

Package ‘RsSimulx’

April 8, 2021

Type Package

Title R Speaks 'Simulx'

Version 1.0.1

Maintainer Clemence Pinaud <clemence.pinaud@lixoft.com>

Description Provide 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 pretilemlx.R simulxR.R
writeData.R exposure.R shinymlx.R statmlx.R stoolsmplx.R
simpop.R smlx-checks.R smlx-tools.R smlx-data.R smlx-project.R
apiTools.R apiManager.R utils.R data.R zzz.R

LazyData true

License BSD_2_clause + file LICENSE

Copyright LIXOFT

RoxygenNote 7.1.1.9000

Suggests testthat

NeedsCompilation no

Author Clemence Pinaud [aut, cre],
Jonathan Chauvin [aut],
Marc Lavielle [ctb]

Repository CRAN

Date/Publication 2021-04-08 13:20:02 UTC

R topics documented:

catplotmlx	2
exposure	4
ggplotmlx	6
initRsSimulx	6
inlineDataFrame	7
inlineModel	8
kmplotmlx	9
lixoft.read.table	10
prctilemlx	11
read.vector	13
rssimulxDemo.model	14
rssimulxDemo.project	14
shinymlx	15
simpopmlx	17
simulx	19
statmlx	22
writeData	24
Index	26

catplotmlx	<i>Plot Categorical Longitudinal Data</i>
------------	---

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:

	<ul style="list-style-type: none"> • 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.webpobox.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$id), a=rep(c(6,10,6,10),each=Ng),
                                     b=rep(c(0.2,0.2,0.4,0.4),each=Ng))
catplotmlx(res$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: <ul style="list-style-type: none"> • 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 steady-state (default=0.01) • ngc: number of doses used for estimating the convergence rate to steady-state (default=5)
group	a list, or a list of lists, with fields: <ul style="list-style-type: none"> • size : size of the group (default=1), • level : level(s) of randomization,

	<ul style="list-style-type: none"> • <code>parameter</code> : if different parameters per group are defined, • <code>output</code> : if different outputs per group are defined, • <code>treatment</code> : if different treatments per group are defined, • <code>regressor</code> : if different regression variables per group are defined.
<code>treatment</code>	<p>a list with fields</p> <ul style="list-style-type: none"> • <code>tfd</code> : time of first dose (default=0), • <code>ii</code> : inter dose interval (mandatory), • <code>amount</code> : the amount for each dose, • <code>rate</code> : the infusion rate (default=Inf), • <code>tin</code> : the time of infusion (default=0), • <code>type</code> : the type of input (default=1), • <code>target</code> : the target compartment (default=NULL).
<code>parameter</code>	a vector of parameters with their names and values
<code>data</code>	a list
<code>project</code>	the name of a Monolix project
<code>settings</code>	<p>a list of optional settings</p> <ul style="list-style-type: none"> • <code>result.file</code> : name of the datafile where the simulated data is written (string), • <code>seed</code> : initialization of the random number generator (integer), • <code>load.design</code> : TRUE/FALSE (if <code>load.design</code> is not defined, a test is automatically performed to check if a new design has been defined), • <code>data.in</code> : TRUE/FALSE (default=FALSE) • <code>id.out</code> : add columns <code>id</code> (when <code>N=1</code>) and <code>group</code> (when <code>#group=1</code>), TRUE/FALSE (default=FALSE) • <code>Nmax</code> : maximum group size used in a single call of <code>mlxCompute</code> (default=100)
<code>regressor</code>	<p>a list, or a list of lists, with fields</p> <ul style="list-style-type: none"> • <code>name</code> : a vector of regressor names, • <code>time</code> : a vector of times, • <code>value</code> : a vector of values.
<code>varlevel</code>	<p>a list, or a list of lists, with fields</p> <ul style="list-style-type: none"> • <code>name</code> : a vector of names of variability levels, • <code>time</code> : 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).

