data contain NAs
`pl`	Specifies if confidence intervals and tests should be based on the profile penalized log likelihood (`pl=TRUE`, the default) or on the Wald method (`pl=FALSE`).
`plconf`	specifies the variables (as vector of their indices) for which profile likelihood confidence intervals should be computed. Default is to compute for all variables.
`lfobject`	A fitted `logistf` object.

Details

FLAC is a simple modification of Firth’s logistic regression which provides average predicted probabilities equal to the observed proportion of events, while preserving the ability to deal with separation. It has been described by Puhr et al (2017).

The modified score equations to estimate coefficients for Firth’s logistic regression can be interpreted as score equations for ML estimates for an augmented data set. This data set can be created by complementing each original observation i with two pseudo-observations weighted by $h_i/2$ with unchanged covariate values and with response values set to $y = 0$ and $y = 1$ respectively. The basic idea of FLAC is to discriminate between original and pseudo-observations in the alternative formulation of Firth’s estimation as an iterative data augmentation procedure. The following generic methods are available for ’ `flac`’s output object: `print`, `summary`, `coef`, `confint`, `anova`, `extractAIC`, `add1`, `drop1`,`profile`, `terms`, `nobs`, `predict`. Furthermore, forward and backward functions perform convenient variable selection. Note that anova, extractAIC, add1, drop1, forward and backward are based on penalized likelihood ratio tests.

Value

A `flac` object with components:

`coefficients`	The coefficients of the parameter in the fitted model.
`predict`	A vector with the predicted probability of each observation
`linear.predictors`	A vector with the linear predictor of each observation.

`prob`	The p-values of the specific parameters
`ci.lower`	The lower confidence limits of the parameter.
`ci.upper`	The upper confidence limits of the parameter.
`call`	The call object.
`alpha`	The significance level: 0.95
`var`	The variance-covariance-matrix of the parameters.
`loglik`	A vector of the (penalized) log-likelihood of the restricted and the full models.
`n`	The number of observations.
`formula`	The formula object.
`augmented.data`	The augmented dataset used
`df`	The number of degrees of freedom in the model.
`method`	depending on the fitting method ’Penalized ML’ or `Standard ML'.} \item{method.ci}{the method in` file likelihood’ or ‘Wald’, depending on the argument pl and plconf.
`control`	a copy of the control parameters.
`modcontrol`	a copy of the modcontrol parameters.
`terms`	the model terms (column names of design matrix).
`model`	if requested (the default), the model frame used.

Methods (by class)

- `flac(default)`: With formula and data
- `flac(logistf)`: With logistf object

References

Puhr R, Heinze G, Nold M, Lusa L, Geroldinger A (2017). Firth’s logistic regression with rare events: accurate effect estimates and predictions? Statistics in Medicine 36: 2302-2317.

See Also

`logistf()` for Firth’s bias-Reduced penalized-likelihood logistic regression.

Examples

```
#With formula and data:
data(sex2)
flac(case ~ age + oc + vic + vicl + vis + dia, sex2)

#With a logistf object:
lf <- logistf(formula = case ~ age + oc + vic + vicl + vis + dia, data = sex2)
flac(lf, data=sex2)
```

`flic` *FLIC - Firth's logistic regression with intercept correction*

Description

`flic` implements Firth's bias-reduced penalized-likelihood logistic regression with intercept correction.

Usage

```
flic(...)

## Default S3 method:
flic(
  formula,
  data,
  model = TRUE,
  control,
  modcontrol,
  weights,
  offset,
  na.action,
  ...
)

## S3 method for class 'logistf'
flic(lfobject, model = TRUE, ...)
```

Arguments

`...`	Further arguments passed to the method or `logistf`-call.
`formula`	A formula object, with the response on the left of the operator, and the model terms on the right. The response must be a vector with 0 and 1 or `FALSE` and `TRUE` for the outcome, where the higher value (1 or `TRUE`) is modeled.
`data`	If using with formula, a data frame containing the variables in the model.
`model`	If TRUE the corresponding components of the fit are returned.
`control`	Controls iteration parameter. Taken from `logistf`-object when specified. Otherwise default is `control= logistf.control()`.
`modcontrol`	Controls additional parameter for fitting. Taken from `logistf`-object when specified. Otherwise default is `logistf.mod.control()`.
`weights`	specifies case weights. Each line of the input data set is multiplied by the corresponding element of weights
`offset`	a priori known component to be included in the linear predictor
`na.action`	a function which indicates what should happen when the data contain NAs
`lfobject`	A fitted `logistf` object.

