Package: babelmixr2 (via r-universe)

September 7, 2024

Type Package

Title Use 'nlmixr2' to Interact with Open Source and Commercial Software

Version 0.1.3

Description Run other estimation and simulation software via the 'nlmixr2' (Fidler et al (2019) <doi:10.1002/psp4.12445>) interface including 'PKNCA', 'NONMEM' and 'Monolix'. While not required, you can get/install the 'lixoftConnectors' package in the 'Monolix' installation, as described at the following url <https://monolix.lixoft.com/monolix-api/lixoftconnectors_installation/>. When 'lixoftConnectors' is available, 'Monolix' can be run directly instead of setting up command line usage.

License GPL (>= 3)

URL https://nlmixr2.github.io/babelmixr2/, https://github.com/nlmixr2/babelmixr2/

NeedsCompilation yes

Encoding UTF-8

Suggests testthat, nlmixr2data, withr, lixoftConnectors, PKNCA (>= 0.10.0), knitr, rmarkdown, spelling, PopED, units, vdiffr, dplyr

Depends R (>= 3.5), nlmixr2 (>= 2.0.8)

Imports checkmate, cli, digest, lotri, nlmixr2est (>= 2.1.6), nonmem2rx (>= 0.1.3), methods, qs, rex, rxode2

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

LinkingTo Rcpp, rxode2, RcppArmadillo, RcppEigen

Language en-US

VignetteBuilder knitr

Repository https://nlmixr2.r-universe.dev

RemoteUrl https://github.com/nlmixr2/babelmixr2

RemoteRef HEAD

RemoteSha 07e7f12ede17e00a3be6386b40fdd0101384250d

Contents

`.setupPopEDdatabase` *Setup the poped database*

Description

Setup the poped database

Usage

```
.setupPopEDdatabase(ui, data, control)
```

Arguments

`ui`	rxode2 ui function
`data`	babelmixr2 design data
`control`	PopED control

Value

PopED database

Author(s)

Matthew L. Fidler

`as.nlmixr2` *Convert an object to a nlmixr2 fit object*

Description

Convert an object to a nlmixr2 fit object

Usage

```
as.nlmixr2(
  x,
  ...,
  table = nlmixr2est::tableControl(),
  rxControl = rxode2::rxControl()
)

as.nlmixr(
  x,
  ...,
  table = nlmixr2est::tableControl(),
  rxControl = rxode2::rxControl()
)
```

Arguments

`x`	Object to convert
`...`	Other arguments
`table`	is the `nlmixr2est::tableControl()` options
`rxControl`	is the `rxode2::rxControl()` options, which is generally needed for how `addl` doses are handled in the translation

Value

nlmixr2 fit object

Author(s)

Matthew L. Fidler

Examples

```
# First read in the model (but without residuals)
mod <- nonmem2rx(system.file("mods/cpt/runODE032.ctl", package="nonmem2rx"),
                 determineError=FALSE, lst=".res", save=FALSE)
```

```
# define the model with residuals (and change the name of the
# parameters) In this step you need to be careful to not change the
# estimates and make sure the residual estimates are correct (could
# have to change var to sd).

 mod2 <-function() {
   ini({
     lcl <- 1.37034036528946
     lvc <- 4.19814911033061
     lq <- 1.38003493562413
     lvp <- 3.87657341967489
     RSV <- c(0, 0.196446108190896, 1)
     eta.cl ~ 0.101251418415006
     eta.v ~ 0.0993872449483344
     eta.q ~ 0.101302674763154
     eta.v2 ~ 0.0730497519364148
   })
   model({
     cmt(CENTRAL)
     cmt(PERI)
     cl <- exp(lcl + eta.cl)
     v <- exp(lvc + eta.v)
     q <- exp(lq + eta.q)
     v2 <- exp(lvp + eta.v2)
     v1 <- v
     scale1 <- v
     k21 <- q/v2
     k12 <- q/v
     d/dt(CENTRAL) <- k21 * PERI - k12 * CENTRAL - cl * CENTRAL/v1
     d/dt(PERI) <- -k21 * PERI + k12 * CENTRAL
     f <- CENTRAL/scale1
     f ~ prop(RSV)
   })
 }

# now we create another nonmem2rx object that validates the model above:

new <- as.nonmem2rx(mod2, mod)

# once that is done, you can translate to a full nlmixr2 fit (if you wish)

fit <- as.nlmixr2(new)

print(fit)
```

---

bblDatToMonolix *Convert nlmixr2-compatible data to other formats (if possible)*

Description

Convert nlmixr2-compatible data to other formats (if possible)

Usage

```
bblDatToMonolix(
  model,
  data,
  table = nlmixr2est::tableControl(),
  rxControl = rxode2::rxControl(),
  env = NULL
)

bblDatToNonmem(
  model,
  data,
  table = nlmixr2est::tableControl(),
  rxControl = rxode2::rxControl(),
  env = NULL
)

bblDatToRxode(
  model,
  data,
  table = nlmixr2est::tableControl(),
  rxControl = rxode2::rxControl(),
  env = NULL
)

bblDatToMrgsolve(
  model,
  data,
  table = nlmixr2est::tableControl(),
  rxControl = rxode2::rxControl(),
  env = NULL
)

bblDatToPknca(
  model,
  data,
  table = nlmixr2est::tableControl(),
  rxControl = rxode2::rxControl(),
  env = NULL
)
```

Arguments

`model`	rxode2 model for conversion

`data`	Input dataset.
`table`	is the table control; this is mostly to figure out if there are additional columns to keep.
`rxControl`	is the rxode2 control options; This is to figure out how to handle the addl dosing information.
`env`	When `NULL` (default) nothing is done. When an environment, the function `nlmixr2est::.foceiPreProce` `env, model, rxControl)` is called on the provided environment.

