Package: nlmixr2targets (via r-universe)

June 5, 2026

Title Targets for 'nlmixr2' Pipelines

Version 0.1.0

Description 'nlmixr2' often has long runtimes. A pipeline toolkit tailored to 'nlmixr2' workflows leverages 'targets' and 'nlmixr2' to ease reproducible workflows. 'nlmixr2targets' ensures minimal rework in model development with 'nlmixr2' and 'targets' by simplifying and standardizing models and datasets.

License GPL (>= 2)

URL https://nlmixr2.github.io/nlmixr2targets/

BugReports https://github.com/nlmixr2/nlmixr2targets/issues

Depends R (>= 4.1)

Imports checkmate, digest, nlmixr2est, rxode2 (>= 2.0.14), targets

Suggests covr, knitr, nlmixr2data, rmarkdown, spelling, tarchetypes, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

Encoding UTF-8

Language en-US

NeedsCompilation no

Author Bill Denney [aut, cre] (ORCID: <https://orcid.org/0000-0002-5759-428X>)

Maintainer Bill Denney <wdenney@humanpredictions.com>

Config/pak/sysreqs cmake libgmp3-dev make libmpfr-dev

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

Date/Publication 2026-06-05 14:40:11 UTC

RemoteUrl https://github.com/cran/nlmixr2targets

RemoteRef HEAD

RemoteSha cff7b89b815724bb52e080b2deb9b1595d4bb26f

Contents

`assign_origData` *Replace the fit data with the original data, then return the modified fit*

Description

This function is intended for use within `nlmixr2targets` target creation, and it's not typically invoked by users.

Usage

```
assign_origData(fit, data)
```

Arguments

`fit`	an estimated `nlmixr2` object
`data`	the data from the original fit

Value

The fit with the data added back in as `fit$env$origData`

`nlmixr_data_simplify` *Standardize and simplify data for nlmixr2 estimation*

Description

This function is typically not needed by end users.

Usage

```
nlmixr_data_simplify(
  data,
  object,
  table = list(),
  directory = file.path(targets::tar_config_get("store"), "user/nlmixr2")
)
```

Arguments

`data`	nlmixr data
`object`	Either an `nlmixr` ui object (e.g. the output of running `nlmixr(object = model)`) or a character md5 hash identifying a simplified ui in the `nlmixr2targets` indirect cache (the form used in targets generated by `tar_nlmixr()`).
`table`	The output table control object (like 'tableControl()')
`directory`	Cache directory to load the simplified `nlmixrui` from. Defaults to `file.path(targets::tar_config_g` `"user/nlmixr2")`, mirroring the convention used by `targets`' own `store = targets::tar_config_get("store")` defaults — the path resolves at call time against whatever store the user has configured for the running `tar_make()` (e.g. a `tempdir()` location set via `targets::tar_config_set()`).

Details

The standardization keeps columns that rxode2 and nlmixr2 use along with the covariates. Column order is standardized (rxode2 then nlmixr2 then alphabetically sorted covariates), and rxode2 and nlmixr2 column names are converted to lower case.

Value

The data with the nlmixr2 column lower case and on the left and the covariate columns on the right and alphabetically sorted.

See Also

Other Simplifiers: `nlmixr_object_complicate()`, `nlmixr_object_simplify()`

`nlmixr_object_complicate` *Re-attach labels, metadata, and original data to a simplified fit*

Description

The inverse of `nlmixr_object_simplify()`. Given a fit produced from the simplified, label-and-meta-stripped model, plus the original model function and the original data, this function:

Usage

```
nlmixr_object_complicate(fit, object, data)
```

Arguments

`fit`	An estimated nlmixr2 fit produced from the simplified model.
`object`	The original model function (or object) the fit was derived from. Its labels and metadata are read back onto `fit`.
`data`	The original data the fit corresponds to (before `nlmixr_data_simplify()` reduced its column set). Must have the same number of rows as the data currently stored on the fit, mirroring `assign_origData()`.

Details

- re-derives parameter labels and the metadata environment from `object` and writes them back onto `fit$ui$iniDf$label` and `fit$ui$meta`, and
- replaces `fit$env$origData` with the original `data`.

This is what makes label/meta edits in the source model cheap under `targets`: the cache hash for the simplified model object is independent of labels and metadata, so `tar_make()` only re-runs this re-attachment step (and the cheap `_object_simple` step) when only labels or metadata change.

