

Package ‘mlxR’

October 11, 2019

Type Package

Version 4.1.0

Title Simulation of Longitudinal Data

Description Simulation and visualization of complex models for longitudinal data. The models are encoded using the model coding language 'Mlxtran' and automatically converted into C++ codes. That allows one to implement very easily complex ODE-based models and complex statistical models, including mixed effects models, for continuous, count, categorical, and time-to-event data.

URL <http://simulx.webpopix.org>

BugReports <https://github.com/MarcLavielle/mlxR/issues>

Depends R (>= 3.0.1), ggplot2

Suggests XML, Rcpp (>= 0.11.3), reshape2, gridExtra, shiny

Imports tools, methods, graphics, grDevices, utils, stats

License BSD_2_clause + file LICENSE

Copyright Inria

RoxygenNote 6.1.1

NeedsCompilation no

Maintainer Marc Lavielle <Marc.Lavielle@inria.fr>

Collate 'apiTools.R' 'apiManager.R' 'mlxComputeRCleaner.R' 'mlxComputeRLibraryBuilder.R' 'addFieldsFromHeader.R' 'catplotmlx.R' 'commentModel.R' 'convertmlx.R' 'exposure.R' 'generateModelFromPkModel.R' 'ggplotmlx.R' 'hformat.R' 'hgdata.R' 'inlineDataFrame.R' 'inlineModel.R' 'isfield.R' 'kmpplotmlx.R' 'mlxplore.R' 'monolix2simulx.R' 'pkmodel.R' 'prctilemlx.R' 'processing_monolix.R' 'processing_setting.R' 'processing_target.R' 'readVector.R' 'readdatamlx.R' 'shinymlx.R' 'simpoplmlx.R' 'simulR.R' 'simulx.R' 'statmlx.R' 'toolsmlx.R' 'toolsmlx.R' 'translateIOV.R' 'translateIOVind.R' 'uuid.R' 'writeDatamlx.R'

Author Marc Lavielle [aut, cre],
 Esther Ilinca [ctb],
 Raphael Kuate [ctb]

Repository CRAN

Date/Publication 2019-10-11 12:20:02 UTC

R topics documented:

catplotmlx	2
exposure	4
ggplotmlx	6
initMlxR	6
inlineDataFrame	7
inlineModel	7
kmplotmlx	8
lixoft.read.table	10
mlxplore	10
monolix2simulx	11
pkmodel	12
prtilemlx	13
read.vector	15
readDatamlx	16
shinymlx	17
simpopmlx	19
simulx	20
statmlx	23
writeDatamlx	25
Index	27

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: <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.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: <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, • parameter : if different parameters per group are defined, • output : if different outputs per group are defined,

	<ul style="list-style-type: none"> • <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>tinf</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).

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

Description

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

Usage

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

Arguments

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

Value

see [ggplot](#)

initMlxR	<i>Initialize mlxR library</i>
----------	--------------------------------

Description

Initialize mlxR library

Usage

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

Arguments

path	<i>(character)</i> [optional] Path to installation directory of the Lixoft suite. If mlxR 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 mlxR is used with a version of Lixoft(/@) software suite higher or equal to 2019R1. <ul style="list-style-type: none"> force <i>(bool)</i> [optional] Should mlxR 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:  
initMlxR(path = "/path/to/lixoftRuntime/")  
  
## End(Not run)
```

inlineDataFrame	<i>inline data frame</i>
-----------------	--------------------------

Description

utility function to inline creation of a data frame

Usage

```
inlineDataFrame(str, header = TRUE, colClasses = NA, ...)
```

Arguments

str	text representation of the data frame
header	see read.table
colClasses	see read.table
...	see read.table

inlineModel	<i>inline model</i>
-------------	---------------------

Description

Define a model "inline"

Usage

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

Arguments

str	model
filename	name of the temporary model file

Details

A temporary model file filename is created. Default name is "tempModel.txt". filename="random" generates a random name.

Value

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

Examples

```
## Not run:
myModel1 <- inlineModel("
[LONGITUDINAL]
EQUATION:
f = 10*exp(-0.2*t)
")

print(myModel1)

r <- simu1x(model=myModel1, output=list(name="f", time=0:100))

myModel2 <- inlineModel("
[LONGITUDINAL]
EQUATION:
f = 10*exp(-0.2*t)
", filename="random")

print(myModel2)

## 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 displayed.
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))
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	<i>Read Lixoft@ files</i>
-------------------	---------------------------

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

mlxplora	<i>Explore and visualize models</i>
----------	-------------------------------------

Description

Explore and visualize 'Mlxtran' models with the 'Mlxplora' software.

