Package: nlmixr2extra (via r-universe)

October 22, 2026

Title Nonlinear Mixed Effects Models in Population PK/PD, Extra Support Functions

Version 5.2.1

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

Description Fit and compare 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>). Differential equation solving is by compiled C code provided in the 'rxode2' package (Wang, Hallow, and James 2015 <doi:10.1002/psp4.12052>). This package is for support functions like preconditioned fits <doi:10.1208/s12248-016-9866-5>, boostrap and stepwise covariate selection.

Depends R (>= 4.1)

License GPL (>= 3)

URL https://nlmixr2.github.io/nlmixr2extra/, https://github.com/nlmixr2/nlmixr2extra/

BugReports https://github.com/nlmixr2/nlmixr2extra/issues/

Imports checkmate, cli (>= 3.4.0), crayon, data.table, digest, dplyr, ggplot2, knitr, lotri, methods, nlme, nlmixr2est (>= 6.0.0), parallel, Rcpp, rxode2 (>= 5.1.0), stats, utils

Suggests brms, nlmixr2data, nlmixr2save (>= 0.2.0), pander, rmarkdown, ggtext, tidyr, nlmixr2lib, testthat (>= 3.0.0), tibble, withr

LinkingTo Rcpp, RcppArmadillo

Biarch true

Config/testthat/edition 3

Encoding UTF-8

NeedsCompilation yes

LazyData true

Config/roxygen2/version 8.1.0

Author Matthew Fidler [aut, cre] (ORCID:
<https://orcid.org/0000-0001-8538-6691>), Vipul Mann [aut],
Vishal Sarsani [aut] (ORCID:
<https://orcid.org/0000-0002-9272-3438>), Christian Bartels
[ctb], Bill Denney [aut] (ORCID:
<https://orcid.org/0000-0002-5759-428X>), Omar Elashkar [ctb]
(ORCID: <https://orcid.org/0000-0002-5505-778X>)

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

Date/Publication 2026-09-21 20:50:02 UTC

RemoteUrl https://github.com/cran/nlmixr2extra

RemoteRef HEAD

RemoteSha 0aeabc9c5baab135c49733404d9711650854d0c3

Contents

.nlmixrFormulaDataPrep . . . 3
.nlmixrFormulaExpandStartParam . . . 4
.nlmixrFormulaParser . . . 5
.nlmixrFormulaParserRanef . . . 5
.nlmixrFormulaSetupIniFixed . . . 6
.renameOrOverwrite . . . 7
aaaNlmixr2ExtraCRAN . . . 7
adaptivelassoCoefficients . . . 8
addAllEtas . . . 9
addCatCovariates . . . 10
addCovariate . . . 11
addData2Rx . . . 12
addorremoveCovariate . . . 12
adjustedlassoCoefficients . . . 13
bootplot . . . 14
bootstrapFit . . . 15
buildcovInfo . . . 17
buildupatedUI . . . 18
covarSearchAuto . . . 18
extractEqHelper.if . . . 20
fixedControl . . . 21
foldgen . . . 21
getMinAICFit . . . 22
horseshoeSummardf . . . 23
iivSearch . . . 25
isBoundaryFit . . . 25
isLinearizeMatch . . . 26
knit_print.nlmixr2FitCore . . . 26
lassoCoefficients . . . 27
lassoSummardf . . . 28

linearize . . . 29
linearizePlot . . . 31
linModGen . . . 31
listModelsTested . . . 32
llpControl . . . 32
multistart . . . 33
multistartControl . . . 36
nlmixrFormula . . . 38
normalizedData . . . 39
optimUnisampling . . . 40
plot.nlmixr2Multistart . . . 41
preconditionFit . . . 42
print.linIIVSearch . . . 42
profile.nlmixr2FitCore . . . 43
profileFixed . . . 44
profileLlp . . . 45
profileNlmixr2FitCoreRet . . . 46
regularmodel . . . 47
rerunTopN . . . 48
resSearch . . . 49
theoFitOde . . . 49

Index **51**

`.nlmixrFormulaDataPrep` *Perform any required data modifications for the nlmixrFormula interface*

Description

Perform any required data modifications for the nlmixrFormula interface

Usage

```
.nlmixrFormulaDataPrep(data, dvName, idName)
```

Arguments

`data`	nlmixr data
`dvName, idName`	The name of the DV and ID columns for the dataset, respectively

Value

A data frame modified, as needed for nlmixrFormula; if data is NULL, return NULL

`.nlmixrFormulaExpandStartParam` *Expand parameters to include their covariate representations, if applicable.*

Description

Expand parameters to include their covariate representations, if applicable.

Usage

```
.nlmixrFormulaExpandStartParam(start, param, paramLink = NULL, data)

.nlmixrFormulaExpandStartParamSingle(
  startName,
  startValue,
  param = NULL,
  link = NULL,
  data = NULL
)
```

Arguments

`start`	the starting values for the model
`param`	The parameter in the model
`paramLink`	Named character vector giving the link function for each parameter that has a `param` entry. Recognised values: `"identity"` (default) emits `<param> <- <linear combination>`; `"log"` wraps the linear combination in `exp()`.
`data`	The dataset
`startName`	The base name for the parameter
`startValue`	The initial value for the base parameter
`link`	Optional link function for this parameter (`"identity"` or `"log"`). Defaults to `"identity"`.

Value

A list with the ini and model parts needed to use the parameter

Functions

- `.nlmixrFormulaExpandStartParamSingle()`: Expand a single parameter in a model using dataset covariates, if applicable

.nlmixrFormulaParser *Parse the formula to extract the dependent variable, predictor, and random effects*

Description

Parse the formula to extract the dependent variable, predictor, and random effects

Usage

```
.nlmixrFormulaParser(object)
```

Arguments

object	The formula to parse

Value

A list with names of "DV", "predictor", and "ranef" which are each part of the formula broken down into calls

Examples

```
## Not run:
.nlmixrFormulaParser(a~b+c~(c|id))

## End(Not run)
```

.nlmixrFormulaParserRanef *Parse the random effects part of a formula*

Description

Parse the random effects part of a formula

Usage

```
.nlmixrFormulaParserRanef(object)
```

Arguments

object	The formula to parse

Value

An unnamed list with one element per random effect. The elements each have names of "ranefVar", "ranefGroup", and "start" indicating the variable with the random effect, the grouping variable for the random effect, and the starting value for the standard deviation of the random effect.

Examples

```
## Not run:
.nlmixrFormulaParserRanef(str2lang("(c|id)+(d|id2)"))

## End(Not run)
```

.nlmixrFormulaSetupIniFixed

Setup the ini() part of the model for fixed effects

Description

Setup the ini() part of the model for fixed effects

Usage

```
.nlmixrFormulaSetupIniFixed(start, base = str2lang("{}"))

.nlmixrFormulaSetupIniRandom(ranefDefinition, base = str2lang("{}"))
```

Arguments

`start`	The starting estimates for the model
`base`	The initial basis for the ini definition
`ranefDefinition`	The random effect definitions

Value

The inside of the ini() part of the model

Functions

- `.nlmixrFormulaSetupIniRandom()`: Setup the ini() part of the model for fixed effects

.renameOrOverwrite *Rename a column in a dataset*

Description

If newName already exists in data and would be overwritten with a different source column, this is treated as a hard error so that user data is never silently clobbered.