Details

FLIC is a simple modification of Firth's logistic regression which provides average predicted probabilities equal to the observed proportion of events, while preserving the ability to deal with separation.

In general the average predicted probability in Firth's logistic regression is not equal to the observed proportion of events. Because the determinant of the Fisher-Information matrix is maximized for $\pi_i = \frac{1}{2}$ it is concluded that Firth's penalization tends to push the predicted probabilities towards one-half compared with ML-estimation. FLIC first applies Firth's logistic regression and then corrects the intercept such that the predicted probabilities become unbiased while keeping all other coefficients constant. The following generic methods are available for `flic`'s output object: `print, summary, coef, confint, anova, extractAIC, add1, drop1,profile, terms, nobs, predict`. Furthermore, forward and backward functions perform convenient variable selection. Note that anova, extractAIC, add1, drop1, forward and backward are based on penalized likelihood ratio tests.

Value

A `flic` object with components:

`coefficients`	The coefficients of the parameter in the fitted model.
`predict`	A vector with the predicted probability of each observation.
`linear.predictors`	A vector with the linear predictor of each observation.
`var`	The variance-covariance-matrix of the parameters.
`prob`	The p-values of the specific parameters.
`ci.lower`	The lower confidence limits of the parameter.
`ci.upper`	The upper confidence limits of the parameter.
`call`	The call object.
`alpha`	The significance level: 0.95.
`method`	depending on the fitting method 'Penalized ML' or `Standard ML'.} \item{method.ci}{the method in` file likelihood' or 'Wald', depending on the argument pl and plconf.
`df`	The number of degrees of freedom in the model.
`loglik`	A vector of the (penalized) log-likelihood of the restricted and the full models.
`n`	The number of observations.
`formula`	The formula object.
`control`	a copy of the control parameters.
`modcontrol`	a copy of the modcontrol parameters.
`terms`	the model terms (column names of design matrix).
`model`	if requested (the default), the model frame used.

Methods (by class)

- `flic(default)`: With formula and data
- `flic(logistf)`: With logistf object

References

Puhr R, Heinze G, Nold M, Lusa L, Geroldinger A (2017). Firth's logistic regression with rare events: accurate effect estimates and predictions? Statistics in Medicine 36: 2302-2317.

See Also

`logistf` for Firth's bias-Reduced penalized-likelihood logistic regression.

Examples

```
#With formula and data:
data(sex2)
flic(case ~ age + oc + vic + vicl + vis + dia, sex2)

#With a logistf object:
lf <- logistf(formula = case ~ age + oc + vic + vicl + vis + dia, data = sex2)
flic(lf)
```

`logistf` *Firth's Bias-Reduced Logistic Regression*

Description

Implements Firth's bias-Reduced penalized-likelihood logistic regression.

Usage

```
logistf(
  formula,
  data,
  pl = TRUE,
  alpha = 0.05,
  control,
  plcontrol,
  modcontrol,
  firth = TRUE,
  init,
  weights,
  na.action,
  offset,
  plconf = NULL,
  flic = FALSE,
  model = TRUE,
  ...
)
```

Arguments

`formula`	A formula object, with the response on the left of the operator, and the model terms on the right. The response must be a vector with 0 and 1 or `FALSE` and `TRUE` for the outcome, where the higher value (1 or `TRUE`) is modeled. It is possible to include contrasts, interactions, nested effects, cubic or polynomial splines and all S features as well, e.g. Y ~ X1*X2 + ns(X3, df=4).
`data`	A data.frame where the variables named in the formula can be found, i. e. the variables containing the binary response and the covariates.
`pl`	Specifies if confidence intervals and tests should be based on the profile penalized log likelihood (`pl=TRUE`, the default) or on the Wald method (`pl=FALSE`).
`alpha`	The significance level (1-α the confidence level, 0.05 as default).
`control`	Controls iteration parameter. Default is `control= logistf.control()`
`plcontrol`	Controls Newton-Raphson iteration for the estimation of the profile likelihood confidence intervals. Default is `plcontrol= logistpl.control()`
`modcontrol`	Controls additional parameter for fitting. Default is `logistf.mod.control()`
`firth`	Use of Firth's penalized maximum likelihood (`firth=TRUE`, default) or the standard maximum likelihood method (`firth=FALSE`) for the logistic regression. Note that by specifying `pl=TRUE` and `firth=FALSE` (and probably a lower number of iterations) one obtains profile likelihood confidence intervals for maximum likelihood logistic regression parameters.
`init`	Specifies the initial values of the coefficients for the fitting algorithm
`weights`	specifies case weights. Each line of the input data set is multiplied by the corresponding element of weights
`na.action`	a function which indicates what should happen when the data contain NAs
`offset`	a priori known component to be included in the linear predictor
`plconf`	specifies the variables (as vector of their indices) for which profile likelihood confidence intervals should be computed. Default is to compute for all variables.
`flic`	If `TRUE`, intercept is altered such that the predicted probabilities become unbiased while keeping all other coefficients constant (see Puhr et al, 2017)
`model`	If TRUE the corresponding components of the fit are returned.
`...`	Further arguments to be passed to `logistf`

