

Package ‘pecme’

October 7, 2026

Type Package

Title Penalized ECME Estimation for Censored Linear Mixed Models

Version 0.1.1

Description Fits Gaussian linear mixed models with a random intercept when the response is subject to left, right, and/or interval censoring, using the Expectation/Conditional Maximization Either (ECME) algorithm of Liu and Rubin (1994) in the spirit of the fast censored-response mixed-model algorithm of Vaida and Liu (2009). Simultaneous estimation and variable selection is supported through coordinate-descent penalized maximization with Lasso, Adaptive Lasso, SCAD, MCP, Elastic Net, and Ridge penalties (no penalty is also supported). The random intercept is integrated out by Gauss-Hermite quadrature at every iteration, and the two ECME conditional-maximization steps respectively maximize the expected penalized complete-data objective (for the regression coefficients) and the actual observed-data marginal likelihood (for the variance components), which is the defining feature of ECME relative to plain ECM/EM. The package provides a single-fit engine, a sequential/parallel penalty-parameter grid search with information-criterion or cross-validated selection, data-dependent or user-supplied lambda grids, and an Expectation-Maximization based treatment of a completely missing (at random) response, sharing the same truncated-normal machinery used for censoring. References: Liu and Rubin (1994) ``The ECME algorithm: A simple extension of EM and ECM with faster monotone convergence" <[doi:10.1093/biomet/81.4.633](https://doi.org/10.1093/biomet/81.4.633)>; Vaida and Liu (2009) ``Fast Implementation for Normal Mixed Effects Models With Censored Response" <[doi:10.1198/jcgs.2009.07130](https://doi.org/10.1198/jcgs.2009.07130)>.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports stats, graphics, parallel, lme4, withr

Suggests testthat (>= 3.0.0), waldo, mice

Config/testthat/edition 3
Config/roxygen2/version 8.0.0
NeedsCompilation no
Author Ali Asghar Haeri-Mehrizi [aut, cre],
Arshia Haeri-Mehrizi [aut],
Adel Mohammadpour [rev]
Maintainer Ali Asghar Haeri-Mehrizi <haeri.stat@gmail.com>
Repository CRAN
Date/Publication 2026-10-07 09:00:08 UTC

Contents

grid_pecme	2
pecme	5
pecme_bootstrap	8
pecme_effective_df	9
pecme_lambda_grid	10
pecme_loo	10
pecme_metrics	11
pecme_mi_combine	13
pecme_quadrature_check	14
plot.pecme_grid	15
predict.pecme	15
simulate_pecme_data	16
Index	18

grid_pecme	<i>Grid search over the penalty parameter lambda</i>
------------	--

Description

Grid search over the penalty parameter lambda

Usage

```
grid_pecme(  
  fixed_formula,  
  random_var,  
  data,  
  censor = "none",  
  left = NULL,  
  right = NULL,  
  penalty = c("lasso", "adaptive", "scad", "mcp", "elastic", "ridge", "none"),  
  lambda = NULL,
```

```

nlambda = 50L,
lambda_min_ratio = NULL,
alpha = 0.5,
gamma = NULL,
adaptive_gamma = 1,
criterion = c("bic", "ebic", "aic", "gcv", "cv"),
ebic_gamma = 0.5,
cv_folds = 5L,
cv_seed = NULL,
missing_y = FALSE,
missing_x = c("error", "cca", "mean", "reg"),
standardize = TRUE,
quadrature = 40L,
workers = 1L,
parallel_over = c("lambda", "groups"),
variance_step = c("ecme", "em"),
refit_at_best = TRUE,
init_method = c("lmer", "zero", "random"),
max_iter = 200L,
tol = 1e-06,
cd_max_iter = 1000L,
cd_tol = 1e-08,
keep_path = TRUE,
verbose = FALSE
)

