Package: rxode2parse (via r-universe)

September 25, 2024

Title Parsing and Code Generation Functions for 'rxode2'

Version 2.0.19

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

Description Provides the parsing needed for 'rxode2' (Wang, Hallow and James (2016) <doi:10.1002/psp4.12052>). It also provides the 'stan' based advan linear compartment model solutions with gradients (Carpenter et al (2015), <doi:10.48550/arXiv.1509.07164>) needed in 'nlmixr2' (Fidler et al (2019) <doi:10.1002/psp4.12445>). This split will reduce computational burden of recompiling 'rxode2'.

License GPL (>= 3)

URL https://nlmixr2.github.io/rxode2parse/, https://github.com/nlmixr2/rxode2parse/

BugReports https://github.com/nlmixr2/rxode2parse/issues/

Depends R (>= 4.0.0)

Imports checkmate, crayon, dparser, knitr, methods, qs, Rcpp (>= 1.0.8), utils, digest, rex, symengine, data.table (>= 1.12.4)

Suggests testthat (>= 3.0.0), dplyr, nlmixr2data, devtools, rmarkdown

LinkingTo BH (>= 1.78.0.0), dparser (>= 1.3.1-10), Rcpp (>= 1.0.8), RcppEigen (>= 0.3.3.9.2), StanHeaders (>= 2.21.0.7)

Biarch true

Config/testthat/edition 3

Encoding UTF-8

Language en-US

NeedsCompilation yes

RoxygenNote 7.3.1

Author Matthew L. Fidler [aut, cre] (<https://orcid.org/0000-0001-8538-6691>), Wenping Wang [aut], Richard Upton [ctb], Gabriel Staples [ctb], Goro Fuji [ctb], Morwenn [ctb], Igor Kushnir [ctb], Kevin Ushey [ctb], David Cooley [ctb]

Date/Publication 2024-05-25 23:10:02 UTC

Additional_repositories https://cranhaven.r-universe.dev

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

RemoteUrl https://github.com/cranhaven/cranhaven.r-universe.dev

RemoteRef package/rxode2parse

RemoteSha 02fc701f938fbada33082cd2efc2bcd40706e904

Contents

`.toClassicEvid` *This converts NONMEM-style EVIDs to classic RxODE events*

Description

This converts NONMEM-style EVIDs to classic RxODE events

Usage

```
.toClassicEvid(cmt = 1L, amt = 0, rate = 0, dur = 0, ii = 0, evid = 0L, ss = 0)
```

Arguments

`cmt`	compartment flag
`amt`	dose amount
`rate`	dose rate
`dur`	dose duration
`ii`	inter-dose interval
`evid`	event id
`ss`	steady state

Value

classic evids, excluding evids that are added (you need to add them manually) or simply use etTran. This is mostly for testing and really shouldn't be used directly.

Author(s)

Matthew L. Fidler

Examples

```
.toClassicEvid(cmt=10, amt=3, evid=1)
.toClassicEvid(cmt=10, amt=3, rate=2, evid=1)
.toClassicEvid(cmt=10, amt=3, rate=-1, evid=1)
.toClassicEvid(cmt=10, amt=3, rate=-2, evid=1)
.toClassicEvid(cmt=10, amt=3, dur=2, evid=1)
.toClassicEvid(cmt=304, amt=3, dur=2, evid=1)
.toClassicEvid(cmt=7, amt=0, rate=2, evid=1, ss=1)
.toClassicEvid(cmt=-10, amt=3, evid=1)
.toClassicEvid(cmt=10, amt=3, evid=5)
.toClassicEvid(cmt=6, amt=3, evid=6)
.toClassicEvid(cmt=6, amt=3, evid=7)
.toClassicEvid(evid=2)
.toClassicEvid(evid=4)
```

---

print.rxModelVars *Print Values*

Description

`print` prints its argument and returns it *invisibly* (via `invisible(x)`). It is a generic function which means that new printing methods can be easily added for new `class`es.

Usage

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

Arguments

`x`	an object used to select a method.
`...`	further arguments passed to or from other methods.

Details

The default method, `print.default` has its own help page. Use `methods("print")` to get all the methods for the `print` generic.

`print.factor` allows some customization and is used for printing `ordered` factors as well.

`print.table` for printing `table`s allows other customization. As of R 3.0.0, it only prints a description in case of a table with 0-extents (this can happen if a classifier has no valid data).

See `noquote` as an example of a class whose main purpose is a specific `print` method.

Value

This returns invisibly the model variables object

References

Chambers, J. M. and Hastie, T. J. (1992) *Statistical Models in S.* Wadsworth & Brooks/Cole.

See Also

The default method `print.default`, and help for the methods above; further `options`, `noquote`.