Usage

```
.renameOrOverwrite(data, newName, oldName)
```

Arguments

data	The dataset to modify
newName, oldName	The new and old column names

Value

data with data[[newName]] <- data[[charOld]]

Examples

```
## Not run:
.renameOrOverwrite(data.frame(A=1), newName="B", oldName="A")

## End(Not run)
```

aaaNlmixr2ExtraCRAN *This function is to set the number of threads to 2*

Description

In general it is a CRAN requirement that packages not use more than 2 threads. This function is to set the number of threads to 2 for CRAN testing. It is not intended for general use.

Usage

```
aaaNlmixr2ExtraCRAN()
```

Details

When testing with devtools::test() or testthat::test_package(), the NOT_CRAN environment variable is set to "true", so the number of threads will not be limited to 2.

Value

nothing, called for side effect of setting the number of threads to 2 for CRAN testing

Author(s)

Matthew L. Fidler

Examples

```
# Set the number of threads to 2 for CRAN testing
aaaNlmixr2ExtraCRAN()
```

`adaptivelassoCoefficients` *Return Adaptive lasso coefficients after finding optimal t*

Description

Return Adaptive lasso coefficients after finding optimal t

Usage

```
adaptivelassoCoefficients(
  fit,
  varsVec,
  covarsVec,
  catvarsVec,
  constraint = 1e-08,
  stratVar = NULL,
  ...
)
```

Arguments

`fit`	nlmixr2 fit.
`varsVec`	character vector of variables that need to be added
`covarsVec`	character vector of covariates that need to be added
`catvarsVec`	character vector of categorical covariates that need to be added
`constraint`	theta cutoff. below cutoff then the theta will be fixed to zero.
`stratVar`	A variable to stratify on for cross-validation.
`...`	Other parameters to be passed to optimalTvaluelasso

Value

return data frame of final lasso coefficients

Author(s)

Vishal Sarsani

Examples

```
## Not run:
oneCmt <- 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)
  })
}

d <- nlmixr2data::theo_sd
d$SEX <-0
d$SEX[d$ID<=6] <-1

fit <-
 nlmixr2(
   oneCmt, d,
   est = "saem",
   control = list(print = 0)
 )
varsVec <- c("ka","cl","v")
covarsVec <- c("WT")
catvarsVec <- c("SEX")

# Adaptive Lasso coefficients:

lassoDf <- adaptivelassoCoefficients(fit, varsVec, covarsVec, catvarsVec)

## End(Not run)
```

---

addAllEtas *Add Individual Random Effects and Fix them to Small Value*

Description

Add Individual Random Effects and Fix them to Small Value

Usage

```
addAllEtas(ui, fix = FALSE)
```

Arguments

ui	model with no individual random effects added
fix	If TRUE, the added etas will be fixed (not estimatable). Default is FALSE.

Value

nlmixr2 fit with individual random effects added on all fixed effects and fixed.

Author(s)

Omar I. Elashkar

---

addCatCovariates *Make dummy variable cols and updated covarsVec*

Description

Make dummy variable cols and updated covarsVec

Usage

```
addCatCovariates(data, covarsVec, catcovarsVec)
```

Arguments

data	data frame used in the analysis
covarsVec	character vector of covariates that need to be added
catcovarsVec	character vector of categorical covariates that need to be added

Value

return updated Data along with the updated covarsVec

Author(s)

Vishal Sarsani

addCovariate *Add Covariate to Model Fit (Generic)*

Description

Add Covariate to Model Fit (Generic)

Usage

```
addCovariate(fit, expr, effect, ref, ...)

## Default S3 method:
addCovariate(fit, expr, effect = "power", ref = "median", ...)

## S3 method for class 'rxUi'
addCovariate(fit, expr, effect = "power", ref = "median", ...)

## S3 method for class 'nlmixr2FitCore'
addCovariate(fit, expr, effect = "power", ref = "median", ...)

## S3 method for class 'nlmixr2Linearize'
addCovariate(fit, expr, effect = "power", ...)
```

Arguments

fit	Model fit object
expr	Expression, or vector or list of expressions. eg. CL ~ WT/70 + AGE/80 + ...
effect	character or list of characters of "linear", "piece_lin", "exp", "power". see details
ref	Default normalization for continuous covariates. "mean", "median" or NA. Default "median"
...	other parameters passed to each specific signature
	effect and normaDefault are only used if covariate is continuous. normaDefault call will be skipped if what covpar expression is normalized by other value

Author(s)

Omar I. Elashkar

addData2Rx *Add Data to RxUi model*

Description

Add Data to RxUi model

Usage

```
addData2Rx(ui, data)
```

Arguments

ui	a model.
data	a data frame to attach. The data can then be accessed via `nlme::getData(ui)`

Value

rxUi.

Author(s)

Omar I. Elashkar

addorremoveCovariate *Add covariate*

Description

Add covariate

Usage

```
addorremoveCovariate(ui, varName, covariate, add = TRUE)
```

Arguments

ui	compiled rxode2 nlmir2 model or fit
varName	the variable name to which the given covariate is to be added
covariate	the covariate that needs string to be constructed
add	boolean indicating if the covariate needs to be added or removed.

Author(s)

Matthew Fidler, Vishal Sarsani

`adjustedlassoCoefficients` *Return Adjusted adaptive lasso coefficients after finding optimal t*

Description

Return Adjusted adaptive lasso coefficients after finding optimal t

Usage

```
adjustedlassoCoefficients(
  fit,
  varsVec,
  covarsVec,
  catvarsVec,
  constraint = 1e-08,
  stratVar = NULL,
  ...
)
```

Arguments

`fit`	nlmixr2 fit.
`varsVec`	character vector of variables that need to be added
`covarsVec`	character vector of covariates that need to be added
`catvarsVec`	character vector of categorical covariates that need to be added
`constraint`	theta cutoff. below cutoff then the theta will be fixed to zero.
`stratVar`	A variable to stratify on for cross-validation.
`...`	Other parameters to be passed to optimalTvaluelasso

Value

return data frame of final lasso coefficients

Author(s)

Vishal Sarsani

Examples

```
## Not run:
oneCmt <- 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)
  })
}

d <- nlmixr2data::theo_sd
d$SEX <-0
d$SEX[d$ID<=6] <-1

fit <- nlmixr2(oneCmt, d, est = "saem", control = list(print = 0))
varsVec <- c("ka","cl","v")
covarsVec <- c("WT")
catvarsVec <- c("SEX")

# Adaptive Lasso coefficients:

lassoDf <- adjustedlassoCoefficients(fit,varsVec,covarsVec,catvarsVec)

## End(Not run)
```

`bootplot` *Produce delta objective function for boostrap*

Description

Produce delta objective function for boostrap

Usage

```
bootplot(x, ...)

## S3 method for class 'nlmixr2FitCore'
bootplot(x, ...)
```

Arguments

`x`	fit object
`...`	other parameters

Value

Fit traceplot or nothing.