```

Arguments

fixed_formula	A formula for the fixed effects, e.g. 'y ~ x1 + x2'.
random_var	Character: name of the grouping (cluster) column in 'data'. Currently a random INTERCEPT only (no random slopes).
data	A data frame.
censor	Either "none"/"left"/"right"/"interval" applied to every row, a column name in 'data' holding per-row labels from that set, or a vector of the same length as 'nrow(data)'. Rows do not need to be pre-labelled "missing": set 'missing_y = TRUE' and leave the response 'NA' for those rows.
left, right	Column names or numeric vectors of censoring bounds (used only for the corresponding 'censor' labels). With censoring, the response must be supplied on its analysis scale (transformed responses such as 'log(y)' are rejected); censoring bounds are never transformed automatically.
penalty	One of "none", "lasso", "adaptive", "scad", "mcp", "elastic", "ridge".
lambda	Either 'NULL' (default: build a data-dependent grid via [pecme_lambda_max()]/[pecme_lambda_grid()]) or a user-supplied numeric vector of candidate lambda values.
nlambda	Number of grid points when 'lambda = NULL'.
lambda_min_ratio	'lambda_min = lambda_min_ratio * lambda_max' when 'lambda = NULL'; defaults to the usual 'n > p' / 'n <= p' heuristic (see [pecme_lambda_grid()]).

alpha	Elastic Net mixing parameter in <code>[0,1]</code> (<code>'1'</code> = Lasso, <code>'0'</code> = Ridge); ignored unless <code>'penalty = "elastic"</code> .
gamma	Extra shape parameter for SCAD (<code>> 2</code> , default <code>'3.7'</code>) or MCP (<code>> 1</code> , default <code>'3'</code>); ignored otherwise.
adaptive_gamma	Exponent used to build Adaptive Lasso weights from a Ridge pilot fit: <code>'weights = 1 / lbeta_pilotl^adaptive_gamma'</code> .
criterion	One of <code>"bic"</code> (default), <code>"ebic"</code> , <code>"aic"</code> , <code>"gcv"</code> , or <code>"cv"</code> . <code>"cv"</code> performs leave-groups-out K-fold cross-validation (refits the whole model <code>'cv_folds'</code> times per lambda) and selects by held-out predictive log-likelihood; this is far more expensive than the closed-form criteria.
ebic_gamma	Extended-BIC tuning constant in <code>[0,1]</code> (Chen and Chen, 2008); <code>'0'</code> reduces EBIC to ordinary BIC. Only used when <code>'criterion = "ebic"</code> .
cv_folds	Number of folds (by group, not by observation) for <code>'criterion = "cv"</code> .
cv_seed	Optional integer seed for the fold assignment.
missing_y	If <code>'TRUE'</code> , rows with a missing response (or a missing bound for a labelled-censored row) are treated as completely missing at random and IMPUTED via the E-step's posterior mean at every iteration, instead of raising an error. Important: such a row contributes exactly <code>'log(1) = 0'</code> to the observed-data log-likelihood – it carries NO direct information about <code>'(beta, sigma, sigma_b)'</code> ; the imputed value is a model- implied *prediction* for that row given the parameters, not evidence that helped estimate them. The parameters themselves are identified entirely by the observed and censored rows.
missing_x	How to handle <code>'NA'</code> in the fixed-effects predictors: <code>"error"</code> (default), <code>"cca"</code> , <code>"mean"</code> , or <code>"reg"</code> . The <code>"mean"</code> and <code>"reg"</code> options impute the numeric model matrix and are intended only for suitable numeric predictors. For factors, interactions, splines, or nonlinear terms, impute raw variables before constructing the model matrix (e.g. with <code>'mice'</code>). See <code>[pecme_handle_missing_x()]</code> and <code>[pecme_mi_combine()]</code> .
standardize	Standardize predictors before penalization (recommended; default <code>'TRUE'</code>).
quadrature	Number of Gauss-Hermite nodes for the random intercept (default <code>'40'</code>). This is a fixed-order numerical approximation to the random-intercept integral, not an exact evaluation of it; use <code>[pecme_quadrature_check()]</code> on a fitted model to check sensitivity to this choice, and increase it if <code>'sigma_b'</code> is large relative to <code>'sigma'</code> , or convergence looks unstable.
workers	Number of parallel worker processes for the E-step (default <code>'1'</code>). Set to e.g. <code>'32'</code> on a 32-core machine.
parallel_over	<code>"lambda"</code> (default: one full ECME fit per worker, embarrassingly parallel – usually the better choice for <code>'nlambda > workers'</code>) or <code>"groups"</code> (parallelize the E-step across groups within each, sequential, lambda – better for a single/small grid on data with many groups). Only one level of parallelism is ever used at a time; <code>'workers'</code> are never oversubscribed by nesting both.
variance_step	<code>"ecme"</code> (default) or <code>"em"</code> ; see Details.
refit_at_best	As in <code>[pecme()]</code> 's <code>'refit'</code> , but applied only to the final selected lambda (grid points themselves are fit with <code>'refit = FALSE'</code> to avoid <code>'nlambda'</code> extra unpenalized re-fits).