For more customizable (but cumbersome) printing, see `cat`, `format` or also `write`. For a simple prototypical print method, see `.print.via.format` in package **tools**.

Examples

```
require(stats)

ts(1:20)  #-- print is the "Default function" --> print.ts(.) is called
for(i in 1:3) print(1:i)

## Printing of factors
attenu$station ## 117 levels -> 'max.levels' depending on width

## ordered factors: levels  "l1 < l2 < .."
esoph$agegp[1:12]
esoph$alcgp[1:12]

## Printing of sparse (contingency) tables
set.seed(521)
t1 <- round(abs(rt(200, df = 1.8)))
t2 <- round(abs(rt(200, df = 1.4)))
table(t1, t2) # simple
print(table(t1, t2), zero.print = ".") # nicer to read

## same for non-integer "table":
T <- table(t2,t1)
T <- T * (1+round(rlnorm(length(T)))/4)
print(T, zero.print = ".") # quite nicer,
print.table(T[,2:8] * 1e9, digits=3, zero.print = ".")
## still slightly inferior to  Matrix::Matrix(T)  for larger T
```

```
## Corner cases with empty extents:
table(1, NA) # < table of extent 1 x 0 >
```

`rxDerived` *Calculate derived parameters for the 1-, 2-, and 3- compartment linear models.*

Description

This calculates the derived parameters based on what is provided in a data frame or arguments

Usage

```
rxDerived(..., verbose = FALSE, digits = 0)
```

Arguments

`...`	The input can be: • A data frame with PK parameters in it; This should ideally be a data frame with one pk parameter per row since it will output a data frame with one PK parameter per row. • PK parameters as either a vector or a scalar
`verbose`	boolean that when TRUE provides a message about the detected pk parameters and the detected compartmental model. By default this is `FALSE`.
`digits`	represents the number of significant digits for the output; If the number is zero or below (default), do not round.

Value

Return a data.frame of derived PK parameters for a 1-, 2-, or 3-compartment linear model given provided clearances and volumes based on the inferred model type.

The model parameters that will be provided in the data frame are:

- `vc`: Central Volume (for 1-, 2- and 3- compartment models)
- `kel`: First-order elimination rate (for 1-, 2-, and 3-compartment models)
- `k12`: First-order rate of transfer from central to first peripheral compartment; (for 2- and 3-compartment models)
- `k21`: First-order rate of transfer from first peripheral to central compartment, (for 2- and 3-compartment models)
- `k13`: First-order rate of transfer from central to second peripheral compartment; (3-compartment model)
- `k31`: First-order rate of transfer from second peripheral to central compartment (3-compartment model)
- `vp`: Peripheral Volume (for 2- and 3- compartment models)

- vp2: Peripheral Volume for 3rd compartment (3- compartment model)
- vss: Volume of distribution at steady state; (1-, 2-, and 3-compartment models)
- t12alpha: $t_{1/2,\alpha}$; (1-, 2-, and 3-compartment models)
- t12beta: $t_{1/2,\beta}$; (2- and 3-compartment models)
- t12gamma: $t_{1/2,\gamma}$; (3-compartment model)
- alpha: α; (1-, 2-, and 3-compartment models)
- beta: β; (2- and 3-compartment models)
- gamma: β; (3-compartment model)
- A: true A; (1-, 2-, and 3-compartment models)
- B: true B; (2- and 3-compartment models)
- C: true C; (3-compartment model)
- fracA: fractional A; (1-, 2-, and 3-compartment models)
- fracB: fractional B; (2- and 3-compartment models)
- fracC: fractional C; (3-compartment model)

Author(s)

Matthew Fidler and documentation from Justin Wilkins, <justin.wilkins@occams.com>

References

Shafer S. L. CONVERT.XLS

Rowland M, Tozer TN. Clinical Pharmacokinetics and Pharmacodynamics: Concepts and Applications (4th). Clipping Williams & Wilkins, Philadelphia, 2010.

Examples

```
## Note that rxode2 parses the names to figure out the best PK parameter

params <- rxDerived(cl = 29.4, v = 23.4, Vp = 114, vp2 = 4614, q = 270, q2 = 73)

## That is why this gives the same results as the value before

params <- rxDerived(CL = 29.4, V1 = 23.4, V2 = 114, V3 = 4614, Q2 = 270, Q3 = 73)

## You may also use micro-constants alpha/beta etc.

params <- rxDerived(k12 = 0.1, k21 = 0.2, k13 = 0.3, k31 = 0.4, kel = 10, v = 10)

## or you can mix vectors and scalars

params <- rxDerived(CL = 29.4, V = 1:3)

## If you want, you can round to a number of significant digits
## with the `digits` argument:

params <- rxDerived(CL = 29.4, V = 1:3, digits = 2)
```

