Package 'PFIM'

June 24, 2022

Type Package

Title Population Fisher Information Matrix

Version 5.0

Date 2022-05-23

NeedsCompilation no

Description Evaluate or optimize designs for nonlinear mixed effects models using the Fisher Information matrix. Methods used in the package refer to
Mentré F, Mallet A, Baccar D (1997) <doi:10.1093/biomet/84.2.429>,
Retout S, Comets E, Samson A, Mentré F (2007) <doi:10.1002/sim.2910>,
Bazzoli C, Retout S, Mentré F (2009) <doi:10.1002/sim.3573>,
Le Nagard H, Chao L, Tenaillon O (2011) <doi:10.1186/1471-2148-11-326>,
Combes FP, Retout S, Frey N, Mentré F (2013) <doi:10.1007/s11095-013-1079-3> and
Seurat J, Tang Y, Mentré F, Nguyen TT (2021) <doi:10.1016/j.cmpb.2021.106126>.

Depends R (>= 4.0.0)

License GPL (>= 2)

Encoding UTF-8

Imports methods, rmarkdown, stats, scales, deSolve, kableExtra, gtable, Deriv, grid, knitr, markdown, Matrix, ggplot2, ggbreak, pracma, Rcpp, filesstrings

VignetteBuilder knitr

Suggests testthat, inline, utils, devtools, htmltools

RoxygenNote 7.1.2

Collate 'Constraint.R' 'Administration.R' 'AdministrationConstraint.R' 'Arm.R' 'Fim.R' 'IndividualFim.R' 'BayesianFim.R' 'ModelError.R' 'Combined1.R' 'Combined1c.R' 'Combined2c.R' 'Combined2.R' 'Constant.R' 'ContinuousConstraint.R' 'Optimization.R' 'Design.R' 'DesignConstraint.R' 'DiscreteConstraint.R' 'Distribution.R' 'FedorovWynnAlgorithm.R' 'ModelEquations.R' 'Model.R' 'PKModel.R' 'PDModel.R' 'LibraryOfModels.R' 'ModelODEquations.R' 'ModelInfusionODEquations.R' 'ModelInfusionEquations.R' 'FillLibraryOfModels.R'

'StandardDistribution.R' 'LogNormalDistribution.R'
'ModelParameter.R' 'ModelVariable.R'
'MultiplicativeAlgorithm.R' 'NormalDistribution.R'
'PFIM-package.R' 'StatisticalModel.R' 'PFIMProject.R'
'PGBOAlgorithm.R' 'PKPDModel.R' 'PSOAlgorithm.R'
'PopulationFim.R' 'Proportional.R' 'ProportionalC.R'
'ReportAndPlots.R' 'Response.R' 'SamplingConstraint.R'
'SamplingTimes.R' 'SimplexAlgorithm.R' 'globals.R'

Author France Mentré [aut] (<https://orcid.org/0000-0002-7045-1275>),
Hervé Le Nagard [aut],
Romain Leroux [aut, cre],
Jérémy Seurat [aut],
Tran Bach Nguyen [ctb],
Caroline Bazzoli [ctb],
Emmanuelle Comets [ctb],
Anne Dubois [ctb],
Cyrielle Dumont [ctb],
Giulia Lestini [ctb],
Thi Huyen Tram Nguyen [ctb],
Thu Thuy Nguyen [ctb],
Sylvie Retout [ctb]

Maintainer Romain Leroux <romain.leroux@inserm.fr>

Repository CRAN

Date/Publication 2022-06-24 08:10:05 UTC

R topics documented:

`PFIM-package` *Fisher Information matrix for design evaluation/optimization for non-linear mixed effects models.*

Description

Evaluate or optimize designs for nonlinear mixed effects models using the Fisher Information matrix. Methods used in the package refer to Mentré F, Mallet A, Baccar D (1997) <doi:10.1093/biomet/84.2.429>, Retout S, Comets E, Samson A, Mentré F (2007) <doi:10.1002/sim.2910>, Bazzoli C, Retout S, Mentré F (2009) <doi:10.1002/sim.3573>, Le Nagard H, Chao L, Tenaillon O (2011) <doi:10.1186/1471-2148-11-326>, Combes FP, Retout S, Frey N, Mentré F (2013) <doi:10.1007/s11095-013-1079-3> and Seurat J, Tang Y, Mentré F, Nguyen TT (2021) <doi:10.1016/j.cmpb.2021.106126>.

Description

Nonlinear mixed effects models (NLMEM) are widely used in model-based drug development and use to analyze longitudinal data. The use of the "population" Fisher Information Matrix (FIM) is a good alternative to clinical trial simulation to optimize the design of these studies. PFIM 4.0 was released in 2018 as a list of R functions [1]. The present version, **PFIM** 5.0, is an R package that uses the S4 object system for evaluating and/or optimizing population designs based on FIM in NLMEMs.

This new version of **PFIM** now includes a library of models implemented also using the object oriented system S4 of R. This new library contains two libraries of pharmacokinetic (PK) and/or pharmacodynamic (PD) models. The PK library includes model with different administration routes (bolus, infusion, first-order absorption), different number of compartments (from 1 to 3), and different types of eliminations (linear or Michaelis-Menten). The PD model library, contains direct immediate models (e.g. Emax and Imax) with various baseline models, and turnover response models. The PK/PD models are obtained with combination of the models from the PK and PD model libraries. **PFIM** handles both analytical and ODE models and offers the possibility to the user to define his/her own model(s).

In **PFIM 5.0**, the FIM is evaluated by first order linearization of the model assuming a block diagonal FIM as in [3]. The Bayesian FIM is also available to give shrinkage predictions [4]. **PFIM 5.0** includes several algorithms to conduct design optimization based on the D-criterion, given design constraints : the simplex algorithm (Nelder-Mead) [5], the multiplicative algorithm [6], the Fedorov-Wynn algorithm [7], PSO (*Particle Swarm Optimization*) and PGBO (*Population Genetics Based Optimizer*) [9].

Validation

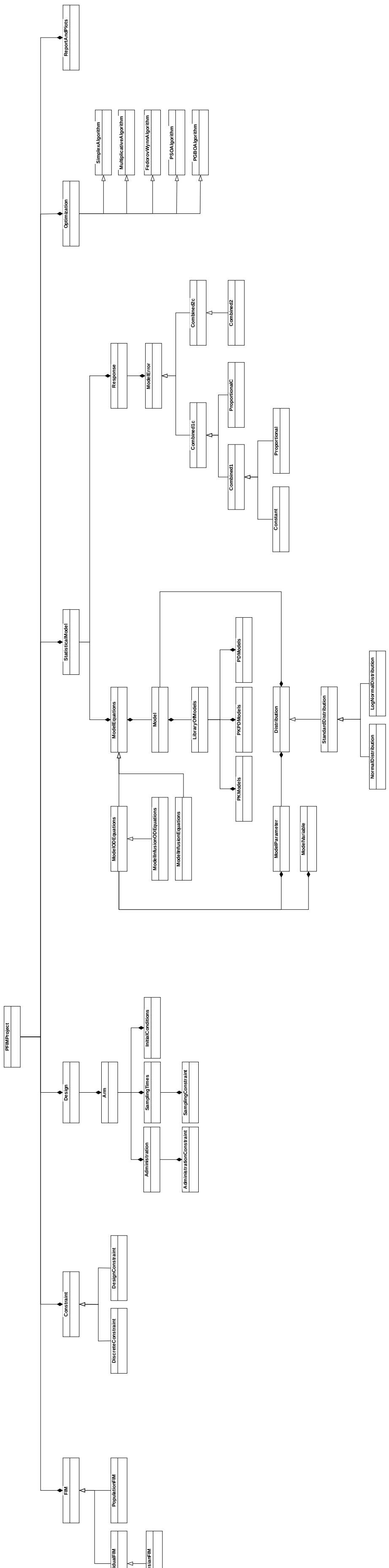
PFIM 5.0 also provides quality control with tests and validation using the evaluated FIM to assess the validity of the new version and its new features. Finally, **PFIM 5.0** displays all the results with both clear graphical form and a data summary, while ensuring their easy manipulation in R. The standard data visualization package ggplot2 for R is used to display all the results with clear graphical form [10]. A quality control using the D-criterion is also provided.

Organization of the source code / files in the `/R` folder

PFIM 5.0 contains a hierarchy of S4 classes with corresponding methods and functions serving as constructors. All of the source code related to the specification of a certain class is contained in a file named `[Name_of_the_class]-Class.R`. These classes include:

- 1. all roxygen `@include` to insure the correctly generated collate for the DESCRIPTION file,
- 2. `\setClass` preceded by a roxygen documentation that describes the purpose and slots of the class,
- 3. specification of an initialize method,
- 4. all getter and setter, respectively returning attributes of the object and associated objects.

The following class diagrams provide an overview on the structure of the package.


ReportAndPlots
Optimization
SimplexAlgorithm
MultiplicativeAlgorithm
FedorovWynnAlgorithm
PSOAlgorithm
PGBOAlgorithm
StatisticalModel
Response
ModelError
Combined2c
Combined2
Combined1c
ProportionalC
Combined1
Proportional
Constant
ModelEquations
Model
LibraryOfModels
PDModels
PKPDModels
PKModels
Distribution
StandardDistribution
LogNormalDistribution
NormalDistribution
ModelODEquations
ModelInfusionODEquations
ModelInfusionEquations
ModelParameter
ModelVariable
PFIMProject
Design
Arm
InitialConditions
SamplingTimes
SamplingConstraint
Administration
AdministrationConstraint
Constraint
DesignConstraint
DiscreteConstraint
FIM
PopulationFIM

Content of the source code and files in the `/R` folder

- Class `Administration`
  - `getAllowedDose`
  - `getAllowedTime`
  - `getAllowedTinf`
  - `getAmountDose`
  - `getNameAdministration`
  - `getTau`
  - `getTimeDose`
  - `getTinf`
  - `is.multidose`
  - `setAllowedDose`
  - `setAllowedTime<-`
  - `setAllowedTinf<-`
  - `setAmountDose`
  - `setTau`
  - `setTimeDose<-`
  - `setTinf`
- Class `AdministrationConstraint`
  - `AllowedDoses`
  - `fixedDoses`
  - `getAllowedDoses`
  - `getDoseOptimisability`
  - `getNumberOfDoses`
  - `getResponseName`
- Class `Arm`
  - `addAdministration`
  - `addSampling`
  - `addSamplings`
  - `EvaluateStatisticalModel`
  - `getAdministration`
  - `getAdministrationByOutcome`
  - `getArmSize`
  - `getNameArm`
  - `getCondInit`
  - `getSamplings`
  - `setArmSize`
  - `setInitialConditions`
  - `setSamplings<-`
  - `getResponseNameByIndice`
  - `addSamplingConstraints`

  - getSamplingConstraintsInArm
  - modifySamplingTimes
  - getNumberOfSamplings
- Class BayesianFim
  - getDescription
  - getShrinkage
- Classes Combined1, Combined1c, Combined2, Combined2c
  - getSigmaNames
  - getSigmaValues
  - show
- Class Constant
  - getSigmaNames
  - getSigmaValues
  - show
- Class Constraint
- Class ContinuousConstraint
  - getRange
  - setRange<-
- Class Design
  - addArm
  - addArms
  - EvaluateDesignForEachArm
  - getAmountOfArms
  - getArms
  - getEvaluationDesign
  - getFimOfDesign
  - getNameDesign
  - getNumberSamples
  - setNumberSamples<-
  - getOptimizationResult
  - getTotalSize
  - modifyArm
  - setAmountOfArms
  - setArms
  - setNameDesign
  - setTotalSize<-
  - show
  - showArmData
  - summary
  - summaryArmData

- Class DesignConstraint
  - addAdministrationConstraint
  - addDesignConstraints
  - addSamplingConstraint
  - getAdministrationConstraint
  - getNameDesignConstraint
  - getTotalNumberOfIndividuals
  - getSamplingConstraints
  - setAmountOfArmsAim
  - setPossibleArms
  - setTotalNumberOfIndividuals
  - show
- Class DiscreteConstraint
  - getDiscret
  - setDiscret<-
- Class Distribution
- Class FedorovWynnAlgorithm
  - FedorovWynnAlgorithm_Rcpp
  - resizeFisherMatrix
  - PrepareFIMs
  - Optimize
- Class Fim
  - FinalizeFIMForOneElementaryDesign
  - getConditionNumberMatrix
  - getCorr
  - getDcriterion
  - getDescription
  - getDeterminant
  - getEigenValue
  - getMfisher
  - getSE
  - getStatisticalModelStandardErrors
  - setMfisher<-
  - setMu
  - setOmega
  - show
  - showStatisticalModelStandardErrors
- Class IndividualFim
  - getDescription
  - getStatisticalModelStandardErrors

  - show
  - showStatisticalModelStandardErrors
- Class LibraryOfModels
  - addModel
  - getContentsLibraryOfModels
  - getModel
  - getModelNameList
  - getPKPDModel
- Class LogNormalDistribution
  - AdjustLogNormalDistribution
- Class Model
  - getEquations
  - getEquationsModel
  - getModelName
  - setParametersModel
- Class ModelEquations
  - convertAnalyticToODE
  - EvaluateModel
  - getDerivate
  - getEquation
  - getEquations
  - getNumberOfParameters
  - getParameters
  - getResponseIndice
  - remplaceDose
- Class ModelError
  - g
  - getCError
  - getDVSigma
  - getEquation
  - getErrorModelParameters
  - getNumberOfParameter
  - getSig
  - getSigmaInter
  - getSigmaNames
  - getSigmaSlope
  - getSigmaValues
  - setCError<-
  - setSigmaInter<-
  - setSigmaSlope<-