Value

With the function `bblDatToMonolix()` return a list with:

- Monolix compatible dataset ($monolix)
- Monolix ADM information ($adm)

With the function `nlmixrDataToNonmem()` return a dataset that is compatible with NONMEM.

With the function `nlmixrDataToMrgsolve()` return a dataset that is compatible with `mrgsolve`. Unlike NONMEM, it supports replacement events with `evid=8` (note with `rxode2` replacement `evid` is `5`).

With the function `nlmixrDataToRxode()` this will normalize the dataset to use newer `evid` definitions that are closer to NONMEM instead of any classic definitions that are used at a lower level

Author(s)

Matthew L. Fidler

Examples

```
pk.turnover.emax3 <- function() {
  ini({
    tktr <- log(1)
    tka <- log(1)
    tcl <- log(0.1)
    tv <- log(10)
    ##
    eta.ktr ~ 1
    eta.ka ~ 1
    eta.cl ~ 2
    eta.v ~ 1
    prop.err <- 0.1
    pkadd.err <- 0.1
    ##
    temax <- logit(0.8)
    tec50 <- log(0.5)
    tkout <- log(0.05)
    te0 <- log(100)
    ##
    eta.emax ~ .5
    eta.ec50  ~ .5
    eta.kout ~ .5
```

```
    eta.e0 ~ .5
    ##
    pdadd.err <- 10
  })
  model({
    ktr <- exp(tktr + eta.ktr)
    ka <- exp(tka + eta.ka)
    cl <- exp(tcl + eta.cl)
    v <- exp(tv + eta.v)
    emax = expit(temax+eta.emax)
    ec50 =  exp(tec50 + eta.ec50)
    kout = exp(tkout + eta.kout)
    e0 = exp(te0 + eta.e0)
    ##
    DCP = center/v
    PD=1-emax*DCP/(ec50+DCP)
    ##
    effect(0) = e0
    kin = e0*kout
    ##
    d/dt(depot) = -ktr * depot
    d/dt(gut) =  ktr * depot -ka * gut
    d/dt(center) =  ka * gut - cl / v * center
    d/dt(effect) = kin*PD -kout*effect
    ##
    cp = center / v
    cp ~ prop(prop.err) + add(pkadd.err)
    effect ~ add(pdadd.err) | pca
  })
}

bblDatToMonolix(pk.turnover.emax3, nlmixr2data::warfarin)

bblDatToNonmem(pk.turnover.emax3, nlmixr2data::warfarin)

bblDatToMrgsolve(pk.turnover.emax3, nlmixr2data::warfarin)

bblDatToRxode(pk.turnover.emax3, nlmixr2data::warfarin)
```

`getStandardColNames` *Determine standardized rxode2 column names from data*

Description

Determine standardized rxode2 column names from data

Usage

```
getStandardColNames(data)
```

Arguments

`data`	A data.frame as the source for column names

Value

A named character vector where the names are the standardized names and the values are either the name of the column from the data or `NA` if the column is not present in the data.

Examples

```
getStandardColNames(data.frame(ID=1, DV=2, Time=3, CmT=4))
```

`modelUnitConversion` *Unit conversion for pharmacokinetic models*

Description

Unit conversion for pharmacokinetic models

Usage

```
modelUnitConversion(
  dvu = NA_character_,
  amtu = NA_character_,
  timeu = NA_character_,
  volumeu = NA_character_
)
```

Arguments

`dvu`, `amtu`, `timeu`	The units for the DV, AMT, and TIME columns in the data
`volumeu`	The units for the volume parameters in the model

Value

A list with names for the units associated with each parameter ("amtu", "clearanceu", "volumeu", "timeu", "dvu") and the numeric value to multiply the modeled estimate (for example, `cp`) so that the model is consistent with the data units.

See Also

Other Unit conversion: `simplifyUnit()`

Examples

```
modelUnitConversion(dvu = "ng/mL", amtu = "mg", timeu = "hr", volumeu = "L")
```

monolixControl *Monolix Controller for nlmixr2*

Description

Monolix Controller for nlmixr2

Usage

```
monolixControl(
  nbSSDoses = 7,
  useLinearization = FALSE,
  stiff = FALSE,
  addProp = c("combined2", "combined1"),
  exploratoryAutoStop = FALSE,
  smoothingAutoStop = FALSE,
  burnInIterations = 5,
  smoothingIterations = 200,
  exploratoryIterations = 250,
  simulatedAnnealingIterations = 250,
  exploratoryInterval = 200,
  exploratoryAlpha = 0,
  omegaTau = 0.95,
  errorModelTau = 0.95,
  variability = c("none", "firstStage", "decreasing"),
  runCommand = getOption("babelmixr2.monolix", ""),
  rxControl = NULL,
  sumProd = FALSE,
  optExpression = TRUE,
  calcTables = TRUE,
  compress = TRUE,
  ci = 0.95,
  sigdigTable = NULL,
  absolutePath = FALSE,
  modelName = NULL,
  muRefCovAlg = TRUE,
  run = TRUE,
  ...
)
```

Arguments

`nbSSDoses`	Number of steady state doses (default 7)
`useLinearization`	Use linearization for log likelihood and fim.
`stiff`	boolean for using the stiff ODE solver

`addProp` specifies the type of additive plus proportional errors, the one where standard deviations add (combined1) or the type where the variances add (combined2).

