Package 'Rsmlx'

January 20, 2025

Type Package

Title R Speaks 'Monolix'

Version 2024.1.0

Maintainer Chloe Bracis <support@lixoft.com>

Description Provides methods for model building and model evaluation of mixed effects models using 'Monolix' <https://monolix.lixoft.com>. 'Monolix' is a software tool for nonlinear mixed effects modeling that must have been installed in order to use 'Rsmlx'.
Among other tasks, 'Rsmlx' provides a powerful tool for automatic PK model building, performs statistical tests for model assessment, bootstrap simulation and likelihood profiling for computing confidence intervals. 'Rsmlx' also proposes several automatic covariate search methods for
mixed effects models.

URL https://monolix.lixoft.com/rsmlx/

SystemRequirements 'Monolix' (<https://monolix.lixoft.com>)

Depends R (>= 3.0.0)

Imports graphics, grDevices, utils, stats, MASS, ggplot2, gridExtra, dplyr, tidyr

Collate mlxConnectors.R bootstrap.R buildmlx.R buildVar.R buildAll.R confintmlx.R correlationModelSelection.R covariateModelSelection.R covariateSearch.R errorModelSelection.R llp.R newConnectors.R setSettings.R testmlx.R zzz.R RsmlxTools.R readDatamlx.R pkbuild.R pkpopini.R whichPKmodel.R

License BSD_2_clause + file LICENSE

NeedsCompilation no

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

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

Repository CRAN

Date/Publication 2024-06-10 13:30:07 UTC

Contents

`bootmlx` *Bootstrapping - case resampling*

Description

Generate replicates of the original data using random sampling with replacement. Population parameters are then estimated from each replicate.

Usage

```
bootmlx(
  project,
  nboot = 100,
  dataFolder = NULL,
  parametric = FALSE,
  tasks = c(populationParameterEstimation = TRUE),
  settings = NULL
)
```

Arguments

`project`	Monolix project
`nboot`	[optional] number of bootstrap replicates (default=100)
`dataFolder`	[optional] folder where already generated datasets are stored, e.g dataFolder="./dummy_project/boot/" (default: data set are generated by bootmlx)
`parametric`	[optional] boolean to define if parametric bootstrap is performed (new data is drawn from the model), (default: false)
`tasks`	[optional] vector of booleans defining the list of tasks to perform (default: estimation of the population parameters) available tasks: populationParameterEstimation, conditionalDistributionSampling, conditionalModeEstimation, standardErrorEstimation, logLikelihoodEstimation, plots
`settings`	[optional] a list of settings for the resampling and the results: • `N` the number of individuals in each bootstrap data set (default value is the number of individuals in the original data set). • `newResampling` boolean to generate the data sets again if they already exist (default=FALSE). • `covStrat` a categorical covariate of the project. The original distribution of this covariate is maintained in each resampled data set if covStrat is defined (default=NULL). Notice that if the categorical covariate is varying within the subject (in case of IOV), it will not be taken into account. • `plot` boolean to choose if the distribution of the bootstraped esimates is displayed (default = FALSE) • `level` level of the bootstrap confidence intervals of the population parameters (default = 0.90) • `seed` a positive integer < 2147483647, seed for the generation of the data sets (default = NA) • `deleteData` delete created data set files after estimation (default = FALSE) • `deleteProjects` delete created Monolix projects after estimation (default = FALSE)

Details

Bootstrap functionality is now available directly in the lixoftConnectors package using the function `runBootstrap`. Please migrate, as this function will be deprecated in the future.

Value

a data frame with the bootstrap estimates

See Also

`getBootstrapSettings` settings for bootstrap with lixoftConnectors
`runBootstrap` run the bootstrap with lixoftConnectors
`getBootstrapResults` results for bootstrap with lixoftConnectors

Examples

```
## Not run:
# RsmlxDemo1.mlxtran is a Monolix project for modelling the PK of warfarin using a PK model
# with parameters ka, V, Cl.

# In this example, bootmlx will generate 100 random replicates of the original data and will
# use Monolix to estimate the population parameters from each of these 100 replicates:
r1 <- bootmlx(project="RsmlxDemo1.mlxtran")

# 5 replicates will now be generated, with 50 individuals in each replicate:
r2 <- bootmlx(project="RsmlxDemo1.mlxtran",  nboot = 5, settings = list(N = 50))

# Proportions of males and females in the original dataset will be preserved
# in each replicate:
r3 <- bootmlx(project="RsmlxDemo1.mlxtran",  settings = list(covStrat = "sex"))

## End(Not run)

# See http://monolix.lixoft.com/rsmlx/bootmlx/ for detailed examples of use of bootmlx
# Download the demo examples here: http://monolix.lixoft.com/rsmlx/installation
```

---

buildAll *Automatic complete statistical model building*

Description

buildAll builds the complete statistical model by iteratively calling functions buildmlx and buildVar

Penalization criterion can be either a custom penalization of the form gamma*(number of parameters), AIC (gamma=2) or BIC (gamma=log(N)).

