Package: ameras (via r-universe)

October 30, 2026

Title Analyze Multiple Exposure Realizations in Association Studies

Version 0.5.2.9000

Depends R (>= 4.1.0), stats, nimble

LazyData true

LazyDataCompression xz

Suggests knitr, nnet, rmarkdown, survival, testthat (>= 3.0.0), future, future.apply, ggplot2, dplyr, tidyr, scales, patchwork

Description Analyze association studies with multiple realizations of a noisy or uncertain exposure. These can be obtained from e.g. a two-dimensional Monte Carlo dosimetry system (Simon et al 2015 <doi:10.1667/RR13729.1>) to characterize exposure uncertainty. The implemented methods are regression calibration (Carroll et al. 2006 <doi:10.1201/9781420010138>), extended regression calibration (Little et al. 2023 <doi:10.1038/s41598-023-42283-y>), Monte Carlo maximum likelihood (Stayner et al. 2007 <doi:10.1667/RR0677.1>), frequentist model averaging (Kwon et al. 2023 <doi:10.1371/journal.pone.0290498>), and Bayesian model averaging (Kwon et al. 2016 <doi:10.1002/sim.6635>). Supported model families are Gaussian, binomial, multinomial, Poisson, proportional hazards, and conditional logistic.

License MIT + file LICENSE

Imports Rcpp (>= 1.0.10), RcppEigen, coda, numDeriv, mvtnorm, methods, MCMCvis, tidyselect, lifecycle, graphics, grDevices

LinkingTo Rcpp, RcppEigen

NeedsCompilation yes

VignetteBuilder knitr

Config/testthat/edition 3

Author Sander Roberti [aut, cre] (ORCID: <https://orcid.org/0000-0002-6275-7442>), William Wheeler [aut], Deukwoo Kwon [aut] (ORCID: <https://orcid.org/0000-0001-5376-5320>), Ruth Pfeiffer [ctb]

(ORCID: <https://orcid.org/0000-0001-7791-2698>), U.S. National Cancer Institute [cph, fnd]

Maintainer Sander Roberti <statspackages@sanderroberti.nl>

URL https://ameras.sanderroberti.com, https://github.com/sanderroberti/ameras

BugReports https://github.com/sanderroberti/ameras/issues

Config/pak/sysreqs cmake libglpk-dev make libuv1-dev libxml2-dev

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

Date/Publication 2026-10-30 16:28:26 UTC

RemoteUrl https://github.com/sanderroberti/ameras

RemoteRef HEAD

RemoteSha 01b2352bb4f958eedca445a02b8beabe985ac6e2

Contents

ameras-package . . . 3
ameras . . . 3
ameras_simulated . . . 11
coef.amerasfit . . . 12
confint.amerasfit . . . 13
convergence . . . 16
dose_lrt . . . 18
ecdfplot . . . 19
included_realizations . . . 20
plot.amerasfit . . . 22
print.amerasfit . . . 24
residuals.amerasfit . . . 25
Rhat . . . 28
summary.amerasfit . . . 29
traceplot . . . 31
transform1 . . . 32
transform1.inv . . . 33
transform1.jacobian . . . 34
vcov.amerasfit . . . 34

Index **36**

`ameras-package` *Analyze Multiple Exposure Realizations in Association Studies*

Description

Analyze association studies with multiple realizations of a noisy or uncertain exposure. These can be obtained from e.g. a two-dimensional Monte Carlo dosimetry system (Simon et al 2015 <doi:10.1667/RR13729.1>) to characterize exposure uncertainty. Methods include regression calibration (Carroll et al. 2006 doi:10.1201/9781420010138), extended regression calibration (Little et al. 2023 doi:10.1038/s4159802342283y), Monte Carlo maximum likelihood (Stayner et al. 2007 doi:10.1667/RR0677.1), frequentist model averaging (Kwon et al. 2023 doi:10.1371/journal.pone.0290498), and Bayesian model averaging (Kwon et al. 2016 doi:10.1002/sim.6635). Supported model families are Gaussian, binomial, multinomial, Poisson, proportional hazards, and conditional logistic.

Details

The function used to fit models is `ameras`. To attach confidence/credible intervals, use the method `confint`. To visualize the exposure uncertainty in the dose realizations, use `ecdfplot`.

Author(s)

Sander Roberti <statspackages@sanderroberti.nl>, William Wheeler <WheelerB@imsweb.com>, Ruth Pfeiffer <pfeiffer@mail.nih.gov>, and Deukwoo Kwon <DKwon@uams.edu>

References

Roberti, S., Kwon D., Wheeler W., Pfeiffer R. (in preparation). ameras: An R Package to Analyze Multiple Exposure Realizations in Association Studies

`ameras` *Analyze Multiple Exposure Realizations*

Description

Fit regression models that account for exposure uncertainty using multiple exposure realizations. Supported outcome models are Gaussian, binomial, Poisson, proportional hazards, multinomial, and conditional logistic regression. Estimation methods include regression calibration (RC), extended regression calibration (ERC), Monte Carlo maximum likelihood (MCML), frequentist model averaging (FMA), and Bayesian model averaging (BMA).

Usage

```
ameras(formula=NULL, data, family="gaussian", methods="RC",
  Y=NULL, dosevars=NULL, doseRRmod=NULL, deg=NULL,
  M=NULL, X=NULL, offset=NULL, entry=NULL, exit=NULL,
  setnr=NULL,
  CI=NULL, params.profCI=NULL,
  maxit.profCI=NULL, tol.profCI=NULL,
  transform=NULL,
  transform.jacobian=NULL, inpar=NULL, loglim=1e-30, MFMA=100000,
  future.chunk.size.FMA=NULL,
  prophaz.numints.BMA=10, ERRprior.BMA="doubleexponential", nburnin.BMA=5000,
  niter.BMA=20000, nchains.BMA=2, thin.BMA=10, included.realizations.BMA=NULL,
  included.replicates.BMA=NULL, optim.method="Nelder-Mead", control=NULL,
  keep.data=TRUE, na.action=getOption("na.action"), ... )
```

Arguments

formula	an object of class "formula" containing the model specification. See Details.
data	input data frame.
family	outcome model family: "gaussian", "binomial", "poisson", "prophaz", "multinomial" or "clogit" (default "gaussian").
methods	character vector of one or multiple methods to apply. Options: "RC", "ERC", "MCML", "FMA", "BMA" (default "RC").
Y	**Deprecated**. Use the formula interface instead. Name or column index of the outcome variable for linear, binomial, Poisson, multinomial and conditional logistic models, or event indicator variable for the proportional hazards model.
dosevars	**Deprecated**. Use the formula interface instead. Names or column indices of exposure realization vectors.
doseRRmod	**Deprecated**. Use the formula interface instead. The functional form of the dose-response relationship; options are exponential RR ("EXP"), linear ERR ("ERR"), or linear-exponential RR ("LINEXP") (default "ERR").
deg	**Deprecated**. Use the formula interface instead. For doseRRmod="ERR" and doseRRmod="EXP", whether to fit a linear (deg=1) or linear-quadratic (deg=2) dose-response model (default linear).
M	**Deprecated**. Use the formula interface instead. Names or column indices of binary effect modifying variables (optional).
X	**Deprecated**. Use the formula interface instead. Names or column indices of other covariates (optional).
offset	**Deprecated**. Use the formula interface instead. Name or column index of offset variable for Poisson regression (optional).
entry	**Deprecated**. Use the formula interface instead. Name or column index of left truncation time variable for proportional hazards regression (optional).
exit	**Deprecated**. Use the formula interface instead. Name or column index of exit time variable, required when family=prophaz.

`setnr` **Deprecated**. Use the formula interface instead. Name or column index of integer-valued matched set variable, required when `family="clogit"`.

`CI` **Deprecated**. Use confint() to compute confidence intervals instead. Method for calculation of 95% confidence or credible intervals (see Details). For RC, ERC, and MCML, options are `"wald.orig"`, `"wald.transformed"`, `"proflik"` (default `"proflik"`). For FMA and BMA, options are `"percentile"` and `"hpd"` (default `"percentile"`). If `methods` contains at least one of RC, ERC, and MCML and at least one of FMA and BMA, `CI` must be length 2 and specify one method for RC, ERC, and MCML, and one for FMA and BMA (see Details).

`params.profCI` **Deprecated**. Use confint() to compute confidence intervals instead. When `CI="proflik"`, whether to obtain profile-likelihood CIs for all parameters (`"all"`) or only dose-related parameters (`"dose"`, default).

`maxit.profCI` **Deprecated**. Use confint() to compute confidence intervals instead. Maximum iterations for determining profile-likelihood CIs; passed to `uniroot` (default 20).

`tol.profCI` **Deprecated**. Use confint() to compute confidence intervals instead. Tolerance for determining profile-likelihood CIs; passed to `uniroot` (default `1e-2`).

`transform` function for parameter transformation during optimization (see Details).

`transform.jacobian`
Jacobian of the transformation function (see Details).