The combined1 error type can be described by the following equation:

$$y = f + (a + b \times f^c) \times \varepsilon$$

The combined2 error model can be described by the following equation:

$$y = f + \sqrt{a^2 + b^2 \times f^{2\times c}} \times \varepsilon$$

Where:
- y represents the observed value
- f represents the predicted value
- a is the additive standard deviation
- b is the proportional/power standard deviation
- c is the power exponent (in the proportional case c=1)

`exploratoryAutoStop`
logical to turn on or off exploratory phase auto-stop of SAEM (default 250)

`smoothingAutoStop`
Boolean indicating if the smoothing should automatically stop (default `FALSE`)

`burnInIterations`
Number of burn in iterations

`smoothingIterations`
Number of smoothing iterations

`exploratoryIterations`
Number of iterations for exploratory phase (default 250)

`simulatedAnnealingIterations`
Number of simulating annealing iterations

`exploratoryInterval`
Minimum number of iterations in the exploratory phase (default 200)

`exploratoryAlpha`
Convergence memory in the exploratory phase (only used when `exploratoryAutoStop` is `TRUE`)

`omegaTau` Proportional rate on variance for simulated annealing

`errorModelTau` Proportional rate on error model for simulated annealing

`variability` This describes the methodology for parameters without variability. It could be: - Fixed throughout (none) - Variability in the first stage (firstStage) - Decreasing until it reaches the fixed value (decreasing)

`runCommand` is a shell command or function to run monolix; You can specify the default by `options("babelmixr2.monolix"="runMonolix")`. If it is empty and 'lixoftConnectors' is available, use lixoftConnectors to run monolix. See details for function usage.

`rxControl` ‘rxode2‘ ODE solving options during fitting, created with ‘rxControl()‘

`sumProd` Is a boolean indicating if the model should change multiplication to high precision multiplication and sums to high precision sums using the PreciseSums package. By default this is `FALSE`.

`optExpression`	Optimize the rxode2 expression to speed up calculation. By default this is turned on.
`calcTables`	This boolean is to determine if the foceiFit will calculate tables. By default this is `TRUE`
`compress`	Should the object have compressed items
`ci`	Confidence level for some tables. By default this is 0.95 or 95% confidence.
`sigdigTable`	Significant digits in the final output table. If not specified, then it matches the significant digits in the ‘sigdig‘ optimization algorithm. If ‘sigdig‘ is NULL, use 3.
`absolutePath`	Boolean indicating if the absolute path should be used for the monolix runs
`modelName`	Model name used to generate the NONMEM output. If `NULL` try to infer from the model name (could be `x` if not clear). Otherwise use this character for outputs.
`muRefCovAlg`	This controls if algebraic expressions that can be mu-referenced are treated as mu-referenced covariates by: 1. Creating a internal data-variable ‘nlmixrMuDerCov#‘ for each algebraic mu-referenced expression 2. Change the algebraic expression to ‘nlmixrMuDerCov# * mu_cov_theta‘ 3. Use the internal mu-referenced covariate for saem 4. After optimization is completed, replace ‘model()‘ with old ‘model()‘ expression 5. Remove ‘nlmixrMuDerCov#‘ from nlmix2 output In general, these covariates should be more accurate since it changes the system to a linear compartment model. Therefore, by default this is ‘TRUE‘.
`run`	Should monolix be run and the results be imported to nlmixr2? (Default is TRUE)
`...`	Ignored parameters

Details

If `runCommand` is given as a string, it will be called with the `system()` command like:

`runCommand mlxtran`.

For example, if `runCommand="'/path/to/monolix/mlxbsub2021' -p "` then the command line used would look like the following:

`'/path/to/monolix/mlxbsub2021' monolix.mlxtran`

If `runCommand` is given as a function, it will be called as `FUN(mlxtran, directory, ui)` to run Monolix. This allows you to run Monolix in any way that you may need, as long as you can write it in R. babelmixr2 will wait for the function to return before proceeding.

If `runCommand` is `NA`, `nlmixr()` will stop after writing the model files and without starting Monolix.

Note that you can get the translated monolix components from a parsed/compiled rxode2 ui object with `ui$monolixModel` and `ui$mlxtran`

Value

A monolix control object

Author(s)

Matthew Fidler

`nlmixr2Est.pknca` *Estimate starting parameters using PKNCA*

Description

Estimate starting parameters using PKNCA

Usage

```
## S3 method for class 'pknca'
nlmixr2Est(env, ...)
```

Arguments

`env`	Environment for the nlmixr2 estimation routines. This needs to have: - rxode2 ui object in ‘$ui‘ - data to fit in the estimation routine in ‘$data‘ - control for the estimation routine’s control options in ‘$ui‘
`...`	Other arguments provided to ‘nlmixr2Est()‘ provided for flexibility but not currently used inside nlmixr

Details

Parameters are estimated as follows:

- `ka` 4 half-lives to Tmax but not higher than 3: `log(2)/(tmax/4)`
- `vc` Inverse of dose-normalized Cmax
- `cl` Estimated as the median clearance
- `vp,vp2`2- and 4-fold the `vc`, respectively by default, controlled by the `vpMult` and `vp2Mult` arguments to `pkncaControl`
- `q,q2` 0.5- and 0.25-fold the `cl`, respectively by default, controlled by the `qMult` and `q2Mult` arguments to `pkncaControl`

The bounds for the parameter estimates are set to 10% of the first percentile and 10 times the 99th percentile. (For ka, the lower bound is set to the lower of 10% of the first percentile or 0.03 and the upper bound is not modified from 10 times the 99th percentile.)