init_method	"lmer" (default), "zero", or "random". With "random", starting values are drawn from the caller's ongoing global random-number stream (unlike [simulate_pecme_data()]s 'seed', this is NOT locally seed-preserved – by design, since the point of "random" is to give a genuinely different starting point on repeated calls, e.g. for a multi-start convergence check). Call 'set.seed()' yourself beforehand if you need the specific starting values to be reproducible.
max_iter, tol	Outer ECME iteration limit/tolerance.
cd_max_iter, cd_tol	Inner coordinate-descent iteration limit/tolerance.
keep_path	Keep the fitted 'beta' at every grid point (for plotting a regularization path)? Default 'TRUE'.
verbose	Print iteration progress.

Value

An object of class "pecme_grid": a list with '\$best' (the selected "pecme" fit, refit per 'refit_at_best'), '\$lambda', '\$scores' (a data frame of one row per lambda with every criterion, 'df', 'logLik', and convergence status), '\$best_index', and '\$criterion'.

pecme	<i>Fit a penalized ECME censored/missing-response linear mixed model</i>
-------	--

Description

Fits $Y_{ij}^* = x'_{ij}\beta + b_j + e_{ij}$, $e_{ij} \sim N(0, \sigma^2)$, $b_j \sim N(0, \sigma_b^2)$, where the response Y^* may be left-, right-, or interval-censored (and/or missing), by the ECME algorithm of Liu and Rubin (1994). The random intercept is integrated out numerically by fixed-order Gauss-Hermite quadrature (a numerical approximation to the continuous integral, not an exact evaluation of it; see [pecme_quadrature_check()] for a sensitivity diagnostic and 'quadrature' below). β is estimated by a penalized CM-step (coordinate descent); (σ, σ_b) are estimated, by default, with a true ECME CM-step that directly maximizes the observed-data marginal likelihood (not merely its EM lower bound), following the strategy of Vaida and Liu (2009) for censored linear mixed models.

Usage

```
pecme(
  fixed_formula,
  random_var,
  data,
  censor = "none",
  left = NULL,
  right = NULL,
  penalty = c("none", "lasso", "adaptive", "scad", "mcp", "elastic", "ridge"),
  lambda = 0,
  alpha = 0.5,
  gamma = NULL,
```

```

adaptive_gamma = 1,
missing_y = FALSE,
missing_x = c("error", "cca", "mean", "reg"),
standardize = TRUE,
quadrature = 40L,
workers = 1L,
variance_step = c("ecme", "em"),
refit = TRUE,
init_method = c("lmer", "zero", "random"),
max_iter = 200L,
tol = 1e-06,
cd_max_iter = 1000L,
cd_tol = 1e-08,
verbose = FALSE
)

```

Arguments

<code>fixed_formula</code>	A formula for the fixed effects, e.g. <code>'y ~ x1 + x2'</code> .
<code>random_var</code>	Character: name of the grouping (cluster) column in <code>'data'</code> . Currently a random INTERCEPT only (no random slopes).
<code>data</code>	A data frame.
<code>censor</code>	Either <code>"none"/"left"/"right"/"interval"</code> applied to every row, a column name in <code>'data'</code> holding per-row labels from that set, or a vector of the same length as <code>'nrow(data)'</code> . Rows do not need to be pre-labelled <code>"missing"</code> : set <code>'missing_y = TRUE'</code> and leave the response <code>'NA'</code> for those rows.
<code>left, right</code>	Column names or numeric vectors of censoring bounds (used only for the corresponding <code>'censor'</code> labels). With censoring, the response must be supplied on its analysis scale (transformed responses such as <code>'log(y)'</code> are rejected); censoring bounds are never transformed automatically.
<code>penalty</code>	One of <code>"none"</code> , <code>"lasso"</code> , <code>"adaptive"</code> , <code>"scad"</code> , <code>"mcp"</code> , <code>"elastic"</code> , <code>"ridge"</code> .
<code>lambda</code>	Non-negative penalty strength.
<code>alpha</code>	Elastic Net mixing parameter in <code>'[0,1]'</code> (<code>'1'</code> = Lasso, <code>'0'</code> = Ridge); ignored unless <code>'penalty = "elastic"'</code> .
<code>gamma</code>	Extra shape parameter for SCAD (<code>'> 2'</code> , default <code>'3.7'</code>) or MCP (<code>'> 1'</code> , default <code>'3'</code>); ignored otherwise.
<code>adaptive_gamma</code>	Exponent used to build Adaptive Lasso weights from a Ridge pilot fit: <code>'weights = 1 / beta_pilot ^adaptive_gamma'</code> .
<code>missing_y</code>	If <code>'TRUE'</code> , rows with a missing response (or a missing bound for a labelled-censored row) are treated as completely missing at random and IMPUTED via the E-step's posterior mean at every iteration, instead of raising an error. Important: such a row contributes exactly <code>'log(1) = 0'</code> to the observed-data log-likelihood – it carries NO direct information about <code>'(beta, sigma, sigma_b)'</code> ; the imputed value is a model- implied <i>*prediction*</i> for that row given the parameters, not evidence that helped estimate them. The parameters themselves are identified entirely by the observed and censored rows.