Usage

```
buildAll(
  project = NULL,
  final.project = NULL,
  model = "all",
```

```
  prior = NULL,
  weight = NULL,
  coef.w1 = 0.5,
  cv.min = 0.001,
  fError.min = 0.001,
  paramToUse = "all",
  covToTest = "all",
  covToTransform = "none",
  center.covariate = FALSE,
  criterion = "BICc",
  linearization = FALSE,
  ll = T,
  test = T,
  direction = NULL,
  steps = 1000,
  max.iter = 20,
  explor.iter = 2,
  seq.cov = FALSE,
  seq.corr = TRUE,
  seq.cov.iter = 0,
  p.max = 0.1,
  p.min = c(0.075, 0.05, 0.1),
  print = TRUE,
  nb.model = 1,
  fix.param1 = NULL,
  fix.param0 = NULL,
  remove = T,
  add = T,
  delta = c(30, 10, 5),
  omega.set = NULL,
  pop.set1 = NULL,
  pop.set2 = NULL
)
```

Arguments

project	a string: the initial Monolix project
final.project	a string: the final Monolix project (default adds "_buildAll" to the original project)
model	components of the model to optimize c("residualError", "covariate", "correlation"), (default="all")
prior	list of prior probabilities for each component of the model (default=NULL)
weight	list of penalty weights for each component of the model (default=NULL)
coef.w1	multiplicative weight coefficient used for the first iteration only (default=0.5)
cv.min	value of the coefficient of variation below which an individual parameter is considered fixed (default=0.001)

fError.min	minimum fraction of residual variance for combined error model (default = 1e-3)
paramToUse	list of parameters possibly function of covariates (default="all")
covToTest	components of the covariate model that can be modified (default="all")
covToTransform	list of (continuous) covariates to be log-transformed (default="none")
center.covariate	TRUE/FALSE center the covariates of the final model (default=FALSE)
criterion	penalization criterion to optimize c("AIC", "BIC", "BICc", gamma)
linearization	TRUE/FALSE whether the computation of the likelihood is based on a linearization of the model (default=FALSE, deprecated)
ll	TRUE/FALSE compute the observe likelihood and the criterion to optimize at each iteration
test	TRUE/FALSE perform additional statistical tests for building the model (default=TRUE)
direction	method for covariate search c("full", "both", "backward", "forward"), (default="full" or "both")
steps	maximum number of iteration for stepAIC (default=1000)
max.iter	maximum number of iterations (default=20)
explor.iter	number of iterations during the exploratory phase (default=2)
seq.cov	TRUE/FALSE whether the covariate model is built before the correlation model
seq.corr	TRUE/FALSE whether the correlation model is built iteratively (default=TRUE)
seq.cov.iter	number of iterations before building the correlation model (only when seq.cov=F, default=0)
p.max	maximum p-value used for removing non significant relationships between covariates and individual parameters (default=0.1)
p.min	minimum p-values used for testing the components of a new model (default=c(0.075, 0.05, 0.1))
print	TRUE/FALSE display the results (default=TRUE)
nb.model	number of models to display at each iteration (default=1)
fix.param1	parameters with variability that cannot be removed (default=NULL)
fix.param0	parameters without variability that cannot be added (default=NULL)
remove	try to remove random effects (default=T)
add	try to add random effects (default=T)
delta	maximum difference in criteria for testing a new model (default=c(30,10,5))
omega.set	settings to define how a variance varies during iterations of SAEM
pop.set1	Monolix settings 1
pop.set2	Monolix settings 2

Details

See https://monolix.lixoft.com/rsmlx/ for more details.

Value

a new Monolix project with a new statistical model.

Examples

```
## Not run:
# Build the complete statistical model using the default settings
r1 <- buildAll(project="warfarinPK_project.mlxtran")

# Force parameter Tlag to be fixed (no variability) and parameter Cl to vary
r2 <- buildAll(project="warfarinPK_project.mlxtran", fix.param0="Tlag", fix.param1="Cl")

# Estimate the log-likelihood by linearization of the model (faster)
r3 <- buildAll(project="warfarinPK_project.mlxtran", linearization=T)


## End(Not run)

# See http://monolix.lixoft.com/rsmlx/buildmlx/ for detailed examples of use of buildmlx
# Download the demo examples here: http://monolix.lixoft.com/rsmlx/installation
```

---

buildmlx *Automatic statistical model building*

Description

Automatic statistical model building is available directly in the lixoftConnectors package using the function `runModelBuilding`.