Parameter estimation methods may be changed in a future version.

Value

A model with updated starting parameters. In the model a new element named "nca" will be available which includes the PKNCA results used for the calculation.

nonmemControl *NONMEM estimation control*

Description

NONMEM estimation control

Usage

```
nonmemControl(
  est = c("focei", "imp", "its", "posthoc"),
  advanOde = c("advan13", "advan8", "advan6"),
  cov = c("r,s", "r", "s", ""),
  maxeval = 1e+05,
  tol = 6,
  atol = 12,
  sstol = 6,
  ssatol = 12,
  sigl = 12,
  sigdig = 3,
  print = 1,
  extension = getOption("babelmixr2.nmModelExtension", ".nmctl"),
  outputExtension = getOption("babelmixr2.nmOutputExtension", ".lst"),
  runCommand = getOption("babelmixr2.nonmem", ""),
  iniSigDig = 5,
  protectZeros = FALSE,
  muRef = TRUE,
  addProp = c("combined2", "combined1"),
  rxControl = NULL,
  sumProd = FALSE,
  optExpression = TRUE,
  calcTables = TRUE,
  compress = TRUE,
  ci = 0.95,
  sigdigTable = NULL,
  readRounding = FALSE,
  readBadOpt = FALSE,
  niter = 100L,
  isample = 1000L,
  iaccept = 0.4,
  iscaleMin = 0.1,
  iscaleMax = 10,
  df = 4,
  seed = 14456,
  mapiter = 1,
  mapinter = 0,
  noabort = TRUE,
```

```
  modelName = NULL,
  muRefCovAlg = TRUE,
  run = TRUE,
  ...
)
```

Arguments

`est`	NONMEM estimation method
`advanOde`	The ODE solving method for NONMEM
`cov`	The NONMEM covariance method
`maxeval`	NONMEM's maxeval (for non posthoc methods)
`tol`	NONMEM tolerance for ODE solving advan
`atol`	NONMEM absolute tolerance for ODE solving
`sstol`	NONMEM tolerance for steady state ODE solving
`ssatol`	NONMEM absolute tolerance for steady state ODE solving
`sigl`	NONMEM sigl estimation option
`sigdig`	the significant digits for NONMEM
`print`	The print number for NONMEM
`extension`	NONMEM file extensions
`outputExtension`	Extension to use for the NONMEM output listing
`runCommand`	Command to run NONMEM (typically the path to "nmfe75") or a function. See the details for more information.
`iniSigDig`	How many significant digits are printed in $THETA and $OMEGA when the estimate is zero. Also controls the zero protection numbers
`protectZeros`	Add methods to protect divide by zero
`muRef`	Automatically mu-reference the control stream
`addProp`, `sumProd`, `optExpression`, `calcTables`, `compress`, `ci`, `sigdigTable`	Passed to `nlmixr2est::foceiControl`
`rxControl`	Options to pass to `rxode2::rxControl` for simulations
`readRounding`	Try to read NONMEM output when NONMEM terminated due to rounding errors
`readBadOpt`	Try to read NONMEM output when NONMEM terminated due to an apparent failed optimization
`niter`	number of iterations in NONMEM estimation methods
`isample`	Isample argument for NONMEM ITS estimation method
`iaccept`	Iaccept for NONMEM ITS estimation methods
`iscaleMin`	parameter for IMP NONMEM method (ISCALE_MIN)
`iscaleMax`	parameter for IMP NONMEM method (ISCALE_MAX)
`df`	degrees of freedom for IMP method

seed	is the seed for NONMEM methods
mapiter	the number of map iterations for IMP method
mapinter	is the MAPINTER parameter for the IMP method
noabort	Add the NOABORT option for $EST
modelName	Model name used to generate the NONMEM output. If NULL try to infer from the model name (could be x if not clear). Otherwise use this character for outputs.
muRefCovAlg	This controls if algebraic expressions that can be mu-referenced are treated as mu-referenced covariates by: 1. Creating a internal data-variable ‘nlmixrMuDerCov#‘ for each algebraic mu-referenced expression 2. Change the algebraic expression to ‘nlmixrMuDerCov# * mu_cov_theta‘ 3. Use the internal mu-referenced covariate for saem 4. After optimization is completed, replace ‘model()‘ with old ‘model()‘ expression 5. Remove ‘nlmixrMuDerCov#‘ from nlmix2 output In general, these covariates should be more accurate since it changes the system to a linear compartment model. Therefore, by default this is ‘TRUE‘.
run	Should NONMEM be run (and the files imported to nlmixr2); default is TRUE, but FALSE will simply create the NONMEM control stream and data file.
...	optional genRxControl argument controlling automatic rxControl generation.

Details

If runCommand is given as a string, it will be called with the system() command like:

runCommand controlFile outputFile.

For example, if runCommand="'/path/to/nmfe75'" then the command line used would look like the following:

'/path/to/nmfe75' one.cmt.nmctl one.cmt.lst

If runCommand is given as a function, it will be called as FUN(ctl, directory, ui) to run NONMEM. This allows you to run NONMEM in any way that you may need, as long as you can write it in R. babelmixr2 will wait for the function to return before proceeding.

If runCommand is NA, nlmixr() will stop after writing the model files and without starting NONMEM.