Author(s)

Vipul Mann, Matthew L. Fidler

References

R Niebecker, MO Karlsson. (2013) *Are datasets for NLME models large enough for a bootstrap to provide reliable parameter uncertainty distributions?* PAGE 2013. https://www.page-meeting.org/Abstracts/are-datasets-for-nlme-models-large-enough-for-a-bootstrap-to-provide-reliable-parame

---

`bootstrapFit` *Bootstrap nlmixr2 fit*

Description

Bootstrap input dataset and rerun the model to get confidence bounds and aggregated parameters

Usage

```
bootstrapFit(
  fit,
  nboot = 200,
  nSampIndiv,
  stratVar,
  stdErrType = c("perc", "sd", "se"),
  ci = 0.95,
  pvalues = NULL,
  restart = FALSE,
  plotHist = FALSE,
  fitName = as.character(substitute(fit)),
  returnType = c("model", "fitList", "modelList")
)
```

Arguments

`fit`	the nlmixr2 fit object
`nboot`	an integer giving the number of bootstrapped models to be fit; default value is 200
`nSampIndiv`	an integer specifying the number of samples in each bootstrapped sample; default is the number of unique subjects in the original dataset
`stratVar`	Variable in the original dataset to stratify on; This is useful to distinguish between sparse and full sampling and other features you may wish to keep distinct in your bootstrap
`stdErrType`	This gives the standard error type for the updated standard errors; The current possibilities are: `"perc"` which gives the standard errors by percentiles (default), `"sd"` which gives the standard errors by the using the normal approximation of the mean with standard devaition, or `"se"` which uses the normal approximation with standard errors calculated with `nSampIndiv`

ci	Confidence interval level to calculate. Default is 0.95 for a 95 percent confidence interval
pvalues	a vector of pvalues indicating the probability of each subject to get selected; default value is NULL implying that probability of each subject is the same
restart	A boolean to try to restart an interrupted or incomplete boostrap. By default this is FALSE
plotHist	A boolean indicating if a histogram plot to assess how well the bootstrap is doing. By default this is turned off (FALSE)
fitName	is the fit name that is used for the name of the boostrap files. By default it is the fit provided though it could be something else.
returnType	this describes the return type • model this is the default and returns an updated model boostrapped confidence intervals and standard errors updated in the estimates. • fitList this returns the list of the full fits from the bootstrap. • modelList this returns the model list (which abbreviates the parameters and messages) used for the bootstrap summary statistics.

Value

Nothing, called for the side effects; The original fit is updated with the bootstrap confidence bands

Author(s)

Vipul Mann, Matthew Fidler

Examples

```
## Not run:

oneCmt <- function() {
  ini({
    tka <- 0.45; label("Ka")
    tcl <- 1; 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 <- nlmixr2(oneCmt, nlmixr2data::theo_sd, est = "saem", control = list(print = 0))
```

```
withr::with_tempdir({ # Run example in temp dir

bootstrapFit(fit, nboot = 5, restart = TRUE) # overwrites any of the existing data or model files
bootstrapFit(fit, nboot = 7) # resumes fitting using the stored data and model files

# Note this resumes because the total number of bootstrap samples is not 10

bootstrapFit(fit, nboot=10)

# Note the boostrap standard error and variance/covariance matrix is retained.
# If you wish to switch back you can change the covariance matrix by

nlmixr2est::setCov(fit, "linFim")

# And change it back again

nlmixr2est::setCov(fit, "boot10")

# This change will affect any simulations with uncertainty in their parameters

# You may also do a chi-square diagnostic plot check for the bootstrap with
bootplot(fit)
})

## End(Not run)
```

`buildcovInfo` *Build covInfo list from varsVec and covarsVec*

Description

Build covInfo list from varsVec and covarsVec

Usage

```
buildcovInfo(varsVec, covarsVec)
```

Arguments

`varsVec`	character vector of variables that need to be added
`covarsVec`	character vector of covariates that need to be added

Value

covInfo list of covariate info

Author(s)

Vishal Sarsani

buildupatedUI *Build updated from the covariate and variable vector list*

Description

Build updated from the covariate and variable vector list

Usage

```
buildupatedUI(ui, varsVec, covarsVec, add = TRUE, indep = FALSE)
```

Arguments

ui	compiled rxode2 nlmir2 model or fit
varsVec	character vector of variables that need to be added
covarsVec	character vector of covariates that need to be added
add	boolean indicating if the covariate needs to be added or removed
indep	a boolean indicating if the covariates should be added independently, or sequentially (append to the previous model). only applicable to adding covariate

Value

updated ui with added covariates

Author(s)

Vishal Sarsani

covarSearchAuto *Stepwise Covariate Model-selection (SCM) method*

Description

Stepwise Covariate Model-selection (SCM) method

Usage

```
covarSearchAuto(
  fit,
  varsVec,
  covarsVec,
  pVal = list(fwd = 0.05, bck = 0.01),
  catvarsVec = NULL,
  searchType = c("scm", "forward", "backward"),
  restart = FALSE
)
```

Arguments

fit	an nlmixr2 'fit' object
varsVec	a list of candidate variables to which the covariates could be added
covarsVec	a list of candidate covariates that need to be tested
pVal	a named list with names 'fwd' and 'bck' for specifying the p-values for the forward and backward searches, respectively
catvarsVec	character vector of categorical covariates that need to be added
searchType	one of 'scm', 'forward' and 'backward' to specify the covariate search method; default is 'scm'
restart	a boolean that controls if the search should be restarted; default is FALSE

Value

A list summarizing the covariate selection steps and output; This list has the "summaryTable" for the overall summary of the covariate selection as well as "resFwd" for the forward selection method and "resBck" for the backward selection method.

Author(s)

Vipul Mann, Matthew Fidler, Vishal Sarsani

Examples

```
## Not run:
oneCmt <- 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 <- nlmixr2(oneCmt, nlmixr2data::theo_sd, est = "saem", control = list(print = 0))
rxode2::.rxWithWd(tempdir(), {# with temporary directory

auto1 <- covarSearchAuto(fit, varsVec = c("ka", "cl"),
    covarsVec = c("WT"))

})
```

```
## Note that this didn't include sex, add it to dataset and restart model


d <- nlmixr2data::theo_sd
d$SEX <-0
d$SEX[d$ID<=6] <-1

fit <- nlmixr2(oneCmt, d, est = "saem", control = list(print = 0))

# This would restart if for some reason the search crashed:

rxode2::.rxWithWd(tempdir(), {# with temporary directory

auto2 <- covarSearchAuto(fit, varsVec = c("ka", "cl"), covarsVec = c("WT"),
                catvarsVec= c("SEX"), restart = TRUE)

auto3 <- covarSearchAuto(fit, varsVec = c("ka", "cl"), covarsVec = c("WT"),
                catvarsVec=  c("SEX"), restart = TRUE,
                searchType = "forward")
})

## End(Not run)
```

`extractEqHelper.if` *Generate LaTeX for if blocks*

Description

Generate LaTeX for if blocks

Usage

```
## S3 method for class '`if`'
extractEqHelper(x, ..., inModel, alignment, indent = 0L, firstIf = 0L, name)
```

Arguments

`x`	language object to convert to LaTeX
`...`	additional arguments passed to methods
`inModel`	logical; whether the expression is inside a model block
`alignment`	LaTeX alignment character used for equation alignment
`indent`	tracks the current indention level
`firstIf`	tracks if this is the first if at the current indent level
`name`	optional name for the expression

fixedControl *Control options for fixed-value likelihood profiling*

Description

Control options for fixed-value likelihood profiling

Usage

```
fixedControl()
```

Value

A validated list of control options for fixed-value likelihood profiling

See Also

`profileFixed()`

Other Profiling: `llpControl()`, `profile.nlmixr2FitCore()`, `profileFixed()`, `profileLlp()`, `profileNlmixr2FitCoreRet()`

foldgen *Stratified cross-validation fold generator function, inspired from the caret*