`inpar` vector of initial values for log-likelihood optimization (optional).

`loglim` parameter used in likelihood computations to avoid taking the log of very small or negative numbers via `log(max(x, loglim))` (default `1e-30`).

`MFMA` number of samples for `"FMA"` to compute estimates and CIs (default 100,000).

`future.chunk.size.FMA`
optional positive number controlling the average number of FMA dose realizations per future. When `NULL` (the default), approximately one future per available worker is used. Parallel execution is controlled by the user's current `future::plan()`; the default plan is sequential.

`prophaz.numints.BMA`
for `methods="BMA"` with `family="prophaz"`, the number of subintervals with constant baseline hazard (default 10). Cut points are determined based on quantiles of the event time distribution among cases.

`ERRprior.BMA` prior for dose-related parameters when `doseRRmod="ERR"` or `"LINEXP"` and `methods="BMA"`. Options: `"truncated_normal"`, `"truncated_horseshoe"`, `"truncated_doubleexponential"`, `"normal"`, `"horseshoe"`, `"doubleexponential"`, see Details (default `"doubleexponential"`).

`nburnin.BMA` number of MCMC burn-in iterations for BMA (default 5,000).

`niter.BMA` number of MCMC iterations per chain for BMA (default 20,000).

`nchains.BMA` number of MCMC chains for BMA (default 2).

`thin.BMA` thinning rate for BMA (default 10).

`included.realizations.BMA`
indices of exposure realizations used in BMA (defaults to all realizations).

`included.replicates.BMA`
Deprecated. Use `included.realizations.BMA` instead. Indices of exposure realizations used in BMA (defaults to all realizations).

optim.method	method used for optimization by optim. Options are "Nelder-Mead" and "BFGS". When using Nelder-Mead, a second optimization with BFGS is normally run to refine the fit. If this automatic BFGS refinement fails because its finite-difference evaluations are non-finite, the Nelder-Mead solution is retained and checked using the usual Hessian and convergence diagnostics. A warning is emitted if the selected final optimizer result has a nonzero convergence code.
control	control list passed to optim (default list(reltol=1e-10)).
keep.data	whether to attach data to the output object (default TRUE). When the data object is large, keep.data can be set to FALSE to preserve memory. The attached data is used to compute profile likelihood confidence intervals, but can also be supplied separately when keep.data=FALSE. See confint.
na.action	function or function name controlling missing-value handling for model inputs after formula terms have been expanded. By default, getOption("na.action") is used, typically na.omit. Use na.fail to reject missing model inputs explicitly, or na.pass to skip row removal; missing model inputs left by na.pass will generally cause an error during input checks or model fitting. na.exclude is accepted for fitting, but residuals and diagnostic plots are returned for the fitted rows rather than padded back to the originally supplied row count.
...	other arguments, passed to functions such as transform.

Details

Missing values in model inputs are handled by na.action after formula terms have been expanded. Thus missing values in raw variables and in derived covariate columns, such as factor contrasts, interactions, and spline bases, are handled consistently before model fitting. The omitted-row action is stored on the fitted object. When keep.data=FALSE and data are supplied later to methods such as confint(), residuals(), and plot(), the same missing-value policy is reapplied. Unlike lm and glm, na.exclude does not pad residuals or diagnostic plots back to the originally supplied row count; those outputs are computed on the fitted rows.

When keep.data=FALSE, later calls that use supplied data apply the same expanded covariate columns as the original fit. If covariates appear poorly scaled or ill-conditioned, ameras emits a diagnostic warning; centering or scaling continuous covariates such as calendar year can improve numerical optimization.

For conditional logistic regression, matched sets of size 1 and matched sets with no cases are excluded, and matched sets with more than one case are not currently supported. For proportional hazards regression with entry times, rows with entry == exit are excluded because they have zero follow-up time under the entry < t <= exit risk-set convention.

A transformation can be used to reparametrize parameters for optimization, such that the likelihoods are evaluated at transform(parameters), where parameters are unconstrained. This should be specified when fitting linear excess relative risk and linear-exponential models to ensure nonnegative odds/risk/hazard. The included function transform1 applies an exponential transformation to the desired parameters, see ?transform1. When supplying a function to transform, this should be a function of the full parameter vector, returning a full (transformed) parameter vector. In particular, the full parameter vector contains parameters in the following order: $\alpha_0, \boldsymbol{\alpha}, \beta_1, \beta_2, \boldsymbol{\beta}_{m1}, \boldsymbol{\beta}_{m2}, \sigma$, where $\boldsymbol{\alpha}$, $\boldsymbol{\beta}_{m1}$ and $\boldsymbol{\beta}_{m2}$ can be vectors, with lengths matching $\boldsymbol{X}$ and $\boldsymbol{M}$, respectively. σ is only included for the linear model (Gaussian family), and no intercept is included for the proportional hazards and conditional logistic models. For the multinomial model, the full parameter

vector is the concatenation of $Z-1$ parameter vectors in the order as given above, where Z is the number of outcome categories, with the last category chosen as the referent category. See `vignette("transformations", package="ameras")` for an example of how to specify a custom transformation function.

When no transformation is specified and the linear ERR model is used, `transform1` is used for ERR parameters β_1 and β_2 by default, with lower limits $-1/max(D)$ for β_1 in the linear dose-response and $(0, -1/max(D^2))$ for (β_1, β_2) in the linear-quadratic dose-response, respectively. For the linear-exponential model, a lower limit of 0 is used for β_1, and no transformation is used for β_2. If reference-plus-contrast effect modifiers are specified, no transformation is used for modifier contrast parameters. With subgroup-specific modifier coding, the default transformation is applied to the subgroup dose-effect parameters. For ERR and linear-exponential models with effect modifiers, subgroup-specific coding can therefore be more numerically robust than reference-plus-contrast coding, because bounds are applied directly to each subgroup dose-effect parameter. When negative RRs are obtained during optimization, an error will be generated and a different transformation or bounds should be used. All output is returned in the original parametrization. The Jacobian of the transformation (`transform.jacobian`) is required when using a transformation. For `transform1`, the Jacobian is given by `transform1.jacobian`. No transformations are used in BMA, and FMA is applied on the parameters using the parametrization as given above with variances obtained using the delta method with the provided Jacobian function.

For BMA, a prior distribution for exposure-response parameters can be chosen when using linear or linear-exponential exposure-response model. The options are normal, horseshoe, and double exponential priors, and the same priors truncated at 0 to yield positive values. In particular:

- Normal: $\beta_j \sim N(0, 1000)$ for all exposure-response parameters β_j
- Horseshoe (shrinkage prior): $\tau \sim \text{Cauchy}(0,1)^+; \lambda_j \sim \text{Cauchy}(0,1)^+; \beta_j \sim N(0, \tau^2\lambda_j^2)$. Here τ is shared across all parameters
- Double exponential (shrinkage prior): $\lambda_j \sim \text{Cauchy}(0,1)^+; \beta_j \sim \text{DoubleExponential}(0, \lambda_j)$

For all other parameters, and when using the exponential exposure-response model or the Gaussian outcome family, the prior is $N(0, 1000)$. For the parameter σ in the Gaussian family, this prior is truncated at 0. When subgroup-specific modifier coding is used in BMA, priors and MCMC sampling remain on the internal reference-plus-contrast scale; reported coefficients, variance matrices, samples, diagnostics, and sample-based intervals are converted to the subgroup-specific scale.

Because the proportional hazards model is not available in `nimble`, `ameras` uses a piecewise constant baseline hazard for Bayesian model averaging. The interval `min(entry), max(exit))` is divided into `prophaz.numints.BMA` subintervals with cutpoints obtained as quantiles of the distribution of event times among cases, and a baseline hazard parameter is estimated for each subinterval.

Value

The output is an object of class `amerasfit`. General components are `call` (the function call to `ameras`), `formula` (the formula object specifying the model), `num.rows` (the number of rows used for fitting after applying `na.action` and any family-specific filtering), `num.rows.original` (the number of rows supplied before applying `na.action`), `na.action` (the omitted-row object returned by the missing-value action, when applicable), `num.realizations` (the number of dose realizations provided), `transform` (the used transformation function, if applicable), `transform.jacobian` (the used Jacobian function for the transformation, if applicable), `other.args` (any other arguments

passed to ...), `model` (a list containing the specified model components parsed from the formula, including `model$data`, which is either the data frame used for model fitting when `keep.data=TRUE` or `NULL` otherwise), and `CI.computed` (logical, whether confidence intervals have been attached by `confint`).

For each method supplied to `methods`, the output contains a list with components:

`coefficients` named vector of model coefficients.

`sd` named vector of standard deviations.

`vcov` covariance matrix for the full parameter vector. Based on the observed information (negative second derivative of log-likelihood) for RC, ERC, and MCML, and on the obtained samples for FMA and BMA.

`timing` list with CPU and elapsed timings for fitting, confidence interval computation, and their total. CPU time is used for printed runtime summaries.