Value

babelmixr2 control option for generating NONMEM control stream and reading it back into babelmixr2/nlmixr2

Author(s)

Matthew L. Fidler

Examples

```
nonmemControl()
```

pkncaControl *PKNCA estimation control*

Description

PKNCA estimation control

Usage

```
pkncaControl(
  concu = NA_character_,
  doseu = NA_character_,
  timeu = NA_character_,
  volumeu = NA_character_,
  vpMult = 2,
  qMult = 1/2,
  vp2Mult = 4,
  q2Mult = 1/4,
  dvParam = "cp",
  groups = character(),
  sparse = FALSE,
  ncaData = NULL,
  ncaResults = NULL,
  rxControl = rxode2::rxControl()
)
```

Arguments

`concu`, `doseu`, `timeu`
: concentration, dose, and time units from the source data (passed to `PKNCA::pknca_units_table()`).

`volumeu`
: compartment volume for the model (if `NULL`, simplified units from source data will be used)

`vpMult`, `qMult`, `vp2Mult`, `q2Mult`
: Multipliers for vc and cl to provide initial estimates for vp, q, vp2, and q2

`dvParam`
: The parameter name in the model that should be modified for concentration unit conversions. It must be assigned on a line by itself, separate from the residual error model line.

`groups`
: Grouping columns for NCA summaries by group (required if `sparse = TRUE`)

`sparse`
: Are the concentration-time data sparse PK (commonly used in small nonclinical species or with terminal or difficult sampling) or dense PK (commonly used in clinical studies or larger nonclinical species)?

`ncaData`
: Data to use for calculating NCA parameters. Typical use is when a subset of the original data are informative for NCA.

`ncaResults`
: Already computed NCA results (a PKNCAresults object) to bypass automatic calculations. At least the following parameters must be calculated in the NCA: tmax, cmax.dn, cl.last

`rxControl` Control options sent to `rxode2::rxControl()`

Value

A list of parameters

`popedControl` *Control for a PopED design task*

Description

Control for a PopED design task

Usage

```
popedControl(
  stickyRecalcN = 4,
  maxOdeRecalc = 5,
  odeRecalcFactor = 10^(0.5),
  maxn = NULL,
  rxControl = NULL,
  sigdig = 4,
  important = NULL,
  unimportant = NULL,
 iFIMCalculationType = c("reduced", "full", "weighted", "loc", "reducedPFIM", "fullABC",
    "largeMat", "reducedFIMABC"),
  iApproximationMethod = c("fo", "foce", "focei", "foi"),
  iFOCENumInd = 1000,
  prior_fim = matrix(0, 0, 1),
  d_switch = c("d", "ed"),
  ofv_calc_type = c("lnD", "d", "a", "Ds", "inverse"),
  strEDPenaltyFile = "",
  ofv_fun = NULL,
  iEDCalculationType = c("mc", "laplace", "bfgs-laplace"),
  ED_samp_size = 45,
  bLHS = c("hypercube", "random"),
  bUseRandomSearch = TRUE,
  bUseStochasticGradient = TRUE,
  bUseLineSearch = TRUE,
  bUseExchangeAlgorithm = FALSE,
  bUseBFGSMinimizer = FALSE,
  EACriteria = c("modified", "fedorov"),
  strRunFile = "",
  poped_version = NULL,
  modtit = "PopED babelmixr2 model",
  output_file = "PopED_output_summary",
  output_function_file = "PopED_output_",
```

```
strIterationFileName = "PopED_current.R",
user_data = NULL,
ourzero = 1e-05,
dSeed = NULL,
line_opta = NULL,
line_optx = NULL,
bShowGraphs = FALSE,
use_logfile = FALSE,
m1_switch = c("central", "complex", "analytic", "ad"),
m2_switch = c("central", "complex", "analytic", "ad"),
hle_switch = c("central", "complex", "ad"),
gradff_switch = c("central", "complex", "analytic", "ad"),
gradfg_switch = c("central", "complex", "analytic", "ad"),
grad_all_switch = c("central", "complex"),
rsit_output = 5,
sgit_output = 1,
hm1 = 1e-05,
hlf = 1e-05,
hlg = 1e-05,
hm2 = 1e-05,
hgd = 1e-05,
hle = 1e-05,
AbsTol = 1e-06,
RelTol = 1e-06,
iDiffSolverMethod = NULL,
bUseMemorySolver = FALSE,
rsit = 300,
sgit = 150,
intrsit = 250,
intsgit = 50,
maxrsnullit = 50,
convergence_eps = 1e-08,
rslxt = 10,
rsla = 10,
cfaxt = 0.001,
cfaa = 0.001,
bGreedyGroupOpt = FALSE,
EAStepSize = 0.01,
EANumPoints = FALSE,
EAConvergenceCriteria = 1e-20,
bEANoReplicates = FALSE,
BFGSProjectedGradientTol = 1e-04,
BFGSTolerancef = 0.001,
BFGSToleranceg = 0.9,
BFGSTolerancex = 0.1,
ED_diff_it = 30,
ED_diff_percent = 10,
line_search_it = 50,
```

```
Doptim_iter = 1,
iCompileOption = c("none", "full", "mcc", "mpi"),
compileOnly = FALSE,
iUseParallelMethod = c("mpi", "matlab"),
MCC_Dep = NULL,
strExecuteName = "calc_fim.exe",
iNumProcesses = 2,
iNumChunkDesignEvals = -2,
Mat_Out_Pre = "parallel_output",
strExtraRunOptions = "",
dPollResultTime = 0.1,
strFunctionInputName = "function_input",
bParallelRS = FALSE,
bParallelSG = FALSE,
bParallelMFEA = FALSE,
bParallelLS = FALSE,
groupsize = NULL,
time = "time",
timeLow = "low",
timeHi = "high",
id = "id",
m = NULL,
x = NULL,
ni = NULL,
maxni = NULL,
minni = NULL,
maxtotni = NULL,
mintotni = NULL,
maxgroupsize = NULL,
mingroupsize = NULL,
maxtotgroupsize = NULL,
mintotgroupsize = NULL,
xt_space = NULL,
a = NULL,
maxa = NULL,
mina = NULL,
a_space = NULL,
x_space = NULL,
use_grouped_xt = FALSE,
grouped_xt = NULL,
use_grouped_a = FALSE,
grouped_a = NULL,
use_grouped_x = FALSE,
grouped_x = NULL,
our_zero = NULL,
auto_pointer = "",
user_distribution_pointer = "",
fixRes = FALSE,
```

```
  script = NULL,
  ...
)
```

