Package 'RsSimulx'

May 7, 2026

Type Package

Title Extension of 'lixoftConnectors' for 'Simulx'

Version 2024.1

Maintainer Chloe Bracis <support@lixoft.com>

Description Provides useful tools which supplement the use of 'Simulx' software and 'R' connectors ('Monolix Suite'). 'Simulx' is an easy, efficient and flexible application for clinical trial simulations. You need 'Simulx' software to be installed in order to use 'RsSimulx' package. Among others tasks, 'RsSimulx' provides the same functions as package 'mlxR' does with a compatibility with 'Simulx' software.

SystemRequirements 'Simulx' (<http://simulx.lixoft.com/>)

Depends R (>= 3.0.0), ggplot2

Imports gridExtra, utils, stats, grDevices

Encoding UTF-8

Collate catplotmlx.R ggplotmlx.R kmplotmlx.R prctilemlx.R simulxR.R exposure.R shinymlx.R statmlx.R stoolsmlx.R simpop.R smlx-checks.R smlx-tools.R smlx-data.R smlx-project.R apiTools.R apiManager.R utils.R data.R RsSimulx-deprecated.R zzz.R

LazyData true

License BSD_2_clause + file LICENSE

Copyright LIXOFT

RoxygenNote 7.3.1

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Clemence Pinaud [aut],
Jonathan Chauvin [aut],
Marc Lavielle [aut],
Frano Mihaljevic [aut],
Chloe Bracis [aut, cre]

Repository CRAN

Date/Publication 2024-06-10 12:30:07 UTC

Contents

`catplotmlx` *Plot Categorical Longitudinal Data*

Description

Plot the empirical distribution of categorical longitudinal data.

Usage

```
catplotmlx(
  r,
  col = NULL,
  breaks = NULL,
  plot = TRUE,
  color = "#194280",
  group = NULL,
  facet = TRUE,
  labels = NULL
)
```

Arguments

`r`	a data frame with a column 'id', a column 'time', a column with values and possibly Hk[ja column 'group'.
`col`	a vector of 3 column numbers: ('id', 'time/x', 'y'. Default = c(1, 2,3).
`breaks`	one of:

- a vector giving the breakpoints,
- a single number giving the number of segments.

plot	if TRUE the empirical distribution is displayed, if FALSE the values are returned
color	a color to be used for the plots (default="#194280")
group	variable to be used for defining groups (by default, 'group' is used when it exists)
facet	makes subplots for different groups if TRUE
labels	vector of strings

Details

See http://simulx.webpopix.org/mlxr/catplotmlx/ for more details.

Value

a ggplot object if plot=TRUE ; otherwise, a list with fields:

- color a vector of colors used for the plot
- y a data frame with the values of the empirical distribution computed at each time point

Examples

```
## Not run:
  catModel <- inlineModel("
  [LONGITUDINAL]
  input =  {a,b}
  EQUATION:
  lp1=a-b*t
  lp2=a-b*t/2
  DEFINITION:
  y = {type=categorical, categories={1,2,3},
  logit(P(y<=1))=lp1, logit(P(y<=2))=lp2}
  ")

  y.out  <- list(name='y', time=seq(0, 100, by=4))

  Ng  <- 1000
  g1 <- list(size=Ng, parameter=c(a=6,b=0.2))
  res <- simulx(model=catModel, output=y.out, group=g1)
  catplotmlx(res$y)
  catplotmlx(res$y, breaks=seq(-2,102,by=8), color="purple")
  catplotmlx(res$y, breaks=5, color="#490917")

  g2 <- list(size=Ng, parameter=c(a=10,b=0.2))
  res <- simulx(model=catModel, output=y.out, group=list(g1,g2))
  catplotmlx(res$y)
  catplotmlx(res$y, group="none")

  g3 <- list(size=Ng, parameter=c(a=6,b=0.4))
  g4 <- list(size=Ng, parameter=c(a=10,b=0.4))
```

```
  res <- simulx(model=catModel, output=y.out, group=list(g1,g2,g3,g4))
  catplotmlx(res$y)

  cov <- data.frame(id=levels(res$y$id), a=rep(c(6,10,6,10),each=Ng),
                    b=rep(c(0.2,0.2,0.4,0.4),each=Ng))
  catplotmlx(res$y, group=cov)

## End(Not run)
```