  - show
- Class ModelInfusionEquations
  - getInfusionEquations
  - getEquationsModelPKPD
  - EvaluateModelInfusion
- Class ModelInfusionODEquations
  - getResponseIndice
  - scaleResponsesEvaluationODEInfusion
  - EvaluateModelODEInfusion
- Class ModelODEquations
  - getDerivatives
  - scaleResponsesEvaluationODE
  - getEquationsModelPKPD
  - EvaluateModelODE
- Class ModelParameter
  - getDerivatesAdjustedByDistribution
  - getDistribution
  - getMu
  - getNameModelParameter
  - getOmega
  - isFixed
  - isFixedMu
  - isNotFixed
  - isNotFixedMu
- Class ModelVariable
  - getNameModelVariable
- Class MultiplicativeAlgorithm
  - getWeightFrame
  - MultiplicativeAlgorithm_Rcpp
  - Optimize
  - PrepareFIMs
  - setDelta
  - setIteration
  - setShowProcess
  - show
- Class NormalDistribution
  - AdjustNormalDistribution
- Class Optimization
  - Combinaison

  - EvaluateFIMsAndDesigns
  - getElementaryProtocols
  - getOptimalDesign
  - setOptimalDesign
  - Optimize
  - PrepareFIMs
  - setShowProcess
  - show
- Class PDModel
- Class PFIMProject
  - addDesign
  - addDesigns
  - defineStatisticalModel
  - EvaluateBayesianFIM
  - EvaluateDesign
  - EvaluateIndividualFIM
  - EvaluatePopulationFIM
  - getDesign
  - getEvaluationResponses
  - getFim
  - getFims
  - getFisherMatrices
  - getNamePFIMProject
  - setNamePFIMProject
  - getParametersOdeSolver
  - setParametersOdeSolver
  - getStatisticalModel
  - getWeights
  - OptimizeDesign
  - plotCriteria
  - plotResponse
  - plotRSE
  - plotSE
  - plotSensitivity
  - plotWeightOptimisation
  - plotFrequenciesOptimisation
  - plotShrinkage
  - setConstraint
  - setDesign
  - show
  - showConstraints
  - showDesigns

  - showFims
  - summary
  - reportPFIMProject
- Class PKModel
  - changeVariablePKModel
  - getEquations
- Class PKPDModel
  - getEquations
  - getPDModel
  - getPKModel
- Class PopulationFim
  - FinalizeFIMForOneElementaryDesign
  - getDescription
  - getStatisticalModelStandardErrors
  - showStatisticalModelStandardErrors
- Class ReportAndPlots
  - knitrModelEquations
  - knitrModelError
  - knitrModelParameters
  - knitrAdministrationParameters
  - knitrInitialDesigns
  - knitrFIM
  - knitrOptimalDesign
  - PFIMProjectReportEvaluation
  - PFIMProjectReportOptimization
- Class Response
  - EvaluateErrorModelDerivatives
  - EvaluateODEErrorModelDerivatives
  - getModelError
  - getNameResponse
  - getSigmaNames
  - IndividualFIMEvaluateVariance
  - PopulationFIMEvaluateVariance
  - setModelError<-
- Class SamplingConstraint
  - allowedContinuousSamplingTimes
  - allowedDiscretSamplingTimes
  - FixTimeValues
  - getallowedDiscretSamplingTimes
  - getfixedTimes

  - getnumberOfSamplingTimes
  - getOptimisability
  - getResponseName
  - isLessThanDelay
  - isTimeInBetweenBounds
  - numberOfSamplingTimesIsOptimisable
- Class SamplingTimes
  - getNameSampleTime
  - getSampleTime
  - setSampleTime
  - getNumberTime
  - getInitialTime
- Class StandardDistribution
- Class StatisticalModel
  - addResponse
  - addResponses
  - CalculatedResidualVariance
  - defineCorrelation
  - defineModelEquations
  - defineParameter
  - Evaluate
  - EvaluationModel
  - getEquationsStatisticalModel
  - getErrorModelStandardErrors
  - getModelParameters
  - getFixedParameters
  - getResponsesStatisticalModel
  - show
  - checkParameterInEquations
  - setParametersForEvaluateModel
  - parametersForComputingGradient
  - defineVariable
  - defineVariables

Author(s)

Maintainer: Romain Leroux <romain.leroux@inserm.fr>

Authors:

- France Mentré <france.mentre@inserm.fr> (ORCID)
- Hervé Le Nagard <herve.lenagard@inserm.fr>
- Jérémy Seurat <jeremy.seurat@inserm.fr>

Other contributors:

- Tran Bach Nguyen [contributor]
- Caroline Bazzoli [contributor]
- Emmanuelle Comets [contributor]
- Anne Dubois [contributor]
- Cyrielle Dumont [contributor]
- Giulia Lestini [contributor]
- Thi Huyen Tram Nguyen [contributor]
- Thu Thuy Nguyen [contributor]
- Sylvie Retout [contributor]

References

[1] Dumont C, Lestini G, Le Nagard H, Mentré F, Comets E, Nguyen TT, et al. PFIM 4.0, an extended R program for design evaluation and optimization in nonlinear mixed-effect models. Comput Methods Programs Biomed. 2018;156:217-29.

[2] Chambers JM. Object-Oriented Programming, Functional Programming and R. Stat Sci. 2014;29:167-80.

[3] Mentré F, Mallet A, Baccar D. Optimal Design in Random-Effects Regression Models. Biometrika. 1997;84:429-42.

[4] Combes FP, Retout S, Frey N, Mentré F. Prediction of shrinkage of individual parameters using the bayesian information matrix in nonlinear mixed effect models with evaluation in pharmacokinetics. Pharm Res. 2013;30:2355-67.

[5] Nelder JA, Mead R. A simplex method for function minimization. Comput J. 1965;7:308–13.

[6] Seurat J, Tang Y, Mentré F, Nguyen, TT. Finding optimal design in nonlinear mixed effect models using multiplicative algorithms. Computer Methods and Programs in Biomedicine, 2021.

[7] Fedorov VV. Theory of Optimal Experiments. Academic Press, New York, 1972.

[8] Eberhart RC, Kennedy J. A new optimizer using particle swarm theory. Proc. of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, 4-6 October 1995, 39-43.

[9] Le Nagard H, Chao L, Tenaillon O. The emergence of complexity and restricted pleiotropy in adapting networks. BMC Evol Biol. 2011;11:326.

[10] Wickham H. ggplot2: Elegant Graphics for Data Analysis, Springer-Verlag New York, 2016.

`addAdministration` *Add an administration to an arm.*

Description

Add an administration to an arm.

Usage

```
addAdministration(object, value)
```

Arguments

`object`	An object `Arm` from the class Arm.
`value`	An object from the class Administration.

Value

The object `Arm` with its administration.

`addAdministrationConstraint`
Add constraints on the administration for a `DesignConstraint` *object.*

Description

Add constraints on the administration for a `DesignConstraint` object.

Usage

```
addAdministrationConstraint(object, value)
```

Arguments

`object`	A `DesignConstraint` object.
`value`	An `AdministrationConstraint` object.

Value

The `DesignConstraint` object with its administration constraints.

addArm *Add an arm to a design.*

Description

Add an arm to a design.

Usage

```
addArm(object, arm)
```

Arguments

object	A Design object.
arm	An Arm object.

Value

The Design object with the new arm.

addArms *Add arms to a design.*

Description

Add arms to a design.

Usage

```
addArms(object, listOfArms)
```

Arguments

object	A Design object.
listOfArms	A list of Arm object.

Value

The Design object with the new arms.

addDesign *Add a design to the* PFIMProject *object.*

Description

Add a design to the PFIMProject object.

Usage

```
addDesign(object, design)
```

Arguments

object	A PFIMProject object.
design	A Design object.

Value

The PFIMProject object with the Design object added.

addDesignConstraints *Add design constraints on the sampling for a design.*

Description

Add design constraints on the sampling for a design.

Usage

```
addDesignConstraints(object, listOfConstraints)
```

Arguments

object	A DesignConstraint object.
listOfConstraints	A list of Constraint object.

Value

The DesignConstraint object constraints with the constraints from listOfConstraints.

addDesigns *Add a list of designs to the* PFIMProject *object.*

Description

Add a list of designs to the PFIMProject object.

Usage

```
addDesigns(object, listOfDesigns)
```

Arguments

object	A PFIMProject object.
listOfDesigns	A list of Design objects.

Value

The PFIMProject object with the Design objects added.

addModel *Add a* Model *object in the* LibraryOfModels.

Description

Add a Model object in the LibraryOfModels.

Usage

```
addModel(object, model)
```

Arguments

object	A LibraryOfModels object.
model	The model to add in the library (PK, PD or PKPD model).

Value

The LibraryOfModels object with the loaded library of models.

addResponse *Add a response to a statistical model.*

Description

Add a response to a statistical model.

Usage

```
addResponse(object, value)
```

Arguments

object	A StatisticalModel object.
value	A character string giving the name of the response to add.

Value

The StatisticalModel object with the added response.

addResponses *Add responses to a statistical model.*

Description

Add responses to a statistical model.

Usage

```
addResponses(object, listOfResponses)
```

Arguments

object	A StatisticalModel object.
listOfResponses	A list of character string giving the names of the responses to add.

Value

The StatisticalModel object with the added responses.

addSampling *Add sampling time for an arm and for a response.*

Description

Add sampling time for an arm and for a response.

Usage

```
addSampling(object, value)
```

Arguments

object	An object Arm from the class Arm.
value	An object from the class SamplingTimes.

Value

The object Arm with its the new sampling times.

addSamplingConstraint *Add a constraint on the sampling for a design.*

Description

Add a constraint on the sampling for a design.

Usage

```
addSamplingConstraint(object, value)
```

Arguments

object	A DesignConstraint object.
value	A SamplingConstraint object.

Value

The DesignConstraint object constraints with the constraints from SamplingConstraint object added.

addSamplingConstraints *Add sampling constraints to an arm.*

Description

Add sampling constraints to an arm.

Usage

```
addSamplingConstraints(object, sampling_constraints)
```

Arguments

object	An object Arm from the class Arm.
sampling_constraints	A SamplingConstraint object giving the sampling constraints added to the Arm object.

Value

The object Arm with the new sampling constraints.

addSamplings *Add sampling times for an arm and for a response.*

Description

Add sampling times for an arm and for a response.

Usage

```
addSamplings(object, listOfSamplings)
```

Arguments

object	An object Arm from the class Arm.
listOfSamplings	The objects from the class SamplingTimes.

Value

The object Arm with its new sampling times.

AdjustLogNormalDistribution

Adjust the mean of a LogNormalDistribution *object.*

Description

Adjust the mean of a LogNormalDistribution object.

Usage

```
AdjustLogNormalDistribution(object, mu, df_total)

## S4 method for signature 'LogNormalDistribution'
AdjustLogNormalDistribution(object, mu, df_total)
```

Arguments

object	A AdjustLogNormalDistribution.
mu	A numeric giving the mean mu.
df_total	numeric giving df_total

Value

A StandardDistribution object giving the adjusted Log Normal distribution.

AdjustNormalDistribution

Adjust the Normal Distribution.

Description

Adjust the Normal Distribution.

Usage

```
AdjustNormalDistribution(object, mu, df_total)

## S4 method for signature 'NormalDistribution'
AdjustNormalDistribution(object, mu, df_total)
```

Arguments

object	A NormalDistribution.
mu	A numeric giving the mean mu.
df_total	numeric giving df_total

Value

A StandardDistribution object giving the adjusted Normal distribution.

Administration-class *Class "Administration"*

Description

The class Administration defines information concerning the parametrization and the type of administration: single dose, multiple doses. Constraints can also be added on the allowed times, doses and infusion duration.

Objects from the class

Objects form the class Administration can be created by calls of the form Administration(...) where (...) are the parameters for the Administration objects.

Slots for Administration objects

outcome: A character string giving the name for the reponse of the model.

time_dose: A numeric vector giving the times when doses are given. By default set to 0.

amount_dose: A numeric vector giving the amount of doses.

tau: A numeric giving the frequency.

Tinf: A numeric vector giving the infusion duration Tinf (Tinf can be null).

allowed_time: Constraint object containing the constraints on allowed times.

allowed_dose: Constraint object containing the constraints on allowed dose.

allowed_tinf: Constraint object containing the constraints on Tinf.

AdministrationConstraint-class *Class "AdministrationConstraint"*

Description

The class AdministrationConstraint represents the constraint of an input to the system. The class stores information concerning the constraints for the dosage regimen : response of the model, type of administration, amount of dose.

Objects from the class

Objects form the class AdministrationConstraint can be created by calls of the form #'AdministrationConstraint(... where (...) are the parameters for the AdministrationConstraint objects.

Slots for the `AdministrationConstraint` **objects**

`response`**:** A character giving the response of the model.

`Optimisability`**:** A boolean giving if a dose is optimisable or not. If not the dose is fixed.

`fixedDoses`**:** A vector giving the fixed doses.

`AllowedDoses`**:** A vector giving the allowed amount of doses.

`allowedContinuousSamplingTimes`
Set the allowed continuous sampling times.

Description

Set the allowed continuous sampling times.

Usage

```
allowedContinuousSamplingTimes(object, allowedTimes)
```

Arguments

`object`	A `SamplingConstraint` object.
`allowedTimes`	A list giving the vectors for the allowed continuous Sampling Times.

Value

The object `SamplingConstraint` object with the allowed continuous Sampling Times.

`allowedDiscretSamplingTimes`
Set the allowed discret sampling times.

Description

Set the allowed discret sampling times.

Usage

```
allowedDiscretSamplingTimes(object, allowedTimes)
```

Arguments

`object`	A `SamplingConstraint` object.
`allowedTimes`	A list giving the vectors for the allowed Continous Sampling Times.

Value

The object `SamplingConstraint` object with the allowed Continous Sampling Times.