This function is typically not invoked directly by end users; it is the command for the final target produced by `tar_nlmixr()`.

Value

The modified `fit`.

See Also

`nlmixr_object_simplify()`, `assign_origData()`.

Other Simplifiers: `nlmixr_data_simplify()`, `nlmixr_object_simplify()`

`nlmixr_object_simplify` *Simplify an nlmixr object*

Description

This function is typically not needed by end users.

Usage

```
nlmixr_object_simplify(
  object,
  directory = file.path(targets::tar_config_get("store"), "user/nlmixr2")
)
```

Arguments

`object`	Fitted object or function specifying the model.
`directory`	Cache directory to load the simplified `nlmixrui` from. Defaults to `file.path(targets::tar_config_g` `"user/nlmixr2")`, mirroring the convention used by `targets`' own `store = targets::tar_config_get("store")` defaults — the path resolves at call time against whatever store the user has configured for the running `tar_make()` (e.g. a `tempdir()` location set via `targets::tar_config_set()`).

Details

The object simplification removes comments (so please use `label()` instead of comments to label parameters) and then converts the `object` to a "nlmixrui" object.

Object metadata (`ui$meta`) and parameter labels (`ui$iniDf$label`) are also stripped from the simplified object before it is written to the indirect cache. They do not affect estimation, and stripping them keeps the cache hash stable across edits to either, so editing only labels or metadata will not invalidate the cached fit. The stripped values are restored on the final fit by `nlmixr_object_complicate()`, which reads them straight back off the original model.

The natural nlmixr2 DSL form for compartment initial conditions (`cmt(0) <- value` inside a `model({...})` block) trips `targets`' static analysis because `codetools::findGlobals()` interprets it as a replacement-function assignment with a non-symbol target. `tar_nlmixr()` auto-rewrites `cmt(0) <- value` to `cmt(initial) <- value` inside `model({...})` blocks at construction time (mutating the user's model function in env), and converts back to `cmt(0) <- value` before nlmixr2 sees the model. The user can therefore write `cmt(0) <- value` directly.

Manual `cmt(initial) <- value` is also accepted, but it is a `nlmixr2targets`-only workaround: bare nlmixr2 does not understand the `cmt(initial)` form, so a model function written this way only fits when routed through `tar_nlmixr()` / `tar_nlmixr_multimodel()`. Note: because the rewrite mutates the function in env, calling the model function directly (outside `tar_make()`) after `tar_nlmixr()` will see `cmt(initial)` in its body.

The simplified model's `model.name` is always set to `"object"`. This keeps the simplified output stable so that the MD5 hash used by the `targets` indirect cache is independent of the symbol the caller bound the model function to.

Value

The MD5 hash used to load the simplified `nlmixrui` object back from the `nlmixr2targets` indirect cache.

See Also

`nlmixr_object_complicate()` for the inverse operation that re-attaches labels, metadata, and the original data on the final fit.

Other Simplifiers: `nlmixr_data_simplify()`, `nlmixr_object_complicate()`

`nlmixr_object_zero_initial_eval` *Runtime helper: undo the* `cmt(0) -> cmt(initial)` *rewrite before evaluating the captured* `object` *expression.*

Description

Wrapped around the captured expression at construction time when `tar_nlmixr_protect_zero_initial()` performed any rewrites. Needed for pipe forms like `pheno |> model({...})` and `pheno |> ini(...)`, because nlmixr2's pipe handlers parse pheno's body before the existing `nlmixr_object_simplify_zero_initial_helper(` could intervene. Harmless for symbol-only forms (the existing helper reaches them at simplify time anyway).

Usage

```
nlmixr_object_zero_initial_eval(expr, envir = parent.frame())
```

Arguments

`expr`	A quoted (unevaluated) language object.
`envir`	The parent environment for evaluation. Defaults to the caller's frame (the targets execution env).

Details

Mechanism:

- Walk the quoted expression, rewriting `name(initial) <- val` back to `name(0) <- val` via the existing inverse helper.
- Walk the (rewritten) expression for symbols. For each that resolves to a function whose body still contains `name(initial) <- val` inside `model({...})`, build a corrected closure with `name(0) <- val` and bind it in an override frame.
- Evaluate the rewritten expression in the override frame (whose parent is `envir`), so any lookup of a rewritten symbol hits the corrected closure first.