Usage

```
buildmlx(
  project = NULL,
  final.project = NULL,
  model = "all",
  prior = NULL,
  weight = NULL,
  coef.w1 = 0.5,
  paramToUse = "all",
  covToTest = "all",
  covToTransform = "none",
  center.covariate = FALSE,
  criterion = "BICc",
  linearization = FALSE,
  ll = T,
  test = T,
  direction = NULL,
```

```
  steps = 1000,
  n.full = 10,
  max.iter = 20,
  explor.iter = 2,
  fError.min = 0.001,
  seq.cov = FALSE,
  seq.cov.iter = 0,
  seq.corr = TRUE,
  p.max = 0.1,
  p.min = c(0.075, 0.05, 0.1),
  print = TRUE,
  nb.model = 1
)
```

Arguments

project	a string: the initial Monolix project
final.project	a string: the final Monolix project (default adds "_built" to the original project)
model	components of the model to optimize c("residualError", "covariate", "correlation"), (default="all")
prior	list of prior probabilities for each component of the model (default=NULL)
weight	list of penalty weights for each component of the model (default=NULL)
coef.w1	multiplicative weight coefficient used for the first iteration only (default=0.5)
paramToUse	list of parameters possibly function of covariates (default="all")
covToTest	components of the covariate model that can be modified (default="all")
covToTransform	list of (continuous) covariates to be log-transformed (default="none")
center.covariate	TRUE/FALSE center the covariates of the final model (default=FALSE)
criterion	penalization criterion to optimize c("AIC", "BIC", "BICc", gamma) (default=BICc)
linearization	TRUE/FALSE whether the computation of the likelihood is based on a linearization of the model (default=FALSE)
ll	TRUE/FALSE compute the observe likelihood and the criterion to optimize at each iteration
test	TRUE/FALSE perform additional statistical tests for building the model (default=TRUE)
direction	method for covariate search c("full", "both", "backward", "forward"), (default="full" or "both")
steps	maximum number of iteration for stepAIC (default=1000)
n.full	maximum number of covariates for an exhaustive comparison of all possible covariate models (default=10)
max.iter	maximum number of iterations (default=20)
explor.iter	number of iterations during the exploratory phase (default=2)

fError.min	minimum fraction of residual variance for combined error model (default = 1e-3)
seq.cov	TRUE/FALSE whether the covariate model is built before the correlation model
seq.cov.iter	number of iterations before building the correlation model (only when seq.cov=F, default=0)
seq.corr	TRUE/FALSE whether the correlation model is built iteratively (default=TRUE)
p.max	maximum p-value used for removing non significant relationships between covariates and individual parameters (default=0.1)
p.min	vector of 3 minimum p-values used for testing the components of a new model (default=c(0.075, 0.05, 0.1))
print	TRUE/FALSE display the results (default=TRUE)
nb.model	number of models to display at each iteration (default=1)

Details

buildmlx uses SAMBA (Stochastic Approximation for Model Building Algorithm), an iterative procedure to accelerate and optimize the process of model building by identifying at each step how best to improve some of the model components. This method allows to find the optimal statistical model which minimizes some information criterion in very few steps.

Penalization criterion can be either a custom penalization of the form gamma*(number of parameters), AIC (gamma=2) or BIC (gamma=log(N)).

Several strategies can be used for building the covariate model at each iteration of the algorithm: `direction="full"` means that all the possible models are compared (default when the number of covariates is less than 10). Othrwise, `direction` is the mode of stepwise search of `stepAIC {MASS}`, can be one of "both", "backward", or "forward", with a default of "both" when there are at least 10 covariates. See https://monolix.lixoft.com/rsmlx/ for more details.

Value

a new Monolix project with a new statistical model.

See Also

`getModelBuildingSettings` settings for model building with lixoftConnectors
`runModelBuilding` run model building with lixoftConnectors
`getModelBuildingResults` results for model building with lixoftConnectors

Examples

```
## Not run:
# RsmlxDemo1.mlxtran is a Monolix project for modelling the pharmacokinetics (PK) of warfarin
# using a PK model with parameters ka, V, Cl.

# By default, buildmlx will compute the best statistical model in term of BIC, i.e ,
# the best covariate model, the best correlation model for the three random effects and the best
# residual error model in terms of BIC.
# In this example, three covariates (wt, age, sex) are available with the data and will be used
```

```
# for building the covariate model for the three PK parameters:
r1 <- buildmlx(project="RsmlxDemo1.mlxtran")

# Here, the covariate model will be built for V and Cl only and log-transformation of all
# continuous covariates will also be considered:
r2 <- buildmlx(project="RsmlxDemo1.mlxtran", paramToUse=c("V", "Cl"), covToTransform="all")

# Only the covariate model will be  built, using AIC instead of BIC:
r3 <- buildmlx(project="RsmlxDemo1.mlxtran", model="covariate", criterion="AIC")

## End(Not run)

# See http://monolix.lixoft.com/rsmlx/buildmlx/ for detailed examples of use of buildmlx
# Download the demo examples here: http://monolix.lixoft.com/rsmlx/installation
```

`buildVar` *Automatic model variance building*

Description

buildVar is designed to build the best variance model for the random effects by selecting which individual parameters vary and which ones are fixed.