`exposure` *Computation of AUC, Cmax and Cmin*

Description

Compute the area under the curve, the maximum and minimum values of a function of time over a given interval or at steady state

Usage

```
exposure(
  model = NULL,
  output = NULL,
  group = NULL,
  treatment = NULL,
  parameter = NULL,
  data = NULL,
  project = NULL,
  settings = NULL,
  regressor = NULL,
  varlevel = NULL
)
```

Arguments

`model`	a `Mlxtran` model used for the simulation
`output`	a list with fields: • `name`: a vector of output names • `time`: = 'steady.state' • `ntp`: number of time points used for computing the exposure (default=100) • `tol`: tolerance number, between 0 and 1, for approximating steaty-state (default=0.01) • `ngc`: number of doses used for estimating the convergence rate to steaty-state (default=5)
`group`	a list, or a list of lists, with fields: • `size` : size of the group (default=1), • `level` : level(s) of randomization,

- parameter : if different parameters per group are defined,
- output : if different outputs per group are defined,
- treatment : if different treatements per group are defined,
- regressor : if different regression variables per group are defined.

treatment — a list with fields

- tfd : time of first dose (default=0),
- ii : inter dose interval (mandatory),
- amount : the amount for each dose,
- rate : the infusion rate (default=Inf),
- tinf : the time of infusion (default=0),
- type : the type of input (default=1),
- target : the target compartment (default=NULL).

parameter — a vector of parameters with their names and values

data — a list

project — the name of a Monolix project

settings — a list of optional settings

- result.file : name of the datafile where the simulated data is written (string),
- seed : initialization of the random number generator (integer),
- load.design : TRUE/FALSE (if load.design is not defined, a test is automatically performed to check if a new design has been defined),
- data.in : TRUE/FALSE (default=FALSE)
- id.out : add columns id (when N=1) and group (when #group=1), TRUE/FALSE (default=FALSE)
- Nmax : maximum group size used in a single call of mlxCompute (default=100)

regressor — a list, or a list of lists, with fields

- name : a vector of regressor names,
- time : a vector of times,
- value : a vector of values.

varlevel — a list, or a list of lists, with fields

- name : a vector of names of variability levels,
- time : a vector of times that define the occasions.

Details

Input arguments are the input arguments of Simulx (http://simulx.webpopix.org)

Specific input arguments can be also used for computing the exposure at steady state, i.e. after the administration of an "infinite" number of doses. See http://simulx.webpopix.org/exposure/ for more details.

Value

A list of data frames. One data frame per output is created with columns `id` (if number of subject >1), `group` (if number of groups >1), `t1` (beginning of time interval), `t2` (end of time interval), `step` (time step), `auc` (area under the curve), `tmax` (time of maximum value), `cmax` (maximum value), `tmin` (time of minimum value), `cmin` (minimum value).

---

ggplotmlx *wrapper for ggplot*

Description

wrapper around `ggplot` with a custom theme

Usage

```
ggplotmlx(...)
```

Arguments

`...`	parameters passed to `ggplot`

Value

see `ggplot`

---

initRsSimulx *Initialize RsSimulx library*

Description

Initialize RsSimulx library

Usage

```
initRsSimulx(path = NULL, ...)
```

Arguments

`path`	(*character*) (*optional*) Path to installation directory of the Lixoft suite. If RsSimulx library is not already loaded and no path is given, the directory written in the lixoft.ini file is used for initialization.
`...`	(*optional*) Extra arguments passed to lixoftConnectors package when RsSimulx is used with a version of Lixoft(/@) software suite. • `force` (*bool*) (*optional*) Should RsSimulx initialization overpass lixoftConnectors software switch security or not. Equals FALSE by default.

Value

A list:

- `software`: the software that is used (should be simulx)
- `path`: the path to MonolixSuite
- `version`: the version of MonolixSuite that is used
- `status`: boolean equaling TRUE if the initialization has been successful.