missing_x	How to handle 'NA' in the fixed-effects predictors: "error" (default), "cca", "mean", or "reg". The "mean" and "reg" options impute the numeric model matrix and are intended only for suitable numeric predictors. For factors, interactions, splines, or nonlinear terms, impute raw variables before constructing the model matrix (e.g. with 'mice'). See [pecme_handle_missing_x()] and [pecme_mi_combine()].
standardize	Standardize predictors before penalization (recommended; default 'TRUE').
quadrature	Number of Gauss-Hermite nodes for the random intercept (default '40'). This is a fixed-order numerical approximation to the random-intercept integral, not an exact evaluation of it; use [pecme_quadrature_check()] on a fitted model to check sensitivity to this choice, and increase it if 'sigma_b' is large relative to 'sigma', or convergence looks unstable.
workers	Number of parallel worker processes for the E-step (default '1'). Set to e.g. '32' on a 32-core machine.
variance_step	"ecme" (default) or "em"; see Details.
refit	If 'TRUE' (default) and 'penalty != "none"', an additional unpenalized refit restricted to the selected coefficients ("relaxed refit") is stored in '\$refit'. This reduces the shrinkage bias of the penalized estimates for reporting, but it is NOT generally valid unconditional post-selection inference: the selection step itself is not accounted for, so treat its 'se' and p-values as approximate/exploratory, not confirmatory.
init_method	"lmer" (default), "zero", or "random". With "random", starting values are drawn from the caller's ongoing global random-number stream (unlike [simulate_pecme_data()]'s 'seed', this is NOT locally seed-preserved – by design, since the point of "random" is to give a genuinely different starting point on repeated calls, e.g. for a multi-start convergence check). Call 'set.seed()' yourself beforehand if you need the specific starting values to be reproducible.
max_iter, tol	Outer ECME iteration limit/tolerance.
cd_max_iter, cd_tol	Inner coordinate-descent iteration limit/tolerance.
verbose	Print iteration progress.

Value

An object of class "pecme".

References

- Liu, C. and Rubin, D. B. (1994). The ECME algorithm. **Biometrika**, 81(4), 633-648.
- Vaida, F. and Liu, L. (2009). Fast implementation for normal mixed effects models with censored response. **JCGS**, 18(1), 151-162.
- Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. **JASA**, 96(456), 1348-1360.
- Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. **Annals of Statistics**, 38(2), 894-942.

See Also

[grid_pecme()], [pecme_mi_combine()]

pecme_bootstrap

Cluster (group-level) bootstrap for a pecme fit

Description

Resamples whole groups with replacement, refits the identical model (same formula, ‘penalty’, ‘lambda’, ‘alpha’, ‘gamma’) to each resampled data set, and summarizes the resulting coefficient distribution – a valid source of standard errors and confidence intervals for a penalized fit, where naive SEs do not apply.

Usage

```
pecme_bootstrap(
  object,
  R = 200L,
  workers = NULL,
  ci_level = 0.95,
  seed = NULL,
  verbose = FALSE
)
```

Arguments

object	A “pecme” fit.
R	Number of bootstrap replicates (default ‘200’; increase for final reporting, e.g. ‘1000+’).
workers	Parallel workers across replicates (default: reuse ‘object\$workers’).
ci_level	Confidence level for the percentile interval.
seed	Optional integer seed (locally preserved; see [simulate_pecme_data()]).
verbose	Print replicate progress.