The corrected copies retain the original closure environment of the user's function; only the body is swapped.

Value

The result of evaluating the corrected expression.

`nlmixr2_indirect` *Estimate an nlmixr2 model loading the model from a targets indirect hash storage*

Description

This is the command used by the `fit_simple` target generated by `tar_nlmixr()`; it is not typically invoked by end users.

Usage

```
nlmixr2_indirect(
  object,
  data,
  est,
  control,
  directory = file.path(targets::tar_config_get("store"), "user/nlmixr2")
)
```

Arguments

`object`	Fitted object or function specifying the model.
`data`	nlmixr data
`est`	estimation method (all methods are shown by 'nlmixr2AllEst()'). Methods can be added for other tools
`control`	The estimation control object. These are expected to be different for each type of estimation method
`directory`	Cache directory to load the simplified `nlmixrui` from. Defaults to `file.path(targets::tar_config_g` `"user/nlmixr2")`, mirroring the convention used by `targets`' own `store = targets::tar_config_get("store")` defaults — the path resolves at call time against whatever store the user has configured for the running `tar_make()` (e.g. a `tempdir()` location set via `targets::tar_config_set()`).

Details

`object` is the md5 hash returned by `nlmixr_object_simplify()` and stored as the `object_simple` target in the pipeline. `nlmixr2_indirect()` loads the simplified `nlmixrui` object from the `nlmixr2targets` indirect cache (under `<targets store>/user/nlmixr2/`) and passes it to `nlmixr2est::nlmixr()` along with `data`, `est`, and `control`.

Routing the simplified ui through a small character-hash target rather than a target whose value is the ui object keeps the dependency hash for the `fit_simple` target independent of large rxUi internals, so cosmetic edits to labels or metadata (which `nlmixr_object_simplify()` strips before caching) do not invalidate the cached fit.

Value

An nlmixr2 fit object, as returned by `nlmixr2est::nlmixr()`.

See Also

`tar_nlmixr()`, `nlmixr_object_simplify()`.

`nlmixr2targets_cache_prune` *Drop orphaned entries from the* `nlmixr2targets` *indirect cache*

Description

Removes cache entries whose md5 hash is not in `keep`. By default `dry_run = TRUE`, so the function reports what would be removed without actually deleting anything; pass `dry_run = FALSE` to delete.

Usage

```
nlmixr2targets_cache_prune(
  keep = NULL,
  dry_run = TRUE,
  directory = file.path(targets::tar_config_get("store"), "user/nlmixr2")
)
```

Arguments

`keep`	Character vector of md5 hashes to retain. If `NULL` (default), the helper auto-discovers reachable hashes from the `targets` metadata in the current store. Set to `character()` to treat every cached entry as an orphan.
`dry_run`	Logical. If `TRUE` (default), do not delete anything; just return the hashes that would be removed. If `FALSE`, delete.
`directory`	Cache directory; same default as `nlmixr2targets_cache_status()`.

Details

When `keep = NULL`, the function reads the `targets` metadata in the current store and treats every value of a built target whose name ends in `_object_simple` (the hashes produced by `nlmixr_object_simplify()`) as reachable. Pass an explicit character vector if you want to override that, e.g. before calling `targets::tar_destroy()` yourself.

Value

Invisibly, a character vector of the orphan hashes (those that were removed when `dry_run = FALSE`, or would be removed otherwise).

See Also

`nlmixr2targets_cache_status()`.

nlmixr2targets_cache_status *Inspect the* nlmixr2targets *indirect cache*

Description

The indirect cache is a directory of serialized, simplified nlmixrui objects keyed by md5. It lives under <targets store>/user/nlmixr2/ and is normally invisible to targets::tar_destroy(); these helpers let you inspect and prune it without losing the rest of the targets store.

Usage

```
nlmixr2targets_cache_status(
  directory = file.path(targets::tar_config_get("store"), "user/nlmixr2")
)
```

Arguments

directory	Cache directory. Defaults to <targets::tar_config_get("store")>/user/nlmixr2.

Value

A data frame with one row per cached object and columns hash (md5 file name), size_bytes (file size), and mtime (modification time). Empty data frame with the same columns if the cache directory does not exist or is empty.

