Package 'RxODE'

October 27, 2019

Version 0.9.1-7

Title Facilities for Simulating from ODE-Based Models

Maintainer Wenping Wang <wwang8198@gmail.com>

Depends R (>= 3.5.0)

Suggests knitr, DT, data.table, nlme, shiny, tcltk, testthat, usethis, devtools, covr, rmarkdown, SnakeCharmR, dplyr (>= 0.8.0), tidyr, tibble, curl, gridExtra, microbenchmark, scales, stringi, htmltools, reticulate, rlang, installr

Imports Matrix, PreciseSums (>= 0.3), Rcpp (>= 0.12.3), brew, cli, crayon, digest, dparser (>= 0.1.8), ggforce, ggplot2, inline, magrittr, memoise, methods, mvnfast, pillar, rex, sys, units(>= 0.6-0), utils, assertthat, lotri, remotes

Description Facilities for running simulations from ordinary differential equation (ODE) models, such as pharmacometrics and other compartmental models. A compilation manager translates the ODE model into C, compiles it, and dynamically loads the object code into R for improved computational efficiency. An event table object facilitates the specification of complex dosing regimens (optional) and sampling schedules. NB: The use of this package requires both C and Fortran compilers, for details on their use with R please see Section 6.3, Appendix A, and Appendix D in the ``R Administration and Installation'' manual. Also the code is mostly released under GPL. The VODE and LSODA are in the public domain. The information is available in the inst/COPYRIGHTS.

BugReports https://github.com/nlmixrdevelopment/RxODE/issues

NeedsCompilation yes

VignetteBuilder knitr

License GPL (>= 3)

URL https://nlmixrdevelopment.github.io/RxODE/

RoxygenNote 6.1.1

Biarch true

LinkingTo dparser(>= 0.1.8), Rcpp (>= 0.12.3), RcppArmadillo(>= 0.9.300.2.0), PreciseSums (>= 0.3)

Encoding UTF-8

LazyData true

Language en-US

Author Matthew L. Fidler [aut] (<https://orcid.org/0000-0001-8538-6691>),
Melissa Hallow [aut],
Wenping Wang [aut, cre],
Zufar Mulyukov [ctb],
Justin Wilkins [ctb] (<https://orcid.org/0000-0002-7099-9396>),
Simon Frost [ctb],
Goro Fuji [ctb],
Morwenn [ctb],
Heng Li [ctb],
Yu Feng [ctb],
Alan Hindmarsh [ctb],
Linda Petzold [ctb],
Ernst Hairer [ctb],
Gerhard Wanner [ctb],
J Colinge [ctb],
Hadley Wickham [ctb],
G Grothendieck [ctb],
Robert Gentleman [ctb],
Daniel C. Dillon [ctb],
Ross Ihaka [ctb],
R core team [cph],
odepack authors [cph]

Repository CRAN

Date/Publication 2019-10-27 08:50:06 UTC

R topics documented:

.clearPipe 4
.rxFindPow 5
.rxRmPrint 6
.rxRmSens 6
.rxSymPyJacobian 7
.rxWinRtoolsPath 7
.setWarnIdSort 8
add.dosing 8
add.sampling 11
as.data.frame.rxEvid 13
as.data.table.rxEt 14
as.et 14
as_tibble.rxEt 15
coef.RxODE 15

cvPost . . . 16
et . . . 17
etExpand . . . 21
etRbind . . . 22
etRep . . . 24
etSeq . . . 27
eventTable . . . 30
genShinyApp.template . . . 32
is.rxEt . . . 34
is.rxSolve . . . 34
print.rxCoefSolve . . . 35
print.RxODE . . . 35
rinvchisq . . . 36
rxAddReturn . . . 36
rxAllowUnload . . . 37
rxAssignPtr . . . 37
rxC14 . . . 38
rxChain . . . 38
rxClean . . . 39
rxCompile . . . 39
rxControl . . . 41
rxCores . . . 48
rxDelete . . . 48
rxDfdy . . . 49
rxEvid . . . 49
rxFoExpandEta . . . 50
rxGetRxODE . . . 51
rxHtml . . . 51
rxInv . . . 52
rxIsCurrent . . . 52
rxLhs . . . 53
rxLock . . . 53
rxNorm . . . 54
RxODE . . . 54
rxOptExpr . . . 59
rxOptions . . . 60
rxParams . . . 61
rxPermissive . . . 62
rxProgress . . . 63
rxRateDur . . . 64
rxSetIni0 . . . 65
rxSetProd . . . 66
rxSetSum . . . 66
rxShiny . . . 67
rxSimThetaOmega . . . 68
rxStack . . . 69
rxState . . . 70
rxSumProdModel . . . 70

rxSymInvChol 71
rxSymPyFix 72
rxSymPySensitivity 72
rxSymPyVersion 73
rxSyncOptions 74
rxTempDir 74
rxTrans 74
rxUnloadAll 76
rxUse 76
rxValidate 77
rxWinPythonSetup 77
rxWinSetup 78
summary.RxODE 78
tibble 79

Index **80**

`.clearPipe` *Clear/Set pipeline*

Description

Clear/Set pipeline

Usage

```
.clearPipe(rx = NULL, inits = NULL, events = NULL, params = NULL,
  iCov = NULL, keep = NULL, thetaMat = NULL, omega = NULL,
  sigma = NULL, dfObs = NULL, dfSub = NULL, nSub = NULL,
  nStud = NULL)
```

Arguments

`rx`	RxODE object
`inits`	a vector of initial values of the state variables (e.g., amounts in each compartment), and the order in this vector must be the same as the state variables (e.g., PK/PD compartments);
`events`	an `eventTable` object describing the input (e.g., doses) to the dynamic system and observation sampling time points (see `eventTable`);
`params`	a numeric named vector with values for every parameter in the ODE system; the names must correspond to the parameter identifiers used in the ODE specification;
`iCov`	A data frame of individual non-time varying covariates to combine with the `params` to form a parameter data.frame.

keep	Columns to keep from either the input dataset or the iCov dataset. With the iCov dataset, the column is kept once per line. For the input dataset, if any records are added to the data LOCF (Last Observation Carried forward) imputation is performed.
thetaMat	Named theta matrix.
omega	Estimate of Covariance matrix. When omega is a list, assume it is a block matrix and convert it to a full matrix for simulations.
sigma	Named sigma covariance or Cholesky decomposition of a covariance matrix. The names of the columns indicate parameters that are simulated. These are simulated for every observation in the solved system.
dfObs	Degrees of freedom to sample the unexplained variability matrix from the inverse Wishart distribution (scaled) or scaled inverse chi squared distribution.
dfSub	Degrees of freedom to sample the between subject variability matrix from the inverse Wishart distribution (scaled) or scaled inverse chi squared distribution.
nSub	Number between subject variabilities (ETAs) simulated for every realization of the parameters.
nStud	Number virtual studies to characterize uncertainty in estimated parameters.

`.rxFindPow` *Find power THETAs for appropriate scaling*

Description

Find power THETAs for appropriate scaling

Usage

```
.rxFindPow(x)
```

Arguments

x	RxODE model that can be access by rxNorm

Value

THETA numbers of x^theta

Author(s)

Matthew L. Fidler

.rxRmPrint *Remove print statements*

Description

Remove print statements

Usage

```
.rxRmPrint(x)
```

Arguments

x	RxODE lines to remove

Value

RxODE with print lines removed.

Author(s)

Matthew L. Fidler

.rxRmSens *Remove sensitivity equations*

Description

Remove sensitivity equations

Usage

```
.rxRmSens(x)
```

Arguments

x	RxODE lines to remove

Value

Lines with d/dt(rx_sens_...._) removed.

Author(s)

Matthew L. Fidler

.rxSymPyJacobian *Calculate the full Jacobian for a model*

Description

This expand the model to calculate the Jacobian. This requires rSymPy.

Usage

```
.rxSymPyJacobian(model)
```

Arguments

model	RxODE family of objects

Value

RxODE syntax for model with Jacobian specified.

Author(s)

Matthew L. Fidler

.rxWinRtoolsPath *Setup Rtools path*

Description

Setup Rtools path

Usage

```
.rxWinRtoolsPath(rm.rtools = TRUE, rm.python = TRUE, retry = FALSE)
```

Arguments

rm.rtools	Remove the Rtools from the current path specs.
rm.python	Remove Python from the current path specs.
retry	Should you retry to find Rtools? If NA, don't throw an error if it isn't found.

Author(s)

Matthew L. Fidler

.setWarnIdSort *Turn on/off warnings for ID sorting.*

Description

Turn on/off warnings for ID sorting.

Usage

```
.setWarnIdSort(warnIdSort = TRUE)
```

Arguments

warnIdSort	Boolean for if the sorting warning is turned on or off.

Value

Nothing

Author(s)

Matthew Fidler

add.dosing *Add dosing to eventTable*

Description

This adds a dosing event to the event table. This is provided for piping syntax through magrittr

Usage

```
add.dosing(eventTable, dose, nbr.doses = 1L, dosing.interval = 24,
  dosing.to = 1L, rate = NULL, amount.units = NA_character_,
  start.time = 0, do.sampling = FALSE, time.units = NA_character_,
  ...)
```

Arguments

eventTable	eventTable object
dose	numeric scalar, dose amount in amount.units;
nbr.doses	integer, number of doses;
dosing.interval	required numeric scalar, time between doses in time.units, defaults to 24 of time.units="hours";

dosing.to	integer, compartment the dose goes into (first compartment by default);
rate	for infusions, the rate of infusion (default is NULL, for bolus dosing;
amount.units	optional string indicating the dosing units. Defaults to NA to indicate as per the original EventTable definition.
start.time	required dosing start time;
do.sampling	logical, should observation sampling records be added at the dosing times? Defaults to FALSE.
time.units	optional string indicating the time units. Defaults to "hours" to indicate as per the original EventTable definition.
...	Other parameters passed to et.

Value

eventTable with updated dosing (note the event table will be updated anyway)

Author(s)

Matthew L. Fidler

Matthew L Fidler, Wenping Wang

References

Wang W, Hallow K, James D (2015). "A Tutorial on RxODE: Simulating Differential Equation Pharmacometric Models in R." CPT: Pharmacometrics \& Systems Pharmacology, 5(1), 3-10. ISSN 2163-8306, <URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4728294/>.

See Also

eventTable, add.sampling, add.dosing, et, etRep, etRbind, RxODE

Examples

```
## Model from RxODE tutorial
mod1 <-RxODE({
    KA=2.94E-01;
    CL=1.86E+01;
    V2=4.02E+01;
    Q=1.05E+01;
    V3=2.97E+02;
    Kin=1;
    Kout=1;
    EC50=200;
    C2 = centr/V2;
    C3 = peri/V3;
    d/dt(depot) =-KA*depot;
    d/dt(centr) = KA*depot - CL*C2 - Q*C2 + Q*C3;
    d/dt(peri)  =                   Q*C2 - Q*C3;
    d/dt(eff)  = Kin - Kout*(1-C2/(EC50+C2))*eff;
```

```
});

## These are making the more complex regimens of the RxODE tutorial

## bid for 5 days
bid <- et(timeUnits="hr") %>%
       et(amt=10000,ii=12,until=set_units(5, "days"))

## qd for 5 days
qd <- et(timeUnits="hr") %>%
      et(amt=20000,ii=24,until=set_units(5, "days"))

## bid for 5 days followed by qd for 5 days

et <- seq(bid,qd) %>% et(seq(0,11*24,length.out=100));

bidQd <- rxSolve(mod1, et)

plot(bidQd, C2)


## Now Infusion for 5 days followed by oral for 5 days

##  note you can dose to a named compartment instead of using the compartment number
infusion <- et(timeUnits = "hr") %>%
      et(amt=10000, rate=5000, ii=24, until=set_units(5, "days"), cmt="centr")


qd <- et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(5, "days"), cmt="depot")

et <- seq(infusion,qd)

infusionQd <- rxSolve(mod1, et)

plot(infusionQd, C2)

## 2wk-on, 1wk-off

qd <- et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(2, "weeks"), cmt="depot")

et <- seq(qd, set_units(1,"weeks"), qd) %>%
     add.sampling(set_units(seq(0, 5.5,by=0.005),weeks))

wkOnOff <- rxSolve(mod1, et)

plot(wkOnOff, C2)

## You can also repeat the cycle easily with the rep function

qd <-et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(2, "weeks"), cmt="depot")

et <- etRep(qd, times=4, wait=set_units(1,"weeks")) %>%
     add.sampling(set_units(seq(0, 12.5,by=0.005),weeks))
```

```
repCycle4 <- rxSolve(mod1, et)

plot(repCycle4, C2)
```

`add.sampling` *Add sampling to eventTable*

Description

This adds a dosing event to the event table. This is provided for piping syntax through magrittr

Usage

```
add.sampling(eventTable, time, time.units = NA)
```

Arguments

`eventTable`	An eventTable object
`time`	a vector of time values (in `time.units`).
`time.units`	an optional string specifying the time units. Defaults to the units specified when the `EventTable` was initialized.

Value

eventTable with updated sampling. (Note the event table will be updated even if you don't reassign the eventTable)

Author(s)

Matthew L Fidler, Wenping Wang

References

Wang W, Hallow K, James D (2015). "A Tutorial on RxODE: Simulating Differential Equation Pharmacometric Models in R." CPT: Pharmacometrics \& Systems Pharmacology, 5(1), 3-10. ISSN 2163-8306, <URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4728294/>.