Details

`logistf` is the main function of the package. It fits a logistic regression model applying Firth's correction to the likelihood. The following generic methods are available for logistf's output object: `print, summary, coef, vcov, confint, anova, extractAIC, add1, drop1,profile, terms, nobs, predict`. Furthermore, forward and backward functions perform convenient variable selection. Note that anova, extractAIC, add1, drop1, forward and backward are based on penalized likelihood ratios.

Value

The object returned is of the class `logistf` and has the following attributes:

`coefficients`	the coefficients of the parameter in the fitted model.
`alpha`	the significance level (1- the confidence level) as specified in the input.
`terms`	the column names of the design matrix
`var`	the variance-covariance-matrix of the parameters.
`df`	the number of degrees of freedom in the model.
`loglik`	a vector of the (penalized) log-likelihood of the restricted and the full models.
`iter`	A vector of the number of iterations needed in the fitting process for the null and full model.
`n`	the number of observations.
`y`	the response-vector, i. e. 1 for successes (events) and 0 for failures.
`formula`	the formula object.
`call`	the call object.
`terms`	the model terms (column names of design matrix).
`linear.predictors`	a vector with the linear predictor of each observation.
`predict`	a vector with the predicted probability of each observation.
`hat.diag`	a vector with the diagonal elements of the Hat Matrix.
`conv`	the convergence status at last iteration: a vector of length 3 with elements: last change in log likelihood, max(abs(score vector)), max change in beta at last iteration.
`method`	depending on the fitting method ’Penalized ML’ or `Standard ML'.} \item{method.ci}{the method in` file likelihood’ or ‘Wald’, depending on the argument pl and plconf.
`ci.lower`	the lower confidence limits of the parameter.
`ci.upper`	the upper confidence limits of the parameter.
`prob`	the p-values of the specific parameters.
`pl.iter`	only if pl==TRUE: the number of iterations needed for each confidence limit.
`betahist`	only if pl==TRUE: the complete history of beta estimates for each confidence limit.
`pl.conv`	only if pl==TRUE: the convergence status (deviation of log likelihood from target value, last maximum change in beta) for each confidence limit.
`control`	a copy of the control parameters.
`modcontrol`	a copy of the modcontrol parameters.
`flic`	logical, is TRUE if intercept was altered such that the predicted probabilities become unbiased while keeping all other coefficients constant. According to input of logistf.
`model`	if requested (the default), the model frame used.
`na.action`	information returned by model.frame on the special handling of NAs

Author(s)

Georg Heinze and Meinhard Ploner

References

Firth D (1993). Bias reduction of maximum likelihood estimates. Biometrika 80, 27-38. Heinze G, Schemper M (2002). A solution to the problem of separation in logistic regression. Statistics in Medicine 21: 2409-2419.

Heinze G, Ploner M (2003). Fixing the nonconvergence bug in logistic regression with SPLUS and SAS. Computer Methods and Programs in Biomedicine 71: 181-187.

Heinze G, Ploner M (2004). Technical Report 2/2004: A SAS-macro, S-PLUS library and R package to perform logistic regression without convergence problems. Section of Clinical Biometrics, Department of Medical Computer Sciences, Medical University of Vienna, Vienna, Austria. http://www.meduniwien.ac.at/user/georg.heinze/techreps/tr2_2004.pdf

Heinze G (2006). A comparative investigation of methods for logistic regression with separated or nearly separated data. Statistics in Medicine 25: 4216-4226.

Puhr R, Heinze G, Nold M, Lusa L, Geroldinger A (2017). Firth's logistic regression with rare events: accurate effect estimates and predictions? Statistics in Medicine 36: 2302-2317.

Venzon DJ, Moolgavkar AH (1988). A method for computing profile-likelihood based confidence intervals. Applied Statistics 37:87-94.