Description

Stratified cross-validation fold generator function, inspired from the caret

Usage

```
foldgen(data, nfold = 5, stratVar = NULL)
```

Arguments

`data`	data frame used in the analysis
`nfold`	number of k-fold cross validations. Default is 5
`stratVar`	Stratification Variable. Default is NULL and ID is used for CV

Value

return data.frame with the fold column attached

Author(s)

Vishal Sarsani, caret

Examples

```
d <- nlmixr2data::theo_sd
d$SEX <-0
d$SEX[d$ID<=6] <-1

covarsVec <- c("WT")

# Stratified cross-validation data with CMT
df1 <- foldgen(d, nfold=5, stratVar="CMT")

# Stratified cross-validation data with ID (individual)
df2 <- foldgen(d, nfold=5, stratVar=NULL)
```

`getMinAICFit` *Get the model with the minimum Akaike Information Criterion (AIC) value*

Description

Get the model with the minimum Akaike Information Criterion (AIC) value

Usage

```
getMinAICFit(..., excludeBoundary = TRUE, k = 2)
```

Arguments

`...`	One or more model fits or lists of model fits
`excludeBoundary`	Exclude a model from selection if it has a parameter at its boundary
`k`	numeric, the *penalty* per parameter to be used; the default `k = 2` is the classical AIC.

Value

The model fit with the minimum AIC; `NULL` if no model fits are available after exclusions. If more than one model has the same AIC, the first is returned.

Examples

```
## Not run:
d_noec50 <-
  data.frame(
    conc = c(rep(0, 10), rep(1:20, each = 10)),
    DV = c(rnorm(n = 10, mean = 1, sd = 1e-5), rnorm(n = 200, mean = 5, sd = 1e-5)),
    TIME = 0
  )
```

```
modEmax <- function() {
  ini({
    e0 = 1
    emax = 5
    ec50 = c(0, 1.1)
    addSd = 0.5
  })
  model({
    effect <- e0 + emax*conc/(ec50 + conc)
    effect ~ add(addSd)
  })
}

modStep <- function() {
  ini({
    e0 = 1
    emax = 5
    addSd = 1e-5
  })
  model({
    effect <- e0 + emax*(conc > 0)
    effect ~ add(addSd)
  })
}

modLinear <- function() {
  ini({
    e0 = 1
    slope = 5
    addSd = 1
  })
  model({
    effect <- e0 + slope*conc
    effect ~ add(addSd)
  })
}

fitEmaxBoundaryIssue <- nlmixr2(modEmax, data = d_noec50, est = "focei", control = list(print = 0))
fitStep <- nlmixr2(modStep, data = d_noec50, est = "focei", control = list(print = 0))
fitLinear <- nlmixr2(modLinear, data = d_noec50, est = "focei", control = list(print = 0))
getMinAICFit(list(fitEmaxBoundaryIssue, fitStep, fitLinear))

## End(Not run)
```

`horseshoeSummardf` *Create Horseshoe summary posterior estimates*

Description

Create Horseshoe summary posterior estimates

Usage

```
horseshoeSummardf(fit, covarsVec, ...)
```

Arguments

fit	compiled rxode2 nlmir2 model fit
covarsVec	character vector of covariates that need to be added
...	other parameters passed to brm(): warmup = 1000, iter = 2000, chains = 4, cores = 4, control = list(adapt_delta = 0.99, max_treedepth = 15)

Value

Horse shoe Summary data frame of all covariates

Author(s)

Vishal Sarsani, Christian Bartels

Examples

```
## Not run:
oneCmt <- 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)
  })
}

d <- nlmixr2data::theo_sd
fit <- nlmixr2(oneCmt, d, est = "saem", control = list(print = 0))
covarsVec <- c("WT")

# Horseshoe summary posterior estimates:

#hsDf <- horseshoeSummardf(fit,covarsVec,cores=2)
#brms sometimes may throw a Error in sink(type = "output")
#Issue Should be fixed by uninstalling and re-installing rstan

## End(Not run)
```

iivSearch *Automated Inter-Individual Variability Search*

Description

Automated Inter-Individual Variability Search

Usage

```
iivSearch(fit, ...)

## Default S3 method:
iivSearch(fit, ...)

## S3 method for class 'nlmixr2Linearize'
iivSearch(fit, sortBy = "BIC", mceta = 5, ...)
```

Arguments

fit	a model fit
...	parameters for sub functions
sortBy	either "BIC" or "OBJ". Default is "BIC"
mceta	an integer specifying mceta during fitting.

Author(s)

Omar I. Elashkar

isBoundaryFit *Detect if a fit has any parameter at the boundary*

Description

Detect if a fit has any parameter at the boundary

Usage

```
isBoundaryFit(object)
```

Arguments

object	A model fit to check

Value

TRUE if the fit is an nlmxir2 fit and it has a parameter at the boundary; FALSE otherwise

isLinearizeMatch *Check Linearization Match*

Description

Check Linearization Match

Usage

```
isLinearizeMatch(linFit, tol = 0.1)
```

Arguments

linFit	fit of linear model.
tol	relative tolerance for matching. Default is 0.05.

Value

match list for OFV, omega, eta, and residual variance terms

Author(s)

Omar I. Elashkar

knit_print.nlmixr2FitCore *Extract the equations from an nlmixr2/rxode2 model to produce a 'LaTeX' equation.*

Description

Extract the equations from an nlmixr2/rxode2 model to produce a 'LaTeX' equation.

Usage

```
## S3 method for class 'nlmixr2FitCore'
knit_print(x, ..., output = "equations")

## S3 method for class 'rxUi'
knit_print(x, ...)
```

Arguments

x	The model to extract equations from
...	Ignored
output	The type of output to request (currently, just "equations")

`lassoCoefficients` *Return Final lasso coefficients after finding optimal t*

Description

Return Final lasso coefficients after finding optimal t

Usage

```
lassoCoefficients(
  fit,
  varsVec,
  covarsVec,
  catvarsVec,
  constraint = 1e-08,
  stratVar = NULL,
  ...
)
```

Arguments

`fit`	nlmixr2 fit.
`varsVec`	character vector of variables that need to be added
`covarsVec`	character vector of covariates that need to be added
`catvarsVec`	character vector of categorical covariates that need to be added
`constraint`	theta cutoff. below cutoff then the theta will be fixed to zero
`stratVar`	A variable to stratify on for cross-validation
`...`	Other parameters to be passed to optimalTvaluelasso

Value

return data frame of final lasso coefficients

Author(s)

Vishal Sarsani

Examples

```
## Not run:
oneCmt <- 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)
  })
}

d <- nlmixr2data::theo_sd
d$SEX <-0
d$SEX[d$ID<=6] <-1

fit <- nlmixr2(oneCmt, d, est = "saem", control = list(print = 0))
varsVec <- c("ka","cl","v")
covarsVec <- c("WT")
catvarsVec <- c("SEX")

# Lasso coefficients:

lassoDf <- lassoCoefficients(fit, varsVec, covarsVec, catvarsVec, constraint=1e-08, stratVar = NULL)

## End(Not run)
```

`lassoSummardf` *Create Lasso summary posterior estimates*

Description

Create Lasso summary posterior estimates

Usage

```
lassoSummardf(fit, covarsVec, ...)
```

Arguments

`fit`	compiled rxode2 nlmir2 model fit
`covarsVec`	character vector of covariates that need to be added
`...`	other parameters passed to brm(): warmup = 1000, iter = 2000, chains = 4, cores = 4, control = list(adapt_delta = 0.99, max_treedepth = 15)