For RC, ERC, and MCML the following additional output is included:

`optim` a list object with results returned by optim. Components are `par` (raw parameters before applying a transformation if applicable), `hessian` (Hessian matrix for `par`), `convergence` (convergence code with 0 indicating convergence and a nonzero value indicating that the optimizer did not converge; code 1 indicates that the maximal number of iterations was reached), and `counts` (the number of likelihood function evaluations used during optimization).

`loglik` log-likelihood value at the optimum.

For RC and ERC, the output additionally contains:

`ERC` logical, whether the output is for ERC (`ERC=TRUE`) or RC (`ERC=FALSE`).

For BMA the output additionally contains:

`samples` MCMC posterior samples, as obtained from `nimble`. With `nchains.BMA > 1`, this is a list with `nchains.BMA` components, each a named matrix with the samples from one chain in its rows and columns corresponding to model parameters. With `nchains.BMA = 1`, this is a single named matrix.

`Rhat` data frame with two columns, `Rhat` and `n.eff`. The first column contains the Gelman-Rubin statistics $\hat{R} \geq 1$ that can be used to assess convergence of MCMC chains. A value of 1 indicates good convergence and values > 1.05 indicate poor convergence. The effective sample size `n.eff` is a measure of how many independent samples the auto-correlated MCMC samples correspond to. A low effective sample size indicates high correlations and/or poor mixing. For single-chain BMA fits, these diagnostics cannot be estimated and are returned as `NA`.

`included.realizations`
indices of realization exposures that were included to obtain the results.

`prophaz.timepoints`
for `family="prophaz"`, time points defining the intervals on which the estimated baseline hazard is constant; these are `prophaz.numints.BMA + 1` time points covering the interval `(min(entry), max(exit))`, based on quantiles among observed event times. See Details.

Finally, for FMA the output additionally contains:

`samples` data frame with samples generated from the normal distributions associated with parameters estimated for each dose realization.

`weights` vector of weights corresponding to the models fit to each included realization

`included.samples` the total number of samples included.

`included.realizations` indices of realization exposures that were included to obtain results. Fits without a valid variance estimate (i.e., non-invertible Hessian or inverse that is not positive definite) or that reach the maximal number of iterations without convergence are filtered out and not used to obtain results.

The class `amerasfit` supports the methods `print`, `coef`, `confint`, `summary`, `summary_table`, `vcov`, `residuals`, `plot`, `traceplot`, `convergence`, `Rhat`, and `included_realizations`.

Formula Syntax

Models are specified through formulas of the form `Y ~ dose(dose_expression, model="ERR", deg=1, modifier=~M1+M2) + X1 + X2`. Exactly one `dose()` term is required. Its unnamed `dose_expression` argument selects the exposure realization columns using `eval_select` syntax. For example, use `D1:D1000` for sequentially named columns or `all_of(dosevars)` for a character vector of column names.

The recognized named arguments inside `dose()` are `model`, `deg`, and `modifier`; their names must be written exactly. For non-Gaussian families, `model` selects the exponential relative risk model (`"EXP"`), linear-exponential model (`"LINEXP"`), or linear excess relative risk model (`"ERR"`, the default). For exponential and linear ERR models, `deg=1` fits a linear exposure effect and `deg=2` adds a quadratic term.

The optional `modifier` argument uses a one-sided formula. Use `modifier=~M` for reference-plus-contrast coding and `modifier=~0+M` or `modifier=~M-1` for subgroup-specific dose coefficients. Modifier variables may be binary numeric, logical, or factor with the first level used as the reference category. Interactions, transformed modifier terms, continuous numeric modifiers, character modifiers, and no-intercept coding with multiple modifier variables are not supported.

The special terms are parsed more narrowly than ordinary model-formula terms. For non-survival models, the response must be a single, untransformed data column. The argument to `strata()` for conditional logistic regression and the argument to `offset()` for Poisson regression must each be one untransformed data column. Proportional hazards models require `Surv(exit, status)` or `Surv(entry, exit, status)`, with each argument naming a data column. Expressions such as `offset(log(person_time))` or `strata(interaction(center, set))` are therefore not supported; create the derived column in `data` first.

Additional right-hand-side covariates are expanded with `model.matrix` and may use standard formula features such as factor contrasts, interactions, `I()` terms, and basis functions. For example, `X1*X2`, `I(age^2)`, `splines::ns(age, df=3)`, and `splines::bs(age, df=4)` are supported. These terms must preserve one value per input row; terms that drop rows internally, such as `stats::na.omit(X)`, are not supported. Removing the overall intercept with `-1` or `+0` is also not supported.

Supported Outcome Models

For the Gaussian family,

$$Y_i \sim N(\mu_i, \sigma^2),$$

with

$$\mu_i = \alpha_0 + \boldsymbol{X}_i^T \boldsymbol{\alpha} + \beta_1 D_i + \beta_2 D_i^2 + \boldsymbol{M}_i^T \boldsymbol{\beta}_{m1} D_i + \boldsymbol{M}_i^T \boldsymbol{\beta}_{m2} D_i^2.$$

Here $\boldsymbol{X}_i$ are covariates, D_i is the exposure with measurement error, and $\boldsymbol{M}_i$ are effect modifier design columns. The quadratic exposure terms and effect modification are optional.

For non-Gaussian families, the supported main-exposure relative risk models are the exponential model

$$RR_i = \exp(\beta_1 D_i + \beta_2 D_i^2 + \boldsymbol{M}_i^T \boldsymbol{\beta}_{m1} D_i + \boldsymbol{M}_i^T \boldsymbol{\beta}_{m2} D_i^2),$$

the linear excess relative risk (ERR) model

$$RR_i = 1 + \beta_1 D_i + \beta_2 D_i^2 + \boldsymbol{M}_i^T \boldsymbol{\beta}_{m1} D_i + \boldsymbol{M}_i^T \boldsymbol{\beta}_{m2} D_i^2,$$

and the linear-exponential model

$$RR_i = 1 + (\beta_1 + \boldsymbol{M}_i^T \boldsymbol{\beta}_{m1}) D_i \exp\{(\beta_2 + \boldsymbol{M}_i^T \boldsymbol{\beta}_{m2}) D_i\}.$$

Quadratic and effect modification terms are optional where applicable.

For binary outcomes, logistic regression uses

$$p_i/(1-p_i) = RR_i \exp(\alpha_0 + \boldsymbol{X}_i^T \boldsymbol{\alpha}).$$

For counts, Poisson regression uses

$$Y_i \sim \text{Poisson}(\mu_i), \qquad \mu_i = RR_i \exp(\alpha_0 + \boldsymbol{X}_i^T \boldsymbol{\alpha}) \times \text{offset}_i,$$

with an optional offset.

For time-to-event outcomes, proportional hazards regression uses

$$h(t) = h_0(t) RR_i \exp(\boldsymbol{X}_i^T \boldsymbol{\alpha}),$$

where h_0 is the baseline hazard.

For a categorical outcome with $Z > 2$ categories, multinomial logistic regression uses the last category as the referent ($\alpha_{0,Z} = \boldsymbol{\alpha}_Z = \beta_{1,Z} = \beta_{2,Z} = \boldsymbol{\beta}_{m1,Z} = \boldsymbol{\beta}_{m2,Z} = 0$):

$$P(Y_i = z) = \frac{RR_i \exp(\alpha_{0,z} + \boldsymbol{X}_i^T \boldsymbol{\alpha}_z)}{1 + \sum_{s=1}^{Z-1} RR_i \exp(\alpha_{0,s} + \boldsymbol{X}_i^T \boldsymbol{\alpha}_s)}.$$

For matched case-control data, conditional logistic regression uses

$$P\left(Y_i = 1, Y_k = 0 \; \forall k \neq i \;\middle|\; \sum_{i \in \mathcal{R}} Y_i = 1\right) = \frac{RR_i \exp(\boldsymbol{X}_i^T \boldsymbol{\alpha})}{\sum_{k \in \mathcal{R}} RR_k \exp(\boldsymbol{X}_k^T \boldsymbol{\alpha})},$$

where $\mathcal{R}$ is the matched set containing individual i.

The estimation methods are described by Carroll et al. (2006) doi:10.1201/9781420010138 for RC, Little et al. (2023) doi:10.1038/s41598023422283y for ERC, Stayner et al. (2007) doi:10.1667/RR0677.1 for MCML, Kwon et al. (2023) doi:10.1371/journal.pone.0290498 for FMA, and Kwon et al. (2016) doi:10.1002/sim.6635 for BMA.

References

Roberti, S., Kwon D., Wheeler W., Pfeiffer R. (in preparation). ameras: An R Package to Analyze Multiple Exposure Realizations in Association Studies

See Also

`confint` for computing confidence intervals, `summary` for a summary of the fitted model including confidence intervals if computed, `coef` for extracting coefficients, `vcov` for extracting variance-covariance matrices, and `included_realizations` for inspecting which realizations contributed to model averaging.