See Also

`add1.logistf()`, `anova.logistf()`

Examples

```
data(sex2)
fit<-logistf(case ~ age+oc+vic+vicl+vis+dia, data=sex2)
summary(fit)
nobs(fit)
drop1(fit)
plot(profile(fit,variable="dia"))
extractAIC(fit)

fit1<-update(fit, case ~ age+oc+vic+vicl+vis)
extractAIC(fit1)
anova(fit,fit1)

data(sexagg)
fit2<-logistf(case ~ age+oc+vic+vicl+vis+dia, data=sexagg, weights=COUNT)
summary(fit2)

# simulated SNP example
set.seed(72341)
snpdata<-rbind(
  matrix(rbinom(2000,2,runif(2000)*0.3),100,20),
  matrix(rbinom(2000,2,runif(2000)*0.5),100,20))
colnames(snpdata)<-paste("SNP",1:20,"_",sep="")
```

```
snpdata<-as.data.frame(snpdata)
snpdata$case<-c(rep(0,100),rep(1,100))

fitsnp<-logistf(data=snpdata, formula=case~1, pl=FALSE)
add1(fitsnp, scope=paste("SNP",1:20,"_",sep=""), data=snpdata)
fitf<-forward(fitsnp, scope = paste("SNP",1:20,"_",sep=""), data=snpdata)
fitf
```

`logistf.control` *Control Parameters for* `logistf`

Description

Sets parameters for iterations in Firth's penalized-likelihood logistic regression.

Usage

```
logistf.control(
  maxit = 25,
  maxhs = 0,
  maxstep = 5,
  lconv = 1e-05,
  gconv = 1e-05,
  xconv = 1e-05,
  collapse = TRUE,
  fit = "NR"
)
```

Arguments

`maxit`	The maximum number of iterations
`maxhs`	The maximum number of step-halvings in one iteration. The increment of the beta vector within one iteration is divided by 2 if the new beta leads to a decrease in log likelihood.
`maxstep`	Specifies the maximum step size in the beta vector within one iteration. Set to -1 for infinite stepsize.
`lconv`	Specifies the convergence criterion for the log likelihood.
`gconv`	Specifies the convergence criterion for the first derivative of the log likelihood (the score vector).
`xconv`	Specifies the convergence criterion for the parameter estimates.
`collapse`	If `TRUE`, evaluates all unique combinations of x and y and collapses data set.
`fit`	Fitting method used. One of Newton-Raphson: "NR" or Iteratively reweighted least squares: "IRLS"

Details

`logistf.control()` is used by `logistf` and `logistftest` to set control parameters to default values. Different values can be specified, e. g., by `logistf(..., control= logistf.control(maxstep=1))`.

Value

`maxit`	The maximum number of iterations
`maxhs`	The maximum number of step-halvings in one iteration. The increment of the beta vector within one iteration is divided by 2 if the new beta leads to a decrease in log likelihood.
`maxstep`	Specifies the maximum step size in the beta vector within one iteration.
`lconv`	Specifies the convergence criterion for the log likelihood.
`gconv`	Specifies the convergence criterion for the first derivative of the log likelihood (the score vector).
`xconv`	Specifies the convergence criterion for the parameter estimates.
`collapse`	If `TRUE`, evaluates all unique combinations of x and y and collapses data set.
`fit`	Fitting method used. One of Newton-Raphson: "NR" or Iteratively reweighted least squares: "IRLS"
`call`	The function call.

Examples

```
data(sexagg)
fit2<-logistf(case ~ age+oc+vic+vicl+vis+dia, data=sexagg, weights=COUNT,
control=logistf.control(maxstep=1))
summary(fit2)
```

`logistf.mod.control` *Controls additional parameters for* `logistf`

Description

Sets parameters for `logistf` calls.

Usage

```
logistf.mod.control(tau = 0.5, terms.fit = NULL)
```

Arguments

`tau`	Penalization parameter (default = 0.5)
`terms.fit`	A numeric vector of terms to fit. Intercept has to be included if needed.

Value

`tau`	Penalization parameter (default = 0.5)
`terms.fit`	A numeric vector of terms to fit. Intercept has to be included if needed.

Examples

```
data(sexagg)
fit2<-logistf(case ~ age+oc+vic+vicl+vis+dia, data=sexagg, weights=COUNT,
modcontrol=logistf.mod.control(terms.fit=c(1,2)))
summary(fit2)
```

`logistftest` *Penalized likelihood ratio test*

Description

This function performs a penalized likelihood ratio test on some (or all) selected factors. The resulting object is of the class logistftest and includes the information printed by the proper print method.