AllowedDoses *Define the vector of allowed amount of dose.*

Description

Define the vector of allowed amount of dose.

Usage

```
AllowedDoses(object, value)
```

Arguments

object	An object AdministrationConstraint from the class AdministrationConstraint.
value	A numeric vector giving the allowed amount of doses.

Value

The AdministrationConstraint object with its new allowed amount of doses.

Arm-class *Class "Arm"*

Description

The class "Arm combines the treatment (the class Administration) and the sampling schedule (the class SamplingTimes).

Objects from the class

Objects form the class Arm can be created by calls of the form Arm(...) where (...) are the parameters for the Arm objects.

Slots for the Arm objects

arm_size: An integer the number of subjects in the arm. By default set to 1.

administrations: A list of Administration objects.

cond_init: A list of the initial conditions.

samplings: A list of SamplingTimes objects.

constraints: A list of SamplingConstraint objects.

sampling_constraints: A list of objects from SamplingConstraint class.

`BayesianFim-class` *Class "BayesianFim" representing the population Fisher information matrix.*

Description

A class storing information regarding the population Fisher computation matrix.

Objects from the class

`BayesianFim` objects are typically created by calls to `{BayesianFim}`:

Bayesianfim: Create a new `BayesianFim`

`CalculatedResidualVariance` *Compute the residual variance thanks to the function g of the model error.*

Description

Compute the residual variance thanks to the function g of the model error.

Usage

```
CalculatedResidualVariance(objectStatisticalModel, objectModelError, x_i)
```

Arguments

`objectStatisticalModel`	A `StatisticalModel` object.
`objectModelError`	A `ModelError` object.
`x_i`	variable x_i of the model error. #' @return CalculatedResidualVariance

changeVariablePKModel *Change variable in a PK Model.*

Description

Change variable in a PK Model.

Usage

```
changeVariablePKModel(object)
```

Arguments

object	PKModel object.

Value

A expression giving the equations of the PK model with variable changed.

checkParameterInEquations
Check the parameters in the model equations.

Description

Check the parameters in the model equations.

Usage

```
checkParameterInEquations(object)
```

Arguments

object	A StatisticalModel object.

Value

The ModelParameter objects of the model parameters in the model equation.

Combinaison *Create all the possible combinaison for each Design and each Arms.*

Description

Create all the possible combinaison for each Design and each Arms.

Usage

```
Combinaison(object, times, nTimesForEachVector, fixedTimes, n, combin)
```

Arguments

object	A Optimization object.
times	A SAmplingTimes object.
nTimesForEachVector	the number of sampling times for each vector of sampling times.
fixedTimes	the fixed sampling times.
n	parameter n
combin	parameter combin

Value

All the possible combination for each Design and each Arms.

Combined1-class *Class "Combined1"*

Description

The class Combined1 defines the the residual error variance according to the formula g(sigma_inter, sigma_slope, c_error, f(x, theta)) = sigma_inter + sigma_slope*f(x,theta)).

Objects from the class

Combined1 objects are typically created by calls to Combined1 and contain the following slots that are herited from the class Combined1c:

.Object: An object of the Class Combined1

sigma_inter: A numeric value giving the sigma inter of the error model.

sigma_slope: A numeric value giving the sigma slope of the error model.

Combined1c-class *Class "Combined1c"*

Description

The class Combined1c defines the the residual error variance according to the formula g(sigma_inter, sigma_slope, c_error, f(x, theta)) = sigma_inter + sigma_slope*f(x,theta)^c_error).

Objects from the class

Combined1c objects are typically created by calls to {Combined1c} and contain the following slots that are heritated from the class ModelError:

`.Object:` An object of the class `ModelError`

`sigma_inter:` A numeric value giving the sigma inter of the error model.

`sigma_slope:` A numeric value giving the sigma slope of the error model.

`c_error:` A numeric value giving the exponant c of the error model.

Combined2-class *Class "Combined2"*

Description

The class {Combined2} defines the the residual error variance according to the formula g(sigma_inter, sigma_slope, c_error, f(x, theta)) = sigma_inter^2 + sigma_slope^2 * f_x_i_theta.

Objects from the class

{Combined2} objects are typically created by calls to {Combined2} and contain the following slots that are heritated from the class Combined2c:

`.Object:` An object of the class ModelError

`sigma_inter:` A numeric value giving the sigma inter of the error model.

`sigma_slope:` A numeric value giving the sigma slope of the error model.

Combined2c-class *Class "Combined2c"*

Description

The class Combined2c defines the the residual error variance according to the formula g(sigma_inter, sigma_slope, c_error, f(x, theta)) = sigma_inter^2 + sigma_slope^2 * f(x,theta)^(2 * c_error)

Objects from the class

Combined2c objects are typically created by calls to Combined2c and contain the following slots that are heritated from the class ModelError:

.Object: An object of the class ModelError

sigma_inter: A numeric value giving the sigma inter of the error model.

sigma_slope: A numeric value giving the sigma slope of the error model.

c_error: A numeric value giving the exponant c of the error model.

Constant-class *Class "Constant"*

Description

The class Constant defines the the residual error variance according to the formula g(sigma_inter, sigma_slope, c_error, f(x, theta)) = sigma_inter.

Objects from the class

Constant objects are typically created by calls to Constant and contain the following slots that are heritated from the class Combined1:

.Object: An object of the class ModelError

sigma_inter: A numeric value giving the sigma inter of the error model.

`Constraint-class` *Class "Constraint"*

Description

The class `Constraint` stores the constraints for a variable. Constraints are given either as: a continuous range, a discrete set of values, or a Design constraint.

Objects from the class `Constraint`

Objects form the class `Constraint` can be created by calls of the objects from the following classes:

- AdministrationConstraint
- ContinuousConstraint
- DesignConstraint
- DiscreteConstraint

`ContinuousConstraint-class`
Class "ContinuousConstraint" representing the constraints for a variable

---

Description

The class `ContinuousConstraint` stores constraints for a variable. Constraints are given either as a continuous range (any value between a min and a max boundary is admissible) or a discrete set of values (any value belonging to the set is admissible).

Objects from the class

Constraint objects are typically created by calls to `constraint` and contain the following slots:

`type`**:** A character string, one of 'continuous' or 'discrete'.

`range`**:** A numeric vector with two values giving the min/max of the continuous range.

`minimalDelay`**:** A numeric value giving the minimal timestep between two sampling times.

convertAnalyticToODE *Convert an equation of a PD model of a* ModelEquations *object from analytic to ODE.*

Description

Convert an equation of a PD model of a ModelEquations object from analytic to ODE.

Usage

```
convertAnalyticToODE(object)
```

Arguments

object	ModelEquations object.

Value

A list of expression output giving the equations of the analytic PD model in ODE form.

defineCorrelation *Set the correlation.*

Description

Set the correlation.

Usage

```
defineCorrelation(object, correlationlist)
```

Arguments

object	A StatisticalModel object.
correlationlist	...

Value

Return correlationlist

defineModelEquations *Define model equations*

Description

Define model equations

Usage

```
defineModelEquations(object, equations)
```

Arguments

object	A StatisticalModel object.
equations	An expression giving the equations of the model.

Value

The StatisticalModel object with the equations.

defineParameter *Define a parameter of a statistical model.*

Description

Define a parameter of a statistical model.

Usage

```
defineParameter(object, parameter)
```

Arguments

object	A StatisticalModel object.
parameter	An expression giving a parameter of the StatisticalModel object.

Value

Return StatisticalModel object with new parameters.

defineParameters *Define the parameters of a statistical model.*

Description

Define the parameters of a statistical model.

Usage

```
defineParameters(object, listOfParameters)
```

Arguments

object	A StatisticalModel object.
listOfParameters	A list of string giving the parameters of the StatisticalModel object.

Value

Return StatisticalModel object with new parameters.

defineStatisticalModel *Define the* StatisticalModel *object of the PFIMProject object.*

Description

Define the StatisticalModel object of the PFIMProject object.

Usage

```
defineStatisticalModel(object, value)
```

Arguments

object	A PFIMProject object.
value	A StatisticalModel or ODEStatisticalModel object.

Value

The StatisticalModel object of the PFIMProject object.

defineVariable *Define a variable in a statistical model.*

Description

Define a variable in a statistical model.

Usage

```
defineVariable(.Object, variable)
```

Arguments

.Object	A StatisticalModel object.
variable	A character string giving the variable to be defined.

Value

The StatisticalModel object with the new variable.

defineVariables *Define variables in a statistical model.*

Description

Define variables in a statistical model.

Usage

```
defineVariables(.Object, listOfVariables)
```

Arguments

.Object	A StatisticalModel object.
listOfVariables	A list of character string giving the variables to be defined.

Value

The StatisticalModel object with the new variables.

Design-class *Class "Design"*

Description

The class Design defines information concerning the parametrization of the designs.

Objects from the class Design

Objects form the class Design can be created by calls of the form Design(...) where (...) are the parameters for the Design objects.

Slots for the Design objects

isOptimalDesign: A Boolean for testing if the Design is optimal (isOptimalDesign=TRUE) or not.

name: A character string giving the name of the design - optional.

total_size: A numeric giving the total number of subjects in the design - optional.

arms: List of objects from the class Arm.

number_samples: A numeric giving the raint on the number of samples for one subject - optional. Default to the set of possible number of sampling points present in the sampling windows defining the different arms or the design spaces.

arms: A list of arm objects from the class Arm.

amountOfArm: A numeric giving the number of arms in the study.

optimizationResult: An optimization object from the class Optimization giving the results from the optimizsation process.

fimOfDesign: A character string giving the Fisher Information Matrix of the design (Population, Individual or Bayesian).

concentration: A list giving the result of the evaluaton for the responses.

sensitivityindices: A list giving the result of the sensitivity indices for the responses.

DesignConstraint-class *Class "DesignConstraint"*

Description

The class DesignConstraint defines information concerning the parametrization of the constraints on a design.

Objects from the class `DesignConstraint`

Objects form the class DesignConstraint can be created by calls of the form `DesignConstraint(...)` where (...) are the parameters for the `DesignConstraint` objects.

Slots for the `DesignConstraint` objects

`name`: A character string giving the name of the design - optional.

`PossibleArms`: A list of arms for optimization.

`totalNumberOfIndividuals`: A numeric giving the total number of individuals in the design.

`amountOfArm`: A numeric giving the number of arms in the design.

`samplingConstraints`: A list giving the sampling constraints for the design.

`administrationConstraints`: A list giving the administration constraints for the design.

`DiscreteConstraint-class` *Class "DiscreteConstraint" representing the constraints for a variable*

Description

The class `DiscreteConstraint` stores constraints for a variable. Constraints are given either as a continuous range (any value between a min and a max boundary is admissible) or a discrete set of values (any value belonging to the set is admissible)

Objects from the class `DiscreteConstraint`

Objects form the class `DiscreteConstraint` can be created by calls of the form `DiscreteConstraint(...)` where (...) are the parameters for the `DiscreteConstraint` Model.

Slots for the `DiscreteConstraint` objects

`type`: A character string, one of 'continuous' or 'discrete'.

`range`: A numeric vector with two values giving the min/max of the continuous range.

`discrete`: A numeric vector giving the set of possible values.

`Distribution-class` *Class "Distribution"*

Description

The class defines all the required methods for a distribution object. However, the actual functionality needs to implemented by the inheriting class.

Evaluate *Evaluate an* StatisticalModel *object.*

Description

Evaluate an StatisticalModel object.

Usage

```
Evaluate(object, administrations, sampling_times, cond_init, fim)
```

Arguments

object	A StatisticalModel object.
administrations	An Administration object.
sampling_times	A SamplingTimes object.
cond_init	A list for the initial conditions of the StatisticalModel object.
fim	FIM object.

Value

A fim object giving the Fisher Information Matrix of the StatisticalModel object.

EvaluateBayesianFIM *Evaluate design for each arm for a Bayesian FIM.*

Description

Evaluate design for each arm for a Bayesian FIM.

Usage

```
EvaluateBayesianFIM(object)
```

Arguments

object	A PFIMProject object.

Value

The PFIMProject object with the list designs that contains the evaluation of the Bayesian FIM of each design for each arm.

EvaluateDesign *Evaluate the design for each arm.*

Description

Evaluate the design for each arm.

Usage

```
EvaluateDesign(object, fimType, TheDesign)
```

Arguments

object	A PFIMProject object.
fimType	A character string giving the type of FIM: "Population", "Individual" or "Bayesian".
TheDesign	A Design object to be evaluated.

Value

The PFIMProject object with the list designs that contains the evaluation of each design for each arm.

EvaluateDesignForEachArm *Evaluate Design for each arm.*

Description

Evaluate Design for each arm.

Usage

```
EvaluateDesignForEachArm(object, statistical_model, fim)
```

Arguments

object	A Design object.
statistical_model	A statisticalModel object.
fim	A fim object.

Value

The object Design evaluated for each of its arm.

EvaluateErrorModelDerivatives

Evaluate the Error Model Derivatives.

Description

Evaluate the Error Model Derivatives.

Usage

```
EvaluateErrorModelDerivatives(object, f_x_i_theta)
```

Arguments

object	A Response object.
f_x_i_theta	The nonlinear structural model f_x_i_theta

Value

A list giving the error variance V_sig and sigma derivatives sigmaDerivatives of the model error.

EvaluateFIMsAndDesigns

Evaluate the FIMs and the Designs.

Description

Evaluate the FIMs and the Designs.