For extended examples, see `vignette("modelfitting", package="ameras")` for the general workflow, `vignette("relativeriskmodels", package="ameras")` for relative risk models, `vignette("effectmodification", package="ameras")` for effect modification, and `vignette("confidenceintervals", package="ameras")` for confidence and credible intervals.

Examples

```
## Fit a Gaussian regression-calibration model using ten exposure realizations
example_data <- ameras_simulated[1:500, ]
fit <- ameras(
  Y.gaussian ~ dose(V1:V10) + X1 + X2,
  data = example_data,
  family = "gaussian",
  methods = "RC"
)

## Extract estimates
coef(fit)

## Add Wald confidence intervals and display a detailed summary
fit <- confint(fit, type = "wald.orig", print = FALSE)
summary(fit)
```

---

ameras_simulated *Simulated Data for ameras Examples*

Description

A simulated dataset containing 10 exposure realizations, covariates, effect modifiers, and outcomes for all six model families supported by **ameras**. The outcomes were generated under linear excess relative risk models. The dataset is intended for examples and demonstrations, not as a benchmark for recovering known simulation parameters.

Format

A data frame with 3,000 rows and 22 variables:

Y.gaussian A numeric continuous outcome.

Y.binomial A binary outcome coded as 0 or 1.

Y.poisson A nonnegative integer count outcome.

time Observed exit or follow-up time for proportional hazards models.

status Event indicator for proportional hazards models, coded as 0 or 1.

setnr Matched-set identifier for conditional logistic regression.

Y.clogit Case indicator for conditional logistic regression, coded as 0 or 1.

Y.multinomial A factor outcome with levels `"1"`, `"2"`, and `"3"`.

X1, X2 Binary covariates coded as 0 or 1.

M1, M2 Binary effect modifiers coded as 0 or 1.

V1, V2, V3, V4, V5, V6, V7, V8, V9, V10 Numeric simulated exposure realizations.

Source

Simulated by the package authors for illustrating the functionality of **ameras**. The dataset contains no real participant data.

Examples

```
# The dataset is available directly because ameras uses lazy data.
head(ameras_simulated)
```

`coef.amerasfit` *Estimated Coefficients for an amerasfit Object*

Description

Returns a data frame with all the parameters of a fitted `amerasfit` object. The resulting object has a column for every method supplied to `methods` when calling `ameras`, with rows corresponding to parameters.

Usage

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

Arguments

`object`	A fitted model object of class `amerasfit`, as returned by `ameras`.
`...`	Additional arguments, currently unused.

Value

Data frame with estimated model parameters. Column names correspond to the `methods` used in the `ameras` call, and row names correspond to parameter names.

See Also

ameras for model fitting, summary for a summary of the fitted model including confidence intervals if computed.

Examples

```
dosevars <- paste0("V", 1:10)

## Fit the model
fit <- ameras(Y.binomial~dose(V1:V10, model="ERR"), data = ameras_simulated, family = "binomial",
methods = c("RC","ERC"))
## Full matrix
coef(fit)

## Vector with RC parameters
coef(fit)$RC
```

---

confint.amerasfit *Confidence Intervals for an amerasfit Object*

Description

Computes and/or prints confidence intervals for the parameters of a fitted amerasfit object. This is a separate step from model fitting, i.e., ameras fits the model and confint computes intervals, attaches them to the fitted object, and prints them. If confint is called a second time on the same object, intervals are only printed and not recomputed, unless force=TRUE.

Usage

```
## S3 method for class 'amerasfit'
confint(object, parm="dose", level=0.95,
        type=c("proflik","percentile"), maxit.profCI=20,
        tol.profCI=1e-2, data=NULL, force=FALSE, print=TRUE,
        digits = max(3, getOption("digits") - 3), ...)
```

Arguments

object	A fitted model object of class amerasfit, as returned by ameras.
parm	Either "dose" to compute intervals for dose-related parameters only, "all" for all parameters, or a character vector of specific parameter names. Only used when type = "proflik" since Wald intervals are cheap to compute for all parameters simultaneously. Defaults to "dose" since profile likelihood computation can be extensive.
level	The confidence level (default 0.95).
type	The type(s) of confidence intervals to determine. For RC, ERC, and MCML, this can be one of:

"wald.orig" Wald intervals on the original parameter scale using the delta method variance-covariance matrix.

"wald.transformed" Wald intervals computed on the transformed (reparametrized) scale and then back-transformed. Only available when a transformation was used during fitting.

"proflik" Profile likelihood intervals based on the chi-squared approximation. More accurate than Wald intervals but computationally intensive, especially for large datasets or complex models.

For FMA and BMA, confidence intervals are based on the generated samples and possible confidence interval types are:

"percentile" Equal-tailed percentile intervals.

"hpd" Highest posterior density intervals via HPDinterval from the **coda** package.

If object contains results for at least one of RC, ERC, and MCML and at least one of FMA and BMA, type must be length 2 and specify one method for RC, ERC, and MCML, and one for FMA and BMA.

maxit.profCI Maximum number of iterations for the root-finding algorithm used to locate profile likelihood interval bounds. Only used when type = "proflik". Defaults to 20.

tol.profCI Tolerance for the root-finding algorithm. Only used when type = "proflik". Defaults to 1e-2. Reduce for more precise bounds at the cost of additional computation.

data The original data frame used for fitting. Only required when type = "proflik" and the model was fitted with keep.data = FALSE

force Logical. If TRUE, confidence intervals are recomputed even if they have already been computed for this object. Defaults to FALSE, in which case a message is printed and the object is returned unchanged if confidence intervals are already present.

print Logical. If TRUE (default), confidence intervals are printed to the console.

digits Number of significant digits to be printed. Default is max(3, getOption("digits") - 3).

... Additional arguments, currently unused.

Details

For (extended) regression calibration and Monte Carlo maximum likelihood, Wald and profile likelihood intervals can be obtained. When a parameter transformation $\boldsymbol{\theta} = h(\boldsymbol{\eta})$ is used, type="wald.transformed" yields the CI at significance level α of $h(\boldsymbol{\eta} \pm z_{1-\alpha/2}\boldsymbol{V})$ where $z_{1-\alpha/2}$ is the $1-\alpha/2$-quantile of the standard normal distribution and $\boldsymbol{V}$ is the vector of standard deviations estimated using the inverse Hessian matrix, and type="wald.orig" uses the delta method to obtain the CI $h(\boldsymbol{\eta}) \pm z_{1-\alpha/2}\boldsymbol{V}_*$ where $\boldsymbol{V}_*$ is the vector of standard deviations estimated using $JH^{-1}J^T$ with J the Jacobian of the transformation and H is the Hessian. When no transformation is used, type="wald.orig" should be used. The third option is proflik, which uses the profile likelihood to compute confidence bounds.

For FMA and BMA, the options for confidence/credible intervals are `type="percentile"` which uses percentiles, and `type="hpd"` which computes highest posterior density intervals using `HPDinterval` from the `coda` package, both using the FMA samples or Bayesian posterior samples.

Profile likelihood intervals (`type="proflik"`) require re-evaluating the likelihood repeatedly and can be time-consuming. The `parm` argument can be used to restrict computation to dose parameters only (the default) when intervals for the other parameters are not of interest. If a profile likelihood bound cannot be reliably bracketed or solved, the corresponding bound is returned as `NA` with a warning that reports the last attempted parameter value and its p-value. For transformed fits, the warning also reports the corresponding internal optimizer-scale value. For models with formula effect modifiers, subgroup-specific profile likelihood intervals can be obtained by fitting subgroup-specific dose coefficients with `modifier=~0+M` or `modifier=~M-1`.

When the model was fitted with `keep.data=FALSE` and `type="proflik"` is used for `confint`, the original data must be supplied via the `data` argument. The missing-value policy used during fitting is reapplied to the supplied data before profile likelihood evaluation. Wald intervals do not require the data and can always be computed from the stored Hessian and parameter estimates alone.

Value

The original `amerasfit` object with a `CI` element added to each fitted method result. The method-specific `timing` element is also updated so that printed runtime summaries include confidence interval computation time. For RC, ERC, and MCML the `CI` element is a data frame with columns:

`lower` Lower confidence bound.

`upper` Upper confidence bound.

When `type = "proflik"`, four additional columns are included:

`pval.lower` P-value at the lower bound, should be close to $1-$ `level`.

`pval.upper` P-value at the upper bound, should be close to $1-$ `level`.

`iter.lower` Number of iterations used by the root-finding algorithm for the lower bound.

`iter.upper` Number of iterations used by the root-finding algorithm for the upper bound.

For FMA and BMA the `CI` element is a data frame with columns `lower` and `upper`.

See Also

`ameras` for model fitting, `summary` for a summary of the fitted model including confidence intervals if computed, `confint` for the generic function.