Usage

```
logistftest(
  object,
  test,
  values,
  firth = TRUE,
  beta0,
  weights,
  control,
  modcontrol,
  ...
)
```

Arguments

`object`	A fitted `logistf` object
`test`	righthand formula of parameters to test (e.g. ~ B + D - 1). As default all parameter apart from the intercept are tested. If the formula includes -1, the intercept is omitted from testing. As alternative to the formula one can give the indexes of the ordered effects to test (a vector of integers). To test only the intercept specify test = ~ - . or test = 1.
`values`	Null hypothesis values, default values are 0. For testing the specific hypothesis B1=1, B4=2, B5=0 we specify test= ~B1+B4+B5-1 and values=c(1, 2,0).

`firth`	Use of Firth's (1993) penalized maximum likelihood (firth=TRUE, default) or the standard maximum likelihood method (firth=FALSE) for the logistic regression. Note that by specifying pl=TRUE and firth=FALSE (and probably lower number of iterations) one obtains profile likelihood confidence intervals for maximum likelihood logistic regression parameters.
`beta0`	Specifies the initial values of the coefficients for the fitting algorithm
`weights`	Case weights
`control`	Controls parameters for iterative fitting
`modcontrol`	Controls additional parameter for fitting. Default is `modcontrol` of `object`.
`...`	further arguments passed to logistf.fit

Details

This function performs a penalized likelihood ratio test on some (or all) selected factors. The resulting object is of the class logistftest and includes the information printed by the proper print method. Further documentation can be found in Heinze & Ploner (2004). In most cases, the functionality of the logistftest function is replaced by anova.logistf, which is a more standard way to perform likelihood ratio tests. However, as shown in the example below, logistftest provides some specials such as testing against non-zero values. (By the way, anova.logistf calls logistftest.

Value

The object returned is of the class logistf and has the following attributes:

`testcov`	A vector of the fixed values of each covariate; NA stands for a parameter which is not tested.
`loglik`	A vector of the (penalized) log-likelihood of the full and the restricted models. If the argument beta0 not missing, the full model isn't evaluated
`df`	The number of degrees of freedom in the model
`prob`	The p-value of the test
`call`	The call object
`method`	Depending on the fitting method 'Penalized ML' or 'Standard ML'
`beta`	The coefficients of the restricted solution

Author(s)

Georg Heinze

References

Firth D (1993). Bias reduction of maximum likelihood estimates. Biometrika 80, 27-38.

Heinze G, Ploner M (2004). Technical Report 2/2004: A SAS-macro, S-PLUS library and R package to perform logistic regression without convergence problems. Section of Clinical Biometrics, Department of Medical Computer Sciences, Medical University of Vienna, Vienna, Austria. http://www.meduniwien.ac.at/user/georg.heinze/techreps/tr2_2004.pdf

Heinze G (2006). A comparative investigation of methods for logistic regression with separated or nearly separated data. Statistics in Medicine 25: 4216-4226

Examples

```
data(sex2)
fit<-logistf(case ~ age+oc+vic+vicl+vis+dia, data=sex2)
logistftest(fit, test = ~ vic + vicl - 1, values = c(2, 0))
```

`logistpl.control` *Control Parameters for logistf Profile Likelihood Confidence Interval Estimation*

Description

Sets parameters for modified Newton-Raphson iteration for finding profile likelihood confidence intervals in Firth's penalized likelihood logistic regression

Usage

```
logistpl.control(
  maxit = 100,
  maxhs = 0,
  maxstep = 5,
  lconv = 1e-05,
  xconv = 1e-05,
  ortho = FALSE,
  pr = FALSE
)
```

Arguments

`maxit`	The maximum number of iterations
`maxhs`	The maximum number of step-halvings in one iteration. The increment of the beta vector within one iteration is divided by 2 if the new beta leads to a decrease in log likelihood.
`maxstep`	Specifies the maximum step size in the beta vector within one iteration. Set to -1 for infinite stepsize.
`lconv`	Specifies the convergence criterion for the log likelihood.
`xconv`	Specifies the convergence criterion for the parameter estimates.
`ortho`	Requests orthogonalization of variable for which confidence intervals are computed with respect to other covariates
`pr`	Request rotation of the matrix spanned by the covariates

Details

`logistpl.control()` is used by `logistf` to set control parameters to default values when computing profile likelihood confidence intervals. Different values can be specified, e. g., by `logistf(..., control= logistf.control(maxstep=1))`.

