Package 'nlmixr2scm'

September 24, 2026

Title Stepwise Covariate Modeling for 'nlmixr2' Models

Version 0.4

Description Stepwise covariate modeling (SCM) for nonlinear mixed-effects models fitted with 'nlmixr2'. Forward inclusion and backward elimination are driven by likelihood-ratio tests, and the covariate terms are generated inside the model body, so continuous covariates are centered and categorical covariates expanded into indicator columns without editing the model by hand. Candidate fits can be cached and resumed, fitted in parallel, and reviewed through per-step and all-candidate summary tables. The approach follows Jonsson and Karlsson (1998) <doi:10.1023/A:1011970125687>, and the implementation in 'Perl-speaks-NONMEM' described by Lindbom, Ribbing and Jonsson (2004) <doi:10.1016/j.cmpb.2003.11.003>.

Depends R (>= 4.1)

License GPL (>= 3)

URL https://github.com/nlmixr2/nlmixr2scm

BugReports https://github.com/nlmixr2/nlmixr2scm/issues

Imports checkmate, cli (>= 3.4.0), data.table, lotri, nlme, nlmixr2est (>= 7.0.0), nlmixr2utils (>= 0.3), rxode2 (>= 5.0.0), stats, tools, utils

Suggests future, knitr, nlmixr2data, pkgload, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Justin Wilkins [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-7099-9396>), Matthew Fidler [aut] (ORCID: <https://orcid.org/0000-0001-8538-6691>), Yaping Liu [aut] (ORCID: <https://orcid.org/0009-0009-2842-3797>),

Bill Denney [aut] (ORCID: <https://orcid.org/0000-0002-5759-428X>),
Vipul Mann [aut],
Vishal Sarsani [aut] (ORCID: <https://orcid.org/0000-0002-9272-3438>),
Christian Bartels [ctb]

Maintainer Justin Wilkins <justin.wilkins@occams.com>

Repository CRAN

Date/Publication 2026-09-24 13:30:14 UTC

Contents

runSCM 2

Index **8**

`runSCM` *Stepwise Covariate Model-selection (SCM) method*

Description

Stepwise Covariate Model-selection (SCM) method

Usage

```
runSCM(
  fit,
  varsVec = NULL,
  covarsVec = NULL,
  pVal = list(fwd = 0.05, bck = 0.01),
  catvarsVec = NULL,
  pairsVec = NULL,
  shapes = "power",
  customShapes = NULL,
  centers = NULL,
  inits = list(),
  missingToken = NA,
  includedRelations = NULL,
  data = NULL,
  catCutoff = 0.05,
  outputDir = NULL,
  saveModels = TRUE,
  verbose = FALSE,
  control = NULL,
  print = 100,
  searchType = c("scm", "forward", "backward"),
  restart = FALSE,
  workers = NULL,
```

```
  confirm = TRUE,
  profileInit = FALSE,
  profileInitOnStall = TRUE,
  stallTol = 0,
  maxRetries = 3L,
  maxDeltaOFV = Inf,
  retryPerturbSD = 0.5,
  retrySmallInit = 0.01,
  retryOFVTolerance = NULL,
  retryFailOnExhaustion = FALSE,
  rxThreads = NULL
)
```

Arguments

fit	an nlmixr2 'fit' object
varsVec	character vector of PK parameters to which covariates may be added (e.g. c("ka", "cl", "v")). Ignored when pairsVec is provided.
covarsVec	character vector of continuous covariate names to test. Ignored when pairsVec is provided.
pVal	a named list with names fwd and bck for the forward and backward p-value thresholds; default list(fwd = 0.05, bck = 0.01)
catvarsVec	character vector of categorical covariate names. Dummy columns are created internally and appended to covarsVec before building the candidate set.
pairsVec	optional data frame with columns var and covar (or a list of list(var=, covar=) items) that explicitly enumerates the parameter-covariate pairs to test. When supplied, the cartesian product of varsVec x covarsVec is bypassed. Categorical dummy columns must already be present in the dataset; use the expanded covariate name (e.g. covar = "sex_male").
shapes	character vector of shape names to test for each continuous covariate-parameter pair. Built-in shapes are "power" (default; log(cov/median)), "lin" (cov - median), "log" (log(cov)), and "identity" (raw covariate). Per-pair overrides can be specified via the shapes element of individual pairsVec list items. Categorical covariates always use the "cat" shape regardless of this setting.
customShapes	named list of additional shape builder functions of the form function(col, center, level) -> character. These are merged with the built-in shapes and take precedence over them.
centers	optional named numeric vector of fixed reference (centering) values for continuous covariates, e.g. c(BW = 70, CrCL = 95). When supplied, the named covariates are centered on these fixed values in every fit instead of the per-dataset median. This keeps the estimated covariate coefficients (and the structural intercept) on the SAME reference across datasets and matches a data-generating model that used the same fixed references. Names are matched against the raw covariate column; unmatched names are ignored and covariates not named here fall back to the median.