Examples

```
## Fit the model
fit <- ameras(Y.binomial~dose(V1:V10, model="ERR"), data = ameras_simulated, family = "binomial",
              methods = "RC")

## Wald intervals (fast)
fit <- confint(fit, type = "wald.orig")
summary(fit)
```

```
## Profile likelihood intervals for dose parameters only (slower)
fit <- confint(fit, type = "proflik", parm = "dose", force = TRUE)
summary(fit)

## With keep.data = FALSE, supply data explicitly for proflik
fit2 <- ameras(Y.binomial~dose(V1:V10, model="ERR"), data = ameras_simulated, family = "binomial",
               methods = "RC", keep.data = FALSE)
fit2 <- confint(fit2, type = "proflik", data = ameras_simulated)

## FMA and BMA with percentile intervals

fit3 <- ameras(Y.binomial~dose(V1:V10, model="ERR"), data = ameras_simulated, family = "binomial",
               methods = c("FMA", "BMA"))
fit3 <- confint(fit3, type = "percentile")
summary(fit3)
```

`convergence` *Optimizer Convergence Diagnostics*

Description

Compute or extract numerical gradient diagnostics for fitted `ameras` models.

Usage

```
convergence(object, ...)

## S3 method for class 'amerasfit'
convergence(
  object,
  methods = c("RC", "ERC", "MCML"),
  data = NULL,
  recompute = FALSE,
  warn = FALSE,
  ...
)
```

Arguments

`object`	an object of class `amerasfit`.
`methods`	character vector of fitted methods for which diagnostics should be returned. Gradient diagnostics are available for `"RC"`, `"ERC"`, and `"MCML"`.
`data`	optional data frame required when diagnostics must be recomputed for a fit created with `keep.data=FALSE`.
`recompute`	logical, whether to recompute the numerical gradient even when stored diagnostics are available.

warn — logical, whether to emit the same warning used during fitting when optim() reported convergence but the optimizer diagnostics suggest that the solution may not be fully stationary.

... — currently unused.

Details

By default, convergence() returns stored diagnostics when available and recomputes them only when needed or requested. If diagnostics must be recomputed for a fit created with keep.data=FALSE, the original analysis data must be supplied through data.

These diagnostics do not change the fitted object. They are intended to help identify fits where optim() returned convergence code 0 but the objective is not locally flat enough to be fully reassuring. The convergence.warning column is based on optim.convergence and, when available, newton.improvement and newton.improvement.relative. Both Newton-improvement diagnostics must exceed their warning thresholds: newton.improvement > 1e-2 and newton.improvement.relative > 1e-6. This is less sensitive to likelihood scale than the raw gradient norm, because raw gradients can be large for likelihoods with many observations even when the fitted coefficients are effectively at the optimum. If the Hessian cannot be used for this calculation, ameras falls back to gradient.rms > 1e-3 and gradient.rms.scaled > 1e-4. Poorly scaled continuous covariates, such as uncentered calendar year variables, can contribute to this kind of numerical behavior.

Gradient diagnostics are not returned for FMA or BMA. For BMA, use MCMC diagnostics such as Rhat.

Value

A data frame with one row per requested fitted method and columns:

method — method name.

optim.convergence — optimizer convergence code, where 0 indicates convergence.

gradient.rms — root mean square of the numerical gradient at the fitted parameter vector.

gradient.rms.scaled — gradient.rms divided by max(1, abs(objective value)).

newton.improvement — approximate objective improvement still available from one Newton step, computed as $0.5 g^T H^{-1} g$.

newton.improvement.relative — newton.improvement divided by max(1, abs(objective value)). This makes the Newton-step diagnostic less sensitive to the scale of the likelihood.

convergence.warning — logical indicator for whether the ameras diagnostic threshold would warn about potentially unreliable optimizer convergence.

See Also

ameras, Rhat

Examples

```
fit <- ameras(Y.gaussian ~ dose(V1:V10) + X1 + X2,
              data = ameras_simulated, family="gaussian", methods="RC")
convergence(fit)
```

`dose_lrt` *Likelihood-ratio tests for dose-related parameters*

Description

Compute global and individual likelihood-ratio tests for dose-related parameters in supported `amerasfit` methods.

Usage

```
dose_lrt(object, ...)

## S3 method for class 'amerasfit'
dose_lrt(
  object,
  methods = c("RC", "ERC", "MCML"),
  type = c("global", "individual"),
  data = NULL,
  ...
)
```

Arguments

`object`	An object of class `amerasfit`.
`methods`	Character vector of fitted methods to test. Supported methods are `"RC"`, `"ERC"`, and `"MCML"`.
`type`	Character vector selecting which tests to return. `"global"` tests all dose-related parameters jointly. `"individual"` tests each dose-related parameter separately. By default, both are returned.
`data`	Optional data frame required when `object` was fitted with `keep.data=FALSE`.
`...`	Currently unused.

Details

`dose_lrt()` computes likelihood-ratio tests for dose-related parameters in fitted RC, ERC, and MCML models. The null hypothesis sets the tested dose-related parameter(s) equal to zero. FMA and BMA are not supported by this function.

For transformed fits, nuisance parameters are evaluated through the fitted transformation. The tested dose parameters are then fixed to zero before the likelihood is evaluated. This is most straightforward for component-wise transformations, including the default transformations used by **ameras**.

Results from coupled custom transformations should be interpreted with caution. For example, if a custom transformation defines one reported parameter as a function of multiple optimizer-scale parameters, or makes a non-dose reported parameter depend directly on a dose parameter, fixing only the reported dose parameter to zero may not correspond to the intended constrained model.

For speed, constrained null fits report only the optimizer convergence code. They do not run the additional numerical-gradient diagnostics stored for the original RC, ERC, and MCML fits.

Value

A data frame with one row per likelihood-ratio test and columns:

`method` The ameras estimation method.

`type` Either `"global"` or `"individual"`.

`term` The tested dose term. For global tests this is `"all dose terms"`.

`df` Degrees of freedom for the likelihood-ratio test.

`logLik` The fitted model log likelihood.

`logLik.null` The constrained null-model log likelihood.

`statistic` The likelihood-ratio statistic.

`p.value` The chi-square reference p-value.

`null.optim.convergence` Optimizer convergence code for the constrained null fit. A warning is issued if this code is nonzero.

Examples

```
set.seed(1)
dat <- data.frame(
  Y = 1 + 0.8 * seq(-1, 1, length.out = 30) + rnorm(30, sd = 0.2),
  D = seq(-1, 1, length.out = 30)
)

fit <- ameras(Y ~ dose(D), data = dat, family = "gaussian",
              methods = "RC")

dose_lrt(fit)
dose_lrt(fit, type = "global")
```

---

ecdfplot *Visualize Multiple Dose Realizations*

Description

Create a descriptive figure to visualize the distribution of dose and its uncertainty.

Usage

```
ecdfplot(data, dosevars, xlab="Dose",
  ylab="Cumulative distribution", show.mean=TRUE, log.xaxis=TRUE)
```

Arguments

data	data frame containing columns with dose vectors
dosevars	names or column indices of dose vectors.
xlab	label for the x-axis, default "Dose".
ylab	label for the y-axis, default "Cumulative distribution".
show.mean	logical, whether to plot the cumulative distribution of the mean dose across realizations and across individuals, default TRUE.
log.xaxis	logical, whether to use a log-scale for the dose axis (default TRUE). When log.xaxis=TRUE, any zeros in the doses are excluded while plotting.

Details

In the left panel, the empirical cumulative distribution function (ECDF) is plotted for each dose realization. In other words, each curve shows one distribution of dose across individuals. The spread within individual curves reflects the dose range across individuals, while the spread between curves reflects between-realization variation on the cohort level. If show.mean=TRUE, the solid black curve is the cumulative distribution of the mean dose for each individual.

In the right panel, ECDFs are plotted for each individual, showing distributions within individuals. A wide spread within individual curves is indicative of large within-individual variation, while the spread between curves reflects between-individual variation. If show.mean=TRUE, the solid black curve is the cumulative distribution of the mean for each dose realization.

When using a log-scale for the x-axis, any zero dose values are excluded before plotting.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
 ecdfplot(ameras_simulated, dosevars=paste0("V", 1:10))
}
```

---

included_realizations *Extract Included Realizations from an amerasfit Object*

Description

Extracts the indices of dose realizations included in model averaging for FMA and/or BMA results of a fitted amerasfit object. Realizations are excluded from FMA if the optimization did not converge or the Hessian was not positive definite, or if the realization receives no samples after integer sample allocation.

Usage

```
included_realizations(x, ...)