Usage

```
buildVar(
  project = NULL,
  final.project = NULL,
  prior = NULL,
  weight = NULL,
  cv.min = 0.001,
  fix.param1 = NULL,
  fix.param0 = NULL,
  criterion = "BICc",
  linearization = F,
  remove = T,
  add = T,
  delta = c(30, 10, 5),
  omega.set = NULL,
  pop.set1 = NULL,
  pop.set2 = NULL,
  print = TRUE
)
```

Arguments

`project`	a string: the initial Monolix project

final.project	a string: the final Monolix project (default adds "_var" to the original project)
prior	named vector of prior probabilities (default=NULL)
weight	named vector of weights (default=NULL)
cv.min	value of the coefficient of variation below which an individual parameter is considered fixed (default=0.001)
fix.param1	parameters with variability that cannot be removed (default=NULL)
fix.param0	parameters without variability that cannot be added (default=NULL)
criterion	penalization criterion to optimize c("AIC", "BIC", "BICc", gamma) (default=BICc)
linearization	TRUE/FALSE whether the computation of the likelihood is based on a linearization of the model (default=FALSE)
remove	TRUE/FALSE try to remove random effects (default=TRUE)
add	TRUE/FALSE try to add random effects (default=TRUE)
delta	maximum difference in criteria for testing a new model (default=c(30,10,5))
omega.set	settings to define how a variance varies during iterations of SAEM
pop.set1	Monolix settings 1
pop.set2	Monolix settings 2
print	TRUE/FALSE display the results (default=TRUE)

Details

Penalization criterion can be either a custom penalization of the form gamma*(number of parameters), AIC (gamma=2) or BIC (gamma=log(N)).

See https://monolix.lixoft.com/rsmlx/ for more details.

Value

a new Monolix project with a new inter individual variability model.

Examples

```
## Not run:
# Build the variability model using the default settings
r1 <- buildVar(project="warfarinPK_project.mlxtran")

# Force parameter Tlag to be fixed (no variability) and parameter Cl to vary
r2 <- buildVar(project="warfarinPK_project.mlxtran", fix.param0="Tlag", fix.param1="Cl")

# Estimate the log-likelihood by linearization of the model (faster)
r3 <- buildVar(project="warfarinPK_project.mlxtran", linearization=T)


## End(Not run)

# See http://monolix.lixoft.com/rsmlx/buildvar/ for detailed examples of use of buildvar
# Download the demo examples here: http://monolix.lixoft.com/rsmlx/installation
```

confintmlx *Confidence intervals for population parameters*

Description

Compute confidence intervals for the population parameters estimated by Monolix.

Usage

```
confintmlx(
  project,
  parameters = "all",
  method = "fim",
  level = 0.9,
  linearization = TRUE,
  nboot = 100,
  parametric = FALSE,
  settings = NULL
)
```

Arguments

project	a Monolix project
parameters	list of parameters for which confidence intervals are computed (default="all")
method	method `c("fim", "proflike", "bootstrap")` (default="fim")
level	confidence level, a real number between 0 and 1 (default=0.90)
linearization	TRUE/FALSE whether the calculation of the standard errors (default=TRUE) or the profile likelihood is based on a linearization of the model (default=TRUE)
nboot	number of bootstrat replicates (default=100, used when method="bootstrap")
parametric	boolean to define if parametric bootstrap is performed (new data is drawn from the model), (default: FALSE)
settings	a list of settings for the profile likelihood method: • `max.iter` maximum number of iterations to find the solution (default=10) • `tol.LL` absolute tolerance for -2LL (default=0.001) • `tol.param` relative tolerance for the parameter (default=0.01) • `print` TRUE/FALSE display the results (default=TRUE)

Details

Most functionality to compute confidence intervals (other than profile likelihood) is now available directly in the lixoftConnectors package. Please migrate the following uses of this function:

confintmlx method		lixoftConnectors function
`"fim"`	`linearizarion = TRUE`	`getEstimatedConfidenceIntervals` (method = "linearization")

"fim"	linearizarion = FALSE	getEstimatedConfidenceIntervals (method = "stochasticApproximation'
"bootstrap"	parametric = TRUE	runBootstrap (method = "parametric")
"bootstrap"	parametric = FALSE	runBootstrap (method = "nonparametric")

For method="proflike", continue using this function.

The method used for computing the confidence intervals can be either based on the standard errors derived from an estimation of the Fisher Information Matrix ("fim"), on the profile likelihood ("proflike") or on nonparametric bootstrap estimate ("bootstrap"). method="fim" is used by default.

When method="fim", the FIM can be either estimated using a linearization of the model or a stochastic approximation. When method="proflike", the observed likelihood can be either estimated using a linearization of the model or an importance sampling Monte Carlo procedure. When method="bootstrap", the bootstrap estimates are obtained using the bootmlx function

Value

a list with the computed confidence intervals, the method used and the level.