Usage

```
EvaluateFIMsAndDesigns(
  object,
  responseNumber,
  responseNames,
  doses,
  times,
  admin,
  samp,
  statistical_model,
  cond_init,
  totalNumberOfFIMs,
  typeFim
)
```

Arguments

object	An Optimization object.
responseNumber	A numeric giving the number of responses.
responseNames	A character string giving the name of the response
doses	A vector of numeric values giving the doses.
times	A vector of numeric values giving the times doses.
admin	An Administration object giving the administration parameters.
samp	An SamplingTimes object giving the sampling times parameters.
statistical_model	A statisticalModel object.
cond_init	cond_init
totalNumberOfFIMs	A numeric giving the total number of FIMs.
typeFim	A character strgin gving the type of the FIM.

Value

An Optimization object giving the results of the evaluation of the FIMs and the Designs.

EvaluateIndividualFIM *Evaluate design for each arm for a Individual FIM.*

Description

Evaluate design for each arm for a Individual FIM.

Usage

```
EvaluateIndividualFIM(object)
```

Arguments

object	PFIMProject object.

Value

The PFIMProject object with the list designs that contains the evaluation of the Individual FIM of each design for each arm.

EvaluateModel *Evaluate an analytic model.*

Description

Evaluate an analytic model.

Usage

```
EvaluateModel(object, samplingTimesModel, inputsModel, computeFIM)
```

Arguments

object	An object EvaluateModel.
samplingTimesModel	A vector containing the sampling times.
inputsModel	A list containing the models input.
computeFIM	A boolean for computing the FIM or not (plot or evaluation).

Value

A list containing the evaluated responses and the gradients.

EvaluateModelInfusion *Evaluate an analytic model in infusion.*

Description

Evaluate an analytic model in infusion.

Usage

```
EvaluateModelInfusion(object, samplingTimesModel, inputsModel, computeFIM)
```

Arguments

object	A ModelInfusionEquations object.
samplingTimesModel	A vector giving the sampling times of the model
inputsModel	A list containing the inputs used for the model evaluation.
computeFIM	A boolean if the FIM is computed or not.

Value

A list containing the evaluated responses and the gradients.

EvaluateModelODE *Evaluate an ODE model.*

Description

Evaluate an ODE model.

Usage

```
EvaluateModelODE(
  object,
  samplingTimesModel,
  cond_init_ode,
  inputsModel,
  parametersGradient,
  computeFIM
)
```

Arguments

`object`	An `ModelODEquations` object.
`samplingTimesModel`	A vector containing the sampling times.
`cond_init_ode`	A vector containing the initial conditions.
`inputsModel`	A list containing the inputs of the models.
`parametersGradient`	A list containing the gradients of the model.
`computeFIM`	A boolean for computing the FIM or not.

Value

A list containing the evaluated responses and the gradients.

EvaluateModelODEInfusion *Evaluate an ODE model in infusion*

Description

Evaluate an ODE model in infusion

Usage

```
EvaluateModelODEInfusion(
  object,
  samplingTimesModel,
  cond_init_ode,
  inputsModel,
  parametersGradient,
  computeFIM
)
```

Arguments

object	An EvaluateModelODEInfusion object.
samplingTimesModel	A vector containing the sampling times.
cond_init_ode	A vector containing the initial conditions.
inputsModel	A list containing the inputs of the models.
parametersGradient	A list containing the gradients of the model.
computeFIM	A boolean for computing the FIM or not.

Value

A list containing the evaluated responses and the gradients.

EvaluateODEErrorModelDerivatives

Evaluate the ODE Error Model Derivatives.

Description

Evaluate the ODE Error Model Derivatives.

Usage

```
EvaluateODEErrorModelDerivatives(object, f_x_i_theta)
```

Arguments

object	Response object.
f_x_i_theta	The nonlinear structural model f_x_i_theta

Value

A list giving the error variance V_sig and sigma derivatives sigmaDerivatives of the model error in ODE.

EvaluatePopulationFIM *Evaluate a design for each arm for a Population FIM.*

Description

Evaluate a design for each arm for a Population FIM.

Usage

```
EvaluatePopulationFIM(object)
```

Arguments

object	A PFIMProject object.

Value

The PFIMProject object with the list designs that contains the evaluation of the Population FIM of each design for each arm.

EvaluateStatisticalModel *Evaluate a statistical model for all the administrations and all the sampling times of an arm.*

Description

Evaluate a statistical model for all the administrations and all the sampling times of an arm.

Usage

```
EvaluateStatisticalModel(object, statistical_model, fim)
```

Arguments

object	An object Arm from the class Arm.
statistical_model	An object from the class StatisticalModel or ODEStatisticalModel.
fim	Character string giving the type of the Fisher Information Matrix: "PopulationFim", "IndividualFim", "BayesianFim".

Value

A list giving the evaluated Fisher Information Matrix, the concentration, the sensitivity indices and to sampling times used for plotting the outputs.

EvaluationModel *Evaluation for the model, analytic, ode, infusion*

Description

Evaluation for the model, analytic, ode, infusion

Usage

```
EvaluationModel(
  object,
  samplingTimesModel,
  cond_init_ode,
  inputsModel,
  parametersGradient,
  computeFIM
)
```

Arguments

object	A StatisticalModel object.
samplingTimesModel	A vector containing the sampling times of the model.
cond_init_ode	A vector containing the initial conditions for an ODE model.
inputsModel	A list containing the parameters used for the model evaluation.
parametersGradient	A list containing the parameters used for the evaluation of the gradient of the model
computeFIM	A boolean giving TRUE id the FIM is compute, FALSE otherwise.

Value

A list contaning a dataframe giving the results of the evaluation and a list giving the gradient of the model.

FedorovWynnAlgorithm-class *Class "FedorovWynnAlgorithm"*

Description

Class FedorovWynnAlgorithm represents an initial variable for ODE model.

Objects from the class `FedorovWynnAlgorithm`

Objects form the class `FedorovWynnAlgorithm` can be created by calls of the form `FedorovWynnAlgorithm(...)` where (...) are the parameters for the `FedorovWynnAlgorithm` objects.

Slots for `FedorovWynnAlgorithm` objects

`initialElementaryProtocols`: A list of vector for the intial elementary protocols.

`numberOfSubjects`: A vector for the number of subjects.

`proportionsOfSubjects`: A vector for the number of subjects.

`OptimalDesign`: A object Design giving the optimal Design.

`showProcess`: A boolean to show the process or not.

`FisherMatrix`: A vector giving the Fisher Information

`optimalFrequencies`: A vector of the optimal frequencies.

`optimalSamplingTimes`: A list of vectors for the optimal sampling times.

`optimalDoses`: A vector for the optimal doses.

`FedorovWynnAlgorithm_Rcpp` *Fedorov-Wynn algorithm in Rcpp.*

Description

Run the FedorovWynnAlgorithm in Rcpp

Usage

```
FedorovWynnAlgorithm_Rcpp(
  protocols_input,
  ndimen_input,
  nbprot_input,
  numprot_input,
  freq_input,
  nbdata_input,
  vectps_input,
  fisher_input,
  nok_input,
  protdep_input,
  freqdep_input
)
```

Arguments

protocols_input	parameter protocols_input
ndimen_input	parameter ndimen_input
nbprot_input	parameter nbprot_input
numprot_input	parameter numprot_input
freq_input	parameter freq_input
nbdata_input	parameter nbdata_input
vectps_input	parameter vectps_input
fisher_input	parameter fisher_input
nok_input	parameter nok_input
protdep_input	parameter protdep_input
freqdep_input	parameter freqdep_input

Value

A list ginving the results of the outputs of the FedorovWynn algorithm.

FillLibraryOfModels *Function "FillLibraryOfModels"*

Description

This function is used to define the models PK, PD and PKPD models and load these models in the library of models.

Usage

```
PFIMLibraryOfModels
```

Format

An object of class `LibraryOfModels` of length 1.

`Fim-class` *Class "Fim" representing the Fisher information matrix, a parent class used by three classes PopulationFim, IndividualFim and BayesianFim.*

Description

A class storing information regarding the Fisher computation matrix. Type of the Fisher information: population ("PopulationFIM"), individual ("IndividualFIM") or Bayesian ("BayesianFIM"). The computation method for population and Bayesian matrix is first order linearisation (FO).

Objects from the class

Objects form the class `Fim` can be created by calls of the form `Fim(...)` where (...) are the parameters for the `Fim` objects.

Slots for `Fim` objects

`isOptimizationResult`: A Boolean giving TRUE for an optimization result and FALSE an evaluation result.

`mfisher`: A matrix of numeric giving the Fisher information.

`omega`: A matrix of numeric giving the variances.

`mu`: A matrix of numeric giving the means.

`fim_comput_method`: Name of the method used to approximate the population matrix : character strings, 'FO'

`FinalizeFIMForOneElementaryDesign` *FinalizeFIMForOneElementaryDesign*

Description

FinalizeFIMForOneElementaryDesign

Finalize the Fim for one elementary design.

Usage

```
FinalizeFIMForOneElementaryDesign(object, arm)

## S4 method for signature 'PopulationFim'
FinalizeFIMForOneElementaryDesign(object, arm)
```

Arguments

`object`	A `Fim` object.
`arm`	A `Arm` object.

Value

A matrix of numeric `mfisher` giving the Fisher Information Matrix.

The Fim times size of the arm.

`fisher.simplex` *Compute the fisher.simplex*

Description

Compute the fisher.simplex

Usage

```
fisher.simplex(samplingTimes, data)
```

Arguments

`samplingTimes`	A `SamplnigTimes` object.
`data`	A list containing the design, arm, response names, statistical model, constraint and FIM.

Value

The fisher.simplex giving the evalation of the optimization criterion (i.e. D-criterion)

`fixedDoses` *Set the value for the fixed doses in the administration constraints.*

Description

Set the value for the fixed doses in the administration constraints.

Usage

```
fixedDoses(object, value)
```

Arguments

`object`	An object `AdministrationConstraint` from the class AdministrationConstraint.
`value`	A numeric vector giving the value of the fixed dose.

Value

The `AdministrationConstraint` object with its new value of the fixed dose.

FixTimeValues *Set the value for the fixed times.*

Description

Set the value for the fixed times.

Usage

```
FixTimeValues(object, value)
```

Arguments

object	A SamplingConstraint object.
value	A vector of numeric giving the values for the fixed times.

Value

The SamplingConstraint object with the values for the fixed times.

fun.amoeba *function fun.amoeba*

Description

function fun.amoeba

Usage

```
fun.amoeba(p, y, ftol, itmax, funk, data)
```

Arguments

p	input is a matrix p whose ndim+1 rows are ndim-dimensional vectors which are the vertices of the starting simplex.
y	vector whose components must be pre-initialized to the values of funk evaluated at the ndim+1 vertices (rows) of p.
ftol	the fractional convergence tolerance to be achieved in the function value.
itmax	maximal number of iterations.
funk	multidimensional function to be optimized.
data	a fixed set of data.

Value

A list containing the components of the optimized simplex.

g *Evaluation of the model error.*

Description

Evaluation of the model error.

Usage

```
g(object, f_x_i_theta)
```

Arguments

object	ModelError object.
f_x_i_theta	the nonlinear structural model f_x_i_theta.

Value

A numeric giving the evaluation of the error model.

getAdministration *Get the parameters of the administration for an arm.*

Description

Get the parameters of the administration for an arm.

Usage

```
getAdministration(object)
```

Arguments

object	An object Arm from the class Arm.

Value

A list administrations of objects from the class Administration class giving the parameters of the administration for the object Arm.

`getAdministrationByOutcome`

Get the parameters of the administration for an arm given the response of the model.

Description

Get the parameters of the administration for an arm given the response of the model.

Usage

```
getAdministrationByOutcome(object, outcome)
```

Arguments

`object`	An object `Arm` from the class Arm.
`outcome`	A character string giving the the name of the response of the model.

Value

A list of objects from `Administration` class giving the parameters of the administration for the object `Arm`.

`getAdministrationConstraint`

Get the constraints on the administration for a `DesignConstraint` *object.*

Description

Get the constraints on the administration for a `DesignConstraint` object.

Usage

```
getAdministrationConstraint(object)
```

Arguments

`object`	A `DesignConstraint` object.

Value

The list of constraints on the administration given by `administrationConstraint` for a `DesignConstraint` object.

getallowedContinuousSamplingTimes

Get the allowed Continuous SamplingTimes

Description

Get the allowed Continuous SamplingTimes

Usage

```
getallowedContinuousSamplingTimes(object)
```

Arguments

object	A SamplingConstraint object.

Value

A list giving the allowed Continuous SamplingTimes for the SamplingConstraint object.

getallowedDiscretSamplingTimes

Get the allowed discret sampling simes

Description

Get the allowed discret sampling simes

Usage

```
getallowedDiscretSamplingTimes(object)
```

Arguments

object	SamplingConstraint object.

Value

The allowed discret sampling times of the SamplingConstraint object.

getAllowedDose *Get the constraints on allowed dose*

Description

Get the constraints on allowed dose

Usage

```
getAllowedDose(object)
```

Arguments

object	An object Administration from the class Administration.

Value

The vector allowed_dose of the numeric values of the constraints on allowed dose.

getAllowedDoses *Get the vector of allowed amount of dose.*

Description

Get the vector of allowed amount of dose.

Usage

```
getAllowedDoses(object)
```

Arguments

object	An object AdministrationConstraint from the class AdministrationConstraint.

Value

A vector AllowedDoses giving the allowed amount of dose.

`getAllowedTime` *Get the constraints on allowed times.*

Description

Get the constraints on allowed times.

Usage

```
getAllowedTime(object)
```

Arguments

`object`	An object `Administration` from the class Administration.

Value

The numeric vector `allowed_time` giving the constraints on allowed times.

`getAllowedTinf` *Get the constraints on Tinf.*

Description

Get the constraints on Tinf.

Usage

```
getAllowedTinf(object)
```

Arguments

`object`	An object `Administration` from the class Administration.

Value

The vector `allowed_tinf` giving the constraints on Tinf.

getAmountDose *Get the amount of doses.*

Description

Get the amount of doses.