<code>ggplotmlx</code>	<i>wrapper for ggplot</i>
------------------------	---------------------------

Description

wrapper around [ggplot](#) with a custom theme

Usage

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

Arguments

... parameters passed to [ggplot](#)

Value

see [ggplot](#)

<code>initRsSimulx</code>	<i>Initialize RsSimulx library</i>
---------------------------	------------------------------------

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 monolix with Rsmlx)
- 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	<i>Inline dataframe</i>
-----------------	-------------------------

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	<i>Inline model</i>
-------------	---------------------

Description

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

Usage

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

Arguments

srtIn	(<i>string</i>) Model in string format,
filename	(<i>string</i>) 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

<code>r</code>	a data frame with a column 'id', a column 'time', a column with values and possibly a column 'group'.
<code>index</code>	an integer: <code>index=k</code> means that the survival function for the k-th event is displayed. Default is <code>index=1</code> .
<code>level</code>	a number between 0 and 1: confidence interval level.
<code>time</code>	a vector of time points where the survival function is evaluated.
<code>cens</code>	if TRUE right censoring times are displayed.
<code>plot</code>	if TRUE the estimated survival function is displayed, if FALSE the values are returned
<code>color</code>	color to be used for the plots (default="#e05969")
<code>group</code>	variable to be used for defining groups (by default, 'group' is used when it exists)
<code>facet</code>	makes subplots for different groups if TRUE
<code>labels</code>	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))
p11 <- kmplotmlx(res1$e, level=0.95)
print(p11)

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))
p12 <- kmplotmlx(res2$e)
print(p12)

## End(Not run)
```

lixoft.read.table *Read Lixoft@ files*

Description

Utility function to read Lixoft@ formatted 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	<i>Percentiles of the empirical distribution of longitudinal data</i>
------------	---

Description

Compute and display percentiles of the empirical 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 <ul style="list-style-type: none"> • 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/prtilemlx/> 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

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 = "shinymlxApp",
  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: <ul style="list-style-type: none"> • name: a vector of output names • time: a vector of times, or a vector (min, max, step)
treatment	a list with fields <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • 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:

	<ul style="list-style-type: none"> • <code>size</code> : size of the group (default=1), • <code>level</code> : level(s) of randomization, • <code>parameter</code> : if different parameters per group are defined, • <code>output</code> : if different outputs per group are defined, • <code>treatment</code> : if different treatments per group are defined, • <code>regressor</code> : if different regression variables per group are defined.
<code>data</code>	a datafile to display with the plot
<code>appname</code>	the name of the application (and possibly its path)
<code>style</code>	the style of the Shiny app <ul style="list-style-type: none"> • <code>"basic"</code>: basic Shiny app with a single side bar (default) • <code>"navbar1"</code>: navigation bar and tabPanels (including outputs) • <code>"navbar2"</code>: navigation bar and tabPanels (outputs separated) • <code>"dashboard1"</code> : Shiny dashboard,
<code>settings</code>	a list of settings <ul style="list-style-type: none"> • <code>"tabstyle"</code> : look of the tabs c("tabs","pills"), • <code>"select.x"</code> : display the list of variables available for the x-axis c(TRUE,FALSE), • <code>"select.y"</code> : display the list of variables available for the y-axis c(TRUE,FALSE), • <code>"select.log"</code> : log scale option c(TRUE,FALSE), • <code>"select.ref"</code> : reference curves option c(TRUE,FALSE)
<code>title</code>	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,C1,Imax,IC50,S0,kout}
EQUATION:
C      = pkmodel(ka, V, C1)
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")

#-----
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)

## 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	(<i>integer</i>) the number of vectors of population parameters (default = 1),
project	(<i>string</i>) a Monolix project, assuming that the Fisher information Matrix was estimated by Monolix.
fim	the (<i>string</i>) Fisher Information Matrix estimated by Monolix. fim="sa", "lin" (default="sa")
parameter	(<i>data.frame</i>) a data frame with the following columns <ul style="list-style-type: none"> • 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	(<i>matrix</i>) correlation matrix of the population parameters (default = identity). Only when project is not used.
kw.max	(<i>integer</i>) maximum number of trials for generating a positive definite covariance matrix (default = 100)
outputFilename	(<i>string</i>) when defined, path where the population parameters dataframe will be saved It must be a file with a csv or txt extension. If no extension is specified, file will be saved by default in csv format
sep	(<i>string</i>) file separator when outputFilename is defined (default = ",")
seed	(<i>integer</i>) initialization of the random number generator (<i>integer</i>) (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,
  output = NULL,
  treatment = NULL,
  regressor = NULL,
  varlevel = NULL,
  group = NULL,
  project = NULL,
  nrep = 1,
  npop = NULL,
  fim = NULL,
  result.folder = NULL,
  result.file = NULL,
  stat.f = "statmlx",
  addlines = NULL,
  settings = NULL,
  data = NULL
)
```