See Also

getEstimatedConfidenceIntervals replaces this function for method = "fim" in lixoftConnectors
runBootstrap replaces this function for method = "bootstrap" in lixoftConnectors

Examples

```
## Not run:
# RsmlxDemo2.mlxtran is a Monolix project for modelling the PK of warfarin using a PK model
# with parameters ka, V, Cl.

# confintmlx will compute a 90% confidence interval for all the population parameters
# using the population estimates obtained by Monolix and the Fisher Information Matrix
# estimated by linearization
r1 <- confintmlx(project="RsmlxDemo2.mlxtran")

# 95% confidence intervals are now computed, using the FIM estimated by Monolix using a
# stochastic approximation algorithm:
r2 <- confintmlx(project="RsmlxDemo2.mlxtran", linearization=FALSE, level=0.95)

# Confidence intervals are computed for ka_pop and omega_ka only,
# using the profile likelihood method:
r <- confintmlx(project    = "RsmlxDemo2.mlxtran",
                method     = "proflike",
                parameters = c("ka_pop","omega_ka"))

# Confidence intervals are computed using 200 bootstrap samples:
r3 <- confintmlx(project="RsmlxDemo2.mlxtran", method="bootstrap", nboot=200)

## End(Not run)
```

```
# See http://monolix.lixoft.com/rsmlx/confintmlx/ for detailed examples of use of confintmlx
# Download the demo examples here: http://monolix.lixoft.com/rsmlx/installation
```

`covariateSearch` *Covariate model building*

Description

Automatic search of the best covariate model. Automatic covariate model building is available directly in the lixoftConnectors package using the function `runModelBuilding`. Please migrate, as this function will be deprecated in the future.

Two methods for covariate model building are proposed

- SCM: stepwise covariate modeling method In the forward selection, at each step, each of the remaining (i.e not yet included) parameter-covariate relationships are added to the model in an univariate model (one model per relationship), and run. Among all models, the model that improves some criteria (LRT, BIC or AIC) most is selected and taken forward to the next step. During backward elimination, parameter-covariate relationships are removed in an univariate manner.
- COSSAC: COnditional Sampling for Stepwise Approach based on Correlation tests method COSSAC makes use of the information contained in the base model run to choose which covariate to try first (instead of trying all covariates "blindly" as in SCM). Indeed, the correlation between the individual parameters (or random effects) and the covariates hints at possibly relevant parameter-covariate relationships. If the EBEs (empirical Bayes estimates) are used, shrinkage may bias the result. COSSAC instead uses samples from the a posteriori conditional distribution (available as "conditional distribution" task in MonolixSuite2018) to calculate the correlation between the random effects and covariates. A p-value can be derived using the Pearson's correlation test for continuous covariate and ANOVA for categorical covariate. The p-values are used to sort all the random effect-covariate relationships. Relationships with the lowest p-value are added first, run and confirmed using a likelihood ratio test, AIC or BIC criteria.

Usage

```
covariateSearch(
  project,
  final.project = NULL,
  method = NULL,
  covToTest = NULL,
  covToTransform = NULL,
  paramToUse = NULL,
  testRelations = NULL,
  settings = NULL
)
```

Arguments

project	a Monolix project
final.project	[optional] string corresponding to the final Monolix project (default: 'runFinal.mlxtran' in covariate search output folder)
method	[optional] string correspondig to the method. It can be 'COSSAC' or 'SCM'. By default, COSSAC' is used.
covToTest	[optional] vector of covariates to test. Cannot be used if testRelations is defined. By default, all covariates are tested.
covToTransform	[optional] vector of covariates to transform. The transformation consists in a log transform of the covariate with centering by the mean value (ex: WT is transformed into log(WT/mean) with mean the mean WT value over the individuals of the data set). Both the transformed and untransformed covariate are tested by the algorithm. By default, no covariate is transformed. Note: adding a non-transformed covariate on a lognormally distributed parameter results in an exponential relationship: log(V) = log(Vpop) + beta*WT + eta <=> V = Vpop * exp(beta*WT) * exp(eta) adding a log-transformed covariate on a lognormally distributed parameter results in a power law relationship: log(V) = log(Vpop) + beta*log(WT/70) + eta <=> V = Vpop * (WT/70)^beta * exp(eta)
paramToUse	[optional] vector of parameters which may be function of covariates. Cannot be used if testRelations is defined. By default, all parameters are tested.
testRelations	[optional] list of parameter-covariate relationships to test, ex: list(V=c("WT","SEX"),Cl=c("CRCL")). Cannot be used if covToTest or paramToUse is defined. By default, all parameter-covariate relationships are tested.
settings	[optional] list of settings for the covariate search:

- pInclusion [positive double] threshold on the LRT p-value to accept the model with the added parameter-covariate relationship during forward selection (default = .1). Only used if criteria="LRT".
- pElimination [positive double] threshold on the LRT p-value to accept the model without the removed parameter-covariate relationship during the backward elimination (default = .05). Only used if criteria="LRT".
- criteriaThreshold [positive double] the threshold on the AIC or BIC difference to accept the model with added/removed parameter-covariate relationship (default = 0). Only used if criteria="BIC" or "AIC.
- linearization [boolean] whether the computation of the likelihood is based on a linearization of the model (default = FALSE).
- criteria [string] criteria to optimize. It can be the "BIC", "AIC", or "LRT" (default="LRT").
- direction [string] method for covariate search. It can be "backward", "forward", or "both" (default = "both").
- updateInit [boolean] whether to update or not the initial parameters using the estimates of the parent model (default = FALSE)
- saveRun [boolean] whether to save or not each run (default = TRUE)

See Also

getModelBuildingSettings settings for model building with lixoftConnectors
runModelBuilding run model building with lixoftConnectors
getModelBuildingResults results for model building with lixoftConnectors