Value

Horse shoe Summary data frame of all covariates

Author(s)

Vishal Sarsani, Christian Bartels

Examples

```
## Not run:
oneCmt <- 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)
  })
}

d <- nlmixr2data::theo_sd
fit <- nlmixr2(oneCmt, d, est = "saem", control = list(print = 0))
covarsVec <- c("WT")

# Horseshoe summary posterior estimates:

#lassoDf <- lassoSummardf(fit,covarsVec,cores=2)
#brms sometimes may throw a Error in sink(type = "output")
#Issue Should be fixed by uninstalling and re-installing rstan

## End(Not run)
```

---

linearize *Perform linearization of a model fitted using FOCEI*

Description

Perform linearization of a model fitted using FOCEI

Usage

```
linearize(
  fit,
  mceta = c(-1, 10, 100, 1000),
  relTol = 0.25,
```

```
  focei = NA,
  addEtas = FALSE,
  derivFct = FALSE,
  plot = FALSE,
  est = "focei"
)
```

Arguments

fit	fit of nonlinear model fitted using any method with at least one eta. See details.
mceta	a numeric vector for mceta to try. See details.
relTol	relative deviation tolerance between original and linearized models objective functions. Used for switching if focei = NA. See details.
focei	Default is NA for automatic switch from FOCEI to FOCE if failed. See details.
addEtas	boolean. If TRUE, add etas on every theta and fix it to small value to get derivatives. Default is FALSE.
derivFct	boolean. If TRUE, turn on derivatives for linearization. Default is FALSE.
plot	boolean. Print plot of linearized vs original
est	Character. The estimation method used for the linearized model fit. Only 'focei' is supported.

Details

The function accepts a fit object fitted using any method. However, if the method is not FOCE+I, the model is going to be evaluated first.

`mceta` vector will be iterated over to find the best linearization if linearization failed. Escalating to next mceta will depend on the relative deviation `relTol` of the original and linearized models objective functions.

If `focei` is set to `NA`, the function will first try to linearize using FOCEI. If the relative deviation between original and linearized models objective functions is greater than `relTol`, it will switch to FOCE where residual linearization is skipped. If `focei` is set to `TRUE`, the function will use FOCEI linearization with individual and residual linearization. If `focei` is set to `FALSE`, the function will use FOCE linearization with residual interaction linearization skipped.

If `derivFct` is set to `TRUE`, the function will use an extra factor for linearization that might help in stabilization. This might be useful to try with FOCEI.

`plot` argument can only print ggplot figure with default settings. If a user wish to capture the plot, one might use `linearizePlot()` call.

Value

a fit object with subclass nlmixr2linearize

Author(s)

Omar I. Elashkar

linearizePlot *Plot Original Versus Linear Models iObj and Etas*

Description

Plot Original Versus Linear Models iObj and Etas

Usage

```
linearizePlot(lin)
```

Arguments

lin	linear fitting object

Details

If the non-linear fit was not FOCEI, the objective functions are not comparable, hence can be ignored upon comparison.

Value

ggplot object

Author(s)

Omar I. Elashkar

linModGen *Generate a Linearization Model From Previous Fit*

Description

Generate a Linearization Model From Previous Fit

Usage

```
linModGen(ui, focei = TRUE, derivFct = FALSE)
```

Arguments

ui	rx model or fit object.
focei	boolean. If TRUE, use FOCEI linearization with individual and residual linearization. Default is TRUE.
derivFct	boolean. If TRUE, use normalization derivative factors. Default is FALSE.

Value

rxUi model

Author(s)

Omar I. Elashkar

`listModelsTested` *Make a list of models tested ready for reporting*

Description

Make a list of models tested ready for reporting

Usage

```
listModelsTested(fitList, caption, excludeBoundary = TRUE, k = 2)
```

Arguments

`fitList`	A named list of models that were tested
`caption`	The caption for the resulting table
`excludeBoundary`	Exclude a model from selection if it has a parameter at its boundary
`k`	numeric, the *penalty* per parameter to be used; the default `k = 2` is the classical AIC.

Value

A data.frame with a "caption" attribute ready for printing in a report with `pander::pander()`. The data.frame will have names of "Description", "AIC", and "dAIC"; if exclusions are applied, the data.frame will also have an "Exclude" column.

`llpControl` *Control options for log-likelihood profiling*

Description

Control options for log-likelihood profiling

Usage

```
llpControl(
  ofvIncrease = qchisq(0.95, df = 1),
  rseTheta = 30,
  itermax = 10,
  ofvtol = 0.005,
  paramDigits = 3,
  extrapolateExpand = 1.5
)
```

Arguments

`ofvIncrease`	The targetted change in objective function value (3.84 corresponds to a Chi-squared test with a 95% confidence interval)
`rseTheta`	The relative standard error (percent) for the model parameters. It can be missing (the default) in which case a default value of 30% will be applied. If given as a single number, it will be applied to all parameters. If given as a named vector of numbers, it will be applied to each named parameter.
`itermax`	Maximum number of likelihood profiling iterations for each bound estimated
`ofvtol`	The relative tolerance for the objective function being close enough to the `ofvIncrease`.
`paramDigits`	The number of significant digits required for the parameter. When interpolation attempts to get smaller than that number of significant digits, it will stop.
`extrapolateExpand`	When extrapolating outside the range previously tested, how far should the step occur as a ratio

Value

A validated list of control options for log-likelihood profiling

See Also

`profileLlp()`

Other Profiling: `fixedControl()`, `profile.nlmixr2FitCore()`, `profileFixed()`, `profileLlp()`, `profileNlmixr2FitCoreRet()`

`multistart` *Estimate a model from many starting points*

Description

Refits a model from several perturbed sets of initial estimates and collects the results, so that a fit which settled in a local optimum can be recognized. Use `plot()` on the result for the objective-function waterfall and the parameter-stability plots.

Usage

```
multistart(object, ...)

## S3 method for class 'nlmixr2FitCore'
multistart(
  object,
  ...,
  data = NULL,
  est = NULL,
  estControl = NULL,
  control = list()
)

## S3 method for class 'rxUi'
multistart(
  object,
  data,
  ...,
  est = "focei",
  estControl = NULL,
  control = list()
)

## S3 method for class '`function`'
multistart(object, data, ...)

## Default S3 method:
multistart(object, ...)
```

Arguments

`object`	A nlmixr2 fit, a nlmixr2 model function, or a `rxUi` model object
`...`	ignored
`data`	The data to estimate with; taken from `object` when it is a fit
`est`	The estimation method; taken from `object` when it is a fit
`estControl`	The control for `est`; taken from `object` when it is a fit, and otherwise the method's default
`control`	A list passed to `multistartControl()`

Value

An object of class `nlmixr2Multistart`, a list with elements `starts` (one row per candidate starting point), `summary` (one row per estimated start), `fits`, `best`, and `bestIndex`

Starting points

The first candidate is always the unperturbed starting point, so the original fit is always represented in the comparison. Every other candidate perturbs the unfixed population parameters on the scale

the model is estimated on (which for a mu-referenced parameter is usually the log scale), and clips the result to the parameter's declared bounds. Between-subject variances are perturbed multiplicatively so they stay positive, and the resulting matrix is made positive-definite with `lotri::lotriNearPD()`.