Arguments

- | | |
|-----------|--|
| model | a Mlxtran model used for the simulation. It can be a text file or an output of the <code>inLine</code> function. You can now create a new simulx project providing a model using <code>newProject</code> <code>lixoft</code> simulx connector. |
| parameter | a vector of parameters with their names and values You can now define a new individual element using <code>defineIndividualElement</code> <code>lixoft</code> simulx connector and a new population element with <code>'definePopulationElement'</code> <code>lixoft</code> simulx connector. |
| output | a list (or list of lists) with fields: <ul style="list-style-type: none"> • name: a vector of output names • time: a vector of times (only for the longitudinal outputs) • lloq: lower limit of quantification (only for the longitudinal outputs) • uloq: upper limit of quantification (only for the longitudinal outputs) • limit: lower bound of the censoring interval (only for the longitudinal outputs) |

You don't need to add individual parameters in output anymore since there are now automatically outputed. You can now define a new output element using `defineOutputElement` lixoft simulx connector.

<code>treatment</code>	<p>a list with fields</p> <ul style="list-style-type: none"> • <code>time</code> : a vector of input times, • <code>amount</code> : a scalar or a vector of amounts, • <code>rate</code> : a scalar or a vector of infusion rates (default=Inf), • <code>timeinf</code> : a scalar or a vector of infusion times (default=0), • <code>type</code> : the type of input (default=1). <p>"target" field is not supported anymore in RsSimulx. You can now define a new treatment element using <code>defineTreatmentElement</code> lixoft simulx connector.</p>
<code>regressor</code>	<p>a list, or a list of lists, with fields</p> <ul style="list-style-type: none"> • <code>name</code> : a vector of regressor names, • <code>time</code> : a vector of times, • <code>value</code> : a vector of values. <p>You can now define a new regressor element using <code>defineOccasionElement</code> lixoft simulx connector.</p>
<code>varlevel</code>	<p>a list (or a dataframe) with fields:</p> <ul style="list-style-type: none"> • <code>name</code> : name of the variable which defines the occasions, • <code>time</code> : a vector of times (beginnings of occasions) <p>You can now define occasion levels using <code>defineOccasionElement</code> lixoft simulx connector.</p>
<code>group</code>	<p>a list, or a list of lists, with fields:</p> <ul style="list-style-type: none"> • <code>size</code> : size of the group (default=1), • <code>parameter</code> : if different parameters per group are defined, • <code>output</code> : if different outputs per group are defined, • <code>treatment</code> : if different treatments per group are defined, • <code>regressor</code> : if different regression variables per group are defined. <p>"level" field is not supported anymore in RsSimulx. You can now define a new simulation group using <code>addGroup</code> lixoft simulx connector and use connectors <code>setGroupElement</code>, <code>setGroupSize</code>, to define groups attributes.</p>
<code>project</code>	<p>the name of a Monolix project You can now import all elements coming from a Monolix project to perform simulations using <code>importMonolixProject</code> lixoft simulx connector.</p>
<code>nrep</code>	<p>number of replicates You can now define the number of replicates if the simulation using <code>setNbReplicates</code> lixoft simulx connector.</p>
<code>npop</code>	<p>number of population parameters to draw randomly</p>
<code>fim</code>	<p>a string with the Fisher Information Matrix to be used</p>
<code>result.folder</code>	<p>the name of the folder where the outputs of simulx should be stored <code>result.folder</code> argument is deprecated</p>
<code>result.file</code>	<p>the name of the single file where the outputs of simulx should be saved</p>