Examples

```
## Not run:
initRsSimulx(path = "/path/to/lixoftRuntime/")

## End(Not run)
```

`inlineDataFrame` *Inline dataframe*

Description

Convert a string in dataframe and save it in a temporary file

Usage

```
inlineDataFrame(str)
```

Arguments

`str`	(*string*) Dataframe in string format

Value

dataframe object

Examples

```
## Not run:
occ <- inlineDataFrame("
 id time occ
 1   0    1
 1   12   2
 1   24   3
 2   0    1
 2   24   2
 3   0    1
")

## End(Not run)
```

`inlineModel` *Inline model*

Description

Save a string in a temporary file to be used as a model file

Usage

```
inlineModel(srtIn, filename = NULL)
```

Arguments

`srtIn`	(*string*) Model in string format,
`filename`	(*string*) name of the model file (by default the model is saved in a temporary file)

Value

Name of the model file

Examples

```
## Not run:
myModel <- inlineModel("
[LONGITUDINAL]
input = {A, k, c, a}
EQUATION:
t0    = 0
f_0   = A
ddt_f = -k*f/(c+f)
DEFINITION:
y = {distribution=normal, prediction=f, sd=a}
[INDIVIDUAL]
input = {k_pop, omega}
DEFINITION:
k = {distribution=lognormal, prediction=k_pop, sd=omega}
")

## End(Not run)
```

kmplotmlx *Kaplan Meier plot*

Description

Plot empirical survival functions using the Kaplan Meier estimate.

Usage

```
kmplotmlx(
  r,
  index = 1,
  level = NULL,
  time = NULL,
  cens = TRUE,
  plot = TRUE,
  color = "#e05969",
  group = NULL,
  facet = TRUE,
  labels = NULL
)
```

Arguments

r	a data frame with a column 'id', a column 'time', a column with values and possibly a column 'group'.
index	an integer: index=k means that the survival function for the k-th event is displayed. Default is index=1.
level	a number between 0 and 1: confidence interval level.
time	a vector of time points where the survival function is evaluated.
cens	if TRUE right censoring times are diplayed.
plot	if TRUE the estimated survival function is displayed, if FALSE the values are returned
color	color to be used for the plots (default="#e05969")
group	variable to be used for defining groups (by default, 'group' is used when it exists)
facet	makes subplots for different groups if TRUE
labels	vector of strings

Details

See http://simulx.webpopix.org/mlxr/kmplotmlx/ for more details.

Value

a ggplot object if `plot=TRUE` ; otherwise, a list with fields:

- `surv` a data frame with columns `T` (time), `S` (survival), possibly `(S1, S2)` (confidence interval) and possibly `group`
- `cens` a data frame with columns `T0` (time), `S0` (survival) and possibly `group`

Examples

```
## Not run:
tteModel1 <- inlineModel("
  [LONGITUDINAL]
  input = {beta,lambda}
  EQUATION:
  h=(beta/lambda)*(t/lambda)^(beta-1)
  DEFINITION:
  e = {type=event, maxEventNumber=1, rightCensoringTime=70, hazard=h}
  ")

  p1   <- c(beta=2.5,lambda=50)
  e    <- list(name='e', time=0)
  res1 <- simulx(model=tteModel1, parameter=p1, output=e, group=list(size=100))
  pl1  <- kmplotmlx(res1$e,level=0.95)
  print(pl1)

  p2   <- c(beta=2,lambda=45)
  g1   <- list(size=50, parameter=p1)
  g2   <- list(size=100, parameter=p2)
  res2 <- simulx(model=tteModel1, output=e, group=list(g1,g2))
  pl2  <- kmplotmlx(res2$e)
  print(pl2)

## End(Not run)
```

---

lixoft.read.table *Read Lixoft@ files*

Description

Utility function to read Lixoft@ formated input/output files

Usage

```
lixoft.read.table(file, sep = "", ...)
```

Arguments

`file`	file path of the file to read
`sep`	separator
`...`	see `read.table`

Value

a dataframe object

`prctilemlx` *Percentiles of the empiricial distribution of longitudinal data*

Description

Compute and display percentiles of the empiricial distribution of longitudinal data.

