Package 'ameras'

May 7, 2026

Title Analyze Multiple Exposure Realizations in Association Studies

Version 0.3.0

Depends R (>= 4.1.0), stats, nimble

Suggests knitr, rmarkdown, testthat (>= 3.0.0), 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

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>), NCI [cph, fnd]

Maintainer Sander Roberti <sander.roberti@nih.gov>

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

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

Repository CRAN

Date/Publication 2026-05-07 09:40:08 UTC

Contents

`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 <sander.roberti@nih.gov>, 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 accounting for exposure uncertainty using multiple Monte Carlo exposure realizations. Six outcome model families are supported. The first is the Gaussian family for continuous outcomes,

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

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 binary effect modifiers. The quadratic exposure terms and effect modification are optional.

For non-Gaussian families, three relative risk models for the main exposure are supported, the usual exponential $RR_i = \exp(\beta_1 D_i + \beta_2 D_i^2 + \boldsymbol{M}_i^T\boldsymbol{\beta}_{m1} D_i + \boldsymbol{M}_i^T\boldsymbol{\beta}_{m2} D_i^2)$ and the linear excess relative risk (ERR) model $RR_i = 1 + \beta_1 D_i + \beta_2 D_i^2 + \boldsymbol{M}_i^T\boldsymbol{\beta}_{m1} D_i + \boldsymbol{M}_i^T\boldsymbol{\beta}_{m2} D_i^2$, where the quadratic and effect modification terms are optional. Finally, the linear-exponential relative risk model $RR_i = 1 + (\beta_1 + \boldsymbol{M}_i^T\boldsymbol{\beta}_{m1}) D_i \exp\{(\beta_2 + \boldsymbol{M}_i^T\boldsymbol{\beta}_{m2}) D_i\}$ is supported.

The second supported family is logistic regression for binary outcomes, with probabilities

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

Third is Poisson regression for counts,

$$Y_i \sim \text{Poisson}(\mu_i),$$

where $\mu_i = RR_i \exp(\alpha_0 + \boldsymbol{X}_i^T\boldsymbol{\alpha}) \times \text{offset}_i$ with optional offset.

Fourth is proportional hazards regression for time-to-event data, with hazard function

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

with h_0 the baseline hazard.

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

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

Sixth is conditional logistic regression for matched case control data, for which

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

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

Methods include regression calibration (Carroll et al. 2006 doi:10.1201/9781420010138), extended regression calibration (Little et al. 2023 doi:10.1038/s41598023422839y), 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).