inits — named list mapping shape names to initial theta estimates (and optionally bounds) for the covariate parameter. Each entry may be either a scalar estimate (e.g. `list(power = 0.1, lin = 0.01, cat = 0.01)`) or a named list with elements `est`, `lower`, and `upper` (e.g. `list(power = list(est = 0.1, lower = -5, upper = 5),lin = 0.01)`). Applied globally to all pairs; a per-pair `inits` element in a `pairsVec` list item takes precedence. Shapes not listed default to `est = 0`, `lower = -Inf`, `upper = Inf`. When a covariate is accepted, the estimated value from that step is automatically used as the starting estimate for all subsequent steps (warm-starting); bounds are preserved across steps.

missingToken — sentinel value used to identify missing covariate observations in addition to `NA`. Default `NA` (only true `NA` values are treated as missing). Set to a numeric value (e.g. `-99`) or a character string (e.g. `"."`) to also flag that sentinel as missing. When missing observations are detected for a covariate, the generated model expression is wrapped in an `ifelse()` guard so that the covariate contribution corresponds to the population-typical value (the observed median for continuous covariates, the mode for categorical). For continuous covariates the fill is shape-aware: it is the shape expression evaluated at `cov = median`, which is `0` for `"power"` and `"lin"` (centered shapes), `log(median)` for `"log"`, and `median` itself for `"identity"`. For categorical covariates the indicator is set to the mode of the observed levels.

includedRelations — optional specification of covariate-parameter relationships that must be present in the model at the start of backward elimination (analogous to PsN's `[included_relations]`). Can be a data frame with columns `var` and `covar` (and optionally `shape` or `shapes`), or a list of `list(var=, covar=, shapes=)` items in the same format as `pairsVec`. Relations already in the forward-search result are left untouched; missing ones are added to the model and a single re-fit is performed before backward elimination begins. Included relations are eligible for removal during backward elimination like any other covariate. Ignored when `searchType = "forward"`.

data — data frame containing all subjects and columns required by the search (covariates, dose/observation records, etc.). When `NULL` (default), `nlme::getData(fit)` is used, which may omit covariate columns that are not referenced by the base model. Pass the data set explicitly whenever the base model does not reference a covariate column that will be used in the search (e.g. a categorical covariate such as `sex`). No column transformations should be applied before passing the data: centering and other shape transformations are generated inside the model body by the SCM machinery.

catCutoff — minimum proportion of subjects that must belong to a non-reference level for it to be tested. Levels below this threshold are lumped with the reference and excluded from the candidate set. Applied per unique subject (not per observation row). Default `0.05` (5\ Set to `0` to test all non-reference levels regardless of frequency.

outputDir — character; path of the subdirectory used to store all fitted model `.rds` files. When `NULL` (default), the name is derived automatically as `<fit_name>_scm_<N>` where `N` is one greater than the number of existing `<fit_name>_scm_*` directories in the current working directory (so repeated runs produce `fit_scm_1`, `fit_scm_2`,

...). If the resolved directory already exists, the search stops with an error; set `restart = TRUE` to back it up and start fresh instead. Ignored when `saveModels = FALSE`.

`saveModels` logical; if `TRUE` (default) every fitted candidate model accepted during forward or backward search is written to `outputDir` as an `.rds` file. Set to `FALSE` to run the search without writing any files.

`verbose` logical; if `TRUE`, print additional detail at each search step: the full candidate table (with covariate expression and starting estimate) before fitting, the complete results table for every candidate after fitting (sorted by p-value), and–when a covariate is accepted–the accepted model's fixed-effect estimates and diagonal omega variances. Default `FALSE` (only the summary line and accepted- model best row are printed, as at present).

`control` optional control object (e.g. `saemControl()`, `foceiControl()`) passed to every `nlmixr2()` call during the search. When `NULL` (default), the control settings are inherited from the base `fit` object. The `print` argument always overrides `control$print` regardless of which source is used.

`print` integer; how often (in iterations) the optimiser prints progress for each candidate model fit. Default `100`. Set to `0` to suppress all iteration output, or `1` to print every iteration. Always applied as `control$print`, overriding any value in a user-supplied `control` object.

`searchType` one of `'scm'`, `'forward'`, or `'backward'`; default `'scm'`

`restart` logical; if `TRUE` the existing cache is backed up and the search restarts from scratch; default `FALSE`

`workers` integer or `NULL`; number of parallel workers to use when fitting candidate models. When `NULL` (default), the current `future` plan is used unchanged. A positive integer temporarily switches to a `multisession` plan with that many workers for the duration of the search.

`confirm` logical; if `TRUE` (default) and the session is interactive, the user is prompted to confirm before the search begins. Set to `FALSE` to skip the prompt (useful in scripts or tests).