See Also

nlmixr2targets_cache_prune() for removing orphaned entries.

tar_nlmixr *Generate a set of targets for nlmixr estimation*

Description

The targets generated will include the name as the final estimation step, paste(name, "object_simple", sep = "_") (e.g. "pheno_object_simple") as the simplified model object, and paste(name, "data_simple", sep = "_") (e.g. "pheno_data_simple") as the simplified data object.

Usage

```
tar_nlmixr(
  name,
  object,
  data,
  est = NULL,
  control = list(),
  table = nlmixr2est::tableControl(),
  env = parent.frame()
)

tar_nlmixr_raw(
  name,
  object,
  data,
  est,
  control,
  table,
  object_simple_name,
  data_simple_name,
  fit_simple_name,
  env
)
```

Arguments

name	Symbol, name of the target. In tar_target(), name is an unevaluated symbol, e.g. tar_target(name = data). In tar_target_raw(), name is a character string, e.g. tar_target_raw(name = "data"). A target name must be a valid name for a symbol in R, and it must not start with a dot. Subsequent targets can refer to this name symbolically to induce a dependency relationship: e.g. tar_target(downstream_target, f(upstream_target)) is a target named downstream_target which depends on a target upstream_target and a function f(). In most cases, The target name is the name of its local data file in storage. Some file systems are not case sensitive, which means converting a name to a different case may overwrite a different target. Please ensure all target names have unique names when converted to lower case. In addition, a target's name determines its random number generator seed. In this way, each target runs with a reproducible seed so someone else running the same pipeline should get the same results, and no two targets in the same pipeline share the same seed. (Even dynamic branches have different names and thus different seeds.) You can recover the seed of a completed target with tar_meta(your_target, seed) and run tar_seed_set() on the result to locally recreate the target's initial RNG state.
object	Fitted object or function specifying the model.
data	nlmixr data

`est` estimation method (all methods are shown by 'nlmixr2AllEst()'). Methods can be added for other tools

`control` The estimation control object. These are expected to be different for each type of estimation method

`table` The output table control object (like 'tableControl()')

`env` The environment where the model is setup (not needed for typical use)

`object_simple_name`, `data_simple_name`, `fit_simple_name`
target names to use for the simplified object, simplified data, fit of the simplified object with the simplified data, and fit with the original data re-inserted.

Details

For the way that the objects are simplified, see `nlmixr_object_simplify()` and `nlmixr_data_simplify()`. To see how to write initial conditions to work with targets, see `nlmixr_object_simplify()`.

Value

A list of targets for the model simplification, data simplification, and model estimation.

Functions

- `tar_nlmixr_raw()`: An internal function to generate the targets

Side effects

When the user's model function body contains `cmt(0) <- value` inside a `model({...})` block, `tar_nlmixr()` rewrites those lines to `cmt(initial) <- value` directly in the function's binding in `env` so that `targets`' static analysis (which walks every function in env via `codetools::findGlobals()`) accepts the model. The rewrite is reversed at evaluation time, so fitting and downstream behaviour are unchanged. The user-visible consequence is that printing `body(my_model)` at the REPL after a call to `tar_nlmixr()` will show `cmt(initial)` rather than the originally-written `cmt(0)`.

Manual `cmt(initial) <- value` written by the user is also accepted, but it is a `nlmixr2targets`-only workaround: bare nlmixr2 does not understand the `cmt(initial)` form, so a model function written that way only fits when routed through `tar_nlmixr()` (or `tar_nlmixr_multimodel()`).

See Also

`tar_nlmixr_multimodel()` for fitting many models against one dataset.

Examples

```
pheno <- function() {
  ini({
    lcl <- log(0.008); label("Typical value of clearance")
    lvc <- log(0.6); label("Typical value of volume of distribution")
    etalcl + etalvc ~ c(1,
                        0.01, 1)
    cpaddSd <- 0.1; label("residual variability")
  })
```

```
  model({
    cl <- exp(lcl + etalcl)
    vc <- exp(lvc + etalvc)
    kel <- cl / vc
    d / dt(central) <- -kel * central
    cp <- central / vc
    cp ~ add(cpaddSd)
  })
}

# Build the four targets that estimate `pheno`. `data` and `est` are
# captured as expressions, so this just returns the target list; the
# estimation step runs only when you call `targets::tar_make()` from a
# project whose targets store you have configured (for example, with
# `targets::tar_config_set(store = file.path(tempdir(), "_targets"))`
# or by running inside a project directory you own).
tar_nlmixr(
  name = pheno_model,
  object = pheno,
  data = nlmixr2data::pheno_sd,
  est = "saem"
)
```