Usage

```
mlxplora(model, parameter = NULL, output = NULL, group = NULL,
         treatment = NULL)
```

Arguments

model	a Mlxtran model
parameter	a vector of parameters with their names and values
output	a list with fields: <ul style="list-style-type: none"> • name: a vector of output names • time: a vector of times
group	a list with unique field: <ul style="list-style-type: none"> • treatment : a list,
treatment	a list with fields <ul style="list-style-type: none"> • 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), • type : the type of input (default=1), • target : the target compartment (default=NULL).

Details

See <http://simulx.webpopix.org/mlxr/mlxplore/> for more details.

monolix2simulx	<i>Convert a Monolix Project into an executable for the simulator Simulx</i>
----------------	--

Description

Convert a Monolix Project into an executable for the simulator Simulx

Usage

```
monolix2simulx(project, parameter = NULL, group = NULL, open = FALSE,
               r.data = TRUE, fim = NULL)
```

Arguments

project	: the name of a Monolix project
parameter	: string $\$(NameOfTypeOfParameter)$, the type of specific parameters to use example: "mode", "mean"...
group	: a list with the number of subjects
open	: load the R script created if open=TRUE
r.data	: read the data if r.data=TRUE
fim	: Fisher information matrix

Value

creates a folder projectNameR containing files :

- projectName.R : executable R code for the simulator,
- treatment.txt : contains the treatment informations,
- populationParameter.txt : contains the population parameters estimated from Monolix,
- individualParameter.txt : contains the individual parameters (mode/mean) estimated from Monolix (if used for the simulation),
- individualCovariate.txt : contains the individual covariates,
- originalId.txt : contains the original id's when group is used with a different size than the original one,
- outputi.txt : contains the output number i informations (time, id),
- $\$(NameOfTypeOfParameter)$ s.txt : contains the specific parameter used.

Examples

```
## Not run:
project.file <- 'monolixRuns/theophylline1_project.mlxtran' #relative path
monolix2simulx(project=project.file,open=TRUE)
monolix2simulx(project=project.file,parameter=list("mean",c(a=0, b=0)),open=TRUE)

## End(Not run)
```

pkmodel

*Easy simulation of PK models Easy simulation of basic PK models See
<http://simulx.webpopix.org/mlxr/pkmodel/> for more details.*

Description

Easy simulation of PK models

Easy simulation of basic PK models

See <http://simulx.webpopix.org/mlxr/pkmodel/> for more details.

Usage

```
pkmodel(time, treatment, parameter)
```

Arguments

time	a vector
treatment	a list with fields <ul style="list-style-type: none"> 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),
parameter	vector of parameters with their names and values

Examples

```
## Not run:
adm <- list(time=c(2,14,20), amount=40)
p <- c(V=8, Cl=0.5,k12=0.3, k21=0.2)
t <- seq(0, 30, by=0.1)

res <- pkmodel(time = t, treatment = adm, parameter = p)

print(ggplot(data=res, aes(x=time, y=cc)) + geom_line(size=1) +
  xlab("time (h)") + ylab("concentration (mg/L)"))

adm <- list(time = c(1,23,37,45), amount = c(1,0.5,2,0.3))
```

```

p <- c(Mtt=5, Ktr=1, ka=0.5, V=10, Vm=1, Km=0.6, p=0.5)
t <- seq(0, 80, by=0.1)

res <- pkmodel(t,adm,p)

print(ggplot(data=res, aes(x=time, y=cc)) + geom_line(size=1) +
      xlab("time (h)") + ylab("concentration (mg/L)"))

adm <- list( time = 2, amount = 40)

p <- inlineDataFrame("
id  ka  V  Cl
1   0.5 4   1
2    1 6   1
3   1.5 6  1.5
")

t <- seq(0, 30, by=0.1)

res <- pkmodel(t,adm,p)

print(ggplot(data=res, aes(x=time, y=cc, colour=id)) + geom_line(size=1) +
      xlab("time (h)") + ylab("concentration (mg/L)"))
adm <- list(time=seq(2, 100, by=24), amount=40, rate=5)
p <- c(V=8, Cl=0.5, k12=0.3, k21=0.2, ke0=0.2)
t <- seq(0, 50, by=0.1)

res <- pkmodel(t,adm,p)

if( require("reshape2") ){
  r <- melt(res, id='time', variable.name='c')
  print(ggplot(r, aes(time,value)) + geom_line(aes(colour = c),size=1) +
        ylab('concentration') + guides(colour=guide_legend(title=NULL)) +
        theme(legend.position=c(.9, .8)))
}

## End(Not run)