Value

`maxit`	The maximum number of iterations
`maxhs`	The maximum number of step-halvings in one iteration. The increment of the beta vector within one iteration is divided by 2 if the new beta leads to a decrease in log likelihood.
`maxstep`	Specifies the maximum step size in the beta vector within one iteration.
`lconv`	Specifies the convergence criterion for the log likelihood.
`xconv`	Specifies the convergence criterion for the parameter estimates.
`ortho`	specifies if orthogonalization is requested.
`pr`	specifies if rotation is requested

Author(s)

Georg Heinze

Examples

```
data(sexagg)
fit2<-logistf(case ~ age+oc+vic+vicl+vis+dia, data=sexagg, weights=COUNT,
    plcontrol=logistpl.control(maxstep=1))
summary(fit2)
```

`plot.logistf.profile` `plot` *Method for* `logistf` *Likelihood Profiles*

Description

Provides the plot method for objects created by `profile.logistf` or `CLIP.profile`

Usage

```
## S3 method for class 'logistf.profile'
plot(
  x,
  type = "profile",
  max1 = TRUE,
  colmain = "black",
  colimp = "gray",
  plotmain = T,
  ylim = NULL,
  ...
)
```

Arguments

`x`	A `profile.logistf` object
`type`	Type of plot: one of c("profile", "cdf", "density")
`max1`	If `type="density"`, normalizes density to maximum 1
`colmain`	Color for main profile line
`colimp`	color for completed-data profile lines (for `logistf.profile` objects that also carry the `CLIP.profile` class attribute)
`plotmain`	if `FALSE`, suppresses the main profile line (for `logistf.profile` objects that also carry the `CLIP.profile` class attribute)
`ylim`	Limits for the y-axis
`...`	Further arguments to be passed to `plot()`.

Details

The plot method provides three types of plots (profile, CDF, and density representation of a profile likelihood). For objects generated by CLIP.profile, it also allows to show the completed-data profiles along with the pooled profile.

Value

The function is called for its side effects

Author(s)

Georg Heinze und Meinhard Ploner

References

Heinze G, Ploner M, Beyea J (2013). Confidence intervals after multiple imputation: combining profile likelihood information from logistic regressions. Statistics in Medicine, to appear.

Examples

```
data(sex2)
fit<-logistf(case ~ age+oc+vic+vicl+vis+dia, data=sex2)
plot(profile(fit,variable="dia"))
plot(profile(fit,variable="dia"), "cdf")
plot(profile(fit,variable="dia"), "density")

#generate data set with NAs
freq=c(5,2,2,7,5,4)
y<-c(rep(1,freq[1]+freq[2]), rep(0,freq[3]+freq[4]), rep(1,freq[5]), rep(0,freq[6]))
x<-c(rep(1,freq[1]), rep(0,freq[2]), rep(1,freq[3]), rep(0,freq[4]), rep(NA,freq[5]),
   rep(NA,freq[6]))
toy<-data.frame(x=x,y=y)

# impute data set 5 times
set.seed(169)
```

```
toymi<-list(0)
for(i in 1:5){
   toymi[[i]]<-toy
   y1<-toymi[[i]]$y==1 & is.na(toymi[[i]]$x)
   y0<-toymi[[i]]$y==0 & is.na(toymi[[i]]$x)
   xnew1<-rbinom(sum(y1),1,freq[1]/(freq[1]+freq[2]))
   xnew0<-rbinom(sum(y0),1,freq[3]/(freq[3]+freq[4]))
   toymi[[i]]$x[y1==TRUE]<-xnew1
   toymi[[i]]$x[y0==TRUE]<-xnew0
 }

# logistf analyses of each imputed data set
fit.list<-lapply(1:5, function(X) logistf(data=toymi[[X]], y~x, pl=TRUE))

# CLIP profile
xprof<-CLIP.profile(obj=fit.list, variable="x", data=toymi, keep=TRUE)
plot(xprof)

#plot as CDF
plot(xprof, "cdf")

#plot as density
plot(xprof, "density")
```

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

Description

Obtains predictions from a fitted `flac` object.

Usage

```
## S3 method for class 'flac'
predict(
  object,
  newdata,
  type = c("link", "response", "terms"),
  se.fit = FALSE,
  ...
)
```

Arguments

`object`	A fitted object of class `flac`.
`newdata`	Optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.