Details

Because each replicate is refit at a FIXED penalty/lambda, this captures sampling variability in the coefficient estimates GIVEN that regularization strength, but not uncertainty in the choice of lambda itself, and not formal selection-event-conditional inference for which predictors end up nonzero. A replicate in which a coefficient is shrunk to exactly zero contributes 0 to that coefficient’s bootstrap distribution (not ‘NA’), so the reported bootstrap SE reflects both estimation and selection variability together for that coefficient.

Value

An object of class `"pecme_bootstrap"`: a list with `'table'` (a data frame of `'estimate'`, `'boot_mean'`, `'boot_se'`, and percentile `'ci_lower'`/`'ci_upper'` per coefficient), `'replicates'` (the raw `'R x p'` matrix of bootstrap coefficient estimates), `'R'`, `'n_failed'` (replicates that errored or failed to converge, excluded), and `'ci_level'`.

pecme_effective_df	<i>Working Ridge-type effective degrees-of-freedom approximation</i>
--------------------	--

Description

For Ridge, the naive nonzero-coefficient count from `[pecme_df()]` is a poor stand-in for effective df. The classical Ridge smoother trace is used as a `CONDITIONAL/WORKING` approximation on the standardized fixed- effects design. It is not claimed to be the exact effective df of the complete PECME estimator, which also contains censoring, random effects, estimated variance components, quadrature, and possibly adaptive penalization.

Usage

```
pecme_effective_df(object)
```

Arguments

object A `"pecme"` fit.

Details

No corresponding closed form is implemented here for Elastic Net, SCAD, or MCP (a generalized-df/local-derivative calculation would be needed, e.g. Zou, Hastie & Tibshirani, 2007); for `"none"`/`"lasso"`/`"adaptive"` the naive nonzero count is already a standard, reasonable approximation and is returned unchanged (plus the 2 variance components).

Value

A working/conditional df approximation, including the intercept if present and the two variance components, for `"ridge"`, `"none"`, `"lasso"`, or `"adaptive"` fits; `'NA_real_'` (with an explanatory `'attr(, "note")'`) for `"elastic"`, `"scad"`, or `"mcp"`, where no closed form is implemented.

References

Hastie, T. and Tibshirani, R. (1990). **Generalized Additive Models**. Chapman and Hall. (Ridge effective df / "trace of the hat matrix".)

Zou, H., Hastie, T. and Tibshirani, R. (2007). On the "degrees of freedom" of the lasso. **Annals of Statistics**, 35(5), 2173-2192.

pecme_lambda_grid	<i>Build a log-spaced lambda grid</i>
-------------------	---------------------------------------

Description

Build a log-spaced lambda grid

Usage

```
pecme_lambda_grid(
  lambda_max,
  nlambda = 50L,
  lambda_min_ratio = NULL,
  n = NULL,
  p = NULL
)
```

Arguments

lambda_max	Upper end of the grid.
nlambda	Number of grid points.
lambda_min_ratio	'lambda_min = lambda_min_ratio * lambda_max'. Defaults follow the usual 'n > p' vs 'n <= p' heuristic ('1e-4' / '1e-2' respectively) unless supplied explicitly.
n	Number of observations.
p	Number of predictors.

Value

A numeric vector of length 'nlambda' containing a decreasing log-spaced sequence from 'lambda_max' to 'lambda_min'.

pecme_loo	<i>Leave-one-group-out (or leave-one-fold-out) cross-validated log-likelihood</i>
-----------	---

Description

Refits the model with each group (or fold of groups) held out in turn and evaluates the held-out marginal log-likelihood at the refit parameters – a genuine, frequentist out-of-sample predictive criterion for clustered data (leave-one-*observation*-out is not meaningful here since observations within a group share a random intercept).

Usage

```
pecme_loo(object, method = c("exact", "kfold"), folds = 10L, workers = NULL)
```

Arguments

object	A "pecme" fit (from [pecme()] or 'grid_pecme()\$best').
method	Character string specifying the resampling method.
folds	Integer specifying the number of folds for K-fold cross-validation.
workers	Parallel workers for the LOO refits (default: reuse 'object\$workers').

Value

A list with 'elpd_loo' (summed held-out log-likelihood, on the same scale as 'logLik'), 'se' (a naive standard error across successful group/fold contributions), 'n_successful_folds', and 'pointwise' (a data frame of one row per group/fold).

pecme_metrics	<i>Comprehensive fit-quality metrics for a pecme model</i>
---------------	--

Description

Collects the quantities typically reported for a mixed-model fit: pointwise prediction error (MSE/RMSE/MAE/MAPE/SMAP), two kinds of pseudo-R-squared, information criteria, and (optionally) a cross-validated predictive log-likelihood.