See Also

`eventTable`, `add.sampling`, `add.dosing`, `et`, `etRep`, `etRbind`, `RxODE`

Examples

```
## Model from RxODE tutorial
mod1 <-RxODE({
    KA=2.94E-01;
    CL=1.86E+01;
    V2=4.02E+01;
    Q=1.05E+01;
    V3=2.97E+02;
    Kin=1;
    Kout=1;
    EC50=200;
    C2 = centr/V2;
    C3 = peri/V3;
    d/dt(depot) =-KA*depot;
    d/dt(centr) = KA*depot - CL*C2 - Q*C2 + Q*C3;
    d/dt(peri)  =                    Q*C2 - Q*C3;
    d/dt(eff)  = Kin - Kout*(1-C2/(EC50+C2))*eff;
});

## These are making the more complex regimens of the RxODE tutorial

## bid for 5 days
bid <- et(timeUnits="hr") %>%
       et(amt=10000,ii=12,until=set_units(5, "days"))

## qd for 5 days
qd <- et(timeUnits="hr") %>%
      et(amt=20000,ii=24,until=set_units(5, "days"))

## bid for 5 days followed by qd for 5 days

et <- seq(bid,qd) %>% et(seq(0,11*24,length.out=100));

bidQd <- rxSolve(mod1, et)

plot(bidQd, C2)


## Now Infusion for 5 days followed by oral for 5 days

##  note you can dose to a named compartment instead of using the compartment number
infusion <- et(timeUnits = "hr") %>%
      et(amt=10000, rate=5000, ii=24, until=set_units(5, "days"), cmt="centr")


qd <- et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(5, "days"), cmt="depot")

et <- seq(infusion,qd)

infusionQd <- rxSolve(mod1, et)
```

```
plot(infusionQd, C2)

## 2wk-on, 1wk-off

qd <- et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(2, "weeks"), cmt="depot")

et <- seq(qd, set_units(1,"weeks"), qd) %>%
     add.sampling(set_units(seq(0, 5.5,by=0.005),weeks))

wkOnOff <- rxSolve(mod1, et)

plot(wkOnOff, C2)

## You can also repeat the cycle easily with the rep function

qd <-et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(2, "weeks"), cmt="depot")

et <- etRep(qd, times=4, wait=set_units(1,"weeks")) %>%
     add.sampling(set_units(seq(0, 12.5,by=0.005),weeks))

repCycle4 <- rxSolve(mod1, et)

plot(repCycle4, C2)
```

`as.data.frame.rxEvid` *Convert to data.frame*

Description

Convert to data.frame

Usage

```
## S3 method for class 'rxEvid'
as.data.frame(x, row.names = NULL, optional = FALSE,
  ..., nm = paste(deparse(substitute(x), width.cutoff = 500L), collapse =
  " "))
```

Arguments

`x`	any R object.
`row.names`	`NULL` or a character vector giving the row names for the data frame. Missing values are not allowed.
`optional`	logical. If `TRUE`, setting row names and converting column names (to syntactic names: see `make.names`) is optional. Note that all of R's **base** package `as.data.frame()` methods use `optional` only for column names treatment, basically with the meaning of `data.frame(*,check.names = !optional)`. See also the `make.names` argument of the `matrix` method.

...	additional arguments to be passed to or from methods.
nm	Name of column in new data frame

`as.data.table.rxEt` *Convert an event table to a data.table*

Description

Convert an event table to a data.table

Usage

```
as.data.table.rxEt(x, keep.rownames = FALSE, ...)
```

Arguments

x	An R object.
keep.rownames	Default is FALSE. If TRUE, adds the input object's names as a separate column named "rn". keep.rownames = "id" names the column "id" instead.
...	Additional arguments to be passed to or from other methods.

`as.et` *Coerce object to data.frame*

Description

Coerce object to data.frame

Usage

```
as.et(x, ...)

## Default S3 method:
as.et(x, ...)
```

Arguments

x	Object to coerce to et.
...	Other parameters

as_tibble.rxEt *Convert to tbl*

Description

Convert to tbl

Usage

```
as_tibble.rxEt(x, ...)

as.tbl.rxEt(x, ...)
```

Arguments

x	RxODE event table
...	Other arguments to as.tbl

Value

tibble

coef.RxODE *Return the RxODE coefficients*

Description

This returns the parameters , state variables

Usage

```
## S3 method for class 'RxODE'
coef(object, ...)

## S3 method for class 'RxCompilationManager'
coef(...)

## S3 method for class 'solveRxODE'
coef(object, ...)

## S3 method for class 'rxDll'
coef(...)
```

Arguments

object	is an RxODE object
...	ignored arguments

Value

a rxCoef object with the following

params	is a list of strings for parameters for the RxODE object
state	is a list of strings for the names of each state in the RxODE object.
ini	is the model specified default values for the parameters.
RxODE	is the referring RxODE object

Author(s)

Matthew L.Fidler

---

cvPost *Sample a covariance Matrix from the Posterior Inverse Wishart distribution.*

Description

Note this Inverse wishart rescaled to match the original scale of the covariance matrix.

Usage

```
cvPost(nu, omega, n = 1L, omegaIsChol = FALSE, returnChol = FALSE)
```

Arguments

nu	Degrees of Freedom (Number of Observations) for covariance matrix simulation.
omega	Estimate of Covariance matrix.
n	Number of Matrices to sample. By default this is 1.
omegaIsChol	is an indicator of if the omega matrix is in the Cholesky decomposition.
returnChol	Return the Cholesky decomposition of the covariance matrix sample.

Details

If your covariance matrix is a 1x1 matrix, this uses an scaled inverse chi-squared which is equivalent to the Inverse Wishart distribution in the uni-directional case.

Value

a matrix (n=1) or a list of matrices (n > 1)

Author(s)

Matthew L.Fidler & Wenping Wang

Examples

```
## Sample a single covariance.
draw1 <- cvPost(3, matrix(c(1,.3,.3,1),2,2))

## Sample 3 covariances
set.seed(42)
draw3 <- cvPost(3, matrix(c(1,.3,.3,1),2,2), n=3)

## Sample 3 covariances, but return the cholesky decomposition
set.seed(42)
draw3c <- cvPost(3, matrix(c(1,.3,.3,1),2,2), n=3, returnChol=TRUE)
```

---

et *Event Table Function*

Description

Event Table Function

Usage

```
et(x, ..., envir = parent.frame())

## S3 method for class 'RxODE'
et(x, ..., envir = parent.frame())

## S3 method for class 'rxSolve'
et(x, ..., envir = parent.frame())

## S3 method for class 'rxParams'
et(x, ..., envir = parent.frame())

## Default S3 method:
et(x, ..., time, amt, evid, cmt, ii, addl, ss, rate, dur,
  until, id, amountUnits, timeUnits, addSampling, envir = parent.frame(),
  by = NULL, length.out = NULL)
```

Arguments

x	This is the first argument supplied to the event table. This is named to allow et to be used in a pipe-line with arbitrary objects.
...	Times or event tables. They can also be one of the named arguments below.

envir — the environment in which expr is to be evaluated. May also be NULL, a list, a data frame, a pairlist or an integer as specified to sys.call.

time — Time is the time of the dose or the sampling times. This can also be unspecified and is determined by the object type (list or numeric/integer).

amt — Amount of the dose. If specified, this assumes a dosing record, instead of a sampling record.

evid — Event ID; This can be:

- 0 An observation. This can also be specified as evid=obs
- 1 A dose observation. This can also be specified as evid=dose
- 2 A non-dose event. This can also be specified as evid=other.
- 3 A reset event. A reset event resets all the compartment values to zero and turns off all infusions. This can also be specified as evid=reset.
- 4 Dose and reset event. This can also be specified as evid=doseReset or evid=resetDose

cmt — Compartment name or number. If a number, this is an integer starting at 1. Negative compartments are not supported (there is no way to turn off a compartment currently). If the compartment is a name, the compartment name is changed to the correct state/compartment number before running the simulation.

Can also specify cmt as dosing.to, dose.to, doseTo, dosingTo, and state.

ii — When specifying a dose, this is the inter-dose interval for ss, addl and until options (described below).

addl — The number of additional doses at a inter-dose interval after one dose.

ss — Steady state flag; It can be one of:

- 0 This dose is not a steady state dose
- 1 This dose is a steady state dose with the between/inter dose interval of ii
- 2 This is a steady state dose that uses the super-position principle to allow more complex steady states, like 10 mg in the morning and 20 mg at night, or dosing at 8 am 12 pm and 8 pm instead of every 12 hours. Since it uses the super positioning principle, it only makes sense when you know the kinetics are linear.

All other values of SS are currently invalid.

rate — When positive, this is the rate of infusion. Otherwise:

- 0 No infusion is on this record.
- -1 Rate of this record is modeled by rate(cmt) = in the RxODE model. You may also specify type or rate by rate=model
- -2 Duration of this record is modeled by dur(cmt) = in the RxODE model. You may also specify this type of rate by dur=model or rate=dur.

When a modeled bioavailability is applied to positive rates (rate > 0), the duration of infusion is changed. This is because the data specify the rate and amount, the only think ghat modeled bioavailability can affect is duration.

If instead you want the modeled bioavailability to increase the rate of infusion instead of the duration of infusion, specify the dur instead or model the duration with rate=2.

dur	Duration of infusion. When amt and dur are specified the rate is calculated from the two data items. When dur is specified instead of rate, the bioavailability changes will increase rate instead of duration.
until	This is the time until the dosing should end. It can be an easier way to figure out how many additional doses are needed over your sampling period.
id	A integer vector of IDs to add or remove from the event table. If the event table is identical for each ID, then you may expand it to include all the IDs in this vector. All the negative IDs in this vector will be removed.
amountUnits	The units for the dosing records (amt)
timeUnits	The units for the time records (time)
addSampling	This is a boolean indicating if a sampling time should be added at the same time as a dosing time. By default this is FALSE.
by	When there are no observations in the event table, this is the amount to increment for the observations between from and to.
length.out	The number of observations to create if there isn't any observations in the event table. By default this is 200.

Value

A new event table

Author(s)

Matthew L Fidler, Wenping Wang

References

Wang W, Hallow K, James D (2015). "A Tutorial on RxODE: Simulating Differential Equation Pharmacometric Models in R." CPT: Pharmacometrics \& Systems Pharmacology, 5(1), 3-10. ISSN 2163-8306, <URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4728294/>.

See Also

eventTable, add.sampling, add.dosing, et, etRep, etRbind, RxODE

Examples

```
## Model from RxODE tutorial
mod1 <-RxODE({
    KA=2.94E-01;
    CL=1.86E+01;
    V2=4.02E+01;
    Q=1.05E+01;
    V3=2.97E+02;
    Kin=1;
    Kout=1;
    EC50=200;
```

```
    C2 = centr/V2;
    C3 = peri/V3;
    d/dt(depot) =-KA*depot;
    d/dt(centr) = KA*depot - CL*C2 - Q*C2 + Q*C3;
    d/dt(peri)  =                    Q*C2 - Q*C3;
    d/dt(eff)  = Kin - Kout*(1-C2/(EC50+C2))*eff;
});

## These are making the more complex regimens of the RxODE tutorial

## bid for 5 days
bid <- et(timeUnits="hr") %>%
       et(amt=10000,ii=12,until=set_units(5, "days"))

## qd for 5 days
qd <- et(timeUnits="hr") %>%
      et(amt=20000,ii=24,until=set_units(5, "days"))

## bid for 5 days followed by qd for 5 days

et <- seq(bid,qd) %>% et(seq(0,11*24,length.out=100));

bidQd <- rxSolve(mod1, et)

plot(bidQd, C2)


## Now Infusion for 5 days followed by oral for 5 days

##  note you can dose to a named compartment instead of using the compartment number
infusion <- et(timeUnits = "hr") %>%
      et(amt=10000, rate=5000, ii=24, until=set_units(5, "days"), cmt="centr")


qd <- et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(5, "days"), cmt="depot")

et <- seq(infusion,qd)

infusionQd <- rxSolve(mod1, et)

plot(infusionQd, C2)

## 2wk-on, 1wk-off

qd <- et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(2, "weeks"), cmt="depot")

et <- seq(qd, set_units(1,"weeks"), qd) %>%
     add.sampling(set_units(seq(0, 5.5,by=0.005),weeks))

wkOnOff <- rxSolve(mod1, et)

plot(wkOnOff, C2)
```

```
## You can also repeat the cycle easily with the rep function

qd <-et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(2, "weeks"), cmt="depot")

et <- etRep(qd, times=4, wait=set_units(1,"weeks")) %>%
     add.sampling(set_units(seq(0, 12.5,by=0.005),weeks))

repCycle4 <- rxSolve(mod1, et)

plot(repCycle4, C2)
```

---

etExpand *Expand additional doses*

Description

Expand additional doses

Usage

```
etExpand(et)
```

Arguments

et	Event table to expand additional doses for.

Value

New event table with ‘addl‘ doses expanded

Author(s)

Matthew Fidler

Examples

```
ev <- et(amt=3,ii=24,until=240);
print(ev)
etExpand(ev) # expands event table, but doesn't modify it

print(ev)

ev$expand() ## Expands the current event table and saves it in ev
```

etRbind *Combining event tables*

Description

Combining event tables

Usage

```
etRbind(..., samples = c("use", "clear"), waitII = c("smart", "+ii"),
  id = c("merge", "unique"))

## S3 method for class 'rxEt'
rbind(..., deparse.level = 1)
```

Arguments

`...`	The event tables and optionally time between event tables, called waiting times in this help document.
`samples`	How to handle samples when repeating an event table. The options are: • "clear" Clear sampling records before combining the datasets • "use" Use the sampling records when combining the datasets
`waitII`	This determines how waiting times between events are handled. The options are: • `"smart"` This "smart" handling of waiting times is the default option. In this case, if the waiting time is above the last observed inter-dose interval in the first combined event table, then the actual time between doses is given by the wait time. If it is smaller than the last observed inter-dose interval, the time between event tables is given by the inter-dose interval + the waiting time between event tables. • `"+ii"` In this case, the wait time is added to the inter-dose interval no matter the length of the wait time or inter-dose interval
`id`	This is how rbind will handle IDs. There are two different types of options: • merge with `id="merge"`, the IDs are merged together, overlapping IDs would be merged into a single event table. • unique with `id="unique"`, the IDs will be renumbered so that the IDs in all the event tables are not overlapping.
`deparse.level`	The `deparse.level` of a traditional `rbind` is ignored.

