

Package ‘nlmixr2save’

September 25, 2026

Title Save 'nlmixr2' Fits in a Format Readable Outside 'nlmixr2'

Version 0.2.1

Description Provides tools to save 'nlmixr2' fitted models in a portable format readable outside of 'nlmixr2' and independent of the package version. 'nlmixr2' fits and compares nonlinear mixed-effects models in differential equations with flexible dosing information commonly seen in pharmacokinetics and pharmacodynamics (Almquist, Leander, and Jirstrand 2015 <[doi:10.1007/s10928-015-9409-1](https://doi.org/10.1007/s10928-015-9409-1)>). Differential equation solving uses compiled C code from the 'rxode2' package (Wang, Hallow, and James 2015 <[doi:10.1002/psp4.12052](https://doi.org/10.1002/psp4.12052)>).

License MIT + file LICENSE

URL <https://nlmixr2.github.io/nlmixr2save/>,
<https://github.com/nlmixr2/nlmixr2save/>

BugReports <https://github.com/nlmixr2/nlmixr2save/issues/>

Encoding UTF-8

Imports checkmate, cli, digest, Rcpp, rxode2 (> 5.0.1), zip

Suggests babelmixr2, data.table, dplyr, knitr, nlmixr2data, nlmixr2est (>= 5.0.2), PopED, rmarkdown, testthat (>= 3.0.0), tidyvpc, withr

VignetteBuilder knitr

Config/testthat/edition 3

LinkingTo Rcpp

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Matthew Fidler [aut, cre] (ORCID:
<<https://orcid.org/0000-0001-8538-6691>>)

Maintainer Matthew Fidler <matthew.fidler@gmail.com>

Repository CRAN

Date/Publication 2026-09-25 17:40:02 UTC

Contents

.assignRestore	2
:=	3
loadFit	5
nlmixr2saveInvalidate	5
nlmixr2saveShare	6
nlmixrDataSimplify	7
saveFit	8
saveFitRandom	10
Index	11

.assignRestore	<i>Return if the last := assignment was actually a restore from a file</i>
----------------	--

Description

Return if the last := assignment was actually a restore from a file

Usage

.assignRestore()

Value

TRUE if the last := assignment was a restore from a file, FALSE otherwise

Author(s)

Matthew L. Fidler

Examples

.assignRestore()

:=	<i>This assignment operator is meant to assign or load a <code>nlmixr2</code> fit (and other objects)</i>
----	---

Description

By default it is equivalent to the standard assignment operator `<=`, but it is a S3 generic so it can have other behaviors for specific classes.

Usage

```
x := value
```

Arguments

<code>x</code>	the name of the object to assign the value to
<code>value</code>	the value to assign to the object, because R can use non-standard evaluation, this expression may not be evaluated when passed to the function. In the case of the <code>nlmixr2</code> function, the expression will be evaluated only if the fit needs to be refit (i.e. if the zip file does not exist or if the sha1 hash of the arguments does not match).

Details

For example, when used with a `nlmixr2` call, say:

```
fit := nlmixr2(one.cmt, theo_sd, est="focei")
```

the `:=` operator will assign the result of the `nlmixr2` call to `fit`, but it will also save the fit to a file named "fit.zip" in the current working directory.

If the "fit.zip" file already exists, it will be loaded instead of running the possibly expensive fitting process (as long as the sha1 hash of the arguments are the same).

This allows for easy saving and loading of fitted models without having to explicitly call a save function.

This S3 generic can be extended to other classes as needed, allowing for custom behaviors when assigning values to objects of those classes.

When trying to save expensive evaluations like the output of a `nlmixr2()` fit, the s3 dispatch would be to `:=.assign_nlmixr2(x, value)` or whatever function is used in the call. This allows checking the arguments to see if there can be a cache that will be loaded.

Otherwise, the default s3 method would be `:=.class` where `class` instead. Unlike the un-evaluated function dispatch there is no way to check the arguments for a cache, so loading from cache is not possible.

Value

the value that was assigned to the object, invisibly. It also has the side effect of assigning the value to the parent environment.

data.table

data.table also exports a :=, which only works inside DT[...] and errors anywhere else. When data.table is attached after nlmixr2save (for example library(nlmixr2) followed by library(data.table)), nlmixr2save re-attaches itself in front of data.table, so fit := nlmixr2(...) keeps working. data.table's own DT[, a := b] is unaffected, since data.table handles := inside [.data.table rather than looking it up. nlmixr2save::`:=` works regardless of the search path. Under a conflicts.policy that makes conflicts errors ("strict", "depends.ok" or list(error=TRUE), see [base::library\(\)](#)) nlmixr2save stays where it is and library() reports the conflict as usual.