Usage

```
pecme_metrics(
  object,
  newdata = NULL,
  newresponse = NULL,
  loo = FALSE,
  loo_method = c("exact", "kfold"),
  loo_folds = 10L,
  workers = NULL,
  ebic_gamma = 0.5
)
```

Arguments

object	A "pecme" fit (from [pecme()] or 'grid_pecme()\$best').
newdata, newresponse	Optional: evaluate prediction-error metrics out-of-sample instead of on the training data. 'newdata' needs the predictor and grouping columns. The observed response is looked up automatically from 'newdata' using the same column name as in the fitted model's formula ('response' has no independent role –

	see [pecme()]); pass 'newresponse' explicitly only if 'newdata' does not itself contain that column (e.g. truly held-out rows with unknown outcomes, being scored on other rows' observed values). Rows with a missing/'NA' response (however obtained) are silently dropped from the error metrics, since there is no single observed value to compare a point prediction against.
loo	If 'TRUE', additionally compute a cross-validated predictive log-likelihood by actually refitting the model with one group ('loo_method = "exact"') or one fold of groups ('loo_method = "kfold"') held out at a time, and evaluating the held-out marginal log-likelihood. This is a genuine, if expensive, out-of-sample criterion; see Details for why it is not the PSIS-LOO of the 'loo' package.
loo_method	"exact" (leave-one-group-out; 'J' refits) or "kfold" (leave-one-fold-out; 'loo_folds' refits).
loo_folds	Number of folds when 'loo_method = "kfold"'.
workers	Parallel workers for the LOO refits (default: reuse 'object\$workers').
ebic_gamma	Extended-BIC tuning constant (see [grid_pecme()]); only affects the recomputed 'EBIC' entry.

Details

Why there is no WAIC WAIC (and DIC) are defined from the variance of the pointwise log-likelihood *across posterior draws* of the parameters. 'pecme' is a frequentist penalized-likelihood estimator: it returns a single point estimate (plus, for an unpenalized fit, an asymptotic Hessian-based covariance), not a posterior sample. Reporting a WAIC number here would therefore not be a real WAIC – rather than approximate it, '\$WAIC' is returned as 'NA' with an explanatory 'attr'. For out-of-sample model comparison, use 'AIC'/'BIC'/'EBIC' (fast, from the fitted likelihood) or 'loo = TRUE' (slower, but a genuine held-out predictive log-likelihood, which is the same quantity WAIC and PSIS-LOO both aim to approximate).

Two R-squared values 'R2_marginal' and 'R2_conditional' follow Nakagawa & Schielzeth (2013): 'R2_marginal = Var(fixed) / (Var(fixed) + sigma_b^2 + sigma^2)' uses only the fixed-effects predictor's variance in the numerator; 'R2_conditional' additionally credits the random intercept, '(Var(fixed) + sigma_b^2) / (Var(fixed) + sigma_b^2 + sigma^2)'. These are the R-squared values normally reported for a mixed model. 'R2_predictive' = '1 - MSE / Var(y)' is the more familiar OLS-style quantity computed directly from predictions; it is provided for intuition/comparability but is not the standard mixed-model R-squared. 'R2_predictive_adj' heuristically applies the usual OLS adjusted-R2 formula with 'object\$df'; this adjustment is not a rigorously derived quantity for mixed models and should be reported as a diagnostic, not a citation-grade statistic.

Value

An object of class "pecme_metrics" (a list; has a 'print' method).

pecme_mi_combine	<i>Combine ‘pecme’ fits across multiply-imputed data sets (Rubin’s rules)</i>
------------------	---

Description

A convenience helper for the recommended workflow when fixed-effects covariates (not the response) have missing values: impute ‘m’ complete data sets externally (e.g. with the ‘mice’ package), fit the **same** ‘pecme()’ model to each, and combine with Rubin’s rules.

Usage

```
pecme_mi_combine(fits, use_refit = FALSE)
```

Arguments

<code>fits</code>	A list of ‘m’ objects of class “pecme”, all fit with identical ‘penalty’/‘lambda’/‘alpha’/‘gamma’ and the same fixed coefficient names/order.
<code>use_refit</code>	If ‘TRUE’, pool each fit’s ‘\$refit’ sub-fit instead of ‘fits’ themselves (see Details); required whenever ‘fits’ are penalized.