Value

An event table

Author(s)

Matthew L Fidler

Matthew L Fidler, Wenping Wang

References

Wang W, Hallow K, James D (2015). "A Tutorial on RxODE: Simulating Differential Equation Pharmacometric Models in R." CPT: Pharmacometrics \& Systems Pharmacology, 5(1), 3-10. ISSN 2163-8306, <URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4728294/>.

See Also

eventTable, add.sampling, add.dosing, et, etRep, etRbind, RxODE

Examples

```
## Model from RxODE tutorial
mod1 <-RxODE({
    KA=2.94E-01;
    CL=1.86E+01;
    V2=4.02E+01;
    Q=1.05E+01;
    V3=2.97E+02;
    Kin=1;
    Kout=1;
    EC50=200;
    C2 = centr/V2;
    C3 = peri/V3;
    d/dt(depot) =-KA*depot;
    d/dt(centr) = KA*depot - CL*C2 - Q*C2 + Q*C3;
    d/dt(peri)  =                    Q*C2 - Q*C3;
    d/dt(eff)  = Kin - Kout*(1-C2/(EC50+C2))*eff;
});

## These are making the more complex regimens of the RxODE tutorial

## bid for 5 days
bid <- et(timeUnits="hr") %>%
       et(amt=10000,ii=12,until=set_units(5, "days"))

## qd for 5 days
qd <- et(timeUnits="hr") %>%
      et(amt=20000,ii=24,until=set_units(5, "days"))

## bid for 5 days followed by qd for 5 days

et <- seq(bid,qd) %>% et(seq(0,11*24,length.out=100));

bidQd <- rxSolve(mod1, et)
```

```
plot(bidQd, C2)


## Now Infusion for 5 days followed by oral for 5 days

##  note you can dose to a named compartment instead of using the compartment number
infusion <- et(timeUnits = "hr") %>%
      et(amt=10000, rate=5000, ii=24, until=set_units(5, "days"), cmt="centr")


qd <- et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(5, "days"), cmt="depot")

et <- seq(infusion,qd)

infusionQd <- rxSolve(mod1, et)

plot(infusionQd, C2)

## 2wk-on, 1wk-off

qd <- et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(2, "weeks"), cmt="depot")

et <- seq(qd, set_units(1,"weeks"), qd) %>%
     add.sampling(set_units(seq(0, 5.5,by=0.005),weeks))

wkOnOff <- rxSolve(mod1, et)

plot(wkOnOff, C2)

## You can also repeat the cycle easily with the rep function

qd <-et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(2, "weeks"), cmt="depot")

et <- etRep(qd, times=4, wait=set_units(1,"weeks")) %>%
     add.sampling(set_units(seq(0, 12.5,by=0.005),weeks))

repCycle4 <- rxSolve(mod1, et)

plot(repCycle4, C2)
```

---

etRep *Repeat an RxODE event table*

Description

Repeat an RxODE event table

Usage

```
etRep(x, times = 1, length.out = NA, each = NA, n = NULL,
  wait = 0, id = integer(0), samples = c("clear", "use"),
  waitII = c("smart", "+ii"), ii = 24)

## S3 method for class 'rxEt'
rep(x, ...)
```

Arguments

x	An RxODE event table
times	Number of times to repeat the event table
length.out	Invalid with RxODE event tables, will throw an error if used.
each	Invalid with RxODE event tables, will throw an error if used.
n	The number of times to repeat the event table. Overrides times.
wait	Waiting time between each repeated event table. By default there is no waiting, or wait=0
id	A integer vector of IDs to add or remove from the event table. If the event table is identical for each ID, then you may expand it to include all the IDs in this vector. All the negative IDs in this vector will be removed.
samples	How to handle samples when repeating an event table. The options are: • "clear" Clear sampling records before combining the datasets • "use" Use the sampling records when combining the datasets
waitII	This determines how waiting times between events are handled. The options are: • "smart" This "smart" handling of waiting times is the default option. In this case, if the waiting time is above the last observed inter-dose interval in the first combined event table, then the actual time between doses is given by the wait time. If it is smaller than the last observed inter-dose interval, the time between event tables is given by the inter-dose interval + the waiting time between event tables. • "+ii" In this case, the wait time is added to the inter-dose interval no matter the length of the wait time or inter-dose interval
ii	When specifying a dose, this is the inter-dose interval for ss, addl and until options (described below).
...	Times or event tables. They can also be one of the named arguments below.

Author(s)

Matthew L Fidler, Wenping Wang

References

Wang W, Hallow K, James D (2015). "A Tutorial on RxODE: Simulating Differential Equation Pharmacometric Models in R." CPT: Pharmacometrics \& Systems Pharmacology, 5(1), 3-10. ISSN 2163-8306, <URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4728294/>.

See Also

eventTable, add.sampling, add.dosing, et, etRep, etRbind, RxODE

Examples

```
## Model from RxODE tutorial
mod1 <-RxODE({
    KA=2.94E-01;
    CL=1.86E+01;
    V2=4.02E+01;
    Q=1.05E+01;
    V3=2.97E+02;
    Kin=1;
    Kout=1;
    EC50=200;
    C2 = centr/V2;
    C3 = peri/V3;
    d/dt(depot) =-KA*depot;
    d/dt(centr) = KA*depot - CL*C2 - Q*C2 + Q*C3;
    d/dt(peri)  =                Q*C2 - Q*C3;
    d/dt(eff)  = Kin - Kout*(1-C2/(EC50+C2))*eff;
});

## These are making the more complex regimens of the RxODE tutorial

## bid for 5 days
bid <- et(timeUnits="hr") %>%
       et(amt=10000,ii=12,until=set_units(5, "days"))

## qd for 5 days
qd <- et(timeUnits="hr") %>%
      et(amt=20000,ii=24,until=set_units(5, "days"))

## bid for 5 days followed by qd for 5 days

et <- seq(bid,qd) %>% et(seq(0,11*24,length.out=100));

bidQd <- rxSolve(mod1, et)

plot(bidQd, C2)


## Now Infusion for 5 days followed by oral for 5 days

##  note you can dose to a named compartment instead of using the compartment number
infusion <- et(timeUnits = "hr") %>%
      et(amt=10000, rate=5000, ii=24, until=set_units(5, "days"), cmt="centr")


qd <- et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(5, "days"), cmt="depot")
```

```
et <- seq(infusion,qd)

infusionQd <- rxSolve(mod1, et)

plot(infusionQd, C2)

## 2wk-on, 1wk-off

qd <- et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(2, "weeks"), cmt="depot")

et <- seq(qd, set_units(1,"weeks"), qd) %>%
     add.sampling(set_units(seq(0, 5.5,by=0.005),weeks))

wkOnOff <- rxSolve(mod1, et)

plot(wkOnOff, C2)

## You can also repeat the cycle easily with the rep function

qd <-et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(2, "weeks"), cmt="depot")

et <- etRep(qd, times=4, wait=set_units(1,"weeks")) %>%
     add.sampling(set_units(seq(0, 12.5,by=0.005),weeks))

repCycle4 <- rxSolve(mod1, et)

plot(repCycle4, C2)
```

---

etSeq *Sequence of event tables*

Description

This combines a sequence of event tables.

Usage

```
etSeq(..., samples = c("clear", "use"), waitII = c("smart", "+ii"),
  ii = 24)

## S3 method for class 'rxEt'
seq(...)
```

Arguments

...	The event tables and optionally time between event tables, called waiting times in this help document.

`samples` How to handle samples when repeating an event table. The options are:

- "clear" Clear sampling records before combining the datasets
- "use" Use the sampling records when combining the datasets

`waitII` This determines how waiting times between events are handled. The options are:

- `"smart"` This "smart" handling of waiting times is the default option. In this case, if the waiting time is above the last observed inter-dose interval in the first combined event table, then the actual time between doses is given by the wait time. If it is smaller than the last observed inter-dose interval, the time between event tables is given by the inter-dose interval + the waiting time between event tables.
- `"+ii"` In this case, the wait time is added to the inter-dose interval no matter the length of the wait time or inter-dose interval

`ii` If there was no inter-dose intervals found in the event table, assume that the interdose interval is given by this `ii` value. By default this is `24`.

Details

This sequences all the event tables in added in the argument list `....` By default when combining the event tables the offset is at least by the last inter-dose interval in the prior event table (or `ii`). If you separate any of the event tables by a number, the event tables will be separated at least the wait time defined by that number or the last inter-dose interval.

Author(s)

Matthew L Fidler, Wenping Wang

References

Wang W, Hallow K, James D (2015). "A Tutorial on RxODE: Simulating Differential Equation Pharmacometric Models in R." CPT: Pharmacometrics \& Systems Pharmacology, 5(1), 3-10. ISSN 2163-8306, <URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4728294/>.

See Also

`eventTable`, `add.sampling`, `add.dosing`, `et`, `etRep`, `etRbind`, `RxODE`

Examples

```
## Model from RxODE tutorial
mod1 <-RxODE({
    KA=2.94E-01;
    CL=1.86E+01;
    V2=4.02E+01;
    Q=1.05E+01;
    V3=2.97E+02;
    Kin=1;
    Kout=1;
```

```
    EC50=200;
    C2 = centr/V2;
    C3 = peri/V3;
    d/dt(depot) =-KA*depot;
    d/dt(centr) = KA*depot - CL*C2 - Q*C2 + Q*C3;
    d/dt(peri)  =                    Q*C2 - Q*C3;
    d/dt(eff)  = Kin - Kout*(1-C2/(EC50+C2))*eff;
});

## These are making the more complex regimens of the RxODE tutorial

## bid for 5 days
bid <- et(timeUnits="hr") %>%
       et(amt=10000,ii=12,until=set_units(5, "days"))

## qd for 5 days
qd <- et(timeUnits="hr") %>%
      et(amt=20000,ii=24,until=set_units(5, "days"))

## bid for 5 days followed by qd for 5 days

et <- seq(bid,qd) %>% et(seq(0,11*24,length.out=100));

bidQd <- rxSolve(mod1, et)

plot(bidQd, C2)


## Now Infusion for 5 days followed by oral for 5 days

##  note you can dose to a named compartment instead of using the compartment number
infusion <- et(timeUnits = "hr") %>%
      et(amt=10000, rate=5000, ii=24, until=set_units(5, "days"), cmt="centr")


qd <- et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(5, "days"), cmt="depot")

et <- seq(infusion,qd)

infusionQd <- rxSolve(mod1, et)

plot(infusionQd, C2)

## 2wk-on, 1wk-off

qd <- et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(2, "weeks"), cmt="depot")

et <- seq(qd, set_units(1,"weeks"), qd) %>%
     add.sampling(set_units(seq(0, 5.5,by=0.005),weeks))

wkOnOff <- rxSolve(mod1, et)

plot(wkOnOff, C2)
```

```
## You can also repeat the cycle easily with the rep function

qd <-et(timeUnits = "hr") %>% et(amt=10000, ii=24, until=set_units(2, "weeks"), cmt="depot")

et <- etRep(qd, times=4, wait=set_units(1,"weeks")) %>%
     add.sampling(set_units(seq(0, 12.5,by=0.005),weeks))

repCycle4 <- rxSolve(mod1, et)

plot(repCycle4, C2)
```

`eventTable` *Create an event table object*

Description

Initializes an object of class 'EventTable' with methods for adding and querying dosing and observation records

Usage

```
eventTable(amount.units = NA, time.units = NA)
```

Arguments

`amount.units`	string denoting the amount dosing units, e.g., "mg", "ug". Default to `NA` to denote unspecified units. It could also be a solved RxODE object. In that case, eventTable(obj) returns the eventTable that was used to solve the RxODE object.
`time.units`	string denoting the time units, e.g., "hours", "days". Default to `"hours"`.
	An `eventTable` is an object that consists of a data.frame storing ordered time-stamped events of an (unspecified) PK/PD dynamic system, units (strings) for dosing and time records, plus a list of functions to add and extract event records.
	Currently, events can be of two types: dosing events that represent inputs to the system and sampling time events that represent observations of the system with 'amount.units' and 'time.units', respectively. In the future, additional events may include resetting of state variables (compartments), for instance, to indicate time after "wash-out", etc.