Usage

```
prctilemlx(
  r = NULL,
  col = NULL,
  project = NULL,
  outputVariableName = NULL,
  number = 8,
  level = 80,
  plot = TRUE,
  color = NULL,
  group = NULL,
  facet = TRUE,
  labels = NULL,
  band = NULL
)
```

Arguments

`r`	a data frame with a column 'id', a column 'time' and a column with values. The times should be the same for each individual.
`col`	a vector with the three column indexes for 'id', 'time/x' and 'y'. Default = c(1, 2,3).
`project`	simulx project filename (with extension ".smlx")
`outputVariableName`	name of the output to consider. By default the first output will be consider. You must define either a 'r' dataframe and the associated 'col' argument or a simulx project and the name of the output 'outputVariableName"
`number`	the number of intervals (i.e. the number of percentiles minus 1).
`level`	the largest interval (i.e. the difference between the lowest and the highest percentile).
`plot`	if TRUE the empirical distribution is displayed, if FALSE the values are returned
`color`	colors to be used for the plots In case of one group or facet = TRUE, only the first color will be used

group	variable to be used for defining groups (by default, 'group' is used when it exists)
facet	makes subplots for different groups if TRUE
labels	vector of strings
band	is deprecated (use number and level instead) ; a list with two fields • number the number of intervals (i.e. the number of percentiles minus 1). • level the largest interval (i.e. the difference between the lowest and the highest percentile).

Details

See http://simulx.webpopix.org/mlxr/prctilemlx/ for more details.

Value

a ggplot object if plot=TRUE ; otherwise, a list with fields:

- proba a vector of probabilities of length band$number+1
- color a vector of colors used for the plot of length band$number
- y a data frame with the values of the empirical percentiles computed at each time point

Examples

```
## Not run:
  myModel <- inlineModel("
  [LONGITUDINAL]
  input = {ka, V, Cl}
  EQUATION:
  C = pkmodel(ka,V,Cl)

  [INDIVIDUAL]
  input = {ka_pop, V_pop, Cl_pop, omega_ka, omega_V, omega_Cl}
  DEFINITION:
  ka = {distribution=lognormal, reference=ka_pop, sd=omega_ka}
  V  = {distribution=lognormal, reference=V_pop,  sd=omega_V }
  Cl = {distribution=lognormal, reference=Cl_pop, sd=omega_Cl}
  ")

  N=2000

  pop.param   <- c(
    ka_pop  = 1,    omega_ka  = 0.5,
    V_pop   = 10,   omega_V   = 0.4,
    Cl_pop  = 1,    omega_Cl  = 0.3)

  res <- simulx(model     = myModel,
                parameter = pop.param,
                treatment = list(time=0, amount=100),
                group     = list(size=N, level='individual'),
                output    = list(name='C', time=seq(0,24,by=0.1)))
```

```
  # res$C is a data.frame with 2000x241=482000 rows and 3 columns
  head(res$C)
  # we can display the empirical percentiles of C using the default
  # settings (i.e. percentiles of order 10%, 20%, ... 90%)
  prctilemlx(res$C)
  # The 3 quartiles (i.e. percentiles of order 25%, 50% and 75%) are displayed by
  # selecting a 50% interval splitted into 2 subintervals
  prctilemlx(res$C, number=2, level=50)
  # A one 90% interval can be displayed using only one interval
  prctilemlx(res$C, number=1, level=90)
  # or 75 subintervals in order to better represent the continuous distribution
  # of the data within this interval
  prctilemlx(res$C, number=75, level=90)
  # prctilemlx produces a ggplot object that can be modified
  pl <- prctilemlx(res$C, number=4, level=80)
  pl + ylab("concentration") + ggtitle("predictive distribution")
  # The percentiles are not plotted by setting plot=FALSE
  pr.out <- prctilemlx(res$C, number=4, level=80, plot=FALSE)
  print(pr.out$proba)
  print(pr.out$color)
  print(pr.out$y[1:5,])

## End(Not run)
```

`read.vector` *Reads a table into a vector*

Description

Reads a table into a vector

Usage

```
read.vector(f, header = FALSE, sep = "", quote = "\"'")
```

Arguments

`f`	: path to table file
`header`	: bool, use the header or not
`sep`	: the separator
`quote`	: the quote character

Value

the vector

rssimulxDemo.model *Simulx project*

Description

Model definition corresponding to rssimulxDemo.smlx project

Usage

```
rssimulxDemo.model
```

Format

A vector of string

Source

Simulx model

rssimulxDemo.project *Simulx project*

Description

rssimulxDemo.smlx is a Simulx project. In this demo three groups with different dose levels are simulated: low, medium and high. Groups have the same number of individuals, population parameters, distribution of covariates and outputs.