Author(s)

Matthew L. Fidler

See Also

[saveFit\(\)](#) for saving fitted model objects to files, [loadFit\(\)](#) for loading fitted model objects from files, and [.assignParent\(\)](#) for getting or setting the environment used in the := operator.

Examples

```
if (requireNamespace("nlmixr2est", quietly=TRUE) &&
    requireNamespace("nlmixr2data", quietly=TRUE) &&
    requireNamespace("withr")) {
  library(nlmixr2est)
  library(nlmixr2data)
  withr::with_tempdir({
    one.cmt <- function() {
      ini({
        tka <- 0.45
        tcl <- log(c(0, 2.7, 100))
        tv <- 3.45
        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)
      })
    }
    # First fit creates fit.zip
    fit := nlmixr2(one.cmt, theo_sd, est="focei")

    # Second fit loads from fit.zip since it had the same options
    fit := nlmixr2(one.cmt, theo_sd, est="focei")

    # Third fit refits since the options are different
```

```

    fit := nlmixr2(one.cmt, theo_sd, est="saem")
  })
}

```

loadFit

*Load a fitted model object from a file***Description**

Load a fitted model object from a file

Usage

```
loadFit(file, checkVersion = .nlmixr2saveCheckVersion())
```

Arguments

file	the fit to load: the base name of the files it was saved to ("fit" loads fit.zip, or fit.R for a fit saved with zip=FALSE), or the path of that .zip or .R file itself. It may include a directory, e.g. "path/to/fit" or "path/to/fit.zip"; the working directory is not changed and nothing is extracted into it.
checkVersion	when TRUE, warn if the fit was produced with a different nlmixr2est/rxode2 version (or remote sha) than the one currently installed. Defaults to <code>getOption("nlmixr2save.checkVersion")</code> (TRUE).

Value

the fitted model object

```
nlmixr2saveInvalidate
```

Delete every := cache entry under the active prefix and directory

Description

Removes the cache files (.zip, .rds), the fit loader scripts (.R) and fit component files (-*.csv, -*.rds, -*.R) whose names begin with the current `getOption("nlmixr2save.prefix")` in the current `getOption("nlmixr2save.dir")`. Use it to force := to re-run cached fits/simulations on the next render when `nlmixr2save.check` is FALSE.

Usage

```
nlmixr2saveInvalidate()
```

Value

invisibly, the character vector of files removed

Author(s)

Matthew L. Fidler

Examples

```
## Not run:
  options(nlmixr2save.dir = "cache", nlmixr2save.prefix = "modelPiping-")
  nlmixr2saveInvalidate() # clears cache/modelPiping-* entries

## End(Not run)
```

nlmixr2saveShare	<i>Save a shareable copy of a fit without the original data</i>
------------------	---

Description

Writes a portable .zip copy of a saved fit that omits the original dataset (the subject-level data), so a fitted model can be shared without its data. The input may be a live fit object or the base name of an existing saved fit .zip; either way the original is left unchanged and new sibling zips are written. Names are resolved through `getOption("nlmixr2save.dir")` and `getOption("nlmixr2save.prefix")` (as for the `:= cache`).

Usage

```
nlmixr2saveShare(x, noFit = FALSE, file = NULL)
```

Arguments

<code>x</code>	a fitted model object, or the base name (no extension) of a saved fit .zip to read.
<code>noFit</code>	Boolean; when TRUE, also strip the output tables (the returned prediction/residual data.frame), writing <code><file>-noData-noFit.zip</code> instead of <code><file>-noData.zip</code> .
<code>file</code>	optional output base name; defaults to the name of <code>x</code> .

Details

- `nlmixr2saveShare("fit")` writes `fit-noData.zip` – the full fit (predictions and tables intact) with the original data removed.
- `nlmixr2saveShare("fit", noFit=TRUE)` writes `fit-noData-noFit.zip` – the core fit only (model, parameter estimates, objective, omega, eta0bf, parHistData), with both the original data and the returned prediction/residual table removed.

Value

the path of the written .zip, invisibly.