Value

A modified data.frame with the following accessible functions:

`get.EventTable`	returns the current event table.

add.dosing	adds dosing records to the event table. Its arguments are dose: numeric scalar, dose amount in amount.units; nbr.doses: integer, number of doses; dosing.interval: required numeric scalar, time between doses in time.units, defaults to 24 of time.units="hours"; dosing.to: integer, compartment the dose goes into (first compartment by default); rate: for infusions, the rate of infusion (default is NULL, for bolus dosing; start.time: required dosing start time; do.sampling: logical, should observation sampling records be added at the dosing times? Defaults to FALSE. amount.units: optional string indicating the dosing units. Defaults to NA to indicate as per the original EventTable definition. time.units: optional string indicating the time units. Defaults to "hours" to indicate as per the original EventTable definition.
get.dosing	returns a data.frame of dosing records.
clear.dosing	clears or deletes all dosing from event table
add.sampling	adds sampling time observation records to the event table. Its arguments are time a vector of time values (in time.units). time.units an optional string specifying the time units. Defaults to the units specified when the EventTable was initialized.
get.sampling	returns a data.frame of sampled observation records.
clear.sampling	removes all sampling from event table.
get.obs.rec	returns a logical vector indicating whether each event record represents an observation or not.
get.nobs	returns the number of observation (not dosing) records.
get.units	returns a two-element character vector with the dosing and time units, respectively.
copy	makes a copy of the current event table. To create a copy of an event table object use qd2 <-qd$copy().
expand	Expands the event table for multi-subject solving. This is done by qd$expand(400) for a 400 subject data expansion

Author(s)

Matthew Fidler, Melissa Hallow and Wenping Wang

See Also

et, RxODE

Examples

```
# create dosing and observation (sampling) events
# QD 50mg dosing, 5 days followed by 25mg 5 days
#
qd <- eventTable(amount.units = "mg", time.units = "days")
#
qd$add.dosing(dose=50, nbr.doses=5, dosing.interval = 1, do.sampling=FALSE)
#
# sample the system's drug amounts hourly the first day, then every 12 hours
# for the next 4 days
qd$add.sampling(seq(from = 0, to = 1, by = 1/24))
qd$add.sampling(seq(from = 1, to = 5, by = 12/24))
#
#print(qd$get.dosing())     # table of dosing records
print(qd$get.nobs())   # number of observation (not dosing) records
#
# BID dosing, 5 days
bid <- eventTable("mg", "days")  # only dosing
bid$add.dosing(dose=10000, nbr.doses=2*5,
               dosing.interval = 12, do.sampling=FALSE)
#
# Use the copy() method to create a copy (clone) of an existing
# event table (simple assignments just create a new reference to
# the same event table object (closure)).
#
bid.ext <- bid$copy()      # three-day extension for a 2nd cohort
bid.ext$add.dosing(dose = 5000, nbr.doses = 2*3,
                   start.time = 120, dosing.interval = 12, do.sampling = FALSE)

# You can also use the Piping operator to create a table

qd2 <- eventTable(amount.units="mg", time.units="days") %>%
    add.dosing(dose=50, nbr.doses=5, dosing.interval=1, do.sampling=FALSE) %>%
    add.sampling(seq(from=0, to=1, by=1 / 24)) %>%
    add.sampling(seq(from=1, to=5, by=12 / 24))
#print(qd2$get.dosing())     # table of dosing records
print(qd2$get.nobs())   # number of observation (not dosing) records

# Note that piping with %>% will update the original table.

qd3 <- qd2 %>% add.sampling(seq(from=5, to=10, by=6 / 24))
print(qd2$get.nobs())
print(qd3$get.nobs())
```

---

genShinyApp.template *Generate an example (template) of a dosing regimen shiny app*

Description

Create a complete shiny application for exploring dosing regimens given a (hardcoded) PK/PD model.

Usage

```
genShinyApp.template(appDir = "shinyExample", verbose = TRUE,
  ODE.config = list(ode = "model", params = c(KA = 0.294), inits = c(eff
  = 1), method = "lsoda", atol = 1e-08, rtol = 1e-06))

write.template.server(appDir)

write.template.ui(appDir, statevars)
```

Arguments

appDir	a string with a directory where to store the shiny app, by default is "shinyExample". The directory appDir will be created if it does not exist.
verbose	logical specifying whether to write messages as the shiny app is generated. Defaults to TRUE.
ODE.config	model name compiled and list of parameters sent to rxSolve.
statevars	List of statevars passed to to the write.template.ui function. This usually isn't called directly.

A PK/PD model is defined using RxODE, and a set of parameters and initial values are defined. Then the appropriate R scripts for the shiny's user interface ui.R and the server logic server.R are created in the directory appDir.

The function evaluates the following PK/PD model by default:

```
C2 = centr/V2;
C3 = peri/V3;
d/dt(depot) =-KA*depot;
d/dt(centr) = KA*depot - CL*C2 - Q*C2 + Q*C3;
d/dt(peri)  =                   Q*C2 - Q*C3;
d/dt(eff)  = Kin - Kout*(1-C2/(EC50+C2))*eff;
```

This can be changed by the ODE.config parameter.

To launch the shiny app, simply issue the runApp(appDir) R command.

Value

None, these functions are used for their side effects.

Note

These functions create a simple, but working example of a dosing regimen simulation web application. Users may want to modify the code to experiment creating shiny applications for their specific RxODE models.

See Also

RxODE,eventTable, and the package **shiny** (https://shiny.rstudio.com).

Examples

```
## Not run:
# create the shiny app example (template)
genShinyApp.template(appDir = "myapp")
# run the shiny app
runApp("myapp")

## End(Not run)
```

---

is.rxEt *Check to see if this is an rxEt object.*

Description

Check to see if this is an rxEt object.

Usage

```
is.rxEt(x)
```

Arguments

x	object to check to see if it is rxEt If this is an rxEt object that has expired strip all rxEt information.

Author(s)

Matthew L.Fidler

---

is.rxSolve *Check to see if this is an rxSolve object.*

Description

Check to see if this is an rxSolve object.

Usage

```
is.rxSolve(x)
```

Arguments

x	object to check to see if it is rxSolve If this is an rxSolve object that has expired strip all rxSolve information.

Author(s)

Matthew L.Fidler

---

print.rxCoefSolve *Print the rxCoefSolve object*

Description

This prints out the user supplied arguments for the rxCoef object

Usage

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

Arguments

x	rxCoefSolve object
...	Other (ignored) parameters.

Author(s)

Matthew L.Fidler

---

print.RxODE *Print information about the RxODE object.*

Description

This prints the model name and its status for being able to be solved

Usage

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

Arguments

x	An rxode object
...	Ignored parameters

Author(s)

Matthew L.Fidler

`rinvchisq` *Scaled Inverse Chi Squared distribution*

Description

Scaled Inverse Chi Squared distribution

Usage

```
rinvchisq(n = 1L, nu = 1, scale = 1)
```

Arguments

`n`	Number of random samples
`nu`	degrees of freedom of inverse chi square
`scale`	Scale of inverse chi squared distribution (default is 1).

Value

a vector of inverse chi squared deviates .

Examples

```
rinvchisq(3, 4, 1) ## Scale = 1, degrees of freedom = 4
rinvchisq(2, 4, 2) ## Scale = 2, degrees of freedom = 4
```

`rxAddReturn` *Add a return statement to a function.*

Description

Add a return statement to a function.

Usage

```
rxAddReturn(fn, ret = TRUE)
```

Arguments

`fn`	Function to deparse
`ret`	boolean stating if a return statement will be added.

Value

Function with parens removed and add a return statement.

Author(s)

Matthew L. Fidler

---

rxAllowUnload *Allow unloading of dlls*

Description

Allow unloading of dlls

Usage

```
rxAllowUnload(allow)
```

Arguments

allow	boolean indicating if garbage collection will unload of RxODE dlls.

Author(s)

Matthew Fidler

Examples

```
# Garbage collection will not unload un-used RxODE dlls
rxAllowUnload(FALSE);

# Garbage collection will unload unused RxODE dlls
rxAllowUnload(TRUE);
```

---

rxAssignPtr *Assign pointer based on model variables*

Description

Assign pointer based on model variables

Usage

```
rxAssignPtr(object = NULL)
```

Arguments

object	RxODE family of objects

rxC14 *Setup C++14 support in windows (required for nlmixr)*

Description

Setup C++14 support in windows (required for nlmixr)

Usage

```
rxC14()
```

Value

nothing

rxChain *rxChain Chain or add item to solved system of equations*

Description

Add item to solved system of equations

Usage

```
rxChain(obj1, obj2)

## S3 method for class 'solveRxDll'
obj1 + obj2
```

Arguments

obj1	Solved object.
obj2	New object to be added/piped/chained to solved object.

Value

When `newObject` is an event table, return a new solved object with the new event table.

Author(s)

Matthew L. Fidler

rxClean *Cleanup anonymous DLLs*

Description

This cleans up any DLLs created by text files

Usage

```
rxClean(wd)
```

Arguments

wd	What directory should be cleaned This cleans up all files named `rx-*.dll` and associated files as well as `call_dvode.o` and associated files

Value

TRUE if successful

Author(s)

Matthew L. Fidler

rxCompile *Compile a model if needed*

Description

This is the compilation workhorse creating the RxODE model DLL files.

Usage

```
rxCompile(model, dir, prefix, extraC = NULL, force = FALSE,
  modName = NULL, package = NULL, ...)

## S3 method for class 'character'
rxCompile(model, dir = NULL, prefix = NULL,
  extraC = NULL, force = FALSE, modName = NULL, package = NULL,
  ...)

## S3 method for class 'rxDll'
rxCompile(model, ...)

## S3 method for class 'RxODE'
rxCompile(model, ...)
```

Arguments

model	This is the ODE model specification. It can be: • a string containing the set of ordinary differential equations (ODE) and other expressions defining the changes in the dynamic system. • a file name where the ODE system equation is contained • An ODE expression enclosed in {} (see also the filename argument). For details, see the sections "Details" and "RxODE Syntax" below.
dir	This is the model directory where the C file will be stored for compiling. If unspecified, the C code is stored in a temporary directory, then the model is compiled and moved to the current directory. Afterwards the C code is removed. If specified, the C code is stored in the specified directory and then compiled in that directory. The C code is not removed after the DLL is created in the same directory. This can be useful to debug the c-code outputs.
prefix	is a string indicating the prefix to use in the C based functions. If missing, it is calculated based on file name, or md5 of parsed model.
extraC	Extra c code to include in the model. This can be useful to specify functions in the model. These C functions should usually take double precision arguments, and return double precision values.
force	is a boolean stating if the (re)compile should be forced if RxODE detects that the models are the same as already generated.
modName	a string to be used as the model name. This string is used for naming various aspects of the computations, including generating C symbol names, dynamic libraries, etc. Therefore, it is necessary that modName consists of simple ASCII alphanumeric characters starting with a letter.
package	Package name for pre-compiled binaries.
...	Other arguments sent to the rxTrans function.

Value

An rxDll object that has the following components

dll	DLL path
model	model specification
.c	A function to call C code in the correct context from the DLL using the .C function.
.call	A function to call C code in the correct context from the DLL using the .Call function.
args	A list of the arguments used to create the rxDll object.

Author(s)

Matthew L.Fidler

See Also

RxODE

---

rxControl *Solving \& Simulation of a ODE/solved system (and solving options) equation*

Description

This uses RxODE family of objects, file, or model specification to solve a ODE system.

Usage

```
rxControl(scale = NULL, method = c("liblsoda", "lsoda", "dop853"),
  transitAbs = NULL, atol = 1e-08, rtol = 1e-06, maxsteps = 70000L,
  hmin = 0L, hmax = NA, hmaxSd = 0, hini = 0, maxordn = 12L,
  maxords = 5L, ..., cores, covsInterpolation = c("locf", "linear",
  "nocb", "midpoint"), addCov = FALSE, matrix = FALSE, sigma = NULL,
  sigmaDf = NULL, sigmaLower = -Inf, sigmaUpper = Inf,
  nCoresRV = 1L, sigmaIsChol = FALSE, nDisplayProgress = 10000L,
  amountUnits = NA_character_, timeUnits = "hours", stiff,
  theta = NULL, thetaLower = -Inf, thetaUpper = Inf, eta = NULL,
  addDosing = FALSE, stateTrim = Inf, updateObject = FALSE,
  omega = NULL, omegaDf = NULL, omegaIsChol = FALSE,
  omegaLower = -Inf, omegaUpper = Inf, nSub = 1L, thetaMat = NULL,
  thetaDf = NULL, thetaIsChol = FALSE, nStud = 1L, dfSub = 0,
  dfObs = 0, returnType = c("rxSolve", "matrix", "data.frame",
  "data.frame.TBS"), seed = NULL, nsim = NULL, minSS = 10L,
  maxSS = 1000L, infSSstep = 12, strictSS = TRUE, params = NULL,
  events = NULL, istateReset = TRUE, subsetNonmem = TRUE,
  linLog = FALSE, maxAtolRtolFactor = 0.1, from = NULL, to = NULL,
  by = NULL, length.out = NULL, iCov = NULL, keep = NULL,
  idFactor = TRUE, mxhnil = 0, hmxi = 0, warnIdSort = TRUE,
  ssAtol = 1e-08, ssRtol = 1e-06)

rxSolve(object, ...)

## Default S3 method:
rxSolve(object, params = NULL, events = NULL,
  inits = NULL, ...)

## S3 method for class 'rxSolve'
update(object, ...)

## S3 method for class 'RxODE'
predict(object, ...)
```

```
## S3 method for class 'rxSolve'
predict(object, ...)

## S3 method for class 'rxEt'
predict(object, ...)

## S3 method for class 'rxParams'
predict(object, ...)

## S3 method for class 'RxODE'
simulate(object, nsim = 1L, seed = NULL, ...)

## S3 method for class 'rxSolve'
simulate(object, nsim = 1L, seed = NULL, ...)

## S3 method for class 'rxParams'
simulate(object, nsim = 1L, seed = NULL, ...)

## S3 method for class 'rxSolve'
solve(a, b, ...)

## S3 method for class 'RxODE'
solve(a, b, ...)

## S3 method for class 'rxParams'
solve(a, b, ...)

## S3 method for class 'rxEt'
solve(a, b, ...)
```

Arguments

`scale`	a numeric named vector with scaling for ode parameters of the system. The names must correspond to the parameter identifiers in the ODE specification. Each of the ODE variables will be divided by the scaling factor. For example `scale=c(center=2)` will divide the center ODE variable by 2.
`method`	The method for solving ODEs. Currently this supports: • `"liblsoda"` thread safe lsoda. This supports parallel thread-based solving, and ignores user Jacobian specification. • `"lsoda"` – LSODA solver. Does not support parallel thread-based solving, but allows user Jacobian specification. • `"dop853"` – DOP853 solver. Does not support parallel thread-based solving nor user Jacobain specification
`transitAbs`	boolean indicating if this is a transit compartment absorption
`atol`	a numeric absolute tolerance (1e-8 by default) used by the ODE solver to determine if a good solution has been achieved; This is also used in the solved linear model to check if prior doses do not add anything to the solution.