Usage

```
rssimulxDemo.project
```

Format

A vector of string

Source

Simulx project

shinymlx *Automatic code generation for Shiny applications*

Description

Creates a Shiny application for longitudinal data model

Usage

```
shinymlx(
  model,
  parameter = NULL,
  output = NULL,
  treatment = NULL,
  regressor = NULL,
  group = NULL,
  data = NULL,
  appname,
  style = "basic",
  settings = NULL,
  title = " "
)
```

Arguments

model	a Mlxtran model used for the simulation
parameter	a vector, or a list of shiny widgets
output	a list - or a list of lists - with fields: • name: a vector of output names • time: a vector of times, or a vector (min, max, step)
treatment	a list with fields • tfd : first time of dose, • amount : amount, • nd : number of doses, • ii : interdose interval, • type : the type of input, Input argument of Simulx can also be used, i.e. a list with fields time, amount, rate, tinf, type, target.
regressor	a list, or a list of lists, with fields • name : a vector of regressor names, • time : a vector of times, • value : a vector of values.
group	a list, or a list of lists, with fields:

- `size` : size of the group (default=1),
- `level` : level(s) of randomization,
- `parameter` : if different parameters per group are defined,
- `output` : if different outputs per group are defined,
- `treatment` : if different treatements per group are defined,
- `regressor` : if different regression variables per group are defined.

`data` a datafile to display with the plot

`appname` the name of the application (and possibly its path)

`style` the style of the Shiny app

- `"basic"`: basic Shiny app with a single side bar (default)
- `"navbar1"`: navigation bar and tabPanels (including outputs)
- `"navbar2"`: navigation bar and tabPanels (outputs separated)
- `"dashboard1"` : Shiny dashboard,

`settings` a list of settings

- `"tabstyle"` : look of the tabs c("tabs","pills"),
- `"select.x"` : display the list of variables available for the x-axis c(TRUE,FALSE),
- `"select.y"` : display the list of variables available for the y-axis c(TRUE,FALSE),
- `"select.log"` : log scale option c(TRUE,FALSE),
- `"select.ref"` : reference curves option c(TRUE,FALSE)

`title` the title of the application

Details

shinymlx automatically generates files ui.R and server.R required for a Shiny application.

Elements of `parameters` and `treatment` can be either scalars or lists. A scalar automatically generates a slider with default minimum and maximum values and default step. A list may contain the type of widget ("slider", "select", "numeric") and the settings defining the widget: (value, min, max, step) for `slider`, (selected, choices) for `select` and value for `numeric`.

See http://simulx.webpopix.org/mlxr/shinymlx/ for more details.

Value

A Shiny app with files ui.R, server.R and model.txt

Examples

```
## Not run:
PKPDmodel <- inlineModel("
[LONGITUDINAL]
input={ka,V,Cl,Imax,IC50,S0,kout}
EQUATION:
C     = pkmodel(ka, V, Cl)
E_0   = S0
ddt_E = kout*((1-Imax*C/(C+IC50))*S0- E)
")
```

```
p1  <- c(ka=0.5, V=10, Cl=1)
p2  <- c(Imax=0.5, IC50=0.03, S0=100, kout=0.1)
adm <- list(tfd=5, nd=15, ii=12, amount=1)
f1  <- list(name = 'C', time = seq(0, 250, by=1))
f2  <- list(name = 'E', time = seq(0, 250, by=1))
f   <- list(f1, f2)

shinymlx(model=PKPDmodel, treatment=adm, parameter=list(p1,p2), output=f,
         style="dashboard1", appname=tempdir())

#-----------------------------------------------------------------------
p1 <- list(
  ka    = list(widget="slider", value=0.5, min=0.1, max=2,  step=0.1),
  V     = list(widget="slider", value=10,  min=2,   max=20, step=2),
  Cl    = list(widget="slider", value=1,   min=0.1, max=2,  step=0.1)
)
adm <- list(
  tfd    = list(widget="slider", value=5,  min=0, max=100, step=5),
  nd     = list(widget="numeric", value=15),
  ii     = list(widget="select", selected=12, choices=c(3,6,12,18,24)),
  amount = list(widget="slider", value=40, min=0, max=50, step=5)
)
s <- list(select.x=FALSE, select.y=FALSE)
shinymlx(model=PKPDmodel, treatment=adm, parameter=list(p1,p2), output=f,
         style="navbar1", settings=s, appname=tempdir())

## End(Not run)
```