`rxode2parse` *Internal translation to get model variables list*

Description

Internal translation to get model variables list

Usage

```
rxode2parse(
  model,
  linear = FALSE,
  linCmtSens = c("linCmtA", "linCmtB", "linCmtC"),
  verbose = FALSE,
  code = NULL,
  envir = parent.frame()
)
```

Arguments

`model`	Model (either file name or string)
`linear`	boolean indicating if linear compartment model should be generated from `linCmt()` (default FALSE)
`linCmtSens`	Linear compartment model sensitivity type
`verbose`	is a boolean indicating the type of model detected with `linCmt()` parsing
`code`	is a file name where the c code is written to (for testing purposes mostly, it needs `rxode2` to do anything fancy)
`envir`	is the environment to look for R user functions (defaults to parent environment)

Value

A rxModelVars object that has the model variables of a rxode2 syntax expression

Examples

```
rxode2parse("a=3")
```

rxode2parseAssignTranslation *This assigns the c level linkages for a roxde2 model*

Description

This assigns the c level linkages for a roxde2 model

Usage

```
rxode2parseAssignTranslation(df)
```

Arguments

df	data frame containing the character column names rxFun, fun, type, package, packageFun and the integer column names argMin and argMax

Value

Nothing called for side effects

Author(s)

Matthew L. Fidler

Examples

```
rxode2parseAssignTranslation(rxode2parseGetTranslation())
```

rxode2parseD *This gives the derivative table for rxode2*

Description

This will help allow registration of functions in rxode2

Usage

```
rxode2parseD()
```

Details

This environment is a derivative table;

For example:

Derivative(f(a,b,c), a) = fa() Derivative(f(a,b,c), b) = fb() Derivative(f(a,b,c), c) = fc()

Then the derivative table for `f` would be:

assign("f", list(fa(a,b,c), fb(a,b,c), fc(a,b,c)), rxode2parseD())

fa translates the arguments to the derivative with respect to a fb translates the arguments to the derivative with respect to b

If any of the list is NULL then rxode2 won't know how to take a derivative with respect to the argument.

If the list is shorter than the length of the arguments then the argument then the derivative of arguments that are not specified cannot be taken.

Value

Derivative table environment for rxode2

Author(s)

Matthew L. Fidler

`rxode2parseGetPackagesToLoad` *Control the packages that are loaded when a* `rxode2` *model dll is loaded*

Description

Control the packages that are loaded when a `rxode2` model dll is loaded

Usage

```
rxode2parseGetPackagesToLoad()

rxode2parseAssignPackagesToLoad(pkgs = rxode2parseGetPackagesToLoad())
```

Arguments

`pkgs`	The packages to make sure are loaded every time you load an rxode2 model.

Value

List of packages to load

Author(s)

Matthew Fidler

Examples

```
rxode2parseGetPackagesToLoad()

rxode2parseAssignPackagesToLoad(rxode2parseGetPackagesToLoad())
```

---

rxode2parseGetPointerAssignment

This function gets the currently assigned function pointer assignments

---

Description

This function gets the currently assigned function pointer assignments

Usage

```
rxode2parseGetPointerAssignment()
```

Value

The currently assigned pointer assignments

Author(s)

Matthew L. Fidler

Examples

```
rxode2parseGetTranslation()
```

---

rxode2parseGetTranslation

This function gets the currently assigned translations

Description

This function gets the currently assigned translations

Usage

```
rxode2parseGetTranslation()
```

Value

The currently assigned translations

Author(s)

Matthew L. Fidler

Examples

```
rxode2parseGetTranslation()
```

---

rxode2parseMd5 *Get the MD5 hash of the current language revision*

Description

Get the MD5 hash of the current language revision

Usage

```
rxode2parseMd5()
```

Value

md5 hash of language revision

Author(s)

Matthew L. Fidler

Examples

```
rxode2parseMd5()
```

rxParseSuppressMsg *Respect suppress messages*

Description

This turns on the silent REprintf in C when `suppressMessages()` is turned on. This makes the `REprintf` act like `messages` in R, they can be suppressed with `suppressMessages()`

Usage

```
rxParseSuppressMsg()
```

Value

Nothing

Author(s)

Matthew Fidler

Examples

```
# rxParseSuppressMsg() is called with rxode2()

# Note the errors are output to the console

try(rxode2parse("d/dt(matt)=/3"), silent = TRUE)

# When using suppressMessages, the output is suppressed

suppressMessages(try(rxode2parse("d/dt(matt)=/3"), silent = TRUE))

# In rxode2, we use REprintf so that interrupted threads do not crash R
# if there is a user interrupt. This isn't captured by R's messages, but
# This interface allows the `suppressMessages()` to suppress the C printing
# as well

# If you  want to suppress messages from rxode2 in other packages, you can use
# this function
```

`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.

Value

the boolean ini0, though this is called for its side effects

Index