```

prctilemlx

Percentiles of the empirical distribution of longitudinal data

Description

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

Usage

```

prctilemlx(r, col = NULL, number = 8, level = 80, plot = TRUE,
  color = "#9a35ff", group = NULL, facet = TRUE, labels = NULL,
  band = NULL)

```

Arguments

<code>r</code>	a data frame with a column 'id', a column 'time' and a column with values. The times should be the same for each individual.
<code>col</code>	a vector of 3 column numbers: ('id', 'time/x', 'y'). Default = c(1, 2,3).
<code>number</code>	the number of intervals (i.e. the number of percentiles minus 1).
<code>level</code>	the largest interval (i.e. the difference between the lowest and the highest percentile).
<code>plot</code>	if TRUE the empirical distribution is displayed, if FALSE the values are returned
<code>color</code>	a color to be used for the plots (default="#9a35ff")
<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
<code>band</code>	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/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 : the table
 header : bool, use the header or not
 sep : the separator
 quote : the quote character

Value

the vector

readDatamlx	<i>Read formatted data file</i>
-------------	---------------------------------

Description

Read data in a Monolix/NONMEM format

Usage

```
readDatamlx(project = NULL, data = NULL, out.data = FALSE,
  nbSSDoses = 10, obs.rows = FALSE, datafile = NULL, header = NULL,
  infoProject = NULL, addl.ss = NULL)
```

Arguments

project a Monolix project
 data a list with fields

- datafile: path of a formatted data file
- headerTypes: a vector of strings

 out.data TRUE/FALSE (default=FALSE) returns the original data as a table and some information about the Monolix project
 nbSSDoses number of additional doses to use for steady-state (default=10)
 obs.rows a list of observation indexes
 datafile (deprecated) a formatted data file
 header (deprecated) a vector of strings
 infoProject (deprecated) an xmlfile
 addl.ss (deprecated) number of additional doses to use for steady-state (default=10)

Details

See <http://simulx.webpopix.org/mlxr/readdatamlx/> for more details.

Value

A list of data frames

Examples

```
## Not run:
# using a Monolix project:
d <- readDatamlx(project='projects/warfarinPK.mlxtran')

# using a data file:
warfarinPK <- list(dataFile = "data/warfarinPK.csv",
                  headerTypes = c("id", "time", "observation", "amount",
                                  "contcov", "contcov", "catcov"),
                  administration = "oral")
d <- readDatamlx(data=warfarinPK)

## End(Not run)
```

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 = "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,

	<ul style="list-style-type: none"> • <code>type</code> : the type of input,
	Input argument of <code>Simulx</code> can also be used, i.e. a list with fields <code>time</code> , <code>amount</code> , <code>rate</code> , <code>tinfn</code> , <code>type</code> , <code>target</code> .
<code>regressor</code>	a list, or a list of lists, with fields <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>group</code>	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 <code>c("tabs","pills")</code>, • <code>"select.x"</code> : display the list of variables available for the x-axis <code>c(TRUE,FALSE)</code>, • <code>"select.y"</code> : display the list of variables available for the y-axis <code>c(TRUE,FALSE)</code>, • <code>"select.log"</code> : log scale option <code>c(TRUE,FALSE)</code>, • <code>"select.ref"</code> : reference curves option <code>c(TRUE,FALSE)</code>
<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:
library(mlxR)
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, C1=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),
  C1  = 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)

```

simpoplmx

*Population parameters simulation***Description**

Draw population parameters using the covariance matrix of the estimates.

Usage

```

simpoplmx(n = 1, project = NULL, fim = "needed", parameter = NULL,
          corr = NULL, kw.max = 100)

```

Arguments

n	the number of vectors of population parameters (default = 1),
project	a Monolix project, assuming that the Fisher information Matrix was estimated by Monolix.
fim	the Fisher Information Matrix estimated by Monolix. fim="sa", "lin" (default="sa")
parameter	a data frame with a column 'pop.param' (no default), a column 'sd' (no default), possibly a column 'trans' (default='N') and possibly columns 'lim.a' (default=0) and 'lim.b' (default=1). Only when project is not used.
corr	correlation matrix of the population parameters (default = identity). Only when project is not used.
kw.max	maximum number of trials for generating a positive definite covariance matrix (default = 100)

Details

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

Examples

```
## Not run:
project.file <- 'monolixRuns/theophylline1_project.mlxtran' #relative path
pop1 <- simpopmlx(n=3, project=project.file)