`rtol` a numeric relative tolerance (1e-6 by default) used by the ODE solver to determine if a good solution has been achieved. This is also used in the solved linear model to check if prior doses do not add anything to the solution.

`maxsteps` maximum number of (internally defined) steps allowed during one call to the solver. (5000 by default)

`hmin` The minimum absolute step size allowed. The default value is 0.

`hmax` The maximum absolute step size allowed. When `hmax=NA` (default), uses the average difference (+hmaxSd*sd) in times and sampling events. When `hmax=NULL` RxODE uses the maximum difference in times in your sampling and events. The value 0 is equivalent to infinite maximum absolute step size.

`hmaxSd` The number of standard deviations of the time difference to add to hmax. The default is 0

`hini` The step size to be attempted on the first step. The default value is determined by the solver (when hini = 0)

`maxordn` The maximum order to be allowed for the nonstiff (Adams) method. The default is 12. It can be between 1 and 12.

`maxords` The maximum order to be allowed for the stiff (BDF) method. The default value is 5. This can be between 1 and 5.

`...` Other arguments including scaling factors for each compartment. This includes S# = numeric will scale a compartment # by a dividing the compartment amount by the scale factor, like NONMEM.

`cores` Number of cores used in parallel ODE solving. This defaults to the number or system cores determined by `rxCores` for methods that support parallel solving (ie thread-safe methods like "liblsoda").

`covsInterpolation`
specifies the interpolation method for time-varying covariates. When solving ODEs it often samples times outside the sampling time specified in `events`. When this happens, the time varying covariates are interpolated. Currently this can be:

- `"linear"` interpolation (the default), which interpolates the covariate by solving the line between the observed covariates and extrapolating the new covariate value.
- `"constant"` – Last observation carried forward.
- `"NOCB"` – Next Observation Carried Backward. This is the same method that NONMEM uses.
- `"midpoint"` Last observation carried forward to midpoint; Next observation carried backward to midpoint.

`addCov` A boolean indicating if covariates should be added to the output matrix or data frame. By default this is disabled.

`matrix` A boolean indicating if a matrix should be returned instead of the RxODE's solved object.

`sigma` Named sigma covariance or Cholesky decomposition of a covariance matrix. The names of the columns indicate parameters that are simulated. These are simulated for every observation in the solved system.

`sigmaDf` Degrees of freedom of the sigma t-distribution. By default it is equivalent to `Inf`, or a normal distribution.

`sigmaLower` Lower bounds for simulated unexplained variability (by default -Inf)

`sigmaUpper` Upper bounds for simulated unexplained variability (by default Inf)

`nCoresRV` Number of cores used for the simulation of the sigma variables. By default this is 1. This uses the package `rmvn` and `rmvt`. To reproduce the results you need to run on the same platform with the same number of cores. This is the reason this is set to be one, regardless of what the number of cores are used in threaded ODE solving.

`sigmaIsChol` Boolean indicating if the sigma is in the Cholesky decomposition instead of a symmetric covariance

`nDisplayProgress`
An integer indicating the minimum number of c-based solves before a progress bar is shown. By default this is 10,000.

`amountUnits` This supplies the dose units of a data frame supplied instead of an event table. This is for importing the data as an RxODE event table.

`timeUnits` This supplies the time units of a data frame supplied instead of an event table. This is for importing the data as an RxODE event table.

`stiff` a logical (`TRUE` by default) indicating whether the ODE system is stiff or not.

For stiff ODE systems (`stiff = TRUE`), `RxODE` uses the LSODA (Livermore Solver for Ordinary Differential Equations) Fortran package, which implements an automatic method switching for stiff and non-stiff problems along the integration interval, authored by Hindmarsh and Petzold (2003).

For non-stiff systems (`stiff = FALSE`), `RxODE` uses DOP853, an explicit Runge-Kutta method of order 8(5, 3) of Dormand and Prince as implemented in C by Hairer and Wanner (1993).

`theta` A vector of parameters that will be named THETA[#] and added to parameters

`thetaLower` Lower bounds for simulated population parameter variability (by default -Inf)

`thetaUpper` Upper bounds for simulated population unexplained variability (by default Inf)

`eta` A vector of parameters that will be named ETA[#] and added to parameters

`addDosing` Boolean indicating if the solve should add RxODE EVID and related columns. This will also include dosing information and estimates at the doses. Be default, RxODE only includes estimates at the observations. (default `FALSE`). When `addDosing` is `NULL`, only include `EVID=0` on solve and exclude any model-times or `EVID=2`. If `addDosing` is `NA` the classic `RxODE` EVID events. When `addDosing` is `TRUE` add the event information in NONMEM-style format; If `subsetNonmem=FALSE` RxODE will also extra event types (`EVID`) for ending infusion and modeled times:

- `EVID=-1` when the modeled rate infusions are turned off (matches `rate=-1`)
- `EVID=-2` When the modeled duration infusions are turned off (matches `rate=-2`)
- `EVID=-10` When the specified `rate` infusions are turned off (matches `rate>0`)
- `EVID=-20` When the specified `dur` infusions are turned off (matches `dur>0`)

- `EVID=101,102,103,...` Modeled time where 101 is the first model time, 102 is the second etc.

`stateTrim` When amounts/concentrations in one of the states are above this value, trim them to be this value. By default Inf. Also trims to -stateTrim for large negative amounts/concentrations

`updateObject` This is an internally used flag to update the RxODE solved object (when supplying an RxODE solved object) as well as returning a new object. You probably should not modify it's `FALSE` default unless you are willing to have unexpected results.

`omega` Estimate of Covariance matrix. When omega is a list, assume it is a block matrix and convert it to a full matrix for simulations.

`omegaDf` The degrees of freedom of a t-distribution for simulation. By default this is `NULL` which is equivalent to `Inf` degrees, or to simulate from a normal distribution instead of a t-distribution.

`omegaIsChol` Indicates if the `omega` supplied is a Cholesky decomposed matrix instead of the traditional symmetric matrix.

`omegaLower` Lower bounds for simulated ETAs (by default -Inf)

`omegaUpper` Upper bounds for simulated ETAs (by default Inf)

`nSub` Number between subject variabilities (ETAs) simulated for every realization of the parameters.

`thetaMat` Named theta matrix.

`thetaDf` The degrees of freedom of a t-distribution for simulation. By default this is `NULL` which is equivalent to `Inf` degrees, or to simulate from a normal distribution instead of a t-distribution.

`thetaIsChol` Indicates if the `theta` supplied is a Cholesky decomposed matrix instead of the traditional symmetric matrix.

`nStud` Number virtual studies to characterize uncertainty in estimated parameters.

`dfSub` Degrees of freedom to sample the between subject variability matrix from the inverse Wishart distribution (scaled) or scaled inverse chi squared distribution.

`dfObs` Degrees of freedom to sample the unexplained variability matrix from the inverse Wishart distribution (scaled) or scaled inverse chi squared distribution.

`returnType` This tells what type of object is returned. The currently supported types are:

- `"rxSolve"` (default) will return a reactive data frame that can change easily change different pieces of the solve and update the data frame. This is the currently standard solving method in RxODE, is used for `rxSolve(object,...)`, `solve(object,...)`,
- `"data.frame"` – returns a plain, non-reactive data frame; Currently very slightly Faster than `returnType="matrix"`
- `"matrix"` – returns a plain matrix with column names attached to the solved object. This is what is used `object$run` as well as `object$solve`

`seed` an object specifying if and how the random number generator should be initialized

`nsim` represents the number of simulations. For RxODE, if you supply single subject event tables (created with eventTable)

`minSS` Minimum number of iterations for a steady-state dose

`maxSS` Maximum number of iterations for a steady-state dose

`infSSstep` Step size for determining if a constant infusion has reached steady state. By default this is large value, 420.

`strictSS` Boolean indicating if a strict steady-state is required. If a strict steady-state is (`TRUE`) required then at least `minSS` doses are administered and the total number of steady states doses will continue until `maxSS` is reached, or `atol` and `rtol` for every compartment have been reached. However, if ODE solving problems occur after the `minSS` has been reached the whole subject is considered an invalid solve. If `strictSS` is `FALSE` then as long as `minSS` has been reached the last good solve before ODE solving problems occur is considered the steady state, even though either `atol`, `rtol` or `maxSS` have not been achieved.

`params` a numeric named vector with values for every parameter in the ODE system; the names must correspond to the parameter identifiers used in the ODE specification;

`events` an `eventTable` object describing the input (e.g., doses) to the dynamic system and observation sampling time points (see `eventTable`);

`istateReset` When `TRUE`, reset the `ISTATE` variable to 1 for lsoda and liblsoda with doses, like `deSolve`; When `FALSE`, do not reset the `ISTATE` variable with doses.

`subsetNonmem` subset to NONMEM compatible EVIDs only. By default TRUE.

`linLog` Boolean indicating if linear compartment models be calculated more accurately in the log-space (slower) By default this is off (`FALSE`)

`maxAtolRtolFactor`
The maximum `atol`/`rtol` that FOCEi and other routines may adjust to. By default 0.1

`from` When there is no observations in the event table, start observations at this value. By default this is zero.

`to` When there is no observations in the event table, end observations at this value. By default this is 24 + maximum dose time.

`by` When there are no observations in the event table, this is the amount to increment for the observations between `from` and `to`.

`length.out` The number of observations to create if there isn't any observations in the event table. By default this is 200.

`iCov` A data frame of individual non-time varying covariates to combine with the `params` to form a parameter data.frame.

`keep` Columns to keep from either the input dataset or the `iCov` dataset. With the `iCov` dataset, the column is kept once per line. For the input dataset, if any records are added to the data LOCF (Last Observation Carried forward) imputation is performed.

`idFactor` This boolean indicates if original ID values should be maintained. This changes the default sequentially ordered ID to a factor with the original ID values in the original dataset. By default this is enabled.

mxhnil	maximum number of messages printed (per problem) warning that T + H = T on a step (H = step size). This must be positive to result in a non-default value. The default value is 0 (or infinite).
hmxi	inverse of the maximum absolute value of H to be used. hmxi = 0.0 is allowed and corresponds to an infinite hmax (default). hmin and hmxi may be changed at any time, but will not take effect until the next change of H is considered. This option is only considered with method=liblsoda.
warnIdSort	Warn if the ID is not present and RxODE assumes the order of the parameters/iCov are the same as the order of the parameters in the input dataset.
ssAtol	Steady state atol convergence factor. Can be a vector based on each state.
ssRtol	Steady state rtol convergence factor. Can be a vector based on each state.
object	is a either a RxODE family of objects, or a file-name with a RxODE model specification, or a string with a RxODE model specification.
inits	a vector of initial values of the state variables (e.g., amounts in each compartment), and the order in this vector must be the same as the state variables (e.g., PK/PD compartments);
a	when using solve, this is equivalent to the object argument. If you specify object later in the argument list it overwrites this parameter.
b	when using solve, this is equivalent to the params argument. If you specify params as a named argument, this overwrites the output

Value

An "rxSolve" solve object that stores the solved value in a matrix with as many rows as there are sampled time points and as many columns as system variables (as defined by the ODEs and additional assignments in the RxODE model code). It also stores information about the call to allow dynamic updating of the solved object.

The operations for the object are similar to a data-frame, but expand the `$` and `[["",]]` access operators and assignment operators to resolve based on different parameter values, initial conditions, solver parameters, or events (by updating the `time` variable).

You can call the `eventTable` methods on the solved object to update the event table and resolve the system of equations.

Author(s)

Matthew Fidler, Melissa Hallow and Wenping Wang

References

Hindmarsh, A. C. *ODEPACK, A Systematized Collection of ODE Solvers*. Scientific Computing, R. S. Stepleman et al. (Eds.), North-Holland, Amsterdam, 1983, pp. 55-64.

Petzold, L. R. *Automatic Selection of Methods for Solving Stiff and Nonstiff Systems of Ordinary Differential Equations*. Siam J. Sci. Stat. Comput. 4 (1983), pp. 136-148.

Hairer, E., Norsett, S. P., and Wanner, G. *Solving ordinary differential equations I, nonstiff problems*. 2nd edition, Springer Series in Computational Mathematics, Springer-Verlag (1993).

See Also

RxODE

`rxCores` *Get the number of cores in a system*

Description

Get the number of cores in a system

Usage

```
rxCores()
```

`rxDelete` *Delete the DLL for the model*

Description

This function deletes the DLL, but doesn't delete the model information in the object.

Usage

```
rxDelete(obj)
```

Arguments

`obj`	RxODE family of objects

Value

A boolean stating if the operation was successful.

Author(s)

Matthew L.Fidler

rxDfdy *Jacobian and parameter derivatives*

Description

Return Jacobain and parameter derivatives

Usage

```
rxDfdy(obj)
```

Arguments

obj	RxODE family of objects

Value

A list of the jacobian parameters defined in this RxODE object.