Arguments

`stickyRecalcN` The number of bad ODE solves before reducing the atol/rtol for the rest of the problem.

`maxOdeRecalc` Maximum number of times to reduce the ODE tolerances and try to resolve the system if there was a bad ODE solve.

`odeRecalcFactor` The ODE recalculation factor when ODE solving goes bad, this is the factor the rtol/atol is reduced

`maxn` Maximum number of design points for optimization; By default this is declared by the maximum number of design points in the babelmixr2 dataset (when `NULL`)

`rxControl` ‘rxode2‘ ODE solving options during fitting, created with ‘rxControl()‘

`sigdig` Optimization significant digits. This controls:

- The tolerance of the inner and outer optimization is `10^-sigdig`
- The tolerance of the ODE solvers is `0.5*10^(-sigdig-2)`; For the sensitivity equations and steady-state solutions the default is `0.5*10^(-sigdig-1.5)` (sensitivity changes only applicable for liblsoda)
- The tolerance of the boundary check is `5 * 10 ^ (-sigdig + 1)`

`important` character vector of important parameters or NULL for default. This is used with Ds-optimality

`unimportant` character vector of unimportant parameters or NULL for default. This is used with Ds-optimality

`iFIMCalculationType` can be either an integer or a named value of the Fisher Information Matrix type:

- 0/"full" = Full FIM
- 1/"reduced" = Reduced FIM
- 2/"weighted" = weighted models
- 3/"loc" = Loc models
- 4/"reducedPFIM" = reduced FIM with derivative of SD of sigma as in PFIM
- 5/"fullABC" = FULL FIM parameterized with A,B,C matrices & derivative of variance
- 6/"largeMat" = Calculate one model switch at a time, good for large matrices
- 7/"reducedFIMABC" = =Reduced FIM parameterized with A,B,C matrices & derivative of variance

`iApproximationMethod` Approximation method for model, 0=FO, 1=FOCE, 2=FOCEI, 3=FOI

`iFOCENumInd` integer; number of individuals in focei solve

`prior_fim` matrix; prior FIM

`d_switch` integer or character option:

- 0/"ed" = ED design
- 1/"d" = D design

`ofv_calc_type` objective calculation type:

- 1/"d" = D-optimality". Determinant of the FIM: det(FIM)
- 2/"a" = "A-optimality". Inverse of the sum of the expected parameter variances: 1/trace_matrix(inv(FIM))
- 4/"lnD" = "lnD-optimality". Natural logarithm of the determinant of the FIM: log(det(FIM))
- 6/"Ds" = "Ds-optimality". Ratio of the Determinant of the FIM and the Determinant of the uninteresting rows and columns of the FIM: det(FIM)/det(FIM_u)
- 7/"inverse" = Inverse of the sum of the expected parameter RSE: 1/sum(get_rse(FIM,poped.db,use_pe

`strEDPenaltyFile`
Penalty function name or path and filename, empty string means no penalty. User defined criterion can be defined this way.

`ofv_fun` User defined function used to compute the objective function. The function must have a poped database object as its first argument and have "..." in its argument list. Can be referenced as a function or as a file name where the function defined in the file has the same name as the file. e.g. "cost.txt" has a function named "cost" in it.

`iEDCalculationType`
ED Integral Calculation type:

- 0/"mc" = Monte-Carlo-Integration
- 1/"laplace" = Laplace Approximation
- 2/"bfgs-laplace" = BFGS Laplace Approximation

`ED_samp_size` Sample size for E-family sampling

`bLHS` How to sample from distributions in E-family calculations. 0=Random Sampling, 1=LatinHyperCube –

`bUseRandomSearch`

- ********START OF Optimization algorithm SPECIFICATION OPTIONS************

Use random search (1=TRUE, 0=FALSE)

`bUseStochasticGradient`
Use Stochastic Gradient search (1=TRUE, 0=FALSE)

`bUseLineSearch` Use Line search (1=TRUE, 0=FALSE)

`bUseExchangeAlgorithm`
Use Exchange algorithm (1=TRUE, 0=FALSE)

`bUseBFGSMinimizer`
Use BFGS Minimizer (1=TRUE, 0=FALSE)

`EACriteria` Exchange Algorithm Criteria:

- 1/"modified" = Modified
- 2/"fedorov" = Fedorov

`strRunFile` Filename and path, or function name, for a run file that is used instead of the regular PopED call.