`tar_nlmixr_multimodel` *Generate a list of models based on a single dataset and estimation method*

Description

Generate a list of models based on a single dataset and estimation method

Usage

```
tar_nlmixr_multimodel(
  name,
  ...,
  data,
  est,
  control = list(),
  table = nlmixr2est::tableControl(),
  env = parent.frame()
)
```

Arguments

`name`	Symbol, name of the target. In `tar_target()`, `name` is an unevaluated symbol, e.g. `tar_target(name = data)`. In `tar_target_raw()`, `name` is a character string, e.g. `tar_target_raw(name = "data")`.

A target name must be a valid name for a symbol in R, and it must not start with a dot. Subsequent targets can refer to this name symbolically to induce a dependency relationship: e.g. `tar_target(downstream_target, f(upstream_target))` is a target named `downstream_target` which depends on a target `upstream_target` and a function `f()`.

In most cases, The target name is the name of its local data file in storage. Some file systems are not case sensitive, which means converting a name to a different case may overwrite a different target. Please ensure all target names have unique names when converted to lower case.

In addition, a target's name determines its random number generator seed. In this way, each target runs with a reproducible seed so someone else running the same pipeline should get the same results, and no two targets in the same pipeline share the same seed. (Even dynamic branches have different names and thus different seeds.) You can recover the seed of a completed target with `tar_meta(your_target, seed)` and run `tar_seed_set()` on the result to locally recreate the target's initial RNG state.

`...`	Named arguments with the format `"Model description" = modelFunction`
`data`	nlmixr data
`est`	estimation method (all methods are shown by 'nlmixr2AllEst()'). Methods can be added for other tools
`control`	The estimation control object. These are expected to be different for each type of estimation method
`table`	The output table control object (like 'tableControl()')
`env`	The environment where the model is setup (not needed for typical use)

Value

A list of targets for the model simplification, data simplification, and model estimation.

See Also

`tar_nlmixr()` for fitting a single model.

Examples

```
pheno <- function() {
  ini({
    lcl <- log(0.008); label("Typical value of clearance")
    lvc <- log(0.6); label("Typical value of volume of distribution")
    etalcl + etalvc ~ c(1,
                        0.01, 1)
    cpaddSd <- 0.1; label("residual variability")
  })
  model({
    cl <- exp(lcl + etalcl)
    vc <- exp(lvc + etalvc)
    kel <- cl / vc
    d / dt(central) <- -kel * central
```

```
    cp <- central / vc
    cp ~ add(cpaddSd)
  })
}
pheno2 <- function() {
  ini({
    lcl <- log(0.008); label("Typical value of clearance")
    lvc <- log(0.6); label("Typical value of volume of distribution")
    etalcl + etalvc ~ c(2,
                        0.01, 2)
    cpaddSd <- 3.0; label("residual variability")
  })
  model({
    cl <- exp(lcl + etalcl)
    vc <- exp(lvc + etalvc)
    kel <- cl / vc
    d / dt(central) <- -kel * central
    cp <- central / vc
    cp ~ add(cpaddSd)
  })
}

# Build the per-model target chains plus the combined list target.
# Estimation runs only when `targets::tar_make()` is invoked from a
# project whose store you have configured (see `?tar_nlmixr` for one
# tempdir-based setup).
tar_nlmixr_multimodel(
  name = all_models,
  data = nlmixr2data::pheno_sd,
  est = "saem",
  "Base model" = pheno,
  "Alternative residual error" = pheno2
)
```

`tar_nlmixr_multimodel_has_self_reference`
Does the model list refer to another model in the model list?

Description

Does the model list refer to another model in the model list?