Author(s)

Matthew L. Fidler

rxEvid *EVID formatting for tibble and other places.*

Description

This is to make an EVID more readable by non pharmacometricians. It displays what each means and allows it to be displayed in a tibble.

Usage

```
rxEvid(x)

as.rxEvid(x)

## S3 method for class 'rxEvid'
c(x, ...)

## S3 method for class 'rxEvid'
x[...]

## S3 method for class 'rxEvid'
as.character(x, ...)
```

```
## S3 method for class 'rxEvid'
format(x, ...)

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

## S3 method for class 'rxEvid'
x[[...]]

## S3 method for class 'rxEvid'
type_sum(x)

pillar_shaft.rxEvid(x, ...)

## S3 method for class 'rxRateDur'
c(x, ...)

## S3 method for class 'rxRateDur'
format(x, ...)
```

Arguments

x	Item to be converted to a RxODE EVID specification.
...	Other parameters

Examples

```
rxEvid(1:7)
```

`rxFoExpandEta` *First Order Expansion of ETA*

Description

First Order Expansion of ETA

Usage

```
rxFoExpandEta(expr)
```

Arguments

expr	RxODE model

Value

Return a RxODE model with first order Taylor expansion around ETA

Author(s)

Matthew L. Fidler

`rxGetRxODE` *Get RxODE model from object*

Description

Get RxODE model from object

Usage

```
rxGetRxODE(obj)
```

Arguments

`obj`	RxODE family of objects

`rxHtml` *Format rxSolve and related objects as html.*

Description

Format rxSolve and related objects as html.

Usage

```
rxHtml(x, ...)

## S3 method for class 'rxSolve'
rxHtml(x, ...)
```

Arguments

`x`	RxODE object
`...`	Extra arguments sent to kable

Author(s)

Matthew L. Fidler

rxInv *Invert matrix using RcppArmadillo.*

Description

Invert matrix using RcppArmadillo.

Usage

```
rxInv(matrix)
```

Arguments

matrix	matrix to be inverted.

Value

inverse or pseudo inverse of matrix.

rxIsCurrent *Checks if the RxODE object was built with the current build*

Description

Checks if the RxODE object was built with the current build

Usage

```
rxIsCurrent(obj)
```

Arguments

obj	RxODE family of objects

Value

boolean indicating if this was built with current RxODE

rxLhs *Left handed Variables*

Description

This returns the model calculated variables

Usage

```
rxLhs(obj)
```

Arguments

obj	RxODE family of objects

Value

a character vector listing the calculated parameters

Author(s)

Matthew L.Fidler

See Also

RxODE

rxLock *Lock/unlocking of RxODE dll file*

Description

Lock/unlocking of RxODE dll file

Usage

```
rxLock(obj)

rxUnlock(obj)
```

Arguments

obj	A RxODE family of objects

rxNorm *Get the normalized model*

Description

This get the syntax preferred model for processing

Usage

```
rxNorm(obj, condition = NULL, removeInis, removeJac, removeSens)
```

Arguments

obj	RxODE family of objects
condition	Character string of a logical condition to use for subsetting the normalized model. When missing, and a condition is not set via rxCondition, return the whole code with all the conditional settings intact. When a condition is set with rxCondition, use that condition.
removeInis	A boolean indicating if parameter initialization will be removed from the model
removeJac	A boolean indicating if the Jacobians will be removed.
removeSens	A boolean indicating if the sensitivities will be removed.

Value

Normalized Normal syntax (no comments)

Author(s)

Matthew L. Fidler

RxODE *Create an ODE-based model specification*

Description

Create a dynamic ODE-based model object suitably for translation into fast C code

Usage

```
RxODE(model, modName = basename(wd), wd = getwd(), filename = NULL,
  extraC = NULL, debug = FALSE, calcJac = NULL, calcSens = NULL,
  collapseModel = FALSE, package = NULL, ...)
```

Arguments

model	This is the ODE model specification. It can be: • a string containing the set of ordinary differential equations (ODE) and other expressions defining the changes in the dynamic system. • a file name where the ODE system equation is contained • An ODE expression enclosed in {} (see also the filename argument). For details, see the sections "Details" and "RxODE Syntax" below.
modName	a string to be used as the model name. This string is used for naming various aspects of the computations, including generating C symbol names, dynamic libraries, etc. Therefore, it is necessary that modName consists of simple ASCII alphanumeric characters starting with a letter.
wd	character string with a working directory where to create a subdirectory according to modName. When specified, a subdirectory named after the "modName.d" will be created and populated with a C file, a dynamic loading library, plus various other working files. If missing, the files are created (and removed) in the temporary directory, and the RxODE DLL for the model is created in the current directory named rx_????_platform, for example rx_129f8f97fb94a87ca49ca8dafe691e1e_i386.dll
filename	A file name or connection object where the ODE-based model specification resides. Only one of model or filename may be specified.
extraC	Extra c code to include in the model. This can be useful to specify functions in the model. These C functions should usually take double precision arguments, and return double precision values.
debug	is a boolean indicating if the executable should be compiled with verbose debugging information turned on.
calcJac	boolean indicating if RxODE will calculate the Jacobain according to the specified ODEs.
calcSens	boolean indicating if RxODE will calculate the sensitivities according to the specified ODEs.
collapseModel	boolean indicating if RxODE will remove all LHS variables when calculating sensitivities.
package	Package name for pre-compiled binaries.
...	ignored arguments. The "Rx" in the name RxODE is meant to suggest the abbreviation *Rx* for a medical prescription, and thus to suggest the package emphasis on pharmacometrics modeling, including pharmacokinetics (PK), pharmacodynamics (PD), disease progression, drug-disease modeling, etc. The ODE-based model specification may be coded inside a character string or in a text file, see Section *RxODE Syntax* below for coding details. An internal RxODE compilation manager object translates the ODE system into C, compiles it, and dynamically loads the object code into the current R session. The call to RxODE produces an object of class RxODE which consists of a list-like structure (closure) with various member functions (see Section *Value* below).

For evaluating RxODE models, two types of inputs may be provided: a required set of time points for querying the state of the ODE system and an optional set of doses (input amounts). These inputs are combined into a single *event table* object created with the function eventTable.

Value

An object (closure) of class "RxODE" (see Chambers and Temple Lang (2001)) consisting of the following list of strings and functions:

modName	the name of the model (a copy of the input argument).
model	a character string holding the source model specification.
get.modelVars	a function that returns a list with 3 character vectors, params, state, and lhs of variable names used in the model specification. These will be output when the model is computed (i.e., the ODE solved by integration).
solve	this function solves (integrates) the ODE. This is done by passing the code to rxSolve. This is as if you called rxSolve(RxODEobject,...), but returns a matrix instead of a rxSolve object. params: a numeric named vector with values for every parameter in the ODE system; the names must correspond to the parameter identifiers used in the ODE specification; events: an eventTable object describing the input (e.g., doses) to the dynamic system and observation sampling time points (see eventTable); inits: a vector of initial values of the state variables (e.g., amounts in each compartment), and the order in this vector must be the same as the state variables (e.g., PK/PD compartments); stiff: a logical (TRUE by default) indicating whether the ODE system is stiff or not. For stiff ODE systems (stiff = TRUE), RxODE uses the LSODA (Livermore Solver for Ordinary Differential Equations) Fortran package, which implements an automatic method switching for stiff and non-stiff problems along the integration interval, authored by Hindmarsh and Petzold (2003). For non-stiff systems (stiff = FALSE), RxODE uses DOP853, an explicit Runge-Kutta method of order 8(5, 3) of Dormand and Prince as implemented in C by Hairer and Wanner (1993). trans_abs: a logical (FALSE by default) indicating whether to fit a transit absorption term (TODO: need further documentation and example); atol: a numeric absolute tolerance (1e-08 by default); rtol: a numeric relative tolerance (1e-06 by default).e The output of "solve" is a matrix with as many rows as there are sampled time points and as many columns as system variables (as defined by the ODEs and additional assignments in the RxODE model code).
isValid	a function that (naively) checks for model validity, namely that the C object code reflects the latest model specification.
version	a string with the version of the RxODE object (not the package).

dynLoad	a function with one force = FALSE argument that dynamically loads the object code if needed.
dynUnload	a function with no argument that unloads the model object code.
delete	removes all created model files, including C and DLL files. The model object is no longer valid and should be removed, e.g., rm(m1).
run	deprecated, use solve.
parse	deprecated.
compile	deprecated.
get.index	deprecated.
getObj	internal (not user callable) function.

RxODE Syntax

An RxODE model specification consists of one or more statements terminated by semi-colons, ';', and optional comments (comments are delimited by # and an end-of-line marker). **NB:** Comments are not allowed inside statements.

A block of statements is a set of statements delimited by curly braces, '{ ... }'. Statements can be either assignments or conditional if statements. Assignment statements can be: (1) "simple" assignments, where the left hand is an identifier (i.e., variable), (2) special "time-derivative" assignments, where the left hand specifies the change of that variable with respect to time e.g., d/dt(depot), or (3) special "jacobian" assignments, where the left hand specifies the change of of the ODE with respect to one of the parameters, e.g. df(depot)/dy(kel). The "jacobian" assignments are not required, and are only useful for very stiff differential systems.

Expressions in assignment and 'if' statements can be numeric or logical (no character expressions are currently supported). Numeric expressions can include the following numeric operators ('+', '-', '*', '/', '^'), and those mathematical functions defined in the C or the R math libraries (e.g., fabs, exp, log, sin). (Notice that the modulo operator '%' is currently not supported.)

Identifiers in an RxODE model specification can refer to:

- state variables in the dynamic system (e.g., compartments in a pharmacokinetics/pharamcodynamics model);
- implied input variable, t (time), podo (oral dose, for absorption models), and tlast (last time point);
- model parameters, (ka rate of absorption, CL clearance, etc.);
- pi, for the constant pi.
- others, as created by assignments as part of the model specification.

Identifiers consists of case-sensitive alphanumeric characters, plus the underscore '_' character. **NB:** the dot '.' character is **not** a valid character identifier.

The values of these variables at pre-specified time points are saved as part of the fitted/integrated/solved model (see eventTable, in particular its member function add.sampling that defines a set of time points at which to capture a snapshot of the system via the values of these variables).

The ODE specification mini-language is parsed with the help of the open source tool dparser, Plevyak (2015).

Author(s)

Melissa Hallow, Wenping Wang and Matthew Fidler

References

Chamber, J. M. and Temple Lang, D. (2001) *Object Oriented Programming in R*. R News, Vol. 1, No. 3, September 2001. https://cran.r-project.org/doc/Rnews/Rnews_2001-3.pdf.

Hindmarsh, A. C. *ODEPACK, A Systematized Collection of ODE Solvers*. Scientific Computing, R. S. Stepleman et al. (Eds.), North-Holland, Amsterdam, 1983, pp. 55-64.

Petzold, L. R. *Automatic Selection of Methods for Solving Stiff and Nonstiff Systems of Ordinary Differential Equations*. Siam J. Sci. Stat. Comput. 4 (1983), pp. 136-148.

Hairer, E., Norsett, S. P., and Wanner, G. *Solving ordinary differential equations I, nonstiff problems*. 2nd edition, Springer Series in Computational Mathematics, Springer-Verlag (1993).

Plevyak, J. dparser, http://dparser.sourceforge.net. Web. 12 Oct. 2015.

See Also

eventTable, et, add.sampling, add.dosing

Examples

```
# Step 1 - Create a model specification
ode <- "
   # A 4-compartment model, 3 PK and a PD (effect) compartment
   # (notice state variable names 'depot', 'centr', 'peri', 'eff')

   C2 = centr/V2;
   C3 = peri/V3;
   d/dt(depot) =-KA*depot;
   d/dt(centr) = KA*depot - CL*C2 - Q*C2 + Q*C3;
   d/dt(peri)  =                    Q*C2 - Q*C3;
   d/dt(eff)  = Kin - Kout*(1-C2/(EC50+C2))*eff;
"

m1 <- RxODE(model = ode)
print(m1)

# Step 2 - Create the model input as an EventTable,
# including dosing and observation (sampling) events

# QD (once daily) dosing for 5 days.

qd <- eventTable(amount.units = "ug", time.units = "hours")
qd$add.dosing(dose = 10000, nbr.doses = 5, dosing.interval = 24)

# Sample the system hourly during the first day, every 8 hours
# then after

qd$add.sampling(0:24)
qd$add.sampling(seq(from = 24+8, to = 5*24, by = 8))
```

```
# Step 3 - set starting parameter estimates and initial
# values of the state

theta <-
    c(KA = .291, CL = 18.6,
      V2 = 40.2, Q = 10.5, V3 = 297.0,
      Kin = 1.0, Kout = 1.0, EC50 = 200.0)

# init state variable
inits <- c(0, 0, 0, 1);

# Step 4 - Fit the model to the data

qd.cp <- m1$solve(theta, events = qd, inits)

head(qd.cp)

# This returns a matrix.  Note that you can also
# solve using name initial values. For example:

inits <- c(eff = 1);

qd.cp <- solve(m1, theta, events = qd, inits);

print(qd.cp)

plot(qd.cp)
```

`rxOptExpr` *Optimize RxODE for computer evaluation*

Description

This optimizes RxODE code for computer evaluation by only calculating redundant expressions once.

Usage

```
rxOptExpr(x)
```

Arguments

`x`	RxODE model that can be access by rxNorm

Value

Optimized RxODE model text. The order and type lhs and state variables is maintained while the evaluation is sped up. While parameters names are maintained, their order may be modified.