`simpopmlx` *Population parameters simulation*

Description

Draw population parameters using the covariance matrix of the estimates

Usage

```
simpopmlx(
  n = 1,
  project = NULL,
  fim = NULL,
  parameter = NULL,
  corr = NULL,
  kw.max = 100,
  outputFilename = NULL,
  sep = ",",
  seed = NULL
)
```

Arguments

n	(*integer*) the number of vectors of population parameters (default = 1),
project	(*string*) a Monolix project, assuming that the Fisher information Matrix was estimated by Monolix.
fim	the (*string*) Fisher Information Matrix estimated by Monolix. fim = c("sa", "lin") (default="sa")
parameter	(*data.frame*) a data frame with the following columns • pop.param (no default) population parameters • sd (no default) standard deviation of the distribution • trans (default ='N') distribution (N: normal, L: logNormal, G: logitnormal, P: probitnormal, R) • lim.a: lower bound of logit distribution (if trans != G set lim.a to NA) • lim.b: upper bound of logit distribution (if trans != G set lim.b to NA) Only when project is not used.
corr	(*matrix*) correlation matrix of the population parameters (default = identity). Only when project is not used.
kw.max	(*integer*) maximum number of trials for generating a positive definite covariance matrix (default = 100)
outputFilename	(*string*) when defined, path where the population parameters dataframe will be saved It must be a a file with a csv or txt extension. If no extension is specified, file will be saved by default in csv format
sep	(*string*) file separator when outputFilename is defined (default = ",")
seed	(*integer*) initialization of the random number generator (integer) (by default a random seed will be generated)

Details

See http://simulx.webpopix.org/mlxr/simpopmlx/ for more details.

Value

dataframe object with generated population parameters

Examples

```
## Not run:
param <- data.frame(pop.param=c(1.5, 0.5, 0.02, 0.4, 0.15, 0.2, 0.7),
                    sd=c(0.2, 0.05, 0.004, 0.05, 0.02, 0.02, 0.05),
                    trans=c('N','N','N','L','L','L','N'))
pop <- simpopmlx(n=3, parameter=param)

## End(Not run)
```

simulx *Simulation of mixed effects models and longitudinal data*

Description

Compute predictions and sample data from Mlxtran and R models

Usage

```
simulx(
  model = NULL,
  parameter = NULL,
  covariate = NULL,
  output = NULL,
  treatment = NULL,
  regressor = NULL,
  occasion = NULL,
  varlevel = NULL,
  group = NULL,
  project = NULL,
  nrep = 1,
  npop = NULL,
  fim = NULL,
  saveSmlxProject = NULL,
  result.file = NULL,
  addlines = NULL,
  settings = NULL
)
```

Arguments

model	a Mlxtran model used for the simulation. It can be a text file or an outut of the inLine function.
parameter	One of • a vector of parameters with their names and values, • a dataframe with parameters defined for each id or each pop • a string, path to a data frame (csv or txt file) • a string corresponding to the parameter elements automatically generated by monolix (only when simulation is based on a Monolix project). One of the following mlx parameter elements: "mlx_Pop", "mlx_PopUncertainSA", "mlx_PopUncertainLin", "mlx_PopIndiv", "mlx_PopIndivCov","mlx_CondMean", "mlx_EBEs","mlx_CondDistSample"
covariate	One of • a vector of covariates with their names and values. • a dataframe with covariates defined for each id

- a string, path to a data frame (csv or txt file)
- a string corresponding to the covariate elements automatically generated by monolix (only when simulation is based on a Monolix project). One of the following mlx covariate elements: "mlx_Cov" and "mlx_CovDist"

`output` output or list of outputs. An output can be defined by

- a string corresponding to the output elements automatically generated by monolix (only when simulation is based on a Monolix project) - the format is mlx_nameofoutput.
- a list with fields
  - `name`: a vector of output names
  - `time`:
    - a vector of times
    - a dataframe with columns id, time (columns lloq, uloq, limit are optional)
    - a string, path to a data frame (csv or txt file)
  - `lloq`: lower limit of quantification (when time is a vector of times)
  - `uloq`: upper limit of quantification (when time is a vector of times)
  - `limit`: lower bound of the censoring interval (when time is a vector of times)