Usage

```
tar_nlmixr_multimodel_has_self_reference(model_list, name)

tar_nlmixr_multimodel_has_self_reference_single(model, name)
```

Arguments

model_list	A named list of calls for model targets to be created
name	Symbol, name of the target. In tar_target(), name is an unevaluated symbol, e.g. tar_target(name = data). In tar_target_raw(), name is a character string, e.g. tar_target_raw(name = "data"). A target name must be a valid name for a symbol in R, and it must not start with a dot. Subsequent targets can refer to this name symbolically to induce a dependency relationship: e.g. tar_target(downstream_target, f(upstream_target)) is a target named downstream_target which depends on a target upstream_target and a function f(). In most cases, The target name is the name of its local data file in storage. Some file systems are not case sensitive, which means converting a name to a different case may overwrite a different target. Please ensure all target names have unique names when converted to lower case. In addition, a target's name determines its random number generator seed. In this way, each target runs with a reproducible seed so someone else running the same pipeline should get the same results, and no two targets in the same pipeline share the same seed. (Even dynamic branches have different names and thus different seeds.) You can recover the seed of a completed target with tar_meta(your_target, seed) and run tar_seed_set() on the result to locally recreate the target's initial RNG state.
model	A single model call for the model target to be created

Value

A logical vector the same length as model_list indicating if the model is self-referential to another model in the list

Functions

- tar_nlmixr_multimodel_has_self_reference_single(): A helper function to look at each call for each model separately

---

tar_nlmixr_multimodel_parse

Generate nlmixr multimodel target set for all models in one call to tar_nlmixr_multimodel()

Description

Generate nlmixr multimodel target set for all models in one call to tar_nlmixr_multimodel()

Usage

```
tar_nlmixr_multimodel_parse(name, data, est, control, table, model_list, env)
```

Arguments

name	Symbol, name of the target. In tar_target(), name is an unevaluated symbol, e.g. tar_target(name = data). In tar_target_raw(), name is a character string, e.g. tar_target_raw(name = "data"). A target name must be a valid name for a symbol in R, and it must not start with a dot. Subsequent targets can refer to this name symbolically to induce a dependency relationship: e.g. tar_target(downstream_target, f(upstream_target)) is a target named downstream_target which depends on a target upstream_target and a function f(). In most cases, The target name is the name of its local data file in storage. Some file systems are not case sensitive, which means converting a name to a different case may overwrite a different target. Please ensure all target names have unique names when converted to lower case. In addition, a target's name determines its random number generator seed. In this way, each target runs with a reproducible seed so someone else running the same pipeline should get the same results, and no two targets in the same pipeline share the same seed. (Even dynamic branches have different names and thus different seeds.) You can recover the seed of a completed target with tar_meta(your_target, seed) and run tar_seed_set() on the result to locally recreate the target's initial RNG state.
data	nlmixr data
est	estimation method (all methods are shown by 'nlmixr2AllEst()'). Methods can be added for other tools
control	The estimation control object. These are expected to be different for each type of estimation method
table	The output table control object (like 'tableControl()')
model_list	A named list of calls for model targets to be created
env	The environment where the model is setup (not needed for typical use)

---

tar_nlmixr_multimodel_single

Generate a single nlmixr multimodel target set for one model

Description

Generate a single nlmixr multimodel target set for one model

Usage

```
tar_nlmixr_multimodel_single(object, name, data, est, control, table, env)
```

Arguments

object	Fitted object or function specifying the model.
name	Symbol, name of the target. In tar_target(), name is an unevaluated symbol, e.g. tar_target(name = data). In tar_target_raw(), name is a character string, e.g. tar_target_raw(name = "data"). A target name must be a valid name for a symbol in R, and it must not start with a dot. Subsequent targets can refer to this name symbolically to induce a dependency relationship: e.g. tar_target(downstream_target, f(upstream_target)) is a target named downstream_target which depends on a target upstream_target and a function f(). In most cases, The target name is the name of its local data file in storage. Some file systems are not case sensitive, which means converting a name to a different case may overwrite a different target. Please ensure all target names have unique names when converted to lower case. In addition, a target's name determines its random number generator seed. In this way, each target runs with a reproducible seed so someone else running the same pipeline should get the same results, and no two targets in the same pipeline share the same seed. (Even dynamic branches have different names and thus different seeds.) You can recover the seed of a completed target with tar_meta(your_target, seed) and run tar_seed_set() on the result to locally recreate the target's initial RNG state.
data	nlmixr data
est	estimation method (all methods are shown by 'nlmixr2AllEst()'). Methods can be added for other tools
control	The estimation control object. These are expected to be different for each type of estimation method
table	The output table control object (like 'tableControl()')
env	The environment where the model is setup (not needed for typical use)

Index