Author(s)

Matthew L. Fidler

`rxOptions` *Options for RxODE*

Description

This is a backend for `rxPermissive` (with `op.rx = 2`) and `rxStrict` (with `op.rx =1`)

Usage

```
rxOptions(expr, op.rx = NULL, silent = .isTestthat(),
  respect = FALSE, cran = FALSE, on.validate = FALSE)
```

Arguments

`expr`	Expression to evaluate in the permissive/strict environment. If unspecified, set the options for the current environment.
`op.rx`	A numeric for strict (1) or permissive (2) syntax.
`silent`	when true, also silence the syntax errors and interactive output (useful in testing).
`respect`	when TRUE, respect any options that are specified. This is called at startup, but really should not be called elsewhere, otherwise the options are not changed.
`cran`	When specified and true, run on CRAN. Otherwise it is skipped on CRAN.
`on.validate`	When TRUE run only when validating.

Details

When `expr` is missing and `op.rx` is NULL, this displays the current RxODE options.

Author(s)

Matthew L. Fidler

rxParams *Parameters specified by the model*

Description

This returns the model's parameters that are required to solve the ODE system, and can be used to pipe parameters into an RxODE solve

Usage

```
rxParams(obj, ...)

## S3 method for class 'RxODE'
rxParams(obj, constants = TRUE, ..., params = NULL,
  inits = NULL, iCov = NULL, keep = NULL, thetaMat = NULL,
  omega = NULL, dfSub = NULL, sigma = NULL, dfObs = NULL,
  nSub = NULL, nStud = NULL)

## S3 method for class 'rxSolve'
rxParams(obj, constants = TRUE, ..., params = NULL,
  inits = NULL, iCov = NULL, keep = NULL, thetaMat = NULL,
  omega = NULL, dfSub = NULL, sigma = NULL, dfObs = NULL,
  nSub = NULL, nStud = NULL)

## S3 method for class 'rxEt'
rxParams(obj, ..., params = NULL, inits = NULL,
  iCov = NULL, keep = NULL, thetaMat = NULL, omega = NULL,
  dfSub = NULL, sigma = NULL, dfObs = NULL, nSub = NULL,
  nStud = NULL)

rxParam(obj, ...)
```

Arguments

obj	RxODE family of objects
...	Other arguments including scaling factors for each compartment. This includes S# = numeric will scale a compartment # by a dividing the compartment amount by the scale factor, like NONMEM.
constants	is a boolean indicting if constants should be included in the list of parameters. Currently RxODE parses constants into variables in case you wish to change them without recompiling the RxODE model.
params	a numeric named vector with values for every parameter in the ODE system; the names must correspond to the parameter identifiers used in the ODE specification;

inits	a vector of initial values of the state variables (e.g., amounts in each compartment), and the order in this vector must be the same as the state variables (e.g., PK/PD compartments);
iCov	A data frame of individual non-time varying covariates to combine with the params to form a parameter data.frame.
keep	Columns to keep from either the input dataset or the iCov dataset. With the iCov dataset, the column is kept once per line. For the input dataset, if any records are added to the data LOCF (Last Observation Carried forward) imputation is performed.
thetaMat	Named theta matrix.
omega	Estimate of Covariance matrix. When omega is a list, assume it is a block matrix and convert it to a full matrix for simulations.
dfSub	Degrees of freedom to sample the between subject variability matrix from the inverse Wishart distribution (scaled) or scaled inverse chi squared distribution.
sigma	Named sigma covariance or Cholesky decomposition of a covariance matrix. The names of the columns indicate parameters that are simulated. These are simulated for every observation in the solved system.
dfObs	Degrees of freedom to sample the unexplained variability matrix from the inverse Wishart distribution (scaled) or scaled inverse chi squared distribution.
nSub	Number between subject variabilities (ETAs) simulated for every realization of the parameters.
nStud	Number virtual studies to characterize uncertainty in estimated parameters.

Value

When extracting the parameters from an RxODE model, a character vector listing the parameters in the model.

Author(s)

Matthew L.Fidler

`rxPermissive` *Permissive or Strict RxODE syntax options*

Description

This sets the RxODE syntax to be permissive or strict

Usage

```
rxPermissive(expr, silent = .isTestthat(), respect = FALSE,
  cran = FALSE, on.validate = FALSE)

rxStrict(expr, silent = .isTestthat(), respect = FALSE, cran = FALSE,
  on.validate = FALSE)
```

Arguments

expr	Expression to evaluate in the permissive/strict environment. If unspecified, set the options for the current environment.
silent	when true, also silence the syntax errors and interactive output (useful in testing).
respect	when TRUE, respect any options that are specified. This is called at startup, but really should not be called elsewhere, otherwise the options are not changed.
cran	When specified and true, run on CRAN. Otherwise it is skipped on CRAN.
on.validate	When TRUE run only when validating.

Author(s)

Matthew L. Fidler

---

rxProgress *RxODE progress bar functions*

Description

rxProgress sets up the progress bar

Usage

```
rxProgress(num, core = 0L)

rxTick()

rxProgressStop(clear = TRUE)

rxProgressAbort(error = "Aborted calculation")
```

Arguments

num	Tot number of operations to track
core	Number of cores to show. If below 1, don't show number of cores
clear	Boolean telling if you should clear the progress bar after completion (as if it wasn't displayed). By default this is TRUE
error	With rxProgressAbort this is the error that is displayed

Details

rxTick is a progress bar tick

rxProgressStop stop progress bar

rxProgressAbort shows an abort if rxProgressStop wasn't called.

Value

All return NULL invisibly.

Author(s)

Matthew L. Fidler

Examples

```
f <- function(){
  on.exit({rxProgressAbort()});
  rxProgress(100)
    for (i in 1:100) {
       rxTick()
       Sys.sleep(1 / 100)
    }
  rxProgressStop();
 }

## Not run:
f();

## End(Not run)
```

---

rxRateDur *Creates a rxRateDur object*

Description

This is primarily to display information about rate

Usage

```
rxRateDur(x)

## S3 method for class 'rxRateDur'
x[...]

as.rxRateDur(x)

## S3 method for class 'rxRateDur'
as.character(x, ...)

## S3 method for class 'rxRateDur'
x[[...]]

## S3 method for class 'rxRateDur'
```

```
type_sum(x)

pillar_shaft.rxRateDur(x, ...)

## S3 method for class 'rxRateDur'
as.data.frame(x, row.names = NULL,
  optional = FALSE, ..., nm = paste(deparse(substitute(x), width.cutoff
  = 500L), collapse = " "))
```

Arguments

x	rxRateDur data
...	additional arguments to be passed to or from methods.
row.names	NULL or a character vector giving the row names for the data frame. Missing values are not allowed.
optional	logical. If TRUE, setting row names and converting column names (to syntactic names: see make.names) is optional. Note that all of R's **base** package as.data.frame() methods use optional only for column names treatment, basically with the meaning of data.frame(*,check.names = !optional). See also the make.names argument of the matrix method.
nm	Name of column in new data frame

---

rxSetIni0 *Set Initial conditions to time zero instead of the first observed/dosed time*

Description

Set Initial conditions to time zero instead of the first observed/dosed time

Usage

```
rxSetIni0(ini0 = TRUE)
```

Arguments

ini0	When TRUE (default), set initial conditions to time zero. Otherwise the initial conditions are the first observed time.

`rxSetProd` *Choose the type of product to use in RxODE. These are used in the RxODE* `prod` *blocks*

Description

Choose the type of product to use in RxODE. These are used in the RxODE `prod` blocks

Usage

```
rxSetProd(type = c("long double", "double", "logify"))
```

Arguments

`type`	Product to use for `prod()` in RxODE blocks `long double` converts to long double, performs the multiplication and then converts back. `double` uses the standard double scale for multiplication.

Value

nothing

Author(s)

Matthew L. Fidler

`rxSetSum` *Choose the type of sums to use for RxODE.*

Description

Choose the types of sums to use in RxODE. These are used in the RxODE `sum` blocks and the `rxSum` function

Usage

```
rxSetSum(type = c("pairwise", "fsum", "kahan", "neumaier", "c"))
```

Arguments

`type`	Sum type to use for `rxSum` and `sum()` in RxODE code blocks. `pairwise` uses the pairwise sum (fast, default) `fsum` uses Python's fsum function (most accurate) `kahan` uses kahan correction `neumaier` uses Neumaier correction `c` uses no correction, bud default/native summing

Value

nothing

Author(s)

Matthew L. Fidler

`rxShiny` *Use Shiny to help develop an RxODE model*

Description

Use Shiny to help develop an RxODE model

Usage

```
rxShiny(object, params = c(), events = NULL, inits = c(), ...,
  data = data.frame())

## S3 method for class 'rxSolve'
rxShiny(object, params = NULL, events = NULL,
  inits = c(), ..., data = data.frame())

## Default S3 method:
rxShiny(object = NULL, params = c(), events = NULL,
  inits = c(), ..., data = data.frame())
```

Arguments

`object`	A RxODE family of objects. If not supplied a 2-compartment indirect effect model is used. If it is supplied, use the model associated with the RxODE object for the model exploration.
`params`	Initial parameters for model
`events`	Event information (currently ignored)
`inits`	Initial estimates for model
`...`	Other arguments passed to rxShiny. Currently doesn't do anything.
`data`	Any data that you would like to plot. If the data has a `time` variable as well as a compartment or calculated variable that matches the RxODE model, the data will be added to the plot of a specific compartment or calculated variable.

Value

Nothing; Starts a shiny server

Author(s)

Zufar Mulyukov and Matthew L. Fidler

`rxSimThetaOmega` *Simulate Parameters from a Theta/Omega specification*

Description

Simulate Parameters from a Theta/Omega specification

Usage

```
rxSimThetaOmega(params = NULL, omega = NULL, omegaDf = NULL,
  omegaLower = as.numeric(c(R_NegInf)),
  omegaUpper = as.numeric(c(R_PosInf)), omegaIsChol = FALSE,
  nSub = 1L, thetaMat = NULL, thetaLower = as.numeric(c(R_NegInf)),
  thetaUpper = as.numeric(c(R_PosInf)), thetaDf = NULL,
  thetaIsChol = FALSE, nStud = 1L, sigma = NULL,
  sigmaLower = as.numeric(c(R_NegInf)),
  sigmaUpper = as.numeric(c(R_PosInf)), sigmaDf = NULL,
  sigmaIsChol = FALSE, nCoresRV = 1L, nObs = 1L, dfSub = 0,
  dfObs = 0, simSubjects = TRUE)
```

Arguments

`params`	Named Vector of RxODE model parameters
`omega`	Named omega matrix.
`omegaDf`	The degrees of freedom of a t-distribution for simulation. By default this is `NULL` which is equivalent to `Inf` degrees, or to simulate from a normal distribution instead of a t-distribution.
`omegaLower`	Lower bounds for simulated ETAs (by default -Inf)
`omegaUpper`	Upper bounds for simulated ETAs (by default Inf)
`omegaIsChol`	Indicates if the `omega` supplied is a Cholesky decomposed matrix instead of the traditional symmetric matrix.
`nSub`	Number between subject variabilities (ETAs) simulated for every realization of the parameters.
`thetaMat`	Named theta matrix.
`thetaLower`	Lower bounds for simulated population parameter variability (by default -Inf)
`thetaUpper`	Upper bounds for simulated population unexplained variability (by default Inf)
`thetaDf`	The degrees of freedom of a t-distribution for simulation. By default this is `NULL` which is equivalent to `Inf` degrees, or to simulate from a normal distribution instead of a t-distribution.
`thetaIsChol`	Indicates if the `theta` supplied is a Cholesky decomposed matrix instead of the traditional symmetric matrix.
`nStud`	Number virtual studies to characterize uncertainty in estimated parameters.

sigma	Matrix for residual variation. Adds a "NA" value for each of the individual parameters, residuals are updated after solve is completed.
sigmaLower	Lower bounds for simulated unexplained variability (by default -Inf)
sigmaUpper	Upper bounds for simulated unexplained variability (by default Inf)
sigmaDf	Degrees of freedom of the sigma t-distribution. By default it is equivalent to Inf, or a normal distribution.
sigmaIsChol	Boolean indicating if the sigma is in the Cholesky decomposition instead of a symmetric covariance
nCoresRV	Number of cores used for the simulation of the sigma variables. By default this is 1. This uses the package rmvn and rmvt. To reproduce the results you need to run on the same platform with the same number of cores. This is the reason this is set to be one, regardless of what the number of cores are used in threaded ODE solving.
nObs	Number of observations to simulate (with sigma matrix)
dfSub	Degrees of freedom to sample the between subject variability matrix from the inverse Wishart distribution (scaled) or scaled inverse chi squared distribution.
dfObs	Degrees of freedom to sample the unexplained variability matrix from the inverse Wishart distribution (scaled) or scaled inverse chi squared distribution.
simSubjects	boolean indicated RxODE should simulate subjects in studies (TRUE, default) or studies (FALSE)

Author(s)

Matthew L.Fidler

---

rxStack *Stack a solved object for things like ggplot*

Description

Stack a solved object for things like ggplot

Usage

```
rxStack(Data, vars = NULL)
```

Arguments

Data	is a RxODE object to be stacked.
vars	Variables to include in stacked data; By default this is all the variables when vars is NULL.

Value

Stacked data with value and trt, where value is the values and trt is the state and lhs variables.

Author(s)

Matthew Fidler

`rxState` *State variables*

Description

This returns the model's compartments or states.

Usage

```
rxState(obj = NULL, state = NULL)
```

Arguments

`obj`	RxODE family of objects
`state`	is a string indicating the state or compartment that you would like to lookup.