Author(s)

Matthew L. Fidler

See Also

[saveFit\(\)](#), [loadFit\(\)](#)

Examples

```
if (requireNamespace("nlmixr2est", quietly=TRUE) &&
    requireNamespace("nlmixr2data", quietly=TRUE) &&
    requireNamespace("withr")) {
  library(nlmixr2est)
  library(nlmixr2data)
  withr::with_tempdir({
    one.cmt <- function() {
      ini({
        tka <- 0.45; tcl <- log(c(0, 2.7, 100)); tv <- 3.45
        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 <- nlmixr2(one.cmt, theo_sd, est="focei")
    saveFit(fit) # fit.zip (with data)
    nlmixr2saveShare("fit") # fit-noData.zip
    nlmixr2saveShare("fit", noFit=TRUE) # fit-noData-noFit.zip
  })
}
```

nlmixrDataSimplify *Standardize and simplify data for nlmixr2 estimation*

Description

This function is typically not needed by end users.

Usage

```
nlmixrDataSimplify(data, object, table = list(), est = NULL, control = NULL)
```

Arguments

data	nlmixr data
object	an nlmixr_ui object (e.g. the output of running <code>nlmixr(object = model)</code>)
table	The output table control object (like <code>'tableControl()'</code>)
est	estimation method (all methods are shown by <code>'nlmixr2AllEst()'</code>). Methods can be added for other tools
control	The estimation control object. These are expected to be different for each type of estimation method

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.

Some estimation methods take their covariates from the data instead of from the model; `est="vae"` searches the data for covariates to select. For those methods the covariate columns the search can pick from are kept as well, since dropping them would change the fit.

Value

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

Author(s)

William S. Denney with minor modifications from Matt Fidler

saveFit

Save a fitted model object to a series of files

Description

Save a fitted model object to a series of files

Usage

```
saveFit(fit, file, zip = TRUE, data = .nlmixr2saveData())

## S3 method for class 'nlmixr2FitCore'
saveFit(fit, file, zip = TRUE, data = .nlmixr2saveData())

## S3 method for class 'nlmixr2FitData'
saveFit(fit, file, zip = TRUE, data = .nlmixr2saveData())

## Default S3 method:
saveFit(fit, file, zip = TRUE, data = .nlmixr2saveData())
```

Arguments

<code>fit</code>	the fitted model object
<code>file</code>	the base name of the files to save the fit to. It may include a directory, e.g. "path/to/fit": the files are written there (the directory is created if needed) and the archive path/to/fit.zip holds them under the bare name fit, so it can be moved and loaded from anywhere.
<code>zip</code>	Boolean indicating if the files should be zipped.

data Boolean indicating whether the original dataset (`origData`) is stored in the saved fit. When FALSE it is omitted, producing a fit that can be shared without the subject-level data (see [nlmixr2saveShare\(\)](#)). Defaults to `getOption("nlmixr2save.data", TRUE)`.

Value

nothing, called for side effects

Author(s)

Matthew L. Fidler

Examples

```
if (requireNamespace("nlmixr2est", quietly=TRUE) &&
    requireNamespace("nlmixr2data", quietly=TRUE) &&
    requireNamespace("withr")) {
  library(nlmixr2est)
  library(nlmixr2data)
  withr::with_tempdir({
    one.cmt <- function() {
      ini({
        tka <- 0.45
        tcl <- log(c(0, 2.7, 100))
        tv <- 3.45
        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 <- nlmixr2(one.cmt, theo_sd, est="focei")

    saveFit(fit) # saved to fit.zip
    fit2 <- loadFit(fit) # load fit.zip

    if (file.exists("fit.zip")) {
      unlink("fit.zip")
    }

    print(fit2)
  })
}
```

saveFitRandom	<i>Manage functions treated as random by :=</i>
---------------	---

Description

Functions registered here are saved with random-state metadata so cached restores behave like the original call was run again.

Usage

```
saveFitRandom(fun = NULL, remove = FALSE)
```

Arguments

fun	Function name(s) to add or remove. If NULL, the current registry is returned without modification.
remove	Boolean indicating if fun should be removed from the registry instead of added.

Value

Character vector of registered function names.

Author(s)

Matthew L. Fidler

Examples

```
saveFitRandom()  
saveFitRandom("myRandomFun")  
saveFitRandom("myRandomFun", remove=TRUE)
```

Index

* **Simplifiers**

- `nlmixrDataSimplify`, 7
- `.assignParent()`, 4
- `.assignRestore`, 2
- `:=`, 3
- `base::library()`, 4
- `loadFit`, 5
- `loadFit()`, 4, 7
- `nlmixr2saveInvalidate`, 5
- `nlmixr2saveShare`, 6
- `nlmixr2saveShare()`, 9
- `nlmixrDataSimplify`, 7
- `saveFit`, 8
- `saveFit()`, 4, 7
- `saveFitRandom`, 10