Details

Rubin’s rules require a (classically) valid standard error for every fit being combined. A penalized fit (‘penalty != “none”’ and ‘lambda > 0’) deliberately stores ‘se = NA’ (see [pecme()]): naive standard errors are not valid for a shrinkage estimate, so silently pooling ‘NA’s here would produce ‘NA’ within-imputation variance and a misleading pooled result. Because of this, ‘pecme_mi_combine()’:

* by default, **REQUIRES** every element of ‘fits’ to have valid SEs (‘se_valid == TRUE’, i.e. ‘penalty == “none”’ or ‘lambda == 0’) and errors otherwise, naming which elements fail and why; * with ‘use_refit = TRUE’, instead pools each fit’s ‘\$refit’ – the automatic unpenalized “relaxed refit” on the selected predictors (see [pecme()])’s ‘refit’ argument) – which does have valid SEs. This is a practical compromise, not a rigorous solution: (a) the relaxed refit is not valid unconditional post-selection inference to begin with (its SEs ignore the selection step); (b) different imputed data sets can select different variable subsets, so a coefficient absent from one imputation’s refit is treated as exactly 0 (with ‘se = 0’) in that imputation for pooling purposes, which folds selection instability into the between-imputation variance rather than separating it out. Report results obtained this way as approximate/exploratory.

For fully rigorous inference on a penalized, multiply-imputed fit, see [pecme_bootstrap()] (a cluster bootstrap) applied within each imputed data set, or consult a specialist reference on selective/ post-selection inference under multiple imputation.

Value

A data frame with the pooled estimate, within-imputation variance, between-imputation variance, total variance, and pooled standard error for each fixed-effect coefficient.

 pecme_quadrature_check

Check sensitivity of a fitted model to the Gauss-Hermite quadrature order

Description

Fixed-order Gauss-Hermite quadrature is a numerical approximation to the (otherwise intractable) integral over the random intercept, not an exact evaluation of it. This function reports how much the fit changes as the number of quadrature nodes changes, so ‘quadrature’ is a checked choice rather than an unexamined default. Approximation error is typically largest when ‘sigma_b’ is large relative to ‘sigma’, or with heavy censoring/many missing responses.

Usage

```
pecme_quadrature_check(
  object,
  quadrature_values = c(15L, 20L, 30L, 50L, 80L, 120L),
  method = c("loglik", "refit")
)
```

Arguments

object	A “pecme” fit.
quadrature_values	Integer vector of node counts to compare against ‘object’s own ‘quadrature’, which is always included.
method	“loglik” (default; cheap: re-evaluates the marginal log-likelihood at ‘object’s already-fitted ‘(beta, sigma, sigma_b)’ under each node count, with no re-optimization) or “refit” (expensive: fully refits the model – including its ‘beta’, ‘sigma’, ‘sigma_b’ – at each node count, showing whether the actual parameter *estimates*, not just the likelihood value at a fixed point, are sensitive to the quadrature order).

Value

An object of class “pecme_quadrature_check”: a data frame (‘quadrature’, ‘logLik’, and, for ‘method = “refit”’, ‘sigma’, ‘sigma_b’, ‘max_abs_beta_change’ relative to ‘object’), with a ‘print’ method that flags whether the range of ‘logLik’ values looks small relative to typical evidence thresholds.

Examples

```
sim <- simulate_pecme_data(n_groups = 25, n_per_group = 8, p = 3, seed = 1)
fit <- pecme(y ~ x1 + x2 + x3, random_var = "group", data = sim$data)
pecme_quadrature_check(fit)
```

plot.pecme_grid	<i>Plot the regularization path and/or selection-criterion curve</i>
-----------------	--

Description

Plot the regularization path and/or selection-criterion curve

Usage

```
## S3 method for class 'pecme_grid'
plot(x, which = c("both", "path", "criterion"), ...)
```

Arguments

x	A "pecme_grid" object.
which	"path" (coefficient paths vs 'log(lambda)'), "criterion" (the selection criterion vs 'log(lambda)'), or "both" (default).
...	Additional graphical arguments. Currently ignored.