`type`	The type of prediction required. The default is on the scale of the linear predictors. The alternative `response` gives the predicted probabilities. Type `terms` returns a matrix with the fitted values of each term in the formula on the linear predictor scale.
`se.fit`	If `TRUE`(default = `FALSE`) standard errors are computed.
`...`	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

A vector or matrix of predictions.

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

Description

Obtains predictions from a fitted `flic` object.

Usage

```
## S3 method for class 'flic'
predict(
  object,
  newdata,
  type = c("link", "response", "terms"),
  se.fit = FALSE,
  ...
)
```

Arguments

`object`	A fitted object of class `flic`.
`newdata`	Optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
`type`	The type of prediction required. The default is on the scale of the linear predictors. The alternative `response` gives the predicted probabilities. Type `terms` returns a matrix with the fitted values of each term in the formula on the linear predictor scale.
`se.fit`	If `TRUE`(default = `FALSE`) standard errors are computed.
`...`	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

A vector or matrix of predictions

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

Description

Obtains predictions from a fitted `logistf` object.

Usage

```
## S3 method for class 'logistf'
predict(
  object,
  newdata,
  type = c("link", "response", "terms"),
  flic = FALSE,
  se.fit = FALSE,
  reference,
  na.action = na.pass,
  ...
)
```

Arguments

`object`	A fitted object of class `logistf`.
`newdata`	Optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
`type`	The type of prediction required. The default is on the scale of the linear predictors. The alternative `response` gives the predicted probabilities. Type `terms` returns a matrix with the fitted values of each term in the formula on the linear predictor scale.
`flic`	If `TRUE`(default = `FALSE`), predictions are computed with intercept correction.
`se.fit`	If `TRUE`(default = `FALSE`) standard errors are computed.
`reference`	A named vector of reference values for each variable for `type="terms"`.
`na.action`	Function determining what should be done with missing values in newdata. The default is to predict NA.
`...`	further arguments passed to or from other methods.

Details

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

Value

A vector or matrix of predictions.

`profile.logistf` *Compute Profile Penalized Likelihood*

Description

Evaluates the profile penalized likelihood of a variable based on a logistf model fit

Usage

```
## S3 method for class 'logistf'
profile(
  fitted,
  which,
  variable,
  steps = 100,
  pitch = 0.05,
  limits,
  alpha = 0.05,
  firth = TRUE,
  legends = TRUE,
  control,
  plcontrol,
  ...
)
```

Arguments

`fitted`	An object fitted by `logistf`
`which`	A righthand formula to specify the variable for which the profile should be evaluated, e.g., which=~X).
`variable`	Alternatively to which, a variable name can be given, e.g., variable="X"
`steps`	Number of steps in evaluating the profile likelihood
`pitch`	Alternatively to steps, one may specify the step width in multiples of standard errors
`limits`	Lower and upper limits of parameter values at which profile likelihood is to be evaluated
`alpha`	The significance level (1-α the confidence level, 0.05 as default).

`firth`	Use of Firth's penalized maximum likelihood (`firth=TRUE`, default) or the standard maximum likelihood method (`firth=FALSE`) for the logistic regression.
`legends`	legends to be included in the optional plot
`control`	Controls Newton-Raphson iteration. Default is `control= logistf.control(maxstep,maxit, maxhs, lconv, gconv, xconv)`
`plcontrol`	Controls Newton-Raphson iteration for the estimation of the profile likelihood confidence intervals. Default is `plcontrol= logistpl.control(maxstep, maxit, maxhs, lconv, xconv, ortho, pr)`
`...`	Further arguments to be passed.

Value

An object of class `logistf.profile` with the following items:

`beta`	Parameter values at which likelihood was evaluated
`stdbeta`	Parameter values divided by standard error
`profile`	profile likelihood, standardized to 0 at maximum of likelihood. The values in profile are given as minus χ^2
`loglik`	Unstandardized profile likelihood
`signed.root`	signed root (z) of χ^2 values (negative for values below the maximum likelihood estimate, positive for values above the maximum likelihood estimate)
`cdf`	profile likelihood expressed as cumulative distribution function, obtained as $\Phi(z)$, where Φ denotes the standard normal distribution function.

References

Heinze G, Ploner M, Beyea J (2013). Confidence intervals after multiple imputation: combining profile likelihood information from logistic regressions. Statistics in Medicine, to appear.