Examples

```
## Not run:
# RsmlxDemo1.mlxtran is a Monolix project for modelling the pharmacokinetics (PK) of warfarin
# using a PK model with parameters ka, V, Cl.

# In this example, three covariates (wt, age, sex) are available with the data
# covariatesearch will compute the best covariate model, in term of BIC,
# for the three PK parameters using the three covariates.
r1 <- covariateSearch(project="RsmlxDemo1.mlxtran")

# Instead of using the COSSAC method, we can use the SCM method:
r2 <- covariateSearch(project="RsmlxDemo1.mlxtran", method = 'SCM')

# Here, the covariate model is built using age and wt only, for V and Cl only:
r3 <- covariateSearch(project    = "RsmlxDemo1.mlxtran",
                      paramToUse = c("V","Cl"),
                      covToTest  = c("age","wt"))

## End(Not run)

# See http://monolix.lixoft.com/rsmlx/covariatesearch/ for detailed examples of covariatesearch
# Download the demo examples here: http://monolix.lixoft.com/rsmlx/installation
```

---

getEstimatedCovarianceMatrix

Get estimated covariance and correlation matrices

Description

Get estimated covariance and correlation matrices

Usage

```
getEstimatedCovarianceMatrix()
```

Value

a list of two matrices.

Examples

```
## Not run:
# Assume that the Monolix project "warfarinPKPD.mlxtran" has been loaded
r = getEstimatedCovarianceMatrix()  # r is a list with elements "cor.matrix" and "cov.matrix"

# See http://monolix.lixoft.com/rsmlx/newconnectors/ for more detailed examples
# Download the demo examples here: http://monolix.lixoft.com/rsmlx/installation

## End(Not run)
```

---

getEstimatedIndividualParameters2

Get estimated individual and population parameters

Description

Get the individual individual parameters, the population parameters with the population covariates and the population parameters with the individual covariates.

Usage

```
getEstimatedIndividualParameters2()
```

Value

a list of data frames.

Examples

```
## Not run:
# Assume that the Monolix project "warfarinPKPD.mlxtran" has been loaded
r = getEstimatedIndividualParameters2()

# r is a list with elements "saem", "conditionalMean", "conditionalSD",  "conditionalMode",
# "popPopCov" and "popIndCov"

# See http://monolix.lixoft.com/rsmlx/newconnectors/ for more detailed examples
# Download the demo examples here: http://monolix.lixoft.com/rsmlx/installation

## End(Not run)
```

getEstimatedPredictions *Get estimated predictions*

Description

Get the individual predictions obtained with the estimated individual parameters :

Usage

```
getEstimatedPredictions()
```

Value

a list of data frames (one data frame per output).

Examples

```
## Not run:
# Assume that the Monolix project "warfarinPKPD.mlxtran" has been loaded
r = getEstimatedPredictions() # r is a list with elements "y1" and "y2"

# See http://monolix.lixoft.com/rsmlx/newconnectors/ for more detailed examples
# Download the demo examples here: http://monolix.lixoft.com/rsmlx/installation

## End(Not run)
```

getEstimatedResiduals *Get estimated residuals*

Description

Get the residuals computed from the individual predictions obtained with the estimated individual parameters:

Usage

```
getEstimatedResiduals()
```

Value

a list of data frames (one data frame per output).

Examples

```
## Not run:
# Assume that the Monolix project "warfarinPKPD.mlxtran" has been loaded
r = getEstimatedResiduals()  # r is a list with elements "y1" and "y2"

# See http://monolix.lixoft.com/rsmlx/newconnectors/ for more detailed examples
# Download the demo examples here: http://monolix.lixoft.com/rsmlx/installation

## End(Not run)
```

---

getSimulatedPredictions *Get simulated predictions*

Description

Get the individual predictions obtained with the simulated individual parameters :

Usage

```
getSimulatedPredictions()
```

Value

a list of data frames (one data frame per output).

Examples

```
## Not run:
# Assume that the Monolix project "warfarinPKPD.mlxtran" has been loaded
r = getSimulatedPredictions()  # r is a list with elements "Cc" and "E"

# See http://monolix.lixoft.com/rsmlx/newconnectors/ for more detailed examples
# Download the demo examples here: http://monolix.lixoft.com/rsmlx/installation

## End(Not run)
```

---

getSimulatedResiduals *Get simulated residuals*

Description

Get the residuals computed from the individual predictions obtained with the simulated individual parameters:

Usage

```
getSimulatedResiduals()
```

Value

a list of data frames (one data frame per output).

Examples

```
## Not run:
# Assume that the Monolix project "warfarinPKPD.mlxtran" has been loaded
r = getSimulatedResiduals()  # r is a list with elements "y1" and "y2"

# See http://monolix.lixoft.com/rsmlx/newconnectors/ for more detailed examples
# Download the demo examples here: http://monolix.lixoft.com/rsmlx/installation

## End(Not run)
```

---

initRsmlx *Initialize Rsmlx library*

Description

Initialize Rsmlx library and lixoftConnectors. Prints information about the versions of Monolix and lixoftConnectors used.