Screening

Fully estimating every candidate is wasteful when many of them start far from anywhere sensible. With the default `screen = "posthoc"` each candidate is first evaluated with an empirical-Bayes step only, which costs a small fraction of a full estimation, and only the best `nFit` candidates are then fully estimated.

Parallel estimation

Each estimation already runs across every available thread, so estimating several starts at once oversubscribes the machine unless the thread budget is divided. `multistartControl(cores=)` therefore restricts each worker to a single thread. Whether that is faster than the serial default depends entirely on the model; a model whose subjects parallelise well is usually better off left serial. Parallel estimation uses `parallel::mclapply()` and is not available on Windows.

Resuming

Each start is cached to `cacheDir` as it completes, so an interrupted run resumes where it left off. Increasing `n` on a later call re-uses the starts already estimated and only estimates the new ones. Pass `restart = TRUE` to discard the cache, or `cacheDir = NA` to never write one.

Changing anything that alters what a starting point *is* (`sampling`, `spread`, `which`, `perturbOmega`, `omegaFold` or `seed`) gives the run its own cache, so a cached estimation is never re-used for a start it did not come from. A Latin hypercube is a design over all `n` candidates at once, so growing an `"lhs"` run moves its earlier starting points; those starts are detected and re-estimated rather than reported against the wrong starting point.

Author(s)

Matthew Fidler

See Also

Other Multistart: `multistartControl()`, `plot.nlmixr2Multistart()`

Examples

```
## Not run:
# Every start is a full estimation, so this takes a few minutes.
fit <- nlmixr2extra::theoFitOde

ms <- multistart(fit, control = list(n = 8, spread = 0.3))
ms

# objective function values, best to worst
plot(ms)
```

```
# how stable each parameter is across the best starts
plot(ms, "parameters")

# the best fit found, ready to use like any other fit
ms$best

## End(Not run)
```

`multistartControl` *Control options for* `multistart()`

Description

Control options for `multistart()`

Usage

```
multistartControl(
  n = 10L,
  nFit = NULL,
  sampling = c("uniform", "lhs", "normal"),
  spread = 0.2,
  which = NULL,
  perturbOmega = TRUE,
  omegaFold = 2,
  around = c("final", "initial"),
  screen = c("posthoc", "none"),
  refitBest = TRUE,
  excludeBoundary = TRUE,
  keepFits = TRUE,
  seed = 1234L,
  cores = 1L,
  cacheDir = NULL,
  restart = FALSE
)
```

Arguments

`n`	Number of candidate starting points to generate. The first candidate is always the unperturbed starting point, so `n = 1` reproduces the original fit.
`nFit`	Number of candidates to fully estimate after screening. `NULL` (the default) estimates every candidate. Ignored when `screen = "none"`.
`sampling`	How the starting points are drawn around the initial estimates: `"uniform"` (the default) draws uniformly within `spread`, `"lhs"` uses a Latin hypercube over the same interval so the range is covered more evenly, and `"normal"` draws normally with a standard deviation of `spread`.

spread	Fractional spread of the perturbation on the estimation scale. A parameter with initial estimate est is perturbed within est +/- spread*max(abs(est), 1).
which	Names of the population parameters to perturb; NULL (the default) perturbs every unfixed population parameter.
perturbOmega	Should the between-subject variability estimates be perturbed as well?
omegaFold	Fold-range for the omega perturbation. A variance v is drawn within v/omegaFold and v*omegaFold.
around	When starting from a fit, perturb around the fit's "final" estimates (the default, which asks "is this a local optimum?") or around the "initial" estimates the fit started from (which asks "how sensitive was this fit to where I started?").
screen	Cheap pre-selection of candidates. "posthoc" (the default) evaluates the objective function at each candidate with an empirical Bayes step only and fully estimates the best nFit; "none" estimates every candidate.
refitBest	Re-run the best start with the full estimation control, so the returned fit has the covariance step and tables the exploratory runs skip.
excludeBoundary	Should fits with a parameter at a boundary be excluded when picking the best start? Matches getMinAICFit().
keepFits	Keep each start's fit in the result. Setting this to FALSE keeps only the summary and the best fit, which is much smaller.
seed	Integer seed. The perturbations are drawn from this seed and each start is estimated with its own derived seed.
cores	Number of starts to estimate at once. Each estimation already uses every available thread internally, so the default of 1 is usually the fastest choice; see the "Parallel estimation" section of multistart().
cacheDir	Directory used to cache the individual starts so that an interrupted run can be resumed. NULL (the default) derives a name from the model; NA disables caching.
restart	Discard any cached results and start over.

Value

A validated list of control options for multistart()

See Also

multistart()

Other Multistart: multistart(), plot.nlmixr2Multistart()

`nlmixrFormula` *A simple formula-based interface for nlmixr2*

Description

A simple formula-based interface for nlmixr2

Usage

```
nlmixrFormula(
  object,
  data,
  start,
  param = NULL,
  paramLink = NULL,
  ...,
  residualModel = ~add(addSd)
)
```

Arguments

`object`	The formula defining the model (see details)
`data`	The data to fit
`start`	A named list of starting estimates. The names define the parameters in the model. If a single parameter estimate is desired, it can be given here. If a parameter estimate per factor level is desired, either a single starting estimate can be given across all factor levels or one estimate may be given per factor level. (Specify the factors with the `param` argument.)
`param`	A formula or list of two-sided formulas giving the model used for parameters. If a parameter is a simple fixed effect, only, then it should not be included here. The right-hand side of a `param` formula names one or more columns in `data`: factor columns get a separate fixed effect per level; numeric columns get a linear slope. Multiple covariates on the same parameter can be combined with `+`, e.g. `b ~ z + w`.
`paramLink`	Optional named character vector mapping parameter names to a link function. `"identity"` (the default) emits `<param> <- <linear combination>`; `"log"` wraps the linear combination in `exp()` so the parameter is strictly-positive on the natural scale. Only parameters that have a `param` entry may appear here.
`...`	Arguments passed on to `nlmixr2est::nlmixr` `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()')

save Boolean to save a nlmixr2 object in a rds file in the working directory. If NULL, uses option "nlmixr2.save"

envir Environment where the nlmixr object/function is evaluated before running the estimation routine.

residualModel The residual model formula to use as a one-sided formula. The default is ~ add(addSd); richer models such as ~ add(addSd) + prop(propSd) are supported as long as the corresponding sigma parameter names appear in start.

Details

The formula is given with different notation than typical formulas. The formula notation is inspired by and similar to lme4::nlmer(). It is a 3-part formula: dependentVariable~predictorEquation~randomEffects.

The dependentVariable is any variable in the dataset. It may not include any math; for example, log(DV) is not allowed.

The predictorEquation is any valid math, and it will be used directly in the nlmixr2 model.

The randomEffects are one or more random effect parameters defined by putting the parameter in parentheses and putting a vertical bar and the grouping parameter. Only one grouping parameter is allowed for all random effects. An example would be (slope|ID) to estimate a random effect parameter named "slope" for each "ID" in the data.

Value

The model fit from nlmixr2est::nlmixr2()

Examples

```
## Not run:
nlmixrFormula(
  height ~ (Asym+AsymRe)+(R0-(Asym+AsymRe))*exp(-exp(lrc)*age) ~ (AsymRe|Seed),
  data = Loblolly,
  start = list(Asym = 103, R0 = -8.5, lrc = -3.3, addSd=1),
  est="focei"
)

## End(Not run)
```

---

normalizedData *Function to return data of normalized covariates*

Description

Function to return data of normalized covariates

Usage

```
normalizedData(data, covarsVec, replace = TRUE)
```

Arguments

data	a dataframe with covariates to normalize
covarsVec	a list of covariate names (parameters) that need to be estimates
replace	replace the original covariate data with normalized data for easier updated model.