`poped_version`
- ********START OF Labeling and file names SPECIFICATION OPTIONS************

The current PopED version

`modtit` The model title

`output_file` Filename and path of the output file during search

`output_function_file`
Filename suffix of the result function file

`strIterationFileName`
Filename and path for storage of current optimal design

`user_data`
- ********START OF Miscellaneous SPECIFICATION OPTIONS************

User defined data structure that, for example could be used to send in data to the model

`ourzero` Value to interpret as zero in design

`dSeed` The seed number used for optimization and sampling – integer or -1 which creates a random seed `as.integer(Sys.time())` or NULL.

`line_opta` Vector for line search on continuous design variables (1=TRUE,0=FALSE)

`line_optx` Vector for line search on discrete design variables (1=TRUE,0=FALSE)

`bShowGraphs` Use graph output during search

`use_logfile` If a log file should be used (0=FALSE, 1=TRUE)

`m1_switch` Method used to calculate M1:
- 1/"central" = Central difference
- 0/"complex" = Complex difference
- 20/"analytic" = Analytic derivative
- 30/"ad" = Automatic differentiation

`m2_switch` Method used to calculate M2:
- 1/"central" = Central difference
- 0/"complex" = Complex difference
- 20/"analytic" = Analytic derivative
- 30/"ad" = Automatic differentiation

`hle_switch` Method used to calculate linearization of residual error:
- 1/"central" = Central difference
- 0/"complex" = Complex difference
- 30/"ad" = Automatic differentiation

`gradff_switch` Method used to calculate the gradient of the model:
- 1/"central" = Central difference
- 0/"complex" = Complex difference
- 20/"analytic" = Analytic derivative
- 30/"ad" = Automatic differentiation

`gradfg_switch` Method used to calculate the gradient of the parameter vector g:
- 1/"central" = Central difference
- 0/"complex" = Complex difference

- 20/"analytic" = Analytic derivative
- 30/"ad" = Automatic differentiation

`grad_all_switch`
: Method used to calculate all the gradients:
  - 1/"central" = Central difference
  - 0/"complex" = Complex difference

`rsit_output` : Number of iterations in random search between screen output

`sgit_output` : Number of iterations in stochastic gradient search between screen output

`hm1` : Step length of derivative of linearized model w.r.t. typical values

`hlf` : Step length of derivative of model w.r.t. g

`hlg` : Step length of derivative of g w.r.t. b

`hm2` : Step length of derivative of variance w.r.t. typical values

`hgd` : Step length of derivative of OFV w.r.t. time

`hle` : Step length of derivative of model w.r.t. sigma

`AbsTol` : The absolute tolerance for the diff equation solver

`RelTol` : The relative tolerance for the diff equation solver

`iDiffSolverMethod`
: The diff equation solver method, NULL as default.

`bUseMemorySolver`
: If the differential equation results should be stored in memory (1) or not (0)

`rsit` : Number of Random search iterations

`sgit` : Number of stochastic gradient iterations

`intrsit` : Number of Random search iterations with discrete optimization.

`intsgit` : Number of Stochastic Gradient search iterations with discrete optimization

`maxrsnullit` : Iterations until adaptive narrowing in random search

`convergence_eps`
: Stochastic Gradient convergence value, (difference in OFV for D-optimal, difference in gradient for ED-optimal)

`rslxt` : Random search locality factor for sample times

`rsla` : Random search locality factor for covariates

`cfaxt` : Stochastic Gradient search first step factor for sample times

`cfaa` : Stochastic Gradient search first step factor for covariates

`bGreedyGroupOpt`
: Use greedy algorithm for group assignment optimization

`EAStepSize` : Exchange Algorithm StepSize

`EANumPoints` : Exchange Algorithm NumPoints

`EAConvergenceCriteria`
: Exchange Algorithm Convergence Limit/Criteria

`bEANoReplicates`
: Avoid replicate samples when using Exchange Algorithm

`BFGSProjectedGradientTol`
: BFGS Minimizer Convergence Criteria Normalized Projected Gradient Tolerance

`BFGSTolerancef`
: BFGS Minimizer Line Search Tolerance f

`BFGSToleranceg`
: BFGS Minimizer Line Search Tolerance g

`BFGSTolerancex`
: BFGS Minimizer Line Search Tolerance x

`ED_diff_it`
: Number of iterations in ED-optimal design to calculate convergence criteria

`ED_diff_percent`
: ED-optimal design convergence criteria in percent

`line_search_it`
: Number of grid points in the line search

`Doptim_iter`
: Number of iterations of full Random search and full Stochastic Gradient if line search is not used

`iCompileOption`
: Compile options for PopED

- "none"/-1 = No compilation
- "full/0 or 3 = Full compilation
- "mcc"/1 or 4 = Only using MCC (shared lib)
- "mpi"/2 or 5 = Only MPI,

When using numbers, option 0,1,2 runs PopED and option 3,4,5 stops after compilation.

When using characters, the option `compileOnly` determines if the model is only compiled (and PopED is not run).