## S3 method for class 'amerasfit'
included_realizations(x,
                              methods = c("FMA", "BMA"),
                              ...)
```

Arguments

x	A fitted model object of class amerasfit, as returned by ameras, with FMA and/or BMA results present.
methods	Character vector specifying which method(s) to extract included realizations for. One or both of "FMA" and "BMA". Defaults to both. Only methods that were run are returned.
...	Additional arguments, currently unused.

Details

For FMA, realizations are excluded if the optimization did not converge or the Hessian was not invertible or not positive definite. Realizations with model-averaging weights too small to receive any samples for the requested MFMA are also excluded from the final FMA result. Warnings and messages are issued during fitting when realizations are excluded or when the final FMA result is based on very few realizations.

For BMA, all realizations are included by default. A subset of realizations can be specified via the included.realizations.BMA argument to ameras, for example by passing the indices returned by included_realizations(fit, methods="FMA") from a prior FMA fit.

Value

If a single method is requested or only one method was run, an integer vector of indices of the included dose realizations. If both FMA and BMA results are present and both are requested, a named list with elements FMA and BMA, each containing an integer vector of included realization indices.

See Also

ameras, Rhat.amerasfit

Examples

```
dosevars <- paste0("V", 1:10)

## FMA only
fit <- ameras(Y.binomial ~ dose(all_of(dosevars), model="ERR"),
              data = ameras_simulated, family="binomial", methods="FMA")
included_realizations(fit)
length(included_realizations(fit))
```

```
## Both FMA and BMA

fit2 <- ameras(Y.binomial ~ dose(all_of(dosevars), model="ERR"),
               data = ameras_simulated, family="binomial", methods=c("FMA", "BMA"))
included_realizations(fit2)
included_realizations(fit2, methods="FMA")
included_realizations(fit2, methods="BMA")
```

`plot.amerasfit` *Diagnostic Plots for an amerasfit Object*

Description

Produces diagnostic plots for a fitted `amerasfit` object, including residuals versus fitted values and normal Q-Q plots. Plots are produced for each estimation method present in the fitted object.

Usage

```
## S3 method for class 'amerasfit'
plot(x,
     methods = c("RC", "ERC", "MCML", "FMA", "BMA"),
     which = NULL, type = NULL, dose.col = NULL,
     add.smooth = getOption("add.smooth", TRUE),
     qqline = TRUE, id.n = 3, ask = NULL, data = NULL, ...)
```

Arguments

`x`	A fitted model object of class `amerasfit`, as returned by `ameras`.
`methods`	Character vector specifying which estimation methods to produce plots for. One or more of `"RC"`, `"ERC"`, `"MCML"`, `"FMA"`, and `"BMA"`. Defaults to all. Methods not present in `x` are skipped; an error is raised if none of the requested methods are present.
`which`	Character vector specifying which plots to produce. For non-`"prophaz"` families, one or both of `"residuals-vs-fitted"` and `"qq"`. For `family="prophaz"`, `"schoenfeld"` produces a scaled Schoenfeld residual plot against event time. Defaults to both for non-`"prophaz"` families and to `"schoenfeld"` for `"prophaz"` families.
`type`	The type of residuals to use. One of `"pearson"`, `"deviance"`, or `"response"` for non-`"prophaz"` models. For `family="prophaz"`, this must be `"schoenfeld"`. Defaults to `"pearson"` for non-`"prophaz"` families and to `"schoenfeld"` for `"prophaz"`. See `residuals.amerasfit` for details.
`dose.col`	The dose realization to use for computation of residuals. If `NULL` (default), a best fitting realization is determined for each method (see Details).

add.smooth	Logical. If TRUE, a loess smooth line is added to the residuals versus fitted plot via panel.smooth. Defaults to getOption("add.smooth", TRUE). Is not used for the Schoenfeld residual plot for which a spline is always shown, mirroring plot.cox.zph.
qqline	Logical. If TRUE, a reference line is added to normal Q-Q plots via qqline. Defaults to TRUE.
id.n	Integer. The number of extreme residuals to label in residuals vs fitted and normal Q-Q plots. Labels show the row index of the observation. Defaults to 3. Set to 0 to suppress labeling.
ask	Logical. If TRUE, the user is prompted before each new plot. Defaults to NULL, in which case prompting occurs automatically when the number of plots exceeds the number of available panels and the session is interactive.
data	The original data frame used for fitting. Only required when the model was fitted with keep.data=FALSE.
...	Additional graphical arguments passed to plot.default and qqnorm.

Details

If not otherwise specified, the dose realization used to compute fitted values is selected for each estimation method as follows. For RC and ERC the mean dose across realizations is used. For MCML and FMA the realization yielding the largest likelihood at the final parameter estimates is used (which for FMA corresponds to the highest model averaging weight). For BMA the realization most frequently selected by the MCMC sampler is used. The selected dose column is shown in the plot title.

For the "residuals-vs-fitted" plot, the smooth line is added via panel.smooth using a loess smoother when add.smooth=TRUE. For the "qq" plot, the residuals are plotted against the theoretical quantiles of the normal distribution when qqline=TRUE. For both plots, the id.n most extreme residuals are labeled. For family="multinomial", one panel is produced per non-reference outcome category.

For proportional hazards models, scaled Schoenfeld residuals are drawn against the observed event times to assess the proportional hazards assumption. Under proportional hazards, the residuals should fluctuate randomly around zero with no systematic trend over time. Systematic patterns or smooth trends may indicate time-varying covariate effects and violation of the proportional hazards assumption. These plots correspond to using transform = "identity" in cox.zph, in contrast to the default setting for that function, which applies a Kaplan–Meier transformation of time.

The data must either be stored on the object (keep.data=TRUE in ameras, the default) or supplied via the data argument. If the model was fitted with na.exclude, diagnostic plots use the fitted rows only; residuals and fitted values are not padded back to the originally supplied row count.

Value

The amerasfit object x is returned invisibly.

See Also

residuals.amerasfit for computing residuals, ameras for model fitting, confint for confidence intervals, plot.lm for the equivalent method for linear models.

Examples

```
dosevars <- paste0("V", 1:10)

## Binomial model
fit <- ameras(Y.binomial ~ dose(all_of(dosevars), model="ERR"),
              data = ameras_simulated, family="binomial", methods="RC")

## Both diagnostic plots for RC
plot(fit)

## Residuals vs fitted only
plot(fit, which="residuals-vs-fitted")

## Deviance residuals
plot(fit, type="deviance")

## Multiple methods

fit2 <- ameras(Y.binomial ~ dose(all_of(dosevars), model="ERR"),
               data = ameras_simulated, family="binomial", methods=c("RC", "ERC"))
plot(fit2)


## With keep.data=FALSE, supply data explicitly
fit3 <- ameras(Y.binomial ~ dose(all_of(dosevars), model="ERR"),
               data = ameras_simulated, family="binomial", methods="RC",
               keep.data=FALSE)
plot(fit3, data = ameras_simulated)

## Schoenfeld residual plot
fit4 <- ameras(Surv(time, status) ~ dose(all_of(dosevars), model = "ERR"),
              data = ameras_simulated, family = "prophaz", methods = "RC")
plot(fit4)
```

`print.amerasfit` *Simple Summary for an amerasfit Object*

Description

Prints a simple summary of a fitted `amerasfit` object.

Usage

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

Arguments

`x`	A fitted model object of class `amerasfit`, as returned by `ameras`.

digits	Number of significant digits to be printed. Default is max(3, getOption("digits") - 3).
...	Additional arguments, currently unused.

Value

Prints the ameras call, number of rows and dose realizations in the data, CPU runtime, and model coefficients. Row counts reflect omissions due to missing values and model-specific exclusions, when applicable. For objects with structured timing information, fitting time, confidence interval computation time, and total time are shown separately.

See Also

ameras for model fitting, summary for a more detailed summary of the fitted models including confidence intervals if computed.

Examples

```
## Fit the model
fit <- ameras(Y.binomial~dose(V1:V10, model="ERR"), data = ameras_simulated, family = "binomial",
              methods = c("RC","ERC"))

## Default print
fit
print(fit)

## More digits
print(fit, digits=5)
```

---

residuals.amerasfit *Residuals for an amerasfit Object*

Description

Computes residuals for a fitted amerasfit object.

Usage