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,
  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, ... )
```

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 internal parameter transformation (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).

`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 1,000).

`niter.BMA` number of MCMC iterations per chain for BMA (default 5,000).

`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 run to ensure an optimal fit.

`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 seperately when `keep.data=FALSE`. See `confint`.

`...`
: other arguments, passed to functions such as `transform`.

Details

Models are specified through formulas of the form `Y~dose(dose_expression, model="ERR", deg=1, modifier=M1+M2)+X1+X2`. Here `dose_expression` specifies the dose realization columns and is parsed by `eval_select` from the **tidyselect** package. Useful examples are `D1:D1000` if the doses are in a sequence of columns with sequential names such as `D1-D1000`, and `all_of(dosevars)` where `dosevars` is a vector with the names of all dose columns. Further, `model` specifies, for non-Gaussian families, whether to use the exponential dose-response model (`model="EXP"`), the linear-exponential model (`model="LINEXP"`) or the linear ERR model (`model="ERR"`). Next, `deg` is used to specify whether a quadratic dose term should (`deg=2`) or should not (`deg=1`) be estimated for the exponential or linear ERR dose-response model. The `modifier` term is optional and used to specify binary effect modification variables. Note that interactions in the modifier term are not allowed, e.g. `M1*M2`. When `deg`, `modifier`, and `model` are not supplied, the defaults are `deg=1`, no effect modifiers, and `model="ERR"`. Finally, `X1` and `X2` above represent optional additional covariates, which can include factor variables and interactions such as `X1*X2`. The matched set variable `setnr` required for conditional logistic regression is specified on the right-hand side of the formula through a term `strata(setnr)`, and an optional offset variable `offset` for Poisson regression similarly through a term `offset(offset)`. For proportional hazards regression, the left-hand side of the formula should have the form `Surv(exit, status)` or `Surv(entry, exit, status)`.

A transformation can be used to reparametrize parameters internally (i.e., such that the likelihoods are evaluated at `transform(parameters)`, where `parameters` are unconstrained), and 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 effect modifiers `M` are specified, no transformation is used for those parameters. 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 in 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, horshoe, 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): $\tau \sim$ Cauchy$(0,1)^+; \lambda_j \sim$ Cauchy$(0,1)^+; \beta_j \sim N(0, \tau^2\lambda_j^2)$. Here τ is shared across all parameters
- Double exponential (shrinkage prior): $\lambda_j \sim$ Cauchy$(0,1)^+; \beta_j \sim$ 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.

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 in `data`), `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), `CI.computed` (logical, whether confidence intervals have been attached by `confint`), and `data` (either the data frame used for model fitting when `keep.data=TRUE` or `NULL` otherwise).

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.
`runtime`	string with the runtime in seconds.

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

`vcov`	covariance matrix for the full parameter vector.

`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 1 indicating 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`. This is a list object with `nchains.BMA` components, each a named matrix with the samples from one chain in its rows, with columns corresponding to model parameters.

`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.

`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 hazards 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` the samples generated from the normal distributions associated with each dose 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`, and `traceplot`.

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.

Examples

```
data(data, package="ameras")
ameras(Y.gaussian~dose(V1:V10, modifier=M1+M2)+X1+X2, data=data, family="gaussian")
```

---

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

```
data("data", package = "ameras")
dosevars <- paste0("V", 1:10)

## Fit the model
fit <- ameras(Y.binomial~dose(V1:V10, model="ERR"), data = data, 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 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 and attaches them to the fitted object.

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, ...)
```

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.

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

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.

... — 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.

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. 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. 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

```
data("data", package = "ameras")

## Fit the model
fit <- ameras(Y.binomial~dose(V1:V10, model="ERR"), data = data, 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")
summary(fit)


## With keep.data = FALSE, supply data explicitly for proflik

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


## FMA and BMA with percentile intervals

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

`data` *Example data*

Description

Data includes outcomes of all six supported types in the appropriately named columns. For proportional hazards regression, the observed exit time is `time` and event status is `event`. For conditional logistic regression, the matched set variable is `setnr`. The data has 10 exposure realizations in columns `V1-V10`.

Examples

```
data(data, package="ameras")

# Display a few rows of the data
data[1:5, ]
```

`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

```
## Not run:
   data(data, package="ameras")
   ecdfplot(data,  dosevars=paste0("V", 1:10))

## End(Not run)
```

`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, runtime, and model coefficients.

See Also

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

Examples

```
data("data", package = "ameras")

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

## Default print
fit
print(fit)

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

`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 '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.

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 `ameras` invocation.

`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`.
- `pval.lower` p-value associated with the lower bound of the profile likelihood CI, the assess validity of the obtained bound. Only included if profile likelihood confidence intervals were computed via `confint`.
- `pval.upper` p-value associated with the upper bound of the profile likelihood CI, the assess validity of the obtained bound. Only included if profile likelihood confidence intervals were 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 with columns `Method` and `Runtime`, reporting the computation time in seconds for each method.

`total_runtime_seconds` The total 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

```
data("data", package = "ameras")

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

## Summary without confidence intervals
summary(fit)

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

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

## Multiple methods
## Not run:
fit2 <- ameras(Y.binomial~dose(V1:V10, model="ERR"), data = data, family = "binomial",
              methods = c("RC", "ERC", "MCML"))
fit2 <- confint(fit2, method = "wald.orig")
summary(fit2)

## End(Not run)
```

`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

```
data(data, package="ameras")
fit <- ameras(Y.gaussian~dose(V1:V10), data, 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)
```

Index