Usage

```
getAmountDose(object)
```

Arguments

object	An object Administration from the class Administration.

Value

The numeric amount_dose giving the amount of doses.

getAmountOfArms *Get the amount of arms in a Design.*

Description

Get the amount of arms in a Design.

Usage

```
getAmountOfArms(object)
```

Arguments

object	A Design object.

Value

A numeric amountOfArm giving the number of arms in the design.

getArms *Get the arms of a design.*

Description

Get the arms of a design.

Usage

```
getArms(object)
```

Arguments

object	A Design object.

Value

A list arms of the arms of a design.

getArmSize *Get the size of an arm.*

Description

Get the size of an arm.

Usage

```
getArmSize(object)
```

Arguments

object	An object Arm from the class Arm.

Value

A numeric arm_size giving the size of the object Arm.

getCError *Get the* CError *of a* ModelError *object.*

Description

Get the CError of a ModelError object.

Usage

```
getCError(object)
```

Arguments

object	ModelError object.

Value

The numeric c_error giving the CError.

getCondInit *Get the initial conditions in a arm for an ODE model*

Description

Get the initial conditions in a arm for an ODE model

Usage

```
getCondInit(object)
```

Arguments

object	An object Arm from the class Arm

Value

A list cond_init giving the initial conditions for ODE model in the object Arm

`getConditionNumberMatrix` *Get the Condition Number Matrix of the Fisher Information Matrix for a* `Fim` *object..*

Description

Get the Condition Number Matrix of the Fisher Information Matrix for a `Fim` object..

Usage

```
getConditionNumberMatrix(object, FixedEffectParameterNumber)
```

Arguments

`object`	A `Fim` object.
`FixedEffectParameterNumber`	A numerical giving the number of Fixed Effect Parameters.

Value

A matrix `conditionNumbers` of numerical values giving the Condition Number Matrix the min, max and min/max for the FixedEffects and VarianceComponents.

`getContentsLibraryOfModels` *Get the content of the* `LibraryOfModels` *object.*

Description

Get the content of the `LibraryOfModels` object.

Usage

```
getContentsLibraryOfModels(object)
```

Arguments

`object`	A `LibraryOfModels` object.

Value

A list `contentsLibraryOfModels` giving the two lists that respectively corresponds to the two librairies of the PK and PD models contained in the `LibraryOfModels`.

getCorr — *Get the correlation matrix of the Fisher Information Matrix for a* Fim *object.*

Description

Get the correlation matrix of the Fisher Information Matrix for a Fim object.

Usage

```
getCorr(object)
```

Arguments

object	A Fim object.

Value

A matrix of numerical values corr_mat giving the correlation matrix from the Fisher Information Matrix.

getDcriterion — *Get the D-criterion for a* Fim *object.*

Description

Get the D-criterion for a Fim object.

Usage

```
getDcriterion(object)
```

Arguments

object	A Fim object.

Value

A numeric Dcriterion giving the D-criterion of a Fisher Information Matrix.

`getDerivate` *Get the derivate of an equation of a* `ModelEquations` *object.*

Description

Get the derivate of an equation of a `ModelEquations` object.

Usage

```
getDerivate(object, equationName, parameter)

## S4 method for signature 'ModelEquations'
getDerivate(object, equationName, parameter)
```

Arguments

`object`	A `ModelEquations` object.
`equationName`	A character string giving the name of the response of the equations.
`parameter`	An `ModelParameter` object.

Value

A list of expression of the derivate of an equation of a `ModelEquations`

`getDerivatesAdjustedByDistribution` *Get the derivates adjusted by distribution of a* `ModelParameter` *object.*

Description

Get the derivates adjusted by distribution of a `ModelParameter` object.

Usage

```
getDerivatesAdjustedByDistribution(object, df_total)
```

Arguments

`object`	`ModelParameter` object.
`df_total`	df_total

Value

A list of expression giving the derivates adjusted by distribution `distribution`, the mu `distribution` and the dftotal `distribution` of a `ModelParameter` object.

getDerivatives *Get the derivatives of a* ModelODEquations *object.*

Description

Get the derivatives of a ModelODEquations object.

Usage

```
getDerivatives(object)
```

Arguments

object	An ModelODEquations object.

Value

A list of expression derivatives giving the derivatives of a ModelODEquations object.

getDescription *Get the description of FIM.*

Description

Get the description of FIM.

Get the description BayesianFim object.

Get the type of the Fim.

Usage

```
getDescription(object)

## S4 method for signature 'IndividualFim'
getDescription(object)

## S4 method for signature 'BayesianFim'
getDescription(object)

## S4 method for signature 'PopulationFim'
getDescription(object)
```

Arguments

object	A Fim object.

Value

A character string that tells you that is a Fisher information matrix.

Return a string giving the type of the Fim.

A string giving the description of the object `BayesianFim`.

A string giving the type of the Fim.

`getDesign` *Get the design of* `PFIMProject` *object.*

Description

Get the design of `PFIMProject` object.

Usage

```
getDesign(object)
```

Arguments

`object`	`PFIMProject` object.

Value

The list `design` of the designs in the `PFIMProject` object.

`getDeterminant` *Get the Determinant of a Fisher Information Matrix.*

Description

Get the Determinant of a Fisher Information Matrix.

Usage

```
getDeterminant(object)
```

Arguments

`object`	`Fim` object.

Value

A numeric `Det` giving the determinant of a Fisher Information Matrix.

getDiscret *Get the set of possible values for a* DiscreteConstraint *object.*

Description

Get the set of possible values for a DiscreteConstraint object.

Usage

```
getDiscret(object)
```

Arguments

object	A DiscreteConstraint object.

Value

A numeric vector discret giving the set of possible values.

getDistribution *Get the distribution of a* ModelParameter *object.*

Description

Get the distribution of a ModelParameter object.

Usage

```
getDistribution(object)
```

Arguments

object	ModelParameter object.

Value

The distribution given by distribution of a ModelParameter object.

getDoseOptimisability *Get the boolean* Optimisability *for optimizable dose.*

Description

Get the boolean Optimisability for optimizable dose.

Usage

```
getDoseOptimisability(object)
```

Arguments

object	An object AdministrationConstraint from the class AdministrationConstraint.

Value

The boolean Optimisability giving FALSE for fixed dose, TRUE for an optimizable dose.

getDVSigma *Get the DV Sigma of a* ModelError *object.*

Description

Get the DV Sigma of a ModelError object.

Usage

```
getDVSigma(object, parameter)
```

Arguments

object	A ModelError object.
parameter	An string giving a parameter of the model error.

Value

A list giving the derivates Sigma for a paramerer.

getEigenValue *Get the eigen values of the Fisher Information Matrix for a* Fim *object.*

Description

Get the eigen values of the Fisher Information Matrix for a Fim object.

Usage

```
getEigenValue(object)
```

Arguments

object	A Fim object.

Value

A vector of numerical values EV giving the eigen values of the Fim object.

getElementaryProtocols *Get the matrix of all the combination of the elementary protocols.*

Description

Get the matrix of all the combination of the elementary protocols.

Usage

```
getElementaryProtocols(object)
```

Arguments

object	An Optimization object.

Value

A matrix giving all the combination of the elementary protocols.

getEquation *Get the equation of a* ModelError *object by their names.*

Description

Get the equation of a ModelError object by their names.

Get the equation of a ModelEquations object with respect to its name.

Usage

```
getEquation(object, equationName)

## S4 method for signature 'ModelEquations'
getEquation(object, equationName)
```

Arguments

object	A ModelEquations object.
equationName	A character string giving the name of the response of the equations.

Value

An expression equation giving the equation of the model error.

An expression equations giving the equation of a model with respect to its name.

getEquations *Get the equations of a* ModelEquations *object.*

Description

Get the equations of a ModelEquations object.

Get the equations of a Model object after changing variable in PK model.

Get the equations of a PKModel object.

Get the equations of a PKPDModel object.

Usage

```
getEquations(object)

## S4 method for signature 'Model'
getEquations(object)

## S4 method for signature 'PKModel'
getEquations(object)
```

```
## S4 method for signature 'PKPDModel'
getEquations(object)
```

Arguments

object	A PKPDModel object.

Value

A list of expression equations giving the equations of the model after the change of variable in the model.

A list of expression giving the equations of a Model object after the changing variable in PK model.

A list of expressions giving the equations of a PKModel object.

A list of expressions giving the equations of a PKPDModel object.

---

getEquationsModel *Get the equations of a* Model *object.*

Description

Get the equations of a Model object.

Usage

```
getEquationsModel(object)
```

Arguments

object	A Model object.

Value

A list of expression equationsModel giving the equations of a Model object.

getEquationsModelPKPD *Get the equations of the PK and PD models of a* ModelEquations *object.*

Description

Get the equations of the PK and PD models of a ModelEquations object.

getEquationsModelPKPD

Usage

```
getEquationsModelPKPD(modelEquationsPKmodel, modelEquationsPDmodel)

getEquationsModelPKPD(modelEquationsPKmodel, modelEquationsPDmodel)
```

Arguments

modelEquationsPKmodel	An expression giving the equation of the PK model.
modelEquationsPDmodel	An expression giving the equation of the PD model.

Value

A list output giving:

- expressions for the equations of the PK and PD models, for an analytic PK and PD models.
- expressions for the equations of the PK and the infusion equations PD models, for a PK model in infusion.
- expressions for the equations of the PK and PD models, for an ODE PK and PD models.

getEquationsStatisticalModel *Get the equations of a statistical model.*

Description

Get the equations of a statistical model.

Usage

```
getEquationsStatisticalModel(object)
```

Arguments

object	A StatisticalModel object.

Value

A ModelEquationsg object giving the equations of the StatisticalModel object.

---

getErrorModelParameters

Get parameters of the error model of a ModelError *object.*

Description

Get parameters of the error model of a ModelError object.

Usage

```
getErrorModelParameters(object)
```

Arguments

object	A ModelError object.

Value

A list of string giving the parameters of the error model.

---

getErrorModelStandardErrors

Get the SE and RSE of the parameters.

Description

Get the SE and RSE of the parameters.

Usage

```
getErrorModelStandardErrors(object, fim)
```

Arguments

object	A StatisticalModel object.
fim	A Fim object giving the Fisher Information Matrix.

Value

A dataframe giving he SE and RSE of the parameters.

getEvaluationDesign *Get the evaluated concentration and sensitivity indices of a design.*

Description

Get the evaluated concentration and sensitivity indices of a design.

Usage

```
getEvaluationDesign(object)
```

Arguments

object	A Design object.

Value

The object Design evaluated for each of its arm.

getEvaluationResponses *Get the evaluated responses of the model.*

Description

Get the evaluated responses of the model.

Usage

```
getEvaluationResponses(object)
```

Arguments

object	A Design object.

Value

The object Design evaluated for each of its arm.

getFim *Get the Fisher Information Matrix.*

Description

Get the Fisher Information Matrix.

Usage

```
getFim(object, ...)
```

Arguments

object	A PFIMProject object.
...	A list giving the index of the Fim.

Value

A Fim object giving the Fisher Information Matrix of a design.

getFimOfDesign *Get the Fisher Information Matrix of a design.*

Description

Get the Fisher Information Matrix of a design.

Usage

```
getFimOfDesign(object)
```

Arguments

object	A Design object.

Value

A Fim object giving the Fisher Information Matrix of a design.

getFims *Get the Fisher Information Matrices.*

Description

Get the Fisher Information Matrices.

Usage

```
getFims(object)
```

Arguments

object	A PFIMProject object.

Value

A list fimList giving the Fisher Information Matrices for all the designs of a PFIMProject project.

getFisherMatrices *Get the fim matrices from all designs of a PFIMProject object.*

Description

Get the fim matrices from all designs of a PFIMProject object.

Usage

```
getFisherMatrices(object)
```

Arguments

object	A PFIMProject object.

Value

A list of matrices fimList giving the Fisher Information Matrix of all the designs of a PFIMProject project.

getFixedParameters *Get the fixed and non fixed model parameters.*

Description

Get the fixed and non fixed model parameters.

Usage

```
getFixedParameters(object)
```

Arguments

object	A StatisticalModel object.

Value

A list that contains the name and indices of the fixed parameters.

getfixedTimes *Get the fixed times.*

Description

Get the fixed times.

Usage

```
getfixedTimes(object)
```

Arguments

object	A SamplingConstraint object.

Value

The fixed times of the SamplingConstraint object.

getInfusionEquations *Get the Infusion Equations.*

Description

Get the Infusion Equations.

Usage

```
getInfusionEquations(object)
```

Arguments

object	A ModelInfusionEquations object.

Value

A list equations of the expressions giving the infusion equations of the ModelInfusionEquations object

getInitialTime *Get the initial time of a* SamplingTimes *object.*

Description

Get the initial time of a SamplingTimes object.

Usage

```
getInitialTime(object)
```

Arguments

object	SamplingTimes object.

Value

A numeric initialTime giving the inital time of a SamplingTimes object.

getMfisher *Get the Fisher Information Matrix.*

Description

Get the Fisher Information Matrix.

Usage

```
getMfisher(object)
```

Arguments

object	A Fim object.

Value

A matrix of numeric mfisher giving the Fisher Information Matrix.

getModel *Get a model of the* LibraryOfModels *object.*

Description

Get a model of the LibraryOfModels object.

Usage

```
getModel(object, ...)
```

Arguments

object	A LibraryOfModels object.
...	The three-dots for passing one name or two names as arguments. One name to get a PK or PD model and two names for the PK and PD models of a PKPD model.

Value

Return a Model object giving a PK or PD model.

getModelError *Get the model error.*

Description

Get the model error.

Usage

```
getModelError(object)
```

Arguments

object	A Response object.