```
## S3 method for class 'amerasfit'
residuals(object,
          method = "RC",
          type = NULL,
          data = NULL,
          dose.col = NULL,
          scaled.schoenfeld = TRUE,
          ...)
```

Arguments

`object`	A fitted model object of class `amerasfit`, as returned by `ameras`.
`method`	Character string specifying which estimation method to compute residuals for. One of `"RC"`, `"ERC"`, `"MCML"`, `"FMA"`, or `"BMA"`. Defaults to `"RC"`.
`type`	The type of residuals to compute. Defaults to `"pearson"` for all families except `"prophaz"`, for which the default is `"schoenfeld"`. For types other than `"schoenfeld"`, let Y_i and μ_i denote the observed response and fitted mean response for individual i, respectively. Then the residuals are defined as follows: `"pearson"` Pearson residuals. For `"gaussian"`: $(Y_i - \mu_i)/\sigma$ where σ is the estimated residual standard deviation (note: `glm` returns raw residuals for Gaussian). For `"binomial"` and `"clogit"`: $(Y_i - \mu_i)/\sqrt{\mu_i(1-\mu_i)}$. For `"poisson"`: $(Y_i - \mu_i)/\sqrt{\mu_i}$. For `"multinomial"`: per-category Pearson residuals $(Y_{iz} - \mu_{iz})/\sqrt{\mu_{iz}(1-\mu_{iz})}$ where μ_{iz} is the fitted probability of category z for individual i. `"deviance"` Deviance residuals, defined as the signed square root of the individual contribution to the deviance. For `"gaussian"`: the raw residual $Y_i - \mu_i$. For `"binomial"` and `"clogit"`: $\pm\sqrt{-2\log(\hat{p}_i)}$ where $\hat{p}_i = \mu_i$ if $Y_i = 1$ and $\hat{p}_i = 1 - \mu_i$ if $Y_i = 0$, with sign equal to $\text{sign}(Y_i - \mu_i)$. For `"poisson"`: $\pm\sqrt{2(Y_i \log(Y_i/\mu_i) - (Y_i - \mu_i))}$ for $Y_i > 0$ and $\pm\sqrt{2\mu_i}$ for $Y_i = 0$, with sign equal to $\text{sign}(Y_i - \mu_i)$. For `"multinomial"`: per-category deviance residuals as for `"binomial"`. `"response"` Raw residuals $Y_i - \mu_i$. For `"multinomial"`: the matrix $\boldsymbol{Y} - \hat{\boldsymbol{P}}$ where $\boldsymbol{Y}$ is the $N \times Z$ indicator matrix of observed categories and $\hat{\boldsymbol{P}}$ is the $N \times Z$ matrix of fitted probabilities. `"schoenfeld"` Schoenfeld residuals for `family="prophaz"` only. For each event time, the unscaled residual for the individual experiencing the event is the observed covariate vector minus the risk-set weighted mean covariate vector at that event time. See Details.
`data`	The original data frame used for fitting. Only required when the model was fitted with `keep.data=FALSE`.
`dose.col`	Character string specifying the dose column to use when computing fitted values. If `NULL` (the default), the dose column is selected automatically: the mean dose across realizations for RC and ERC, the realization with the highest likelihood for MCML and FMA, and the most frequently selected realization for BMA. Can be set to any dose column present in the data to override the automatic selection.
`scaled.schoenfeld`	Logical. If `TRUE` (the default), scaled Schoenfeld residuals are returned, obtained by multiplying the raw Schoenfeld residuals by the estimated coefficient variance-covariance matrix following Grambsch and Therneau (1994). If `FALSE`, raw Schoenfeld residuals are returned. Only used when `type="schoenfeld"`.
`...`	Additional arguments, currently unused.

Details

Fitted values are computed using the dose column specified by `dose.col`.

If the model was fitted with `na.exclude`, residuals are returned for the fitted rows only; they are not padded with `NA` values back to the originally supplied row count. For proportional hazards models, Schoenfeld residuals are event-level quantities and are returned only for event rows.

For `family="clogit"`, fitted values μ_i are the conditional probabilities of being a case within each matched set, computed as $R_i / \sum_{j \in \text{set}(i)} R_j$ where R_i is the relative risk for individual i and the sum is over all individuals in the same matched set.

For `family="prophaz"` and `type="schoenfeld"`, the Schoenfeld residual for the individual failing at time t is:

$$\boldsymbol{r} = \boldsymbol{x} - \bar{\boldsymbol{x}}(t)$$

where $\boldsymbol{x}$ is the observed covariate vector for the failing individual and

$$\bar{\boldsymbol{x}}(t) = \frac{\sum_{j \in \mathcal{R}(t)} \boldsymbol{x}_j R_j}{\sum_{j \in \mathcal{R}(t)} R_j}$$

is the weighted mean covariate vector in the risk set $\mathcal{R}(t)$ at time t, with weights equal to the relative risks R_j. Ties in event times are handled using the Breslow approximation.

Value

For families `"gaussian"`, `"binomial"`, `"poisson"`, and `"clogit"`, a numeric vector of length N containing the residuals, where N is the number of rows used for fitting after any `na.action` handling.

For `family="multinomial"`, a numeric matrix of dimension $N \times Z$ where Z is the number of outcome categories. Column names correspond to the factor levels. Note that when plotting via `plot`, the reference category is excluded since its residuals are a linear combination of the other categories.

For `family="prophaz"` with `type="schoenfeld"`, a data frame with columns `id` (row index of the event), `time` (event time), and one additional column per model parameter containing the Schoenfeld residuals. Only rows corresponding to events (`status=1`) are included.

References

Schoenfeld, D. (1982). Partial residuals for the proportional hazards regression model. *Biometrika*, **69**(1), 239–241. doi:10.1093/biomet/69.1.239

Grambsch, P. M. and Therneau, T. M. (1994). Proportional hazards tests and diagnostics based on weighted residuals. *Biometrika*, **81**(3), 515–526. doi:10.1093/biomet/81.3.515

See Also

`plot` for diagnostic plots using these residuals, `ameras` for model fitting, `residuals` for the generic function.

Examples

```
dosevars <- paste0("V", 1:10)

fit <- ameras(Y.binomial ~ dose(all_of(dosevars), model="ERR"),
              data = ameras_simulated, family="binomial", methods="RC")
```

```
## Pearson residuals (default)
res <- residuals(fit)
summary(res)

## Deviance residuals
res_dev <- residuals(fit, type="deviance")

## Response residuals
res_raw <- residuals(fit, type="response")

## Specific dose column
res_v1 <- residuals(fit, dose.col="V1")

## Multiple methods

fit2 <- ameras(Y.binomial ~ dose(all_of(dosevars), model="ERR"),
               data = ameras_simulated, family="binomial", methods=c("RC", "ERC"))
res_erc <- residuals(fit2, method="ERC")


## With keep.data=FALSE
fit3 <- ameras(Y.binomial ~ dose(all_of(dosevars), model="ERR"),
               data = ameras_simulated, family="binomial", methods="RC",
               keep.data=FALSE)
res <- residuals(fit3, data = ameras_simulated)

## Schoenfeld residuals for proportional hazards model
fit4 <- ameras(Surv(time, status) ~ dose(all_of(dosevars), model="ERR"),
               data = ameras_simulated, family="prophaz", methods="RC")
res_sch <- residuals(fit4)
res_raw_sch <- residuals(fit4, scaled.schoenfeld=FALSE)
```

`Rhat` *Extract MCMC Convergence Diagnostics from an amerasfit Object*

Description

Extracts the Gelman-Rubin convergence diagnostics from the BMA results of a fitted `amerasfit` object.

Usage

```
Rhat(x, ...)

## S3 method for class 'amerasfit'
Rhat(x, ...)
```

Arguments

x	A fitted model object of class amerasfit, as returned by ameras, with BMA results present.
...	Additional arguments, currently unused.

Value

A data frame with columns Rhat and n.eff, containing the Gelman-Rubin statistic and effective sample size for each parameter. Values of Rhat substantially above 1.05 indicate potential convergence problems, in which case longer chains via niter.BMA and nburnin.BMA are recommended. For single-chain BMA fits, convergence diagnostics cannot be estimated and these columns are returned as NA.

See Also

ameras, summary.amerasfit, included_realizations.amerasfit

Examples

```
dosevars <- paste0("V", 1:10)
fit <- ameras(Y.binomial ~ dose(all_of(dosevars), model="ERR"),
              data = ameras_simulated, family="binomial", methods="BMA")
Rhat(fit)
```

---

summary.amerasfit *Summarize an amerasfit Object*

Description

Produces a summary of a fitted amerasfit object, including parameter estimates, standard errors, and confidence intervals if computed via confint.

Usage

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

## S3 method for class 'amerasfit'
summary_table(object, ...)

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

Arguments

object	A fitted model object of class amerasfit, as returned by ameras.
x	An object of class summary.amerasfit, as returned by summary.amerasfit.
digits	The number of significant digits to use. Defaults to max(3, getOption("digits") - 3).
...	Additional arguments, currently unused.

Details

summary.amerasfit collects results from all estimation methods present in the fitted object into a single summary table. Columns for confidence intervals are only printed if they have been computed by confint. When BMA results are present in the fitted object, the summary table includes columns Rhat and n.eff, with NA values for all other methods. Profile likelihood diagnostic columns such as pval.lower and pval.upper remain available in the stored CI component but are not printed in the summary table.

Printing the summary prints the original call to ameras, row counts, the CPU runtime (total and by method), and the table described above. Row counts reflect omissions due to missing values and model-specific exclusions, when applicable. For conditional logistic regression, model-specific exclusions are rows in uninformative matched sets, i.e., matched sets of size 1 or with no cases. For proportional hazards regression with entry times, model-specific exclusions are rows with zero follow-up time (entry == exit). When confidence intervals have been computed, CI runtime is included in the total runtime. This table can also be accessed directly (i.e., to retrieve confidence intervals) using summary_table.

Value

summary.amerasfit returns an object of class summary.amerasfit, which is a list containing the following elements:

call The matched call from the original call to ameras.

row_info A list describing row handling during model fitting, including the number of rows supplied, omitted by na.action, and used for fitting. For conditional logistic regression, this also includes the number of rows excluded because they belong to uninformative matched sets. For proportional hazards regression with entry times, this also includes the number of rows excluded because they have zero follow-up time (entry == exit).

summary_table A data frame with one row per parameter per method, containing columns:

- Method The estimation method (RC, ERC, MCML, FMA, or BMA).
- Term The parameter name.
- Estimate The parameter estimate.
- SE The standard error.
- CI.lower The lower confidence bound, if confidence intervals have been computed via confint.
- CI.upper The upper confidence bound, if confidence intervals have been computed via confint.
- Rhat The Gelman-Rubin convergence diagnostic, included only when BMA results are present. Values above 1.05 indicate potential convergence problems.
- n.eff The effective sample size, included only when BMA results are present.

`runtime_table` A data frame reporting computation time in seconds for each method. When structured timing information is available this has columns `Method`, `Fit`, `CI`, and `Total`, with CPU time reported for each timing component. Older fitted objects that only contain the former `runtime` field are still supported and produce columns `Method` and `Runtime`.

`total_runtime_seconds` The total CPU computation time in seconds across all methods.

`CI.computed` Logical. `TRUE` if confidence intervals have been computed via `confint`, `FALSE` otherwise.

See Also

`ameras` for model fitting, `confint` for computing confidence intervals, `print` for a shorter printed summary, `coef` for extracting coefficients.

Examples