`compileOnly`
: logical; only compile the model, do not run PopED (in conjunction with `iCompileOption`)

`iUseParallelMethod`
: Parallel method to use

- 0/"matlab"= Matlab PCT
- 1/"mpi" = MPI

`MCC_Dep`
: Additional dependencies used in MCC compilation (mat-files), if several space separated

`strExecuteName`
: Compilation output executable name

`iNumProcesses`
: Number of processes to use when running in parallel (e.g. 3 = 2 workers, 1 job manager)

`iNumChunkDesignEvals`
: Number of design evaluations that should be evaluated in each process before getting new work from job manager

`Mat_Out_Pre`
: The prefix of the output mat file to communicate with the executable

`strExtraRunOptions`
: Extra options send to e$g. the MPI executable or a batch script, see execute_parallel$m for more information and options

`dPollResultTime`
: Polling time to check if the parallel execution is finished

`strFunctionInputName`
: The file containing the popedInput structure that should be used to evaluate the designs

`bParallelRS` If the random search is going to be executed in parallel

`bParallelSG` If the stochastic gradient search is going to be executed in parallel

`bParallelMFEA` If the modified exchange algorithm is going to be executed in parallel

`bParallelLS` If the line search is going to be executed in parallel

`groupsize` Vector defining the size of the different groups (num individuals in each group). If only one number then the number will be the same in every group.

`time` string that represents the time in the dataset (ie xt)

`timeLow` string that represents the lower design time (ie minxt)

`timeHi` string that represents the upper design time (ie maxmt)

`id` The id variable

`m` Number of groups in the study. Each individual in a group will have the same design.

`x` A matrix defining the initial discrete values for the model Each row is a group/individual.

`ni` Vector defining the number of samples for each group.

`maxni`

- ********START OF DESIGN SPACE OPTIONS************

Max number of samples per group/individual

`minni` Min number of samples per group/individual

`maxtotni` Number defining the maximum number of samples allowed in the experiment.

`mintotni` Number defining the minimum number of samples allowed in the experiment.

`maxgroupsize` Vector defining the max size of the different groups (max number of individuals in each group)

`mingroupsize` Vector defining the min size of the different groups (min num individuals in each group) –

`maxtotgroupsize`

The total maximal groupsize over all groups

`mintotgroupsize`

The total minimal groupsize over all groups

`xt_space` Cell array `cell` defining the discrete variables allowed for each xt value. Can also be a vector of values `c(1:10)` (same values allowed for all xt), or a list of lists `list(1:10, 2:23, 4:6)` (one for each value in xt in row major order or just for one row in xt, and all other rows will be duplicated).

`a` Matrix defining the initial continuous covariate values. n_rows=number of groups, n_cols=number of covariates. If the number of rows is one and the number of groups > 1 then all groups are assigned the same values.

`maxa` Vector defining the max value for each covariate. If a single value is supplied then all a values are given the same max value

`mina` Vector defining the min value for each covariate. If a single value is supplied then all a values are given the same max value

`a_space` Cell array `cell` defining the discrete variables allowed for each a value. Can also be a list of values `list(1:10)` (same values allowed for all a), or a list of lists `list(1:10, 2:23, 4:6)` (one for each value in a).

x_space	Cell array cell defining the discrete variables for each x value.
use_grouped_xt	Group sampling times between groups so that each group has the same values (TRUE or FALSE).
grouped_xt	Matrix defining the grouping of sample points. Matching integers mean that the points are matched. Allows for finer control than use_grouped_xt
use_grouped_a	Group continuous design variables between groups so that each group has the same values (TRUE or FALSE).
grouped_a	Matrix defining the grouping of continuous design variables. Matching integers mean that the values are matched. Allows for finer control than use_grouped_a.
use_grouped_x	Group discrete design variables between groups so that each group has the same values (TRUE or FALSE).
grouped_x	Matrix defining the grouping of discrete design variables. Matching integers mean that the values are matched. Allows for finer control than use_grouped_x.
our_zero	Value to interpret as zero in design.
auto_pointer	Filename and path, or function name, for the Autocorrelation function, empty string means no autocorrelation.
user_distribution_pointer	Filename and path, or function name, for user defined distributions for E-family designs
fixRes	boolean; Fix the residuals to what is specified by the model
script	write a PopED/rxode2 script that can be modified for more fine control. The default is NULL. When script is TRUE, the script is returned as a lines that would be written to a file and with the class babelmixr2popedScript. This allows it to be printed as the script on screen. When script is a file name (with an R extension), the script is written to that file.
...	other parameters for PopED control

Value

popedControl object

Author(s)

Matthew L. Fidler

rxToMonolix — *Convert RxODE syntax to monolix syntax*

Description

Convert RxODE syntax to monolix syntax

Usage

```
rxToMonolix(x, ui)
```

Arguments

x	Expression
ui	rxode2 ui

Value

Monolix syntax

Author(s)

Matthew Fidler

rxToNonmem — *Convert RxODE syntax to NONMEM syntax*

Description

Convert RxODE syntax to NONMEM syntax

Usage

```
rxToNonmem(x, ui)
```

Arguments

x	Expression
ui	rxode2 ui

Value

NONMEM syntax

Author(s)

Matthew Fidler

`simplifyUnit` *Simplify units by removing repeated units from the numerator and denominator*

Description

Simplify units by removing repeated units from the numerator and denominator

Usage

```
simplifyUnit(numerator = "", denominator = "")
```

Arguments

`numerator`	The numerator of the units (or the whole unit specification)
`denominator`	The denominator of the units (or NULL if `numerator` is the whole unit specification)

Details

`NA` or `""` for `numerator` and `denominator` are considered unitless.

Value

The units specified with units that are in both the numerator and denominator cancelled.

See Also

Other Unit conversion: `modelUnitConversion()`

Examples

```
simplifyUnit("kg", "kg/mL")
# units that don't match exactly are not cancelled
simplifyUnit("kg", "g/mL")
```

Index