## 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, data = NULL, project = NULL, nrep = 1,
        npop = NULL, fim = NULL, result.folder = NULL,
        result.file = NULL, stat.f = "statmlx", addlines = NULL,
        settings = NULL)
```

Arguments

model	a Mlxtran, or R model used for the simulation
parameter	a vector of parameters with their names and values
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)
treatment	a list with fields <ul style="list-style-type: none"> • 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), • type : the type of input (default=1), • target : the target compartment (default=NULL).
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.
varlevel	(IOV supported by mlxR >= 3.2.2) a list (or a dataframe) with fields: <ul style="list-style-type: none"> • name : name of the variable which defines the occasions, • time : a vector of times (beginnings of occasions) ,
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, • parameter : if different parameters 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.
data	a list (output of simulx when settings\$data.in==TRUE)
project	the name of a Monolix project
nrep	number of replicates
npop	number of population parameters to draw randomly
fim	a string with the Fisher Information Matrix to be used
result.folder	the name of the folder where the outputs of simulx should be stored
result.file	the name of the single file where the outputs of simulx should be saved
stat.f	a R function for computing some summary (mean, quantiles, survival,...) of the simulated data. Default = "statmlx".

addlines	<p>a list with fields:</p> <ul style="list-style-type: none"> • section: a string (default = "[LONGITUDINAL]"), • block: a string (default = "EQUATION:"), • formula: string, or vector of strings, to be inserted .
settings	<p>a list of optional settings</p> <ul style="list-style-type: none"> • 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) • kw.max : maximum number of trials for generating a positive definite covariance matrix (default = 100) • sep : the field separator character (default = ",") • digits : number of decimal digits in output files (default = 5) • disp.iter : TRUE/FALSE (default = FALSE) display replicate and population numbers • replacement : TRUE/FALSE (default = FALSE) sample id's with/without replacement • out.trt : TRUE/FALSE (default = TRUE) output of simulx includes treatment • format.original : TRUE/FALSE (default = FALSE) with a Monolix project, write data in result.file using the original format of the data file

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', or 'R'. 'Mlxtran' models are automatically converted into C++ codes, compiled on the fly and linked to R using the 'Rcpp' 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.webpopix.org> 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      = 'model/home.txt',
              parameter = c(A=100, k_pop=6, omega=0.3, c=10, a=2),
              output    = list(f,y,"k"),
              group      = list(size=4, level='individual'))

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.p <- c("V", "C1")
out <- list(out.y, out.e, out.p)
g <- list(size=100, level='individual')
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)
```

writeDatamlx	<i>Write formatted data file</i>
--------------	----------------------------------

Description

Write data contained in a list of dataframes in a single file (NONMEM/Monolix format) or in several files as tables

Usage

```
writeDatamlx(r, result.file = NULL, result.folder = NULL, sep = ",",
  ext = NULL, digits = 5, app.file = F, app.dir = F,
  project = NULL)
```

Arguments

r	a list of dataframes
result.file	a string with the name of the file
result.folder	a string with the name of the folder
sep	(default = ",")
ext	a string with the extension of the file names
digits	(default = 5)
app.file	TRUE/FALSE (default=FALSE) append to file
app.dir	TRUE/FALSE (default=FALSE) append to dir
project	A Monolix project

Details

See <http://simulx.webpopix.org/mlxr/writedatamlx/> for more details.

Examples

```
## Not run:
modelPK <- inlineModel("
[LONGITUDINAL]
input = {V, Cl, a1}
EQUATION:
Cc = pkmodel(V, Cl)
DEFINITION:
y1 = {distribution=lognormal, prediction=Cc, sd=a1}
")
adm <- list(amount=100, time=seq(0,50,by=12))
p <- c(V=10, Cl=1, a1=0.1)
y1 <- list(name=c('y1'), time=seq(5,to=50,by=5))
res <- simulx(model=modelPK, treatment=adm, parameter=p, output=y1)
writeDatamlx(res, result.file="res.csv")
```

```
writeDatamlx(res, result.file="res.txt", sep="\t")  
writeDatamlx(res, result.folder="res")  
  
## End(Not run)
```

Index

catplotmlx, 2

exposure, 4

ggplot, 6

ggplotmlx, 6

initMlxR, 6

inlineDataFrame, 7

inlineModel, 7

kmpplotmlx, 8

lixoft.read.table, 10

mlxplore, 10

monolix2simulx, 11

pkmodel, 12

prctilemlx, 13

read.table, 7, 10

read.vector, 15

readDatamlx, 16

shinymlx, 17

simpopmlx, 19

simulx, 20

statmlx, 23

writeDatamlx, 25