Usage

```
initRsmlx(path = NULL)
```

Arguments

path	Monolix path

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:
initRsmlx()
initRsmlx(path="C:/ProgramData/Lixoft/MonolixSuite2024R1")  # specifiy a specific path

## End(Not run)
```

pkbuild *Automatic PK model building*

Description

Fit several structural PK models and select the best one based on a Bayesian Information Criterion. Models to compare can be defined by rate constants and/or clearances and can include or not nonlinear elimination models. See https://monolix.lixoft.com/rsmlx/pkbuild/ for more details.

Usage

```
pkbuild(
  data = NULL,
  project = NULL,
  stat = FALSE,
  param = "clearance",
  new.dir = ".",
  MM = FALSE,
  linearization = F,
  criterion = "BICc",
  level = NULL,
  settings.stat = NULL
)
```

Arguments

data	a list with fields • dataFile: path of a formatted data file • headerTypes: a vector of strings • administration ("iv", "bolus", "infusion", "oral", "ev"): route of administration
project	a Monolix project
stat	(FALSE, TRUE): the statistical model is also built (using buildmlx) (default=FALSE)
param	("clearance", "rate", "both): parametrization (default="clearance")
new.dir	name of the directory where the created files are stored (default is the current working directory))
MM	(FALSE, TRUE): tested models include or not Michaelis Menten elimination models (default=FALSE)
linearization	TRUE/FALSE whether the computation of the likelihood is based on a linearization of the model (default=FALSE)
criterion	penalization criterion to optimize c("AIC", "BIC", "BICc", gamma) (default="BICc")
level	an integer between 1 and 9 (used by setSettings)
settings.stat	list of settings used by buildmlx (only if stat=TRUE)

Value

A list of results

Examples

```
## Not run:
# Build a PK model for the warfarin PK data.
# By default, only models using clearance (and inter compartmental clearances) are used
warf.pk1 <- pkbuild(data=warfarin)

# Models using elimination and transfer rate constants are used,
# as well as nonlinear elimination models
warf.pk2 <- pkbuild(data=warfarin,  new.dir="warfarin", param="rate", MM=TRUE)

# Both models using clearances and rates are used.
# Level is set to 7 in order to get accurate results.
warf.pk3 <- pkbuild(data=warfarin,  new.dir="warfarin", param="both", level=7)

## End(Not run)
```

---

pkpopini *Compute initial population PK parameters*

Description

Use the pooled PK data to derive population PK parameters for a "standard" PK model (i.e. a model of the Monolix PK library). The structural model is automatically defined using the names of the PK parameters. Allowed names are: 'Tlag', 'Mtt', 'Ktr', 'ka', 'Tk0', 'V', 'V1', 'V2', 'V3', 'Q', 'Q2', 'Q3', 'Cl', 'k', 'k12', 'k21', 'k13', 'k31', 'Vm', 'Km'.

Usage

```
pkpopini(
  data = NULL,
  project = NULL,
  parameter = NULL,
  new.project = NULL,
  new.dir = NULL,
  par.ini = NULL
)
```

Arguments

`data`	a list with fields • `dataFile`: path to a formatted data file • `headerTypes`: a vector of strings
`project`	a Monolix project

parameter	a vector of strings (names of the PK parameters)
new.project	name of the new Monolix project (a default name is created if not provided)
new.dir	name of the directory where the created files are stored (default is the current working directory))
par.ini	a vector of PK parameter values

Details

A Monolix project is then automatically created using these values as initial population parameters.

See https://monolix.lixoft.com/rsmlx/pkpopini/ for more details.

Value

A list of results

Examples

```
## Not run:
# Create in the working directory a Monolix project for a 1 cpt model with
# lag time, 0 order absorption and linear elimination
warf.ini1 <- pkpopini(data=warfarin, param=c("Tlag", "Tk0", "V", "Cl"))

# Create in directory 'warfarin' a Monolix project called 'warfPK2.mlxtran'
# for a 2 cpt model with 1st order absorption and nonlinear elimination
warf.ini3 <- pkpopini(data=warfarin, param=c("ka", "V", "k12", "k21", "Vm", "Km"),
                      new.dir="warfarin", new.project="warfPK2.mlxtran")

## End(Not run)
```

---

readDatamlx *Read formatted data file*

Description

Read data in a Monolix/NONMEM format

Usage

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

Arguments

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

Value

A list of data frames

Examples

```
## Not run:
# 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)
```

---

resMonolix *Monolix results*

Description

Monolix results used by the Rsmlx examples

Usage

```
resMonolix
```

Format

A R list

Source

Monolix demos

References

Rsmlx website: http://rsmlx.webpopix.org

`RsmlxDemo1.project` *Monolix project for warfarin PK - 1*

Description

RsmlxDemo2.mlxtran is a Monolix project for modelling the pharmacokinetics (PK) of warfarin using a PK model with parameters ka, V, Cl. There is no covariate in the model.

Usage