Value

data frame with all normalized covariates

Author(s)

Vishal Sarsani

Examples

```
d <- nlmixr2data::theo_sd
d$SEX <-0
d$SEX[d$ID<=6] <-1

covarsVec <- c("WT")

# Normalized covariate (replaced)
df1 <- normalizedData(d,covarsVec,replace=TRUE)

# Normalized covariate (without replacement)
df2 <- normalizedData(d,covarsVec,replace=FALSE)
```

---

optimUnisampling *Sample from uniform distribution by optim*

Description

Sample from uniform distribution by optim

Usage

```
optimUnisampling(xvec, N = 1000, medValue, floorT = TRUE)
```

Arguments

xvec	A vector of min,max values . Ex:c(10,20)
N	Desired number of values
medValue	Desired Median
floorT	boolean indicating whether to round up

Value

Samples with approx desired median.

Author(s)

Vishal Sarsani

Examples

```
# Simulate 1000 creatine clearance values with median of 71.7 within range of c(6.7,140)
creatCl <- optimUnisampling(xvec=c(6.7,140), N=1000, medValue = 71.7, floorT=FALSE)
```

`plot.nlmixr2Multistart`

Plot a multistart result

Description

Plot a multistart result

Usage

```
## S3 method for class 'nlmixr2Multistart'
plot(x, type = c("waterfall", "parameters"), kBest = 20L, dOfvMax = NULL, ...)
```

Arguments

`x`	A `nlmixr2Multistart` object from `multistart()`
`type`	`"waterfall"` plots each start's objective function relative to the best one, worst to best; `"parameters"` plots how each parameter estimate varies across the best starts.
`kBest`	Number of starts to show in the `"parameters"` plot
`dOfvMax`	Upper limit for the waterfall's objective-function axis, for zooming in when one start is far worse than the rest
`...`	ignored

Value

A ggplot2 object

See Also

Other Multistart: `multistart()`, `multistartControl()`

preconditionFit — *Linearly re-parameterize the model to be less sensitive to rounding errors*

Description

Linearly re-parameterize the model to be less sensitive to rounding errors

Usage

```
preconditionFit(fit, estType = c("full", "posthoc", "none"), ntry = 10L)
```

Arguments

fit	A nlmixr2 fit to be preconditioned
estType	Once the fit has been linearly reparameterized, should a "full" estimation, "posthoc" estimation or simply a estimation of the covariance matrix "none" before the fit is updated
ntry	number of tries before giving up on a pre-conditioned covariance estimate

Value

A nlmixr2 fit object that was preconditioned to stabilize the variance/covariance calculation

References

Aoki Y, Nordgren R, Hooker AC. Preconditioning of Nonlinear Mixed Effects Models for Stabilisation of Variance-Covariance Matrix Computations. AAPS J. 2016;18(2):505-518. doi:10.1208/s12248-016-9866-5

print.linIIVSearch — *Print Summary Table For Linearized IIV Search*

Description

Print Summary Table For Linearized IIV Search

Usage

```
## S3 method for class 'linIIVSearch'
print(x, ...)
```

Arguments

x	IIV search results object
...	Other arguments passed to print

Author(s)

Omar I. Elashkar

`profile.nlmixr2FitCore` *Perform likelihood profiling on nlmixr2 focei fits*

Description

Perform likelihood profiling on nlmixr2 focei fits

Usage

```
## S3 method for class 'nlmixr2FitCore'
profile(
  fitted,
  ...,
  which = NULL,
  method = c("llp", "fixed"),
  control = list()
)
```

Arguments

`fitted`	The fit model
`...`	ignored
`which`	The parameter names to perform likelihood profiling on (`NULL` indicates all parameters)
`method`	Method to use for profiling (see the details)
`control`	Control arguments for the `method`

Value

A data.frame with one column named `Parameter` indicating the parameter being fixed on that row, one column for the `OFV` indicating the OFV estimated for the model at that step, one column named `profileBound` indicating the estimated value for the profile likelihood and its step above the minimum profile likelihood value, and columns for each parameter estimate (or fixed) in the model.

Log-likelihood profiling

`method = "llp"`

The search will stop when either the OFV is within `ofvtol` of the desired OFV change or when the parameter is interpolating to more significant digits than specified in `paramDigits`. The "llp" method uses the `profileLlp()` function. See its help for more details.

Fixed points

`method = "fixed"`

Estimate the OFV for specific fixed values. The "fixed" method uses the `profileFixed()` function. See its help for more details.

See Also

Other Profiling: `fixedControl()`, `llpControl()`, `profileFixed()`, `profileLlp()`, `profileNlmixr2FitCoreRet()`

Examples

```
## Not run:
# Likelihood profiling takes a long time to run each model multiple times, so
# be aware that running this example may take a few minutes.
oneCmt <- function() {
  ini({
    tka <- log(1.57)
    tcl <- log(2.72)
    tv <- fixed(log(31.5))
    eta.ka ~ 0.6
    add.sd <- 0.7
  })
  model({
    ka <- exp(tka + eta.ka)
    cl <- exp(tcl)
    v <- exp(tv)
    cp <- linCmt()
    cp ~ add(add.sd)
  })
}

fit <-
  nlmixr2(
    oneCmt, data = nlmixr2data::theo_sd, est="focei", control = list(print=0)
  )
# profile all parameters
profall <- profile(fit)

# profile a single parameter
proftka <- profile(fit, which = "tka")

## End(Not run)
```

---

profileFixed — *Estimate the objective function values for a model while fixing defined parameter values*

Description

Estimate the objective function values for a model while fixing defined parameter values

Usage

```
profileFixed(fitted, which, control = list())

profileFixedSingle(fitted, which)
```

Arguments

fitted	The fit model
which	A data.frame with column names of parameters to fix and values of the fitted value to fix (one row only).
control	A list passed to fixedControl() (currently unused)

Value

which with a column named OFV added with the objective function value of the fitted estimate fixing the parameters in the other columns

Functions

- profileFixedSingle(): Estimate the objective function value for a model while fixing a single set of defined parameter values (for use in parameter profiling)

Author(s)

Bill Denney (changed by Matt Fidler to take out R 4.1 specific code)

See Also

Other Profiling: fixedControl(), llpControl(), profile.nlmixr2FitCore(), profileLlp(), profileNlmixr2FitCoreRet()

---

profileLlp *Profile confidence intervals with log-likelihood profiling*

Description

Profile confidence intervals with log-likelihood profiling

Usage

```
profileLlp(fitted, which, control)
```

Arguments

fitted	The fit model
which	Either NULL to profile all parameters or a character vector of parameters to estimate
control	A list passed to llpControl()

Value

A data.frame with columns named "Parameter" (the parameter name(s) that were fixed), OFV (the objective function value), and the current estimate for each of the parameters. In addition, if any boundary is found, the OFV increase will be indicated by the absolute value of the "profileBound" column and if that boundary is the upper or lower boundary will be indicated by the "profileBound" column being positive or negative, respectively.