`treatment` treatment or list of treatments. A treatment can be defined by

- A list with fields
  - `time` : a vector of input times,
  - `amount` : a scalar or a vector of amounts,
  - `rate` : a scalar or a vector of infusion rates (default=`Inf`),
  - `tinf` : a scalar or a vector of infusion times (default=0),
  - `washout` : a scalar or a vector of boolean (default=F),
  - `adm` : (or type) the administration type (default=1),
  - `repeats` : the treatment cycle (optional), a vector with fields
    - `cycleDuration` : the duration of a cycle,
    - `NumberOfRepetitions` : the number of cycle repetition
  - `probaMissDose`: the probability to miss each dose (optional).
- a dataframe with treatments defined for each id
- a string, path to a data frame (csv or txt file)
- a string corresponding to the treatment elements automatically generated by monolix (only when simulation is based on a Monolix project) - for example mlx_Adm1.

`regressor` treatment or list of treatments. A treatment can be defined by

- A list with fields
  - `name` : a vector of regressor names,
  - `time` : a vector of times,
  - `value` : a vector of values.
- A dataframe with columns id, time, and regressor values

- a string, path to a data frame (csv or txt file)

`occasion` An occasion can be defined by

- A list with fields
  - `name` : name of the variable which defines the occasions,
  - `time` : a vector of times (beginnings of occasions)
- A dataframe with columns id, time, occasion
- a string, path to a data frame (csv or txt file)
- "none", to delete an occasion structure

`varlevel` deprecated, use occasion instead.

`group` a list, or a list of lists, with fields:

- `size` : size of the group (default=1),
- `parameter` : if different parameters per group are defined,
- `covariate` : if different covariates per group are defined,
- `output` : if different outputs per group are defined,
- `treatment` : if different treatments per group are defined,
- `regressor` : if different regression variables per group are defined.

"level" field is not supported anymore in RsSimulx.

`project` the name of a Monolix project

`nrep` Samples with or without uncertainty depending which element is given as "parameter".

`npop` deprecated, Set parameter = "mlx_popUncertainSA" or "mlx_popUncertainLin" instead.

`fim` deprecated, Set parameter = "mlx_popUncertainSA" or "mlx_popUncertainLin" instead.

`saveSmlxProject`

If specified, smlx project will be save in the path location (by default smlx project is not saved)

`result.file` deprecated

`addlines` a list with fields:

- `formula`: string, or vector of strings, to be inserted .

"section", "block" field are not supported anymore in RsSimulx. You only need to specify a formula. The additional lines will be added in a new section EQUATION.

`settings` a list of optional settings

- `seed`: initialization of the random number generator (integer) (by default a random seed will be generated)
- `id.out`: add (TRUE) / remove (FALSE) columns id and group when only one element (N = 1 or group = 1) (default=FALSE)
- `kw.max`: deprecated.
- `replacement` : deprecated, use samplingMethod instead
- `samplingMethod`: str, Sampling method used for the simulation. One of "keepOrder", "withReplacement", "withoutReplacement" (default "keepOrder")

- out.trt: TRUE/FALSE (default = TRUE) output of simulx includes treatment
- out.reg: TRUE/FALSE (default = TRUE) output of simulx includes regressors
- sharedIds: Vector of Elements that share ids. Available types are "covariate", "output", "treatment", "regressor", "population", "individual" (default c())
- sameIndividualsAmongGroups: boolean, if True same individuals will be simulated among all groups (default False)
- exportData: boolean, if True and if a path to save the smlx project (saveSmlxProject) is specified, export the simulated dataset (smlx project directory/Simulation/simulatedData.txt) (default False)
- regressorInterpolationMethod: interpolation method for missing regressors. One of "lastCarriedForward" (default), "linearInterpolation"

Details

simulx takes advantage of the modularity of hierarchical models for simulating different components of a model: models for population parameters, individual covariates, individual parameters and longitudinal data.

Furthermore, simulx allows to draw different types of longitudinal data, including continuous, count, categorical, and time-to-event data.

The models are encoded using either the model coding language 'Mlxtran'. 'Mlxtran' models are automatically converted into C++ codes, compiled on the fly and linked to R using the 'RJSONIO' package. That allows one to implement very easily complex models and to take advantage of the numerical sovers used by the C++ 'mlxLibrary'.

See http://simulx.lixoft.com for more details.