Value

Invisibly returns the input 'x', an object of class "pecme_grid".

predict.pecme	<i>Predict from a fitted pecme model</i>
---------------	--

Description

Predict from a fitted pecme model

Usage

```
## S3 method for class 'pecme'
predict(
  object,
  newdata = NULL,
  level = c("population", "group"),
  use_refit = FALSE,
  se_fit = FALSE,
  ...
)
```

Arguments

object	A "pecme" fit.
newdata	Optional data frame with the predictor columns (and, for 'level = "group"', the grouping column) used to fit 'object'. If omitted, fitted values on the training data are returned.
level	"population" (fixed effects only, 'X beta') or "group" ('X beta + b_j' for known groups; groups absent from the training data effectively use 'b_j = 0', i.e. the population mean).
use_refit	Use the post-selection refit coefficients (see 'object\$refit') instead of the penalized estimates, if available.
se_fit	Also return standard errors (only available when 'object\$sse_valid' is 'TRUE', i.e. an unpenalized/lambda=0 fit; a naive delta-method approximation that ignores 'sigma_b' uncertainty).
...	Additional arguments passed to the prediction method.

Value

If 'se_fit = FALSE', a numeric vector of predictions. If 'se_fit = TRUE', a list with components 'fit' and 'se.fit', containing the predicted values and their estimated standard errors, respectively.

simulate_pecme_data	<i>Simulate data from a censored/missing random-intercept linear mixed model</i>
---------------------	--

Description

Generates $Y_{ij}^* = \beta_0 + x'_{ij}\beta + b_j + e_{ij}$, $e_{ij} \sim N(0, \sigma^2)$, $b_j \sim N(0, \sigma_b^2)$, with 'p' standard-normal (optionally correlated) predictors, of which only 'n_active' have a non-zero coefficient – convenient for checking variable-selection performance. Censoring and/or missingness can be injected on top of the latent 'y_star'.

Usage

```
simulate_pecme_data(
  n_groups = 30L,
  n_per_group = 10L,
  p = 10L,
  n_active = 4L,
  beta_active = 1.5,
  rho = 0,
  sigma = 1,
  sigma_b = 0.7,
  censor_type = c("none", "left", "right", "interval"),
  censor_prob = 0,
  missing_prob = 0,
  seed = NULL
)
```

Arguments

<code>n_groups</code>	Number of clusters/groups.
<code>n_per_group</code>	Observations per group (recycled if a vector).
<code>p</code>	Number of predictors (in addition to the intercept).
<code>n_active</code>	Number of predictors with a non-zero true coefficient (the rest are exactly zero – useful for checking variable selection).
<code>beta_active</code>	Non-zero coefficient magnitude used for the ‘ <code>n_active</code> ’ active predictors (sign alternates).
<code>rho</code>	Pairwise predictor correlation (compound symmetry), ‘0’ for independent predictors.
<code>sigma, sigma_b</code>	Residual and random-intercept standard deviations.
<code>censor_type</code>	“none”, “left”, “right”, or “interval” applied to a ‘ <code>censor_prob</code> ’ fraction of observations (chosen independently at random); the rest are “none”.
<code>censor_prob</code>	Fraction of observations to censor.
<code>missing_prob</code>	Fraction of (non-censored) observations to set completely missing (‘NA’) in the response.
<code>seed</code>	Optional integer seed for reproducibility. The caller’s global RNG state is saved and restored on exit, so passing ‘seed’ has no side effect on random draws made elsewhere in the calling session (unlike a bare ‘set.seed()’ call).

Value

A list with ‘`data`’ (a data frame with columns ‘`y`’, ‘`x1..xp`’, ‘`group`’, ‘`censor`’, ‘`left`’, ‘`right`’), ‘`beta_true`’ (length ‘`p + 1`’, intercept first), ‘`sigma_true`’, ‘`sigma_b_true`’, and ‘`y_star`’ (the uncensored, non-missing latent response, for validation).

Examples

```
sim <- simulate_pecme_data(n_groups = 20, n_per_group = 8, p = 6,
                          n_active = 3, censor_type = "left",
                          censor_prob = 0.2, seed = 1)

head(sim$data)
```

Index

`grid_pecme`, [2](#)

`pecme`, [5](#)

`pecme_bootstrap`, [8](#)

`pecme_effective_df`, [9](#)

`pecme_lambda_grid`, [10](#)

`pecme_loo`, [10](#)

`pecme_metrics`, [11](#)

`pecme_mi_combine`, [13](#)

`pecme_quadrature_check`, [14](#)

`plot.pecme_grid`, [15](#)

`predict.pecme`, [15](#)

`simulate_pecme_data`, [16](#)