```
## Fit the model
fit <- ameras(Y.binomial~dose(V1:V10, model="ERR"), data = ameras_simulated, family = "binomial",
              methods = "RC")

## Summary without confidence intervals
summary(fit)

## Summary with confidence intervals
fit <- confint(fit, type = "wald.orig")
summary(fit)

## Access the summary table directly
s <- summary_table(fit)

## Multiple methods

fit2 <- ameras(Y.binomial~dose(V1:V10, model="ERR"), data = ameras_simulated, family = "binomial",
               methods = c("RC", "ERC", "MCML"))
fit2 <- confint(fit2, type = "wald.orig")
summary(fit2)
```

---

traceplot *Traceplots for MCMC Samples*

Description

Produce MCMC traceplots for `amerasfit` objects.

Usage

```
traceplot(object, ...)

## S3 method for class 'amerasfit'
traceplot(object, iter = 5000, Rhat = TRUE, n.eff = TRUE, pdf = FALSE, ...)
```

Arguments

object	a amerasfit object containing BMA output to be plotted
iter	number of iterations to include in the traceplot (defaults to last 5000)
Rhat	logical; whether to include R-hat diagnostics in the plot (default TRUE)
n.eff	logical; whether to include effective sample size in the plot (default TRUE)
pdf	logical; whether to save the output as a PDF (default FALSE)
...	additional arguments passed to MCMCtrace

Details

Wrapper for MCMCvis::MCMCtrace to produce MCMC diagnostic plots. See ?MCMCtrace for more plotting options that can be provided through ....

Value

Traceplots and posterior density plots.

See Also

MCMCtrace

Examples

```
fit <- ameras(Y.gaussian~dose(V1:V10), data = ameras_simulated,
              methods="BMA")
traceplot(fit)
```

---

transform1 *Exponential Parameter Transformation*

Description

Applies exponential transformation $f(\theta_i) = \exp(\theta_i) + L_i$ to one or multiple components of parameter vector $\boldsymbol{\theta}$, where L_i are lower limits that can be different for each component

Usage

```
transform1(params, index.t=1:length(params), lowlimit=rep(0,length(index.t)),
  boundcheck=FALSE, boundtol=1e-3, ... )
```

Arguments

params	full input parameter vector
index.t	indices of parameters to be transformed (default all)
lowlimit	lower limits to be applied (default zero), where the k-th component of lowlimit is applied to the k-th index in index.t
boundcheck	whether to produce a warning when any of the transformed parameters are within boundtol of lowlimit
boundtol	tolerance for producing a warning for reaching the boundary
...	not used

Value

Transformed parameter vector.

Examples

```
params <- c(.1, .5, 1)
transform1(params, lowlimit=c(0, -1, 1))
```

---

transform1.inv *Inverse of Exponential Parameter Transformation*

Description

Inverse of transform1 for the purpose of deriving initial values.

Usage

```
transform1.inv(params, index.t=1:length(params), lowlimit=rep(0,length(index.t)), ... )
```

Arguments

params	full input parameter vector
index.t	indices of parameters to be transformed (default all)
lowlimit	lower limits to be applied (default zero), where the k-th component of lowlimit is applied to the k-th index in index.t
...	not used

Value

Transformed parameter vector.

Examples

```
params <- c(.1, .5, 1) # Desired initial values on original scale
transform1.inv(params, lowlimit=c(0, -1, 1)) # Initial values to use on transformed scale
```

transform1.jacobian *Jacobian of the Exponential Parameter Transformation*

Description

Computes the Jacobian matrix of transform1. Note that lower limits do not need to be specified as the Jacobian is independent of those

Usage

```
transform1.jacobian(params, index.t=1:length(params), ... )
```

Arguments

params	input parameter vector (before transformation) to evaluate the Jacobian at
index.t	indices of parameters to be transformed (default all)
...	not used

Value

Jacobian matrix.

Examples

```
params <- c(.1, .5, 1)
transform1.jacobian(params)
```

vcov.amerasfit *Extract Variance-Covariance Matrices from an amerasfit Object*

Description

Extracts the variance-covariance matrices of the parameter estimates from a fitted amerasfit object.

Usage

```
## S3 method for class 'amerasfit'
vcov(object, methods = c("RC", "ERC", "MCML", "FMA", "BMA"), ...)
```

Arguments

object	A fitted model object of class amerasfit, as returned by ameras.
methods	Character vector specifying which estimation methods to extract variance-covariance matrices for. One or more of "RC", "ERC", "MCML", "FMA", and "BMA". Defaults to all. Methods that were not run are skipped; an error is raised if none of the requested methods are present.
...	Additional arguments, currently unused.

Value

If a single method is requested or only one method was run, a named numeric matrix with rows and columns corresponding to model parameters. If multiple methods are requested and multiple methods were run, a named list of such matrices, one per method.

Note that for FMA and BMA the variance-covariance matrix is based on the generated samples rather than a Hessian-based approximation, and may differ in interpretation from the RC, ERC, and MCML variance-covariance matrices. For BMA, the reliability of the variance-covariance matrix depends on MCMC convergence as indicated by Rhat.

See Also

ameras for model fitting, coef for extracting coefficients, summary.amerasfit for a summary including standard errors, vcov for the generic function.

Examples

```
dosevars <- paste0("V", 1:10)

fit <- ameras(Y.binomial ~ dose(all_of(dosevars), model="ERR"),
              data = ameras_simulated, family="binomial", methods=c("RC", "ERC"))

## Extract vcov for all available methods
vcov(fit)

## Extract vcov for a single method, returns a matrix
vcov(fit, methods="RC")

## Extract vcov for multiple methods, returns a named list
vcov(fit, methods=c("RC", "ERC"))
```

Index

∗ **Bayesian model averaging**
ameras-package, 3
∗ **Monte Carlo Maximum Likelihood**
ameras-package, 3
∗ **datasets**
ameras_simulated, 11
∗ **exposure realizations**
ameras-package, 3
∗ **exposure uncertainty**
ameras-package, 3
∗ **frequentist model averaging**
ameras-package, 3
∗ **measurement error**
ameras-package, 3
∗ **regression calibration**
ameras-package, 3

ameras, 3, *3*, *12, 13*, *15*, *17*, *21–27*, *29–31*, *35*
ameras-package, 3
ameras_simulated, 11

coef, *9*, *11*, *31*, *35*
coef.amerasfit, 12
confint, *3*, *6*, *8, 9*, *11*, *15*, *23*, *29–31*
confint.amerasfit, 13
convergence, *9*, 16
cox.zph, *23*

dose_lrt, 18

ecdfplot, *3*, 19
eval_select, *9*

glm, *6*, *26*

HPDinterval, *14*

included_realizations, *9*, *11*, 20
included_realizations.amerasfit, *29*

lm, *6*

MCMCtrace, *32*
model.matrix, *9*

panel.smooth, *23*
plot, *9*, *27*
plot.amerasfit, 22
plot.cox.zph, *23*
plot.default, *23*
plot.lm, *23*
print, *9*, *31*
print.amerasfit, 24
print.summary.amerasfit
*(*summary.amerasfit*)*, 29

qqline, *23*
qqnorm, *23*

residuals, *9*, *27*
residuals.amerasfit, *22, 23*, 25
Rhat, *9*, *17*, 28
Rhat.amerasfit, *21*

summary, *9*, *11*, *13*, *15*, *25*
summary.amerasfit, 29, *29*, *35*
summary_table, *9*
summary_table *(*summary.amerasfit*)*, 29

traceplot, *9*, 31
transform1, *6, 7*, 32, *34*
transform1.inv, 33
transform1.jacobian, *7*, 34

vcov, *9*, *11*, *35*
vcov.amerasfit, 34