Examples

```
data(sex2)
fit<-logistf(case ~ age+oc+vic+vicl+vis+dia, data=sex2)
plot(profile(fit,variable="dia"))
plot(profile(fit,variable="dia"), "cdf")
plot(profile(fit,variable="dia"), "density")
```

`PVR.confint` *Pseudo Variance Modification of Rubin's Rule*

Description

The pseudo-variance modification proposed by Heinze, Ploner and Beyea (2013) provides a quick way to adapt Rubin's rules to situations of a non-normal distribution of a regression coefficient. However, the approxiation is less accurate than that of the CLIP method.

Usage

```
PVR.confint(obj, variable, skewbeta = FALSE)
```

Arguments

`obj`	A fitted `logisf` object
`variable`	The variable(s) to compute the PVR confidence intervals, either provided as names or as numbers
`skewbeta`	If `TRUE`, incorporates information on the skewness of the parameter estimates across the imputed data sets.

Details

The pseudo-variance modification computes a lower and an upper pseudo-variance, which are based on the distance between profile likelihood limits and the parameter estimates. These are then plugged into the usual Rubin's rules method of variance combination

Value

An object of class `PVR.confint` with items:

`estimate`	the pooled parameter estimate(s) (the average across completed-data estimates)
`ci`	the confidence intervals based on the PVR method
`lower.var`	the lower pseudo-variance(s)
`upper.var`	the upper pseudo-variance(s)
`conflev`	the confidence level: this is determined by the confidence level (1-alpha) used in the input fit objects
`call`	the function call
`variable`	the variable(s) for which confidence intervals were computed

Author(s)

Georg Heinze

References

Heinze G, Ploner M, Beyea J (2013). Confidence intervals after multiple imputation: combining profile likelihood information from logistic regressions. Statistics in Medicine, to appear.

Examples

```
#generate data set with NAs
freq=c(5,2,2,7,5,4)
y<-c(rep(1,freq[1]+freq[2]), rep(0,freq[3]+freq[4]), rep(1,freq[5]), rep(0,freq[6]))
x<-c(rep(1,freq[1]), rep(0,freq[2]), rep(1,freq[3]), rep(0,freq[4]), rep(NA,freq[5]),
   rep(NA,freq[6]))
toy<-data.frame(x=x,y=y)

# impute data set 5 times
set.seed(169)
toymi<-list(0)
for(i in 1:5){
  toymi[[i]]<-toy
  y1<-toymi[[i]]$y==1 & is.na(toymi[[i]]$x)
  y0<-toymi[[i]]$y==0 & is.na(toymi[[i]]$x)
  xnew1<-rbinom(sum(y1),1,freq[1]/(freq[1]+freq[2]))
  xnew0<-rbinom(sum(y0),1,freq[3]/(freq[3]+freq[4]))
  toymi[[i]]$x[y1==TRUE]<-xnew1
  toymi[[i]]$x[y0==TRUE]<-xnew0
  }

# logistf analyses of each imputed data set
fit.list<-lapply(1:5, function(X) logistf(data=toymi[[X]], y~x, pl=TRUE))

# CLIP confidence limits
PVR.confint(obj=fit.list)
```

`sex2` *Urinary Tract Infection in American College Students*

Description

This data set deals with urinary tract infection in sexually active college women, along with covariate information on age an contraceptive use. The variables are all binary and coded in 1 (condition is present) and 0 (condition is absent).

Usage

```
sex2
```

Format

sex2: a data.frame containing 239 observations

case urinary tract infection, the study outcome variable

age >= 24 years

dia use of diaphragm

oc use of oral contraceptive

vic use of condom

vicl use of lubricated condom

vis use of spermicide

Source

`https://cytel.com/`

References

Cytel Inc., (2010) LogXact 9 user manual, Cambridge, MA:Cytel Inc

---

`sexagg` *Urinary Tract Infection in American College Students*

Description

This data set deals with urinary tract infection in sexually active college women, along with covariate information on age an contraceptive use. The variables are all binary and coded in 1 (condition is present) and 0 (condition is absent): case (urinary tract infection, the study outcome variable), age (>= 24 years), dia (use of diaphragm), oc (use of oral contraceptive), vic (use of condom), vicl (use of lubricated condom), and vis (use of spermicide).

Usage

```
sexagg
```

Format

sexagg: an aggregated data.frame containing 31 observations with case weights (COUNT).

case urinary tract infection, the study outcome variable

age >= 24 years

dia use of diaphragm

oc use of oral contraceptive

vic use of condom

vicl use of lubricated condom

vis use of spermicide

Source

https://cytel.com/

References

Cytel Inc., (2010) LogXact 9 user manual, Cambridge, MA:Cytel Inc

Index