`profileInit` logical; if `TRUE`, each forward candidate's new covariate coefficient is warm-started with a cheap 1-D FOCEi profile on a frozen base (all structural thetas fixed at their parent estimates and the between-subject variability FIXED – not zeroed – at its parent values) before the real estimator runs. Keeping the random effects intact matters: profiling a covariate on a fixed-effect-only model is misspecified and can return the wrong sign. This supplies gradient optimisers (`nlminb`, `lbfgsb3c`) with a nonzero, gradient-informative starting value so they do not stall at the flat zero-effect point. bobyqa is never involved in the caller; the profiled value is handed back to the fit's own estimator. Default `FALSE`.

`profileInitOnStall`
logical; if `TRUE` (default), a forward candidate whose ordinary fit *stalls* – i.e. the nested model's OFV improvement over its parent is `<= stallTol` (a nested model can never be genuinely worse than its parent at a true optimum, so this signals the outer optimiser never left the covariate init) – triggers a one-shot `.profileCovInit()` frozen 1-D profile. The profiled coefficient is then used as the init for a rescue refit. Unlike `profileInit`, this fires only when a stall is

detected, so healthy candidates (e.g. analytic `linCmt()` fits) pay no extra cost. It is the fix for ODE covariate candidates that stall at their init under solver-noise-flattened outer objectives. Default `TRUE`.

`stallTol` numeric; OFV-improvement threshold below which a forward candidate is considered stalled and eligible for the profile rescue. Default `0` (any nested model no better than its parent).

`maxRetries` integer; maximum number of retry attempts per candidate when the OFV is deemed unrealistic. Default `3L`. Set to `0` to disable the retry mechanism entirely.

`maxDeltaOFV` numeric; absolute ceiling on plausible OFV change. A candidate whose `|dOFV|` exceeds this value is flagged as unrealistic and retried. Default `Inf` (disabled; only criterion 1 and 2 apply).

`retryPerturbSD` numeric; standard deviation of the log-normal perturbation applied to `cov_init` on odd-numbered retries. Default `0.5`.

`retrySmallInit` numeric; covariate theta init used on even-numbered retries as a near-zero safe-harbour. Default `0.01`.

`retryOFVTolerance`
numeric or `NULL`; the margin (in OFV units) by which the candidate OFV must exceed the parent before a retry is triggered (criterion 1). `NULL` (default) auto-detects: `10` for SAEM (stochastic noise), `0` for all other estimators.

`retryFailOnExhaustion`
logical; when `TRUE` and all retry attempts are exhausted with an unrealistic OFV, the candidate is marked as failed and excluded from the search. When `FALSE` (default) a warning is emitted and the best available result is accepted.

`rxThreads` integer, `"auto"`, or `NULL`; number of rxode2 OpenMP threads to use per worker while fitting candidate models. When `NULL` (default), the current `rxode2::getRxThreads()` value is used and applied consistently to every worker. `"auto"` divides the total core count evenly across the effective number of workers. Whenever more than one worker is involved, `workers * rxThreads` must not exceed the machine's core count – `runSCM()` aborts with an explanatory error before any fitting starts if it would, since each parallel worker is a separate process that runs its own independent rxode2 thread pool. A single worker is never subject to this check. Note this means an existing `workers > 1` or `workers = "auto"` call that does not also set `rxThreads` may now abort where it previously ran silently oversubscribed; set `rxThreads` explicitly (e.g. `rxThreads = 1`) to restore the prior behavior.

Value

A list with elements `summaryTable` (combined forward and backward results), `resFwd` (list of final fit and step table from forward search), and `resBck` (same for backward search).

Author(s)

Vipul Mann, Matthew Fidler, Vishal Sarsani, Justin Wilkins

Examples

```
# Fitting the base model and running even a one-pair search takes longer
# than a few seconds, so this example is wrapped in \donttest{}.
one.cmt <- function() {
  ini({
    tka <- 0.45
    label("Ka")
    tcl <- log(c(0, 2.7, 100))
    label("Cl")
    tv <- 3.45
    label("V")
    eta.ka ~ 0.6
    eta.cl ~ 0.3
    eta.v ~ 0.1
    add.sd <- 0.7
  })
  model({
    ka <- exp(tka + eta.ka)
    cl <- exp(tcl + eta.cl)
    v <- exp(tv + eta.v)
    linCmt() ~ add(add.sd)
  })
}

fit <- nlmixr2est::nlmixr2(
  one.cmt, nlmixr2data::theo_sd,
  est = "focei", control = nlmixr2est::foceiControl(print = 0)
)

# Test one parameter-covariate pair.  saveModels = FALSE keeps the search
# from writing anything to disk.
res <- runSCM(fit,
  pairsVec = list(list(var = "cl", covar = "WT", shapes = "power")),
  searchType = "forward",
  saveModels = FALSE,
  confirm = FALSE,
  print = 0,
  workers = 1L,
  rxThreads = 2L
)
res$summaryTable
```

Index

runSCM, 2