Value

If state is missing, return a character vector of all the states.

If state is a string, return the compartment number of the named state.

Author(s)

Matthew L.Fidler

See Also

`RxODE`

`rxSumProdModel` *Recast model in terms of sum/prod*

Description

Recast model in terms of sum/prod

Usage

```
rxSumProdModel(model, expand = FALSE, sum = TRUE, prod = TRUE)
```

Arguments

model	RxODE model
expand	Boolean indicating if the expression is expanded.
sum	Use sum(...)
prod	Use prod(...)

Value

model string with prod(.) and sum(.) for all these operations.

Author(s)

Matthew L. Fidler

---

rxSymInvChol *Get Omega^-1 and derivatives*

Description

Get Omega^-1 and derivatives

Usage

```
rxSymInvChol(invObjOrMatrix, theta = NULL, type = "cholOmegaInv",
  thetaNumber = 0L)
```

Arguments

invObjOrMatrix	Object for inverse-type calculations. If this is a matrix, setup the object for inversion by rxSymInvCholCreate with the default arguments and return a reactive s3 object. Otherwise, use the inversion object to calculate the requested derivative/inverse.
theta	Thetas to be used for calculation. If missing (NULL), a special s3 class is created and returned to access Omega^1 objects as needed and cache them based on the theta that is used.
type	The type of object. Currently the following types are supported: • cholOmegaInv gives the Cholesky decomposition of the Omega Inverse matrix. • omegaInv gives the Omega Inverse matrix. • d(omegaInv) gives the d(Omega^-1) withe respect to the theta parameter specified in thetaNumber. • d(D) gives the d(diagonal(Omega^-1)) with respect to the theta parameter specified in the thetaNumber parameter
thetaNumber	For types d(omegaInv) and d(D), the theta number that the derivative is taken against. This must be positive from 1 to the number of thetas defining the Omega matrix.

Value

Matrix based on parameters or environment with all the matrixes calculated in variables omega, omegaInv, dOmega, dOmegaInv.

Author(s)

Matthew L. Fidler

`rxSymPyFix` *Fix SymPy expressions to be R parsable expressions*

Description

Fix SymPy expressions to be R parsable expressions

Usage

```
rxSymPyFix(var)
```

Arguments

`var`	sympy expression

Value

R valid expression

Author(s)

Matthew L. Fidler

`rxSymPySensitivity` *Calculate the sensitivity equations for a model*

Description

This expands the model to calculate sensitivities. This requires rSymPy.

Usage

```
rxSymPySensitivity(model, calcSens, calcJac = FALSE, keepState = NULL,
  collapseModel = FALSE)
```

Arguments

model	RxODE family of objects
calcSens	Either a logical or list of sensitivity parameters to calculate. When TRUE, calculate the sensitivities of all the known parameters. When FALSE raise an error.
calcJac	A boolean that determines if the Jacobian should be calculated.
keepState	State parameters to keep the sensitivities for.
collapseModel	A boolean to collapse the model that each expression only depends on the unspecified parameters (instead on LHS quantities).

Value

Model syntax that includes the sensitivity parameters.

Author(s)

Matthew L. Fidler

---

rxSymPyVersion *Return the version of SymPy that is running*

Description

Return the version of SymPy that is running

Usage

```
rxSymPyVersion(numeric = TRUE)
```

Arguments

numeric	boolean that specifies if the major and minor release should be a number.

Value

Version of sympy that is running.

Author(s)

Matthew L. Fidler

rxSyncOptions *Sync options with RxODE variables*

Description

Accessing RxODE options via getOption slows down solving. This allows the options to be synced with variables.

Usage

```
rxSyncOptions()
```

Author(s)

Matthew L. Fidler

rxTempDir *Get the RxODE temporary directory*

Description

Get the RxODE temporary directory

Usage

```
rxTempDir()
```

Value

RxODE temporary directory.

rxTrans *Translate the model to C code if needed*

Description

This function translates the model to C code, if needed

Usage

```
rxTrans(model, extraC = NULL, modelPrefix = "", md5 = "",
  modName = NULL, modVars = FALSE, ...)

## Default S3 method:
rxTrans(model, extraC = NULL, modelPrefix = "",
  md5 = "", modName = NULL, modVars = FALSE, ...)

## S3 method for class 'character'
rxTrans(model, extraC = NULL, modelPrefix = "",
  md5 = "", modName = NULL, modVars = FALSE, ...)
```

Arguments

model	This is the ODE model specification. It can be: • a string containing the set of ordinary differential equations (ODE) and other expressions defining the changes in the dynamic system. • a file name where the ODE system equation is contained • An ODE expression enclosed in {} (see also the filename argument). For details, see the sections "Details" and "RxODE Syntax" below.
extraC	Extra c code to include in the model. This can be useful to specify functions in the model. These C functions should usually take double precision arguments, and return double precision values.
modelPrefix	Prefix of the model functions that will be compiled to make sure that multiple RxODE objects can coexist in the same R session.
md5	Is the md5 of the model before parsing, and is used to embed the md5 into DLL, and then provide for functions like rxModelVars.
modName	a string to be used as the model name. This string is used for naming various aspects of the computations, including generating C symbol names, dynamic libraries, etc. Therefore, it is necessary that modName consists of simple ASCII alphanumeric characters starting with a letter.
modVars	returns the model variables instead of the named vector of translated properties.
...	Ignored parameters.

Value

a named vector of translated model properties including what type of jacobian is specified, the C function prefixes, as well as the C functions names to be called through the compiled model.

Author(s)

Matthew L.Fidler

See Also

RxODE, rxCompile.

rxUnloadAll *Unload all RxODE Dlls that are not locked for solving.*

Description

Unload all RxODE Dlls that are not locked for solving.

Usage

```
rxUnloadAll()
```

rxUse *Use model object in your package*

Description

Use model object in your package

Usage

```
rxUse(obj, overwrite = TRUE, compress = "bzip2", internal = FALSE)
```

Arguments

obj	model to save.
overwrite	By default, use_data() will not overwrite existing files. If you really want to do so, set this to TRUE.
compress	Choose the type of compression used by save(). Should be one of "gzip", "bzip2", or "xz".
internal	If this is run internally. By default this is FALSE

rxValidate *Validate RxODE*

Description

This allows easy validation/qualification of nlmixr by running the testing suite on your system.

Usage

```
rxValidate(full = TRUE)

rxTest(full = TRUE)
```

Arguments

full	Should a full validation be performed? (By default TRUE)

Author(s)

Matthew L. Fidler

rxWinPythonSetup *Setup Python and SymPy for windows*

Description

Setup Python and SymPy for windows

Usage

```
rxWinPythonSetup()
```

Author(s)

Matthew L. Fidler

rxWinSetup *Setup Windows components for RxODE*

Description

Setup Windows components for RxODE

Usage

```
rxWinSetup(rm.rtools = TRUE, rm.python = TRUE)
```

Arguments

rm.rtools	Remove the Rtools from the current path specs.
rm.python	Remove Python from the current path specs.

Author(s)

Matthew L. Fidler

summary.RxODE *Print expanded information about the RxODE object.*

Description

This prints the expanded information about the RxODE object.

Usage

```
## S3 method for class 'RxODE'
summary(object, ...)
```

Arguments

object	RxODE object
...	Ignored parameters

Author(s)

Matthew L.Fidler

tibble *type_sum function for units*

Description

type_sum function for units

Usage

```
## S3 method for class 'units'
type_sum(x, ...)

format_type_sum.type_sum_units(x, width, ...)

pillar_shaft.units(x, ...)

## S3 method for class 'mixed_units'
type_sum(x, ...)

pillar_shaft.mixed_units(x, ...)

## S3 method for class 'units'
scale_type(x)
```

Arguments

x	see type_sum
...	see type_sum
width	see type_sum

Index

∗Topic **Internal**
print.rxCoefSolve, 35
∗Topic **data**
eventTable, 30
∗Topic **intenral**
.clearPipe, 4
∗Topic **models**
eventTable, 30
RxODE, 54
∗Topic **nonlinear**
genShinyApp.template, 32
RxODE, 54
∗Topic **simulation**
genShinyApp.template, 32
+.solveRxDll *(*rxChain*)*, 38
.C, *40*
.Call, *40*
.clearPipe, 4
.rxFindPow, 5
.rxRmPrint, 6
.rxRmSens, 6
.rxSymPyJacobian, 7
.rxWinRtoolsPath, 7
.setWarnIdSort, 8
[.rxEvid *(*rxEvid*)*, 49
[.rxRateDur *(*rxRateDur*)*, 64
[[.rxEvid *(*rxEvid*)*, 49
[[.rxRateDur *(*rxRateDur*)*, 64

add.dosing, 8, *9*, *11*, *19*, *23*, *26*, *28*, *58*
add.sampling, *9*, *11*, 11, *19*, *23*, *26*, *28*, *58*
as.character.rxEvid *(*rxEvid*)*, 49
as.character.rxRateDur *(*rxRateDur*)*, 64
as.data.frame.rxEvid, 13
as.data.frame.rxRateDur *(*rxRateDur*)*, 64
as.data.table.rxEt, 14
as.et, 14
as.rxEvid *(*rxEvid*)*, 49
as.rxRateDur *(*rxRateDur*)*, 64
as.tbl.rxEt *(*as_tibble.rxEt*)*, 15
as_tibble.rxEt, 15

c.rxEvid *(*rxEvid*)*, 49
c.rxRateDur *(*rxEvid*)*, 49
coef.RxCompilationManager *(*coef.RxODE*)*, 15
coef.rxDll *(*coef.RxODE*)*, 15
coef.RxODE, 15
coef.solveRxODE *(*coef.RxODE*)*, 15
cvPost, 16

data.frame, *13*, *65*

environment, *18*
et, *9*, *11*, 17, *19*, *23*, *26*, *28*, *31*, *58*
etExpand, 21
etRbind, *9*, *11*, *19*, 22, *23*, *26*, *28*
etRep, *9*, *11*, *19*, *23*, 24, *26*, *28*
etSeq, 27
eventTable, *4*, *9*, *11*, *19*, *23*, *26*, *28*, 30, *34*, *46*, *47*, *56–58*

format.rxEvid *(*rxEvid*)*, 49
format.rxRateDur *(*rxEvid*)*, 49
format_type_sum.type_sum_units *(*tibble*)*, 79

genShinyApp.template, 32

is.rxEt, 34
is.rxSolve, 34

make.names, *13*, *65*

pillar_shaft.mixed_units *(*tibble*)*, 79
pillar_shaft.rxEvid *(*rxEvid*)*, 49
pillar_shaft.rxRateDur *(*rxRateDur*)*, 64
pillar_shaft.units *(*tibble*)*, 79
predict.rxEt *(*rxControl*)*, 41
predict.RxODE *(*rxControl*)*, 41
predict.rxParams *(*rxControl*)*, 41

predict.rxSolve (rxControl), 41
print.rxCoefSolve, 35
print.rxEvid (rxEvid), 49
print.RxODE, 35

rbind.rxEt (etRbind), 22
rep.rxEt (etRep), 24
rinvchisq, 36
rmvn, *44*, *69*
rmvt, *44*, *69*
rxAddReturn, 36
rxAllowUnload, 37
rxAssignPtr, 37
rxC14, 38
rxChain, 38
rxClean, 39
rxCompile, 39, *75*
rxControl, 41
rxCores, *43*, 48
rxDelete, 48
rxDfdy, 49
rxEvid, 49
rxFoExpandEta, 50
rxGetRxODE, 51
rxHtml, 51
rxInv, 52
rxIsCurrent, 52
rxLhs, 53
rxLock, 53
rxModelVars, *75*
rxNorm, 54
RxODE, *9*, *11*, *19*, *23*, *26*, *28*, *31*, *33*, *34*, *41*, *48*, *53*, 54, *70*, *75*
rxOptExpr, 59
rxOptions, 60
rxParam (rxParams), 61
rxParams, 61
rxPermissive, 62
rxProgress, 63
rxProgressAbort (rxProgress), 63
rxProgressStop (rxProgress), 63
rxRateDur, 64
rxSetIni0, 65
rxSetProd, 66
rxSetSum, 66
rxShiny, 67
rxSimThetaOmega, 68
rxSolve, *33*, *56*
rxSolve (rxControl), 41
rxStack, 69
rxState, 70
rxStrict (rxPermissive), 62
rxSumProdModel, 70
rxSymInvChol, 71
rxSymInvCholCreate, *71*
rxSymPyFix, 72
rxSymPySensitivity, 72
rxSymPyVersion, 73
rxSyncOptions, 74
rxTempDir, 74
rxTest (rxValidate), 77
rxTick (rxProgress), 63
rxTrans, *40*, 74
rxUnloadAll, 76
rxUnlock (rxLock), 53
rxUse, 76
rxValidate, 77
rxWinPythonSetup, 77
rxWinSetup, 78

save(), *76*
scale_type.units (tibble), 79
seq.rxEt (etSeq), 27
simulate.RxODE (rxControl), 41
simulate.rxParams (rxControl), 41
simulate.rxSolve (rxControl), 41
solve.rxEt (rxControl), 41
solve.RxODE (rxControl), 41
solve.rxParams (rxControl), 41
solve.rxSolve (rxControl), 41
summary.RxODE, 78
sys.call, *18*

tibble, 79
type_sum, *79*
type_sum.mixed_units (tibble), 79
type_sum.rxEvid (rxEvid), 49
type_sum.rxRateDur (rxRateDur), 64
type_sum.units (tibble), 79

update.rxSolve (rxControl), 41

write.template.server (genShinyApp.template), 32
write.template.ui, *33*
write.template.ui (genShinyApp.template), 32