Value

The object

getModelName *Get the name of the* Model *object.*

Description

Get the name of the Model object.

Usage

```
getModelName(object)
```

Arguments

object	A Model object.

Value

A character string modelName giving the name of the Model object.

getModelNameList *Get the list of all the models in the* LibraryOfModels *object.*

Description

Get the list of all the models in the LibraryOfModels object.

Usage

```
getModelNameList(object)
```

Arguments

object	A LibraryOfModels object.

Value

The list ModelNameList of the names of the PK, PD and PKPD models in the LibraryOfModels object.

getModelParameters *Get the model parameters of a statistical model.*

Description

Get the model parameters of a statistical model.

Usage

```
getModelParameters(object)
```

Arguments

object	getModelParameters object.

Value

A list giving the model parameters.

getMu *Get* mu *for a* ModelParameter *object.*

Description

Get mu for a ModelParameter object.

Usage

```
getMu(object)
```

Arguments

object	ModelParameter object.

Value

A numeric mu giving the value of the mean mu for a ModelParameter object.

getNameAdministration *Get the name of the outcome of an object* Administration.

Description

Get the name of the outcome of an object Administration.

Usage

```
getNameAdministration(object)
```

Arguments

object	An object Administration from the class Administration.

Value

A character string giving the name for the response of the object Administration.

getNameArm *Get the name of the arm.*

Description

Get the name of the arm.

Usage

```
getNameArm(object)
```

Arguments

object	An object Arm from the class Arm.

Value

A character string name giving the name of the object Arm.

getNameDesign *Get the name of the design.*

Description

Get the name of the design.

Usage

```
getNameDesign(object)
```

Arguments

object	Design object.

Value

A character string name giving the name of design.

getNameDesignConstraint *Get the name of the* DesignConstraint *object.*

Description

Get the name of the DesignConstraint object.

Usage

```
getNameDesignConstraint(object)
```

Arguments

object	A DesignConstraint object.

Value

A character string name giving the name of DesignConstraint object.

getNameModelParameter *Get the name of a* ModelParameter *object.*

Description

Get the name of a ModelParameter object.

Usage

```
getNameModelParameter(object)
```

Arguments

object	ModelParameter object.

Value

A character string name giving the name of a ModelParameter object.

getNameModelVariable *Get the name of the initial variable for an ODE model.*

Description

Get the name of the initial variable for an ODE model.

Usage

```
getNameModelVariable(object)
```

Arguments

object	ModelVariable object.

Value

A character string name giving the name of the initial variable for an ODE model.

getNamePFIMProject *Get the name of a PFIMProject project.*

Description

Get the name of a PFIMProject project.

Usage

```
getNamePFIMProject(object)
```

Arguments

object	A PFIMProject object.

Value

The character string name giving the name of a PFIMProject project.

getNameResponse *Get the name of the response of the model.*

Description

Get the name of the response of the model.

Usage

```
getNameResponse(object)
```

Arguments

object	A Response object.

Value

A character string name giving the name of the response of the model.

getNameSampleTime *Get the name of the response of the* SamplingTimes *object.*

Description

Get the name of the response of the SamplingTimes object.

Usage

```
getNameSampleTime(object)
```

Arguments

object	A SamplingTimes object.

Value

A character string outcome giving the name of the response of the model.

getNumberOfDoses *Get the vector* AllowedDoses *of allowed amount of dose.*

Description

Get the vector AllowedDoses of allowed amount of dose.

Usage

```
getNumberOfDoses(object)
```

Arguments

object	An object AdministrationConstraint from the class AdministrationConstraint.

Value

The numeric AllowedDoses giving the number of allowed amount of doses in the object AdministrationConstraint.

getNumberOfParameter *Get the number of parameters of a* ModelError *object.*

Description

Get the number of parameters of a ModelError object.

Usage

```
getNumberOfParameter(object)
```

Arguments

object	A ModelError object.

Value

A numeric giving the number of parameters.

getNumberOfParameters *Get the number of parameters of a* ModelEquations *object.*

Description

Get the number of parameters of a ModelEquations object.

Usage

```
getNumberOfParameters(object)
```

Arguments

object	A ModelEquations object.

Value

A numeric allParameters giving the number of parameters.

getNumberOfSamplings *Get the number of sampling times in a arm.*

Description

Get the number of sampling times in a arm.

Usage

```
getNumberOfSamplings(object)
```

Arguments

object	An object Arm from the class Arm.

Value

A numeric giving the number of sampling times in the Arm object.

getnumberOfSamplingTimes *Get the number of sampling times.*

Description

Get the number of sampling times.

Usage

```
getnumberOfSamplingTimes(object)
```

Arguments

object	A SamplingConstraint object.

Value

A numeric giving the number of sampling times in the SamplingConstraint object.

getNumberSamples *Get the number of sampled in a Design.*

Description

Get the number of sampled in a Design.

Usage

```
getNumberSamples(object)
```

Arguments

object	Design object.

Value

A numeric number_samples giving the number of sample of the design.

getNumberTime *Get the number of times in a* SamplingTimes *object.*

Description

Get the number of times in a SamplingTimes object.

Usage

```
getNumberTime(object)
```

Arguments

object	A SamplingTimes object.

Value

A numeric giving the number of times.

getOmega *Get* Omega *of a* ModelParameter *object.*

Description

Get Omega of a ModelParameter object.

Usage

```
getOmega(object)
```

Arguments

object	ModelParameter object.

Value

A numeric omega giving the variance Omega of a ModelParameter object.

getOptimalDesign *Get the optimal design.*

Description

Get the optimal design.

Usage

```
getOptimalDesign(object)
```

Arguments

object	An Design object.

Value

An Design object giving the optimal design.

getOptimisability *Get the optimisability of a* SamplingConstraint *object.*

Description

Get the optimisability of a SamplingConstraint object.

Usage

```
getOptimisability(object)
```

Arguments

object	SamplingConstraint object.

Value

A boolean giving the optimisability of a SamplingConstraint object.

getOptimizationResult *Get the results of the optimization process.*

Description

Get the results of the optimization process.

Usage

```
getOptimizationResult(object)
```

Arguments

object	Design object.

Value

A Optimization object giving the results of the optimization process.

getParameters *Get the parameters of a* ModelEquations *object.*

Description

Get the parameters of a ModelEquations object.

Usage

```
getParameters(object)
```

Arguments

object	A ModelEquations object.

Value

A vector allParameters giving the parameters of the model.

getParametersOdeSolver *Get parameters for the ode solver*

Description

Get parameters for the ode solver

Usage

```
getParametersOdeSolver(object)
```

Arguments

object	A StatisticalModel object.

Value

The parameters for the ode solver.

getPDModel *Get a PD model from a* PKPDModel *object.*

Description

Get a PD model from a PKPDModel object.

Usage

```
getPDModel(object)
```

Arguments

object	PKPDModel object.

Value

A Model object giving the pdModel from the PKPD model.

getPKModel *Get a PK model from a* PKPDModel *object.*

Description

Get a PK model from a PKPDModel object.

Usage

```
getPKModel(object)
```

Arguments

object	PKPDModel object.

Value

A Model object giving the pkModel from the PKPD model.

getPKPDModel *Get a PKPD model of the* LibraryOfModels *object.*

Description

Get a PKPD model of the LibraryOfModels object.

Usage

```
getPKPDModel(object, namePKModel, namePDModel)
```

Arguments

object	A LibraryOfModels object.
namePKModel	A character string giving the name of the PK model.
namePDModel	A character string giving the name of the PD model.

Value

Return a Model giving the PKPD model consisting of the PK and PD models named namePKModel and namePDModel respectively.

getRange *Get the range of a* ContinuousConstraint *object.*

Description

Get the range of a ContinuousConstraint object.

Usage

```
getRange(object)
```

Arguments

object	AContinuousConstraint object.

Value

A numeric range giving the range of a ContinuousConstraint object.

getResponseIndice *Get the index of the response of a* ModelEquations *object.*

Description

Get the index of the response of a ModelEquations object.

Usage

```
getResponseIndice(object, equationName)
```

Arguments

object	A ModelEquations object.
equationName	A character string giving the name of the response of the equations.

Value

A numeric giving the index of the equation in the model.

getResponseName *Get the name of the response for the administration constraints.*

Description

Get the name of the response for the administration constraints.

Get the name of the response for the SamplingConstraint.

Usage

```
getResponseName(object)

## S4 method for signature 'SamplingConstraint'
getResponseName(object)
```

Arguments

object	SamplingConstraint object.

Value

The character string response giving the name of the response of the object AdministrationConstraint object.

The character string response giving the name of the response for the SamplingConstraint object.

getResponseNameByIndice *Get the response name given the indice of the response.*

Description

Get the response name given the indice of the response.

Usage

```
getResponseNameByIndice(object, outcomeIndice)
```

Arguments

object	An object Arm from the class Arm.
outcomeIndice	A numeric giving the indice of the response in the Arm object.

Value

A character string giving the name of the response.

getResponsesStatisticalModel *Get the responses of a statistical model.*

Description

Get the responses of a statistical model.

Usage

```
getResponsesStatisticalModel(object)
```

Arguments

object	A getResponsesStatisticalModel object.

Value

A list giving the responses of a statistical model.

getSampleTime *Get the sample time of the response of the* SamplingTimes *object.*

Description

Get the sample time of the response of the SamplingTimes object.

Usage

```
getSampleTime(object)
```

Arguments

object	A getSampleTime object.

Value

A vector sample_time giving the sample time.

getSamplingConstraints

Get the constraints on the sampling for a `DesignConstraint` *object.*

Description

Get the constraints on the sampling for a `DesignConstraint` object.

Usage

```
getSamplingConstraints(object, responseName)
```

Arguments

`object`	`DesignConstraint` object.
`responseName`	A character string giving the name of the response.

Value

The lists of constraints `samplingConstraint` for a `DesignConstraint` object.

getSamplingConstraintsInArm

Get the sampling constraints of an arm.

Description

Get the sampling constraints of an arm.

Usage

```
getSamplingConstraintsInArm(object, responseName)
```

Arguments

`object`	An object `Arm` from the class Arm.
`responseName`	A character string giving the name of the response.

Value

An object `constraintsOfTheResponse` from the class `SamplingConstraint`. giving the sampling constraints of the object `Arm`.

getSamplings *Get the vectors of sampling times for an arm.*

Description

Get the vectors of sampling times for an arm.

Usage

```
getSamplings(object)
```

Arguments

object	An object Arm from the class Arm.

Value

A list of objects samplings from the class SamplingTimes giving the vector. of sampling times for the object Arm.

getSE *Get the Standard Errors for a* Fim *object..*

Description

Get the Standard Errors for a Fim object..

Usage

```
getSE(object)
```

Arguments

object	A Fim object.

Value

A vector of numerical values giving the Standard Errors from the Fisher Information Matrix.

getShrinkage *Calculates the shrinkage of individual parameters from a* BayesianFim *object.*

Description

Calculates the shrinkage of individual parameters from a BayesianFim object.

Usage

```
getShrinkage(object)
```

Arguments

object	An object BayesianFim from the class BayesianFim.

Value

A numeric vector giving the shrinkage of individual parameters from a Bayesian matrix.

getSig *Get the values for Sigma derivatives* DVSigma *for the* ModelError *object.*

Description

Get the values for Sigma derivatives DVSigma for the ModelError object.

Usage

```
getSig(object, f_x_i_theta)
```

Arguments

object	ModelError object.
f_x_i_theta	the nonlinear structural model f_x_i_theta

Value

A list indexed with the parameters giving the values for the derivatives with respect to each parameters of the model error.

getSigmaInter *Get the* sigma_inter *of a* ModelError *object.*

Description

Get the sigma_inter of a ModelError object.

Usage

```
getSigmaInter(object)
```

Arguments

object	A ModelError object.

Value

A numeric sigma_inter giving the sigma_inter.

getSigmaNames *Get the names for the error sigma inter.*

Description

Get the names for the error sigma inter.

Get the names of the variances.

Get the names of the variances.

Get the names of the variances.

Get the names of the variances.

Get the variances sigma_{inter} and sigma_{slope}.

Get the Sigma Names.

Usage

```
getSigmaNames(object)

## S4 method for signature 'Combined1'
getSigmaNames(object)

## S4 method for signature 'Combined1c'
getSigmaNames(object)

## S4 method for signature 'Combined2c'
getSigmaNames(object)
```

```
## S4 method for signature 'Combined2'
getSigmaNames(object)

## S4 method for signature 'Constant'
getSigmaNames(object)

## S4 method for signature 'Response'
getSigmaNames(object)
```

Arguments

object	A Response object.

Value

A character giving the names for the sigma inter of the error model.

The character string sigmaNames giving the names of the variances.

The character string sigmaNames giving the names of the variances.

The character string sigmaNames giving the names of the variances.

The character string sigmaNames giving the names of the variances.

The variances sigma_{inter} and sigma_{slope}.

A character string sigmaNames giving the names of the sigma.

---

getSigmaSlope *Get the* sigma_slope *of a* ModelError *object.*

Description

Get the sigma_slope of a ModelError object.

Usage

```
getSigmaSlope(object)
```

Arguments

object	An ModelError object.

Value

The numeric sigma_slope giving the sigma_slope.

getSigmaValues *Get the values of the variances* sigma_inter *and* sigma_slope.

Description

Get the values of the variances sigma_inter and sigma_slope.

Get the values of the variances sigma_inter and sigma_slope.

Get the values of the variances sigma_inter and sigma_slope.

Get the values of the variances sigma_inter and sigma_slope.

Get the values of the variances sigma_inter and sigma_slope.

Get the values of the variances sigma_{inter} and sigma_{slope}.