<code>stat.f</code>	a R function for computing some summary (mean, quantiles, survival,...) of the simulated data. Default = "statmlx". <code>stat.f</code> argument is deprecated
<code>addlines</code>	a list with fields: <ul style="list-style-type: none"> • <code>formula</code>: string, or vector of strings, to be inserted . <p>"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. You can now add lines to the model file using <code>setAddLines</code> lixoft simulx connector.</p>
<code>settings</code>	a list of optional settings <ul style="list-style-type: none"> • <code>seed</code> : initialization of the random number generator (integer) (by default a random seed will be generated) • <code>id.out</code> : add (TRUE) / remove (FALSE) columns <code>id</code> and <code>group</code> when only one element (<code>N = 1</code> or <code>group = 1</code>) (default=FALSE) • <code>kw.max</code> : maximum number of trials for generating a positive definite covariance matrix (default = 100) • <code>sep</code> : the field separator character (default = ",") • <code>digits</code> : number of decimal digits in output files (default = 5) • <code>replacement</code> : TRUE/FALSE (default = FALSE) sample id's with/without replacement • <code>out.trt</code> : TRUE/FALSE (default = TRUE) output of simulx includes treatment
<code>data</code>	a list (output of simulx when <code>settings\$data.in==TRUE</code>) <code>data</code> argument is deprecated

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 solvers 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))
res <- simulx(model      = myModel,
              parameter = c(A=100, k_pop=6, omega=0.3, c=10, a=2),
              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,C1,alpha, beta,b}

EQUATION:
C = pkmodel(V, C1)
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,C1_pop,omega_V,omega_C1}

DEFINITION:
V     = {distribution=lognormal, prediction=V_pop, sd=omega_V}
C1    = {distribution=lognormal, prediction=C1_pop, sd=omega_C1}
")

adm <- list(amount=100, time=0)
p <- c(V_pop=10, C1_pop=1, omega_V=0.2, omega_C1=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)
```

writeData	<i>Write Simulx Dataset</i>
-----------	-----------------------------

Description

Format outputs of simulx simulations and write datasets in monolix and pkanalix project format.

Usage

```
writeData(
  project = NULL,
  filename = "simulated_dataset.csv",
  sep = ",",
  ext = "csv",
  nbdigits = 5
)
```

Arguments

project	(<i>string</i>) a simulx project. If no project specified, the function will run on the project that is already loaded.
filename	(<i>string</i>) (<i>optional</i>) file path to dataset. (default "simulated_dataset.csv") In case of multiple replicates, the function creates one dataset per replicate with name \$filename_repi If filename contains an extension, it must be "csv" or "txt". If it does not, extension is defined by ext argument.
sep	(<i>string</i>) (<i>optional</i>) Separator used to write dataset file. (default ",") It must be one of "\t", " ", ";", ","
ext	(<i>bool</i>) (<i>optional</i>) Extension used to write dataset file. (default "csv") It must be one of "csv", "txt" To defined only if filename with no extension
nbdigits	(<i>integer</i>) (<i>optional</i>) number of decimal digits in output file. (default = 5)

Details

WARNING: writeData function is not implemented for simulx project with regressors in MonolixSuite version 2020R1

Value

a dataframe if one single simulation, a list of dataframe if multiple replicates.

Examples

```
## Not run:
# rssidemo.smlx is a Simulx project. This demo simulates three groups
# with different dose levels: low, medium and high.
# Groups have the same number of individuals, population parameters,
# distribution of covariates and outputs.
```



```
# In this example we write data in the home directory with the name demo_simulx
# and a txt extension
writeData("rssimulxDemo.smlx", filename = "demo_simulx", ext = "txt")

## End(Not run)
```

Index

* datasets

rssimulxDemo.model, 14
rssimulxDemo.project, 14

catplotmlx, 2

exposure, 4

ggplot, 6
ggplotmlx, 6

initRsSimulx, 6
inlineDataFrame, 7
inlineModel, 8

kmplotmlx, 9

lixoft.read.table, 10

prctilemlx, 11

read.table, 10
read.vector, 13
rssimulxDemo.model, 14
rssimulxDemo.project, 14

shinymlx, 15
simpopmlx, 17
simulx, 19
statmlx, 22

writeData, 24