```
RsmlxDemo1.project
```

Format

A text file

Source

Monolix project

References

Rsmlx doumentation

`RsmlxDemo2.project` *Monolix project for warfarin PK - 2*

Description

RsmlxDemo2.mlxtran is a Monolix project for modelling the pharmacokinetics (PK) of warfarin using a PK model with parameters ka, V, Cl. Here, V and Cl are function of weight.

Usage

```
RsmlxDemo2.project
```

Format

A text file

Source

Monolix project

References

Rsmlx doumentation

`setSettings` *Easy tuning of the settings of a Monolix project*

Description

Use a single accuracy level, between 1 and 9, to automatically tune all the settings of a Monolix project. When the accuray level is equal to 1, the algorithms are very fast but the results may be not precise. When the accuray level is equal to 9, the algorithms are slow but the results are accurate. Default Monolix settings are obtained with level=5.

Usage

```
setSettings(project = NULL, new.project = NULL, level = 5)
```

Arguments

`project`	a string: a Monolix project (the loaded project if NULL)
`new.project`	a string: the new created Monolix project (default is the original project)
`level`	an integer between 1 and 9 (default=5)

Examples

```
## Not run:
# RsmlxDemo1.mlxtran is a Monolix project for modelling the PK of warfarin.
# All settings of the project are set so that algorithms used by Monolix converge as
# quickly as possible possible:
setSettings(project="RsmlxDemo1.mlxtran", level=1)

# A new project will be created with settings set in order to obtain the most
# precise results possible:
new.project= file.path(tempdir(),"RsmlxDemoNew.mlxtran")
setSettings(project="RsmlxDemo1.mlxtran", new.project=new.project, level=9)

# See http://monolix.lixoft.com/rsmlx/setSettings/ for detailed examples of use of setSettings
# Download the demo examples here: http://monolix.lixoft.com/rsmlx/installation

## End(Not run)
```

testmlx *Statistical tests for model assessment*

Description

Perform several statistical tests using the results of a Monolix run to assess the statistical components of the model in use.

Usage

```
testmlx(
  project = NULL,
  tests = c("covariate", "randomEffect", "correlation", "residual"),
  plot = FALSE,
  adjust = "edf",
  n.sample = NULL
)
```

Arguments

project	a Monolix project
tests	a vector of strings: the list of tests to perform among c("covariate","randomEffect","correlation","residual'
plot	FALSE/TRUE display some diagnostic plots associated to the tests (default=FALSE)
adjust	method to take into account the dependency of MCMC sample c("edf","BH") (default="edf")
n.sample	number of samples from the conditional distribution to be used (default = number of available samples in the project)

Details

The tests used are: 1) F-tests (or, equivalently, correlation tests) to evaluate the effect of each covariate on each parameter ("covariate"), 2) correlation tests to assess the correlation structure of the random effects ("correlation"), 3) Shapiro-Wilk and Miao-Gel-Gastwirth tests to assess, respectively the normality and the symmetry of the distribution of the random effects (""randomEffect"), 4) Shapiro-Wilk and Miao-Gel-Gastwirth tests to assess, respectively the normality and the symmetry of the distribution of residual errors ("residual").

By default, the four tests are performed.

When several samples of the conditional distributions are used, two methods are proposed in order to take into the dependance of the samples for the Shapiro-Wilk and Miao-Gel-Gastwirth tests: "edf" computes an effective degrees of freedom, "BH" performs one test per replicates and adjust the smallest p-value using the Benjamini-Hochberg correction.

Value

a list of data frames and ggplot objects if plot=TRUE

Examples

```
## Not run:
# RsmlxDemo2.mlxtran is a Monolix project for modelling the PK of warfarin using a PK model
# with parameters ka, V, Cl.

#testmlx will perform statistical tests for the different component of the statistical model:
r1 <- testmlx(project="RsmlxDemo2.mlxtran")

#testmlx will perform statistical tests for the covariate model and the correlation model only.
r2 <- testmlx(project="RsmlxDemo2.mlxtran", tests=c("covariate","correlation"))

## End(Not run)

# See http://monolix.lixoft.com/rsmlx/testmlx/ for detailed examples of use of testmlx
# Download the demo examples here: http://monolix.lixoft.com/rsmlx/installation
```

---

warfarin.data *warfarin PKPD data*

Description

The warfarin PK and PD data for 32 patients

Usage

```
warfarin.data
```

Format

A csv file

Source

Monolix demos

References

O'Reilly (1968). Studies on coumarin anticoagulant drugs. Initiation of warfarin therapy without a loading dose. Circulation 1968, 38:169-177.

whichPKmodel *Find a Monolix PK model*

Description

Return the path of the Monolix PK model defined by a list of parameter names See https://monolix.lixoft.com/rsmlx/whichPK
for more details.

Usage

```
whichPKmodel(parameter, mlxPath = NULL, pkPath = NULL, lib = FALSE)
```

Arguments

parameter	a vector of PK parameter names
mlxPath	path to Monolix install
pkPath	path to the Monolix PK library
lib	boolean to define if the absolute path is returned

Examples

```
## Not run:
whichPKmodel(parameter=c("Tlag", "Tk0", "V", "Cl"))

## End(Not run)
```

Index