Value

A list of data frames. Each data frame is an output of simulx

Examples

```
## Not run:
myModel <- inlineModel("
[LONGITUDINAL]
input = {A, k, c, a}
EQUATION:
t0    = 0
f_0   = A
ddt_f = -k*f/(c+f)
DEFINITION:
y = {distribution=normal, prediction=f, sd=a}
[INDIVIDUAL]
input = {k_pop, omega}
DEFINITION:
k = {distribution=lognormal, prediction=k_pop, sd=omega}
")
```

```
f <- list(name='f', time=seq(0, 30, by=0.1))
y <- list(name='y', time=seq(0, 30, by=2))
parameter <- c(A=100, k_pop=6, omega=0.3, c=10, a=2)

res <- simulx(model     = myModel,
              parameter = parameter,
              occasion  = data.frame(time=c(0, 0), occ=c(1, 2)),
              output    = list(f,y),
              group     = list(size=4),
              saveSmlxProject = "./project.smlx")

res <- simulx(model     = myModel,
              parameter = parameter,
              occasion  = data.frame(time = c(0, 0, 0, 0),
                                     occ1 = c(1, 1, 2, 2),
                                     occ2 = c(1, 2, 3, 4)),
              output    = list(f,y),
              group     = list(size=4))

res <- simulx(model     = myModel,
              parameter = parameter,
              output    = list(f,y),
              group     = list(size=4))

plot(ggplotmlx() + geom_line(data=res$f, aes(x=time, y=f, colour=id)) +
     geom_point(data=res$y, aes(x=time, y=y, colour=id)))
print(res$parameter)


## End(Not run)
```

`statmlx` *Summary of data*

Description

Compute statistical summaries (mean, quantile, variance, survival rate,...)

Usage

```
statmlx(r, FUN = "mean", probs = c(0.05, 0.5, 0.95), surv.time = NULL)
```

Arguments

`r`	a data frame
`FUN`	a string, or a vector of strings, with the name of the functions to apply to the result of the simulation

probs	a vector of quantiles between 0 and 1. Only used when "quantile" has been defined in FUN
surv.time	a scalar or a vector of times. Only used when "event" has been defined in type

Details

See http://simulx.webpopix.org/stamlx for more details.

Value

A data frame.

Examples

```
## Not run:
modelPK <- inlineModel("
[LONGITUDINAL]
input={V,Cl,alpha, beta,b}

EQUATION:
C = pkmodel(V, Cl)
h = alpha*exp(beta*C)
g = b*C

DEFINITION:
y = {distribution=normal, prediction=C, sd=g}
e = {type=event, maxEventNumber=1, rightCensoringTime=30, hazard=h}

[INDIVIDUAL]
input={V_pop,Cl_pop,omega_V,omega_Cl}

DEFINITION:
V     = {distribution=lognormal,   prediction=V_pop,    sd=omega_V}
Cl    = {distribution=lognormal,   prediction=Cl_pop,   sd=omega_Cl}
")

adm  <- list(amount=100, time=0)
p <- c(V_pop=10, Cl_pop=1, omega_V=0.2, omega_Cl=0.2, alpha=0.02, beta=0.1, b=0.1)
out.y <- list(name=c('y'), time=seq(0,to=25,by=5))
out.e <- list(name=c('e'), time=0)
out   <- list(out.y, out.e)
g <- list(size=100)
res1 <- simulx(model=modelPK, treatment=adm, parameter=p, output=out, group=g)

statmlx(res1$parameter, FUN = "mean", probs = c(0.05, 0.5, 0.95))
statmlx(res1$parameter, FUN = "quantile", probs = c(0.05, 0.5, 0.95))
statmlx(res1$parameter, FUN = c("sd", "quantile"), probs = c(0.05, 0.95))
statmlx(res1$y, FUN = c("mean", "sd", "quantile"), probs = c(0.05, 0.95))
statmlx(res1$e, surv.time=c(10,20))

res2 <- simulx(model=modelPK, treatment=adm, parameter=p, output=out, group=g, nrep=3)
statmlx(res2$parameter, FUN = c("sd", "quantile"), probs = c(0.05, 0.95))
```

```
statmlx(res2$y, FUN = c("mean", "sd", "quantile"), probs = c(0.05, 0.95))
statmlx(res2$e, surv.time=c(10,20,30))

## End(Not run)
```

Index