Usage

```
getSigmaValues(object)

## S4 method for signature 'Combined1'
getSigmaValues(object)

## S4 method for signature 'Combined1c'
getSigmaValues(object)

## S4 method for signature 'Combined2c'
getSigmaValues(object)

## S4 method for signature 'Combined2'
getSigmaValues(object)

## S4 method for signature 'Constant'
getSigmaValues(object)
```

Arguments

object	A Combined1 object.

Value

A numeric giving the values for the sigma inter of the error model.

A numeric vector giving the values of the variances sigma_inter and sigma_slope.

A numeric vector giving the values of the variances sigma_inter and sigma_slope.

A numeric vector giving the values of the variances sigma_inter and sigma_slope.

A numeric vector giving the values of the variances sigma_inter and sigma_slope.

A vector giving the values of the variances sigma_{inter} and sigma_{slope}.

getStatisticalModel *Get the* StatisticalModel *object of the PFIMProject object.*

Description

Get the StatisticalModel object of the PFIMProject object.

Usage

```
getStatisticalModel(object)
```

Arguments

object	A PFIMProject object.

Value

Return the object statistical_model of the StatisticalModel of the PFIMProject object.

getStatisticalModelStandardErrors *Get the SE of* IndividualFim *object.*

Description

Get the SE of IndividualFim object.

Compute expected standard error data frame.

Usage

```
getStatisticalModelStandardErrors(object, modelParameters)

## S4 method for signature 'IndividualFim'
getStatisticalModelStandardErrors(object, modelParameters)

## S4 method for signature 'PopulationFim'
getStatisticalModelStandardErrors(object, modelParameters)
```

Arguments

object	A Fim object.
modelParameters	A character string giving the model parameters.

Value

A data frame giving the expected standard error.

A list giving the fixed effects of `IndividualFim` object.

A data frame giving the expected standard error.

---

getTau *Get the frequency* tau.

Description

Get the frequency `tau`.

Usage

```
getTau(object)
```

Arguments

`object`	An object `Administration` from the class Administration.

Value

The numeric `tau` giving the frequency `tau`.

---

getTimeDose *Get the times vector when doses are given.*

Description

Get the times vector when doses are given.

Usage

```
getTimeDose(object)
```

Arguments

`object`	An object `Administration` from the class Administration.

Value

The vector `time_dose` giving the times when the doses are given.

getTinf *Get the infusion duration.*

Description

Get the infusion duration.

Usage

```
getTinf(object)
```

Arguments

object	An object Administration from the class Administration.

Value

The numeric Tinf giving the infusion duration Tinf.

getTotalNumberOfIndividuals *get the total number of individuals in a* DesignConstraint *object.*

Description

get the total number of individuals in a DesignConstraint object.

Usage

```
getTotalNumberOfIndividuals(object)
```

Arguments

object	DesignConstraint object.

Value

The DesignConstraint object with the total number of individual.

getTotalSize *Get the total size of a design.*

Description

Get the total size of a design.

Usage

```
getTotalSize(object)
```

Arguments

object	Design object.

Value

A numeric total_size giving the size of a design.

getWeightFrame *Get the frame with weight vector after optimisation.*

Description

Get the frame with weight vector after optimisation.

Usage

```
getWeightFrame(object)
```

Arguments

object	A MultiplicativeAlgorithm object.

Value

The data frame armFrame with weight vector after optimisation.

getWeights *Get the weights for the optimal designs.*

Description

Get the weights for the optimal designs.

Usage

```
getWeights(object)
```

Arguments

object	A PFIMProject object.

Value

A data frame weights giving the weights of the optimal designs.

IndividualFim-class *Class "individualFim" representing the individual Fisher information matrix*

Description

A class storing information regarding the individual Fisher computation matrix.

Objects from the class

IndividualFim objects are typically created by calls to {fim} or {pfim} and contain the following slots:

IndividualFim Create a new object {Fim}

IndividualFIMEvaluateVariance
Evaluate the individual FIM variance.

Description

Evaluate the individual FIM variance.

Usage

```
IndividualFIMEvaluateVariance(
  object,
  equations,
  model_parameters,
  administrations,
  sampling_times,
  df_total,
  errorVariances,
  sigmaDerivatives
)
```

Arguments

object	A Response object.
equations	An object of class Response containing the name of the reponse and the equation of the model error.
model_parameters	An object of class ModelParameters containing the values and the distributions of the model parameters.
administrations	An object of class Administration containing the parametrization for the administration of the model.
sampling_times	An object of class SamplingTimes containing the parametrization for the sampling times of the model.
df_total	parameter df_total
errorVariances	parameter errorVariances
sigmaDerivatives	parameter sigmaDerivatives

Value

A list giving VDist and MF_var.

is.multidose *Test if an object* Administration*for a model is multi-doses or not.*

Description

Test if an object Administrationfor a model is multi-doses or not.

Usage

```
is.multidose(object)
```

Arguments

object	An object Administration from the class Administration.

Value

A boolean that gives TRUE if the administration is multi-doses, FALSE otherwise.

isFixed *Boolean to set if a model parameters is fixed or not.*

Description

Boolean to set if a model parameters is fixed or not.

Usage

```
isFixed(object)
```

Arguments

object	ModelParameter object.

Value

A boolean fixed giving TRUE if the model parameters is fixed, or FALSE is this parameters remain to be estimated.

isFixedMu *Boolean to set if* mu *is fixed or not.*

Description

Boolean to set if mu is fixed or not.

Usage

```
isFixedMu(object)
```

Arguments

object	ModelParameter object.

Value

A boolean isFixedMu giving TRUE if mu is fixed, FALSE otherwise.

isLessThanDelay *Set the constraint on minimal time delay in sampling times.*

Description

Set the constraint on minimal time delay in sampling times.

Usage

```
isLessThanDelay(object, samplingTimes)
```

Arguments

object	A SamplingConstraint object.
samplingTimes	A SamplingTimes object.

Value

A boolean that give TRUE/FALSE if the constraint on minimal delay is satisfied.

isNotFixed *Boolean to set if a model parameters is not fixed or not.*

Description

Boolean to set if a model parameters is not fixed or not.

Usage

```
isNotFixed(object)
```

Arguments

object	ModelParameter object.

Value

A boolean isNotFixed giving TRUE if the model parameters is not fixed, FALSE otherwise.

isNotFixedMu *Boolean to set if* mu *is not fixed or not.*

Description

Boolean to set if mu is not fixed or not.

Usage

```
isNotFixedMu(object)
```

Arguments

object	ModelParameter object.

Value

A boolean isNotFixedMu giving TRUE if mu is not fixed, FALSE otherwise.

isTimeInBetweenBounds *Set the constraint on the sampling times bounds.*

Description

Set the constraint on the sampling times bounds.

Usage

```
isTimeInBetweenBounds(object, time)
```

Arguments

object	A SamplingConstraint object.
time	A SamplingTimes object.

Value

A boolean that give TRUE/FALSE if the constraint on on the sampling times bounds are satisfied.

knitrAdministrationParameters *Set the table knitrAdministrationParameters.*

Description

Set the table knitrAdministrationParameters.

Usage

```
knitrAdministrationParameters(object)
```

Arguments

object	An object knitrAdministrationParameters from the class ReportAndPlots.

Value

The table knitrAdministrationParameters.

knitrFIM *Set the table knitrFIM.*

Description

Set the table knitrFIM.

Usage

```
knitrFIM(object)
```

Arguments

object	An object knitrFIM from the class ReportAndPlots.

Value

The table knitrFIM.

knitrInitialDesigns *Set the table knitrInitialDesigns.*

Description

Set the table knitrInitialDesigns.

Usage

```
knitrInitialDesigns(object)
```

Arguments

object	An object knitrInitialDesigns from the class ReportAndPlots.

Value

The table kknitrInitialDesigns.

knitrModelEquations *Set the table knitrModelEquations.*

Description

Set the table knitrModelEquations.

Usage

```
knitrModelEquations(object)
```

Arguments

object	An object knitrModelEquations from the class ReportAndPlots.

Value

The table knitrModelEquations.

knitrModelError *Set the table knitrModelError.*

Description

Set the table knitrModelError.

Usage

```
knitrModelError(object)
```

Arguments

object	An object knitrModelError from the class ReportAndPlots.

Value

The table knitrModelError.

knitrModelParameters *Set the table knitrModelParameters.*

Description

Set the table knitrModelParameters.

Usage

```
knitrModelParameters(object)
```

Arguments

object	An object knitrModelParameters from the class ReportAndPlots.

Value

The table knitrModelParameters.

knitrOptimalDesign *Set the table knitrOptimalDesign.*

Description

Set the table knitrOptimalDesign.

Usage

```
knitrOptimalDesign(object)
```

Arguments

object	An object knitrFIM from the class ReportAndPlots.

Value

The table knitrOptimalDesign.

LibraryOfModels-class *Class for the library of models.*

Description

`LibraryOfModels` is an S4 class that implements the library of models, consisting of two libraries of PK and PD models respectively.

The PK library includes model with different administration routes (bolus, infusion, first-order absorption), different number of compartments (from 1 to 3), and different types of eliminations (linear or Michaelis-Menten). The PD model library, contains direct immediate models (e.g. Emax and Imax) with various baseline models, and turnover response models. The PK/PD models, on the other hand, are obtained with combination of the models from the PK and PD model libraries. Throught the use of the `LibraryOfModels` PFIM handles both analytical and ODE models and offers the possibility to the user to define his own models.

The library of pharmacokinetic (PK) and pharmacodynamic (PD) models is described in the vignette LibraryOfModels.

Objects from the class

`LibraryOfModels` objects are created by calls to `LibraryOfModels` and contain the following slots:

`nameLibraryOfModels`**:** A character string giving the name of the library of models.

`contentsLibraryOfModels`**:** A list of the PK, PD and PKPD models that are in the library of models.

LogNormalDistribution-class *Class "LogNormalDistribution"*

Description

Class `LogNormalDistribution` represent a Log-Normal distribution with mean `mean_log_Gaussian` and standard deviation `sd_log_Gaussian`.

Objects from the Class

LogNormalDistribution objects are typically created by calls to \ `{pfim}` and contain the following slots:

`mean_log_Gaussian`**:** A numeric givnig the mean of the log Normal Distribution.

`sd_log_Gaussian`**:** A numeric giving the standard deviation of the log Normal Distribution.

Model-class *Class "Model" representing a Model*

Description

A class storing information concerning the models in the `LibraryOfModels`.

Objects from the Class `Model`

Objects form the Class `Model` can be created by calls of the form `Model(...)` where (...) are the parameters for the `Model` object.

Slots for the `Model` objects

`nameModel`**:** A character string giving the name of the model.

`descriptionModel`**:** A list of character string giving the characterisation of the model (name, administration, number of compartment)

`equationsModel`**:** A object `ModelEquations` giving the equations of the model.

ModelEquations-class *Class "ModelEquations" representing a the equations of a model.*

Description

A class storing information concerning the model equations of the models in the LibraryOfModels.

Objects from the class `ModelEquations`

Objects form the Class `ModelEquations` can be created by calls of the form `ModelEquations(...)` where (...) are the parameters for the `ModelEquations` objects.

Slots for `ModelEquations` objects

`equations`**:** A list giving the equations of the model.

`allParameters`**:** A vector giving all the parameters of the model.

`ModelError-class` *Class "ModelError" representing a Model error.*

Description

A class storing information concerning the model errors for the models in the LibraryOfModels.

Objects from the class

Objects form the class `ModelError` can be created by calls of the form `ModelError(...)` where (...) are the parameters for the `ModelError` objects.

Slots for Administration objects

`equation`**:** Expression giving the equations of the model.

`derivates`**:** Expression giving the derivatives of the model.

`sigma_inter, sigma_slope`**:** Numerics giving the parameters for the residual variance error model.

`c_error`**:** A numeric taking the values 0 or 1. The ModelError is Proportional when sigma_inter = 0 and c_error = 1. The ModelError isProportionalC: When sigma_inter = 0 and c_error != 1.

`ModelInfusionEquations-class`
Class "ModelInfusionEquations" representing a model with infusion equations

Description

A class giving information on the infusion equations regarding the equations of the model.

Objects from the class

`{ModelInfusionEquations}` objects are typically created by calls to `{ModelInfusionEquations}` and contain the following slots:

`object`**:** An object from the class `ModelEquations`

ModelInfusionODEquations-class

Class "ModelInfusionODEquations" representing a model with infusion equations in ODE model.

Description

A class giving information on the infusion equations regarding the equations of the model.

Objects from the class

`ModelInfusionODEquations` objects are typically created by calls to `ModelInfusionODEquations` and contain the following slots:

`duringInfusionResponsesEquations`**:** A list containing the equations during the infusion.

`afterInfusionResponsesEquations`**:** A list containing the equations after the infusion.

`duringInfusionDerivatives`**:** A list containing the derivatives during the infusion.

`afterInfusionDerivatives`**:** A list containing the derivatives after the infusion.

`derivatives`**:** A list contaning the derivatives of the model.

ModelODEquations-class

Class "ModelODEquations" representing the equations of an ODE model

Description

A class storing information concerning the equations for the ODE models in the `LibraryOfModels`.

Objects from the class

`ModelODEquations` objects are typically created by calls to `ModelODEquations` and contain the following slots:

`derivatives`**:** A list of expression giving the the derivatives of the model.

ModelParameter-class *Class "ModelParameter"*

Description

Class ModelParameter represents a parameters theta included in f(x,theta) theta = mu, covariance_matrix mu - parameter that acts in the individual model covariance_matrix - additional parameter for the population model theta_distribution - Distribution.

Objects from the class

Objects form the class ModelParameter can be created by calls of the form ModelParameter(...) where (...) are the parameters for the ModelParameter objects.