See Also

Other Profiling: `fixedControl()`, `llpControl()`, `profile.nlmixr2FitCore()`, `profileFixed()`, `profileNlmixr2FitCoreRet()`

`profileNlmixr2FitCoreRet` *Give the output data.frame for a single model for profile.nlmixr2FitCore*

Description

Give the output data.frame for a single model for profile.nlmixr2FitCore

Usage

```
profileNlmixr2FitCoreRet(fitted, which, fixedVal)
```

Arguments

`fitted`	The fit model
`which`	The parameter names to perform likelihood profiling on (`NULL` indicates all parameters)
`fixedVal`	The value that `which` is fixed to in case the model does not converge.

Value

A data.frame with columns named "Parameter" (the parameter name(s) that were fixed), OFV (the objective function value), and the current estimate for each of the parameters. Omega values are given as their variances and covariances.

See Also

Other Profiling: `fixedControl()`, `llpControl()`, `profile.nlmixr2FitCore()`, `profileFixed()`, `profileLlp()`

`regularmodel` *Regular lasso model*

Description

Regular lasso model

Usage

```
regularmodel(
  fit,
  varsVec,
  covarsVec,
  catvarsVec,
  constraint = 1e-08,
  lassotype = c("regular", "adaptive", "adjusted"),
  stratVar = NULL,
  ...
)
```

Arguments

`fit`	nlmixr2 fit.
`varsVec`	character vector of variables that need to be added
`covarsVec`	character vector of covariates that need to be added
`catvarsVec`	character vector of categorical covariates that need to be added
`constraint`	theta cutoff. below cutoff then the theta will be fixed to zero.
`lassotype`	must be 'regular' , 'adaptive', 'adjusted'
`stratVar`	A variable to stratify on for cross-validation.
`...`	Other parameters to be passed to optimalTvaluelasso

Value

return fit of the selected lasso coefficients

Author(s)

Vishal Sarsani

Examples

```
## Not run:
oneCmt <- 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)
  })
}

d <- nlmixr2data::theo_sd
d$SEX <-0
d$SEX[d$ID<=6] <-1

fit <- nlmixr2(oneCmt, d, est = "saem", control = list(print = 0))
varsVec <- c("ka","cl","v")
covarsVec <- c("WT")
catvarsVec <- c("SEX")

# Model fit with regular lasso coefficients:

lassoDf <- regularmodel(fit,varsVec,covarsVec,catvarsVec)
# Model fit with adaptive lasso coefficients:

lassoDf <- regularmodel(fit,varsVec,covarsVec,catvarsVec,lassotype='adaptive')
# Model fit with adaptive-adjusted lasso coefficients:

lassoDf <- regularmodel(fit,varsVec,covarsVec,catvarsVec, lassotype='adjusted')

## End(Not run)
```

`rerunTopN` *Rerun Top N Original Models From A Search*

Description

Rerun Top N Original Models From A Search

Usage

```
rerunTopN(x, ...)

## S3 method for class 'linIIVSearch'
rerunTopN(x, n = 5, ...)
```

Arguments

x	a Search object
...	Other Parameters
n	number of models to rerun

Author(s)

Omar I. Elashkar

---

resSearch *Exhaustively Search for Residual Error Model*

Description

Exhaustively Search for Residual Error Model

Usage

```
resSearch(fit)
```

Arguments

fit	Fitted model object

Value

list of original model and summary for different residual models searched

Author(s)

Omar I. Elashkar

---

theoFitOde *Example single dose Theophylline ODE model*

Description

This is a nlmixr2 model that is pre-run so that it can be used in package testing and development. It is regenerated when `devtools::document()` is run on a source checkout. If there is a binary incompatability between the fit objects, re-documenting (or running `build/build.R`) will fix this nlmixr2 fit object.

Format

A (modified) data frame with 132 rows and 22 columns.

ID Patient identifier

TIME Time (hr)

DV Dependent variable (concentration)

PRED Predictions without any between subject variability

RES Population Residual

WRES Weighted Residuals under the FO assumption

IPRED Individual Predictions

IRES Individual Residuals

IWRES Individual Weighted Residuals

CPRED Conditional Prediction under the FOCE assumption

CRES Conditional Residuals under the FOCE assumption

CWRES Conditional Weighted Residuals under the FOCE assumption

eta.ka Between subject changes for ka

eta.cl Between subject changes for v

depot amount in the depot compartment

center amount in the central compartment

ka Individual ka values

cl Individual cl values

v Individual volume of distribution

tad Time after dose

dosenum Dose number

Index

∗ **Internal**
.nlmixrFormulaExpandStartParam, 4
.nlmixrFormulaParser, 5
.nlmixrFormulaParserRanef, 5
.nlmixrFormulaSetupIniFixed, 6
.renameOrOverwrite, 7
∗ **Multistart**
multistart, 33
multistartControl, 36
plot.nlmixr2Multistart, 41
∗ **Profiling**
fixedControl, 21
llpControl, 32
profile.nlmixr2FitCore, 43
profileFixed, 44
profileLlp, 45
profileNlmixr2FitCoreRet, 46
.nlmixrFormulaDataPrep, 3
.nlmixrFormulaExpandStartParam, 4
.nlmixrFormulaExpandStartParamSingle *(.nlmixrFormulaExpandStartParam)*, 4
.nlmixrFormulaParser, 5
.nlmixrFormulaParserRanef, 5
.nlmixrFormulaSetupIniFixed, 6
.nlmixrFormulaSetupIniRandom *(.nlmixrFormulaSetupIniFixed)*, 6
.renameOrOverwrite, 7

aaaNlmixr2ExtraCRAN, 7
adaptivelassoCoefficients, 8
addAllEtas, 9
addCatCovariates, 10
addCovariate, 11
addData2Rx, 12
addorremoveCovariate, 12
adjustedlassoCoefficients, 13

bootplot, 14
bootstrapFit, 15
buildcovInfo, 17
buildupatedUI, 18

covarSearchAuto, 18

extractEqHelper.if, 20

fixedControl, 21
fixedControl(), *33*, *44–46*
foldgen, 21

getMinAICFit, 22
getMinAICFit(), *37*

horseshoeSummardf, 23

iivSearch, 25
isBoundaryFit, 25
isLinearizeMatch, 26

knit_print.nlmixr2FitCore, 26
knit_print.rxUi *(knit_print.nlmixr2FitCore)*, 26

lassoCoefficients, 27
lassoSummardf, 28
linearize, 29
linearizePlot, 31
linModGen, 31
listModelsTested, 32
llpControl, 32
llpControl(), *21*, *44–46*
lotri::lotriNearPD(), *35*

multistart, 33
multistart(), *37*, *41*
multistartControl, 36
multistartControl(), *34, 35*, *41*

nlmixr2est::nlmixr, *38*

nlmixrFormula, 38
normalizedData, 39

optimUnisampling, 40

parallel::mclapply(), *35*
plot(), *33*
plot.nlmixr2Multistart, 41
plot.nlmixr2Multistart(), *35*, *37*
preconditionFit, 42
print.linIIVSearch, 42
profile.nlmixr2FitCore, 43
profile.nlmixr2FitCore(), *21*, *33*, *45, 46*
profileFixed, 44
profileFixed(), *21*, *33*, *44*, *46*
profileFixedSingle *(*profileFixed*)*, 44
profileLlp, 45
profileLlp(), *21*, *33*, *44–46*
profileNlmixr2FitCoreRet, 46
profileNlmixr2FitCoreRet(), *21*, *33*, *44–46*

regularmodel, 47
rerunTopN, 48
resSearch, 49

theoFitOde, 49