Slots for ModelParameter objects

name: A character string giving the name of the parameter.

mu: A numeric giving the value of the mean mu.

omega: A numeric giving the value of the variance.

distribution: An object of the class Distribution.

fixed: A boolean giving if the parameter is fixed or remain to be estimated.

fixedMu: A boolean giving if the mean mu is fixed or remain to be estimaed.

ModelVariable-class *Class "ModelVariable"*

Description

Class "ModelVariable" represents an initial variable for ODE model.

Objects from the class

ModelVariable objects are typically created by calls to ModelVariable and contain the following slots:

Slots for ModelVariable objects

name: A character string giving the name of the initial variable of an ODE model.

value: A numeric giving the value of the initial variable of an ODE model.

modifyArm *Modify an arm of a design.*

Description

Modify an arm of a design.

Usage

```
modifyArm(object, name, arm)
```

Arguments

object	A Design object.
name	A character string giving the name of the Arm object to be modified in the Design object.
arm	An Arm object.

Value

The Design object with the modified arm.

modifySamplingTimes *Modify the sampling times of an arm.*

Description

Modify the sampling times of an arm.

Usage

```
modifySamplingTimes(object, outcome, samplingTimes)
```

Arguments

object	An object Arm from the class Arm.
outcome	A character string giving the name of the outcome ie the name of the response.
samplingTimes	A vector of numeric giving the new sampling times.

Value

The object Arm object with its new sampling times.

MultiplicativeAlgorithm-class

Class "MultiplicativeAlgorithm"

Description

Class "MultiplicativeAlgorithm" implements the Multiplicative algorithm.

Objects from the class

Objects form the class `MultiplicativeAlgorithm` can be created by calls of the form `MultiplicativeAlgorithm(...)` where (...) are the parameters for the `MultiplicativeAlgorithm` objects.

Slots for `MultiplicativeAlgorithm` objects

`lambda`: A numeric giving the lambda parameter of the multiplicative algorithm.

`delta`: A numeric giving the delta parameter of the multiplicative algorithm.

`iteration_init`: A numeric giving the first iteration of the optimization process.

`iteration_fin`: A numeric giving the last iteration of the optimization process.

`FinalWeights`: A vector givint hte optimal weights.

`showProcess`: A boolean for showing or not the process of optimization.

`OptimalDesign`: A object from the class `Design`

`allArms`: A list of all arms.

MultiplicativeAlgorithm_Rcpp

Function MultiplicativeAlgorithm_Rcpp

Description

Run the MultiplicativeAlgorithm_Rcpp in Rcpp

Usage

```
MultiplicativeAlgorithm_Rcpp(
  fisherMatrices_input,
  numberOfFisherMatrices_input,
  weights_input,
  numberOfParameters_input,
  dim_input,
  lambda_input,
  delta_input,
  iterationInit_input
)
```

Arguments

fisherMatrices_input	fisherMatrices_input
numberOfFisherMatrices_input	numberOfFisherMatrices_input
weights_input	weights_input
numberOfParameters_input	numberOfParameters_input
dim_input	dim_input
lambda_input	lambda_input
delta_input	delta_input
iterationInit_input	iterationInit_input

NormalDistribution-class

Class "NormalDistribution"

Description

Class LogNormalDistribution represent a Normal distribution

Objects from the class

LogNormalDistribution objects are typically created by calls to the class NormalDistribution.

---

numberOfSamplingTimesIsOptimisable

Set the number of sampling times that are optimisable.

Description

Set the number of sampling times that are optimisable.

Usage

```
numberOfSamplingTimesIsOptimisable(object, FixedNumberTimes)
```

Arguments

object	A SamplingConstraint object.
FixedNumberTimes	A numeric giving the number of sampling times to be fixed.

Value

Set the number of sampling times that are optimisable in the constraints.

Optimization-class *Class "Optimization"*

Description

A class storing information concerning Optimization.

Objects from the class `Optimization`

Objects form the class `Optimization` can be created by calls of the form `Optimization(...)` where (...) are the parameters for the `Optimization` objects.

Slots for `Optimization` objects

`showProcess`: A logical if the optimization process is shown or not.

`FisherMatrices`: A list of all the Fisher matrices used in the optimization process.

`combinedTimes`: A list giving all he combinaison of n elements for a vector of times.

`arms`: A list giving all the arms on which the optimization processis done.

Optimize *Set the optimization process.*

Description

Set the optimization process.

Optimization with the Fedorov-Wynn algorithm.

Optimization with the Multiplicative Algorithm.

Optimization with the PGBO Algorithm.

Design optimization withe the Simplex algorithm.

Usage

```
Optimize(
  object,
  pfimProject,
  designs,
  statistical_model,
  cond_init,
  constraint,
  typeFim
)

## S4 method for signature 'FedorovWynnAlgorithm'
```

```
Optimize(object, statistical_model, cond_init, constraint, typeFim)

## S4 method for signature 'MultiplicativeAlgorithm'
Optimize(object, statistical_model, cond_init, constraint, typeFim)

## S4 method for signature 'PGBOAlgorithm'
Optimize(object, pfimProject, designs, statistical_model, constraint, typeFim)

## S4 method for signature 'SimplexAlgorithm'
Optimize(object, designs, statistical_model, constraint, typeFim)
```

Arguments

object	An Optimize object.
pfimProject	A PFIMProject object.
designs	A Design object.
statistical_model	A StatisticalModel object.
cond_init	: A list of numeric giving the values of the initial conditions.
constraint	A Constraint object.
typeFim	A FIM object.

Value

A design object giving the optimal design.

The FedorovWynnAlgorithm object with:

- {OptimalDesign}: The optimal Design.
- {optimalDoses}: A vector giving the optimal doses.
- {FisherMatrix}: A matrix giving The Fisher Information Matrix.
- {optimalFrequencies}: A vector of the optimal frequencies.
- {optimalSamplingTimes}: A list of vectors of the optimal sampling times.

The MultiplicativeAlgorithm object with:

- {OptimalDesign}: A Design object giving the optimal design.
- {FinalWeights}: A list of the optimal weights.
- {iteration_final }: A numeric of the final iteration of the process.
- {allArms}: A list of all the arms in the optimal design.

The PGBOAlgorithm object with:

- {resultsOptimization}: A dataframe giving the results for each iteration.
- {OptimalDesign}: A Design object giving the optimal design.
- {iteration_fin }: A numeric of the final iteration of the process.

A Design object giving the optimal design.

`OptimizeDesign` *Optimize the designs for each arms.*

Description

Optimize the designs for each arms.

Usage

```
OptimizeDesign(object, optimizer, typeOfFim)
```

Arguments

`object`	A `PFIMProject` object.
`optimizer`	A `Optimization` object.
`typeOfFim`	A character string giving the type of Fisher Information Matrix (Population, Indidivual or Bayesian).

Value

The `PFIMProject` object with the optimized designs for each arms.

`parametersForComputingGradient`
Parameters used for computing the model gradient by finite-differences.

Description

Parameters used for computing the model gradient by finite-differences.

Usage

```
parametersForComputingGradient(object)
```

Arguments

`object`	A `StatisticalModel` object.

Value

A list containing the parameters used for computing the gradient of the model.

PDModel-class *Class "PDModel" representing a PD model.*

Description

A class storing information concerning the PD models in the LibraryOfModels.

Objects from the class

PDModel objects are typically created by calls to PDModel.

Slots for the PDModel objects, that are heritated from the class Model

nameModel: A character string giving the name of the model.

descriptionModel: A list of character string giving the characterisation of the model (name, administration, number of compartment)

equationsModel: A object ModelEquations giving the equations of the model.

PFIMProject-class *Class "PFIMProject"*

Description

The class PFIMProject implements the evaluation of the Fisher Information Matrix through the use of a statistical model. This class also plot the graphic for the evolution over time of the concentraton, the sensitivity indices and the standard errors (SE, RSE) of a model.

Objects from the class PFIMProject

Objects form the class PFIMProject can be created by calls of the form PFIMProject(...) where (...) are the parameters for the PFIMProject objects.

The slots for the PFIMProject objects

name: A character strings giving the name of the project.

previous_fim: A matrix of numerical values giving the information matrix obtained from a previous study.

fim: A list of Fims (population or individual or Bayesian information).

statistical_model: A list of StatisticalModels

designs: A list of all designs.

constraints: design constraint.

graph_options: List of graphical options.

PFIMProjectReportEvaluation *Generate the html report for the evaluation.*

Description

Generate the html report for the evaluation.

Usage

```
PFIMProjectReportEvaluation(object, inputPath, outputPath, plotOptions)
```

Arguments

object	PFIMProject object.
inputPath	A string giving the input path.
outputPath	A string giving the output path.
plotOptions	A list giving the options.

Value

The html report for the evaluation.

PFIMProjectReportOptimization *Generate the html report for the optimization.*

Description

Generate the html report for the optimization.

Usage

```
PFIMProjectReportOptimization(object, inputPath, outputPath, plotOptions)
```

Arguments

object	PFIMProject object.
inputPath	A string giving the input path.
outputPath	A string giving the output path.
plotOptions	A list giving the options.

Value

The html report for the optimization.

PGBOAlgorithm-class *Class "PGBOAlgorithm"*

Description

The Class "PGBOAlgorithm" implements the PGBO algortihm : Population Genetics Based Optimizer, developped by Hervé Le Nagard [1].

Objects from the Class PGBOAlgorithm

Objects form the Class PGBOAlgorithm can be created by calls of the form PGBOAlgorithm(...) where (...) are the parameters for the PGBOAlgorithm objects.

Slots for PGBOAlgorithm objects

N: A numeric giving the population size.

muteEffect: A numeric giving the mutation effect.

max_iteration: A numeric giving the maximum of iterations.

iteration_fin: A numeric giving the last iteration.

showProcess: A boolean to show or not the process.

OptimalDesign: A Design object giving the optimal design.

resultsPGBO: A list giving the optimal D-criterion computed during the process.

References

[1] Rebecca Bauer, France Mentré, Halima Kaddouri, Jacques Le Bras, Hervé Le Nagard, Benefits of a new Metropolis-Hasting based algorithm, in non-linear regression for estimation of ex vivo antimalarial sensitivity in patients infected with two strains, Computers in Biology and Medicine, Volume 55, 2014, Pages 16-25, ISSN 0010-4825

PKModel-class *Class "PKModel" representing a PK model.*

Description

A class storing information concerning the PK models in the LibraryOfModels.

Objects from the class PKModel

objects are typically created by calls to PKModel.

Slots for the PKModel objects, that are heritated from the class Model

nameModel: A character string giving the name of the model.

descriptionModel: A list of character string giving the characterisation of the model (name, administration, number of compartment)

equationsModel: A object ModelEquations giving the equations of the model.

PKPDModel-class *Class "PKPDModel" representing a PKPDModel model.*

Description

A class storing information concerning the PKPDModel models in the LibraryOfModels.

Objects from the class PKPDModel

objects are typically created by calls to PKPDModel.

Slots for the PKPDModel objects, that are heritated from the class Model

nameModel: A character string giving the name of the model.

descriptionModel: A list of character string giving the characterisation of the model (name, administration, number of compartment)

equationsModel: A object ModelEquations giving the equations of the model.

---

plotCriteria *Plot the D criteria over time.*

Description

Plot the D criteria over time.

Usage

```
plotCriteria(object, ...)
```

Arguments

object	PFIMProject object.
...	A list giving the plot options.

Value

A plot of the D criteria over iterations for a Design optimization.

plotFrequenciesOptimisation *Plot the frequencies for the FedorovWynn algorithm.*

Description

Plot the frequencies for the FedorovWynn algorithm.

Usage

```
plotFrequenciesOptimisation(object)
```

Arguments

object	A PFIMProject object.

Value

A barplot of the the frequencies for the FedorovWynn algorithm..

plotResponse *Plot the concentration over time of a model.*

Description

Plot the concentration over time of a model.

Usage

```
plotResponse(object, plotOptions)
```

Arguments

object	PFIMProject object.
plotOptions	A list giving the plot options.

Value

A list containing the plots of the concentration over time of a model.

plotRSE — *Plot the relative standard errors RSE of the model parameters.*

Description

Plot the relative standard errors RSE of the model parameters.

Usage

```
plotRSE(object)
```

Arguments

object	PFIMProject object.

Value

A list containing the plots of the RSE of the model parameters.

plotSE — *Plot the standard errors SE of the model parameters.*

Description

Plot the standard errors SE of the model parameters.

Usage

```
plotSE(object)
```

Arguments

object	PFIMProject object.

Value

A list containing the plots of the standard errors SE of the model parameters.

plotSensitivity *Plot the sensitivity indices of a model over time.*

Description

Plot the sensitivity indices of a model over time.

Usage

```
plotSensitivity(object, plotOptions)
```

Arguments

object	PFIMProject object.
plotOptions	A list giving the plot options.

Value

A list containing the plots of the sensitivity indices of a model over time.

plotShrinkage *Plot the shrinkage data.*

Description

Plot the shrinkage data.

Usage

```
plotShrinkage(object)
```

Arguments

object	PFIMProject object.

Value

A list containing the plots of the shrinkage of the model parameters.

plotWeightOptimisation *Plot the optimal weights for the Multiplicative algorithm.*

Description

Plot the optimal weights for the Multiplicative algorithm.

Usage

```
plotWeightOptimisation(object, threshold)
```

Arguments

object	A PFIMProject object.
threshold	A numeric giving the threshold for the weights.

Value

A barplot of the optimal weights above the threshold.

PopulationFim-class *Class "PopulationFim"*

Description

Class "PopulationFim" representing the population Fisher information matrix

A class storing information regarding the population Fisher computation matrix (computation method: first order linearisation (FO))

Objects from the class PopulationFim

Objects form the class PopulationFim can be created by calls of the form PopulationFim(...) where (...) are the parameters for the PopulationFim objects.

Slots for PopulationFim objects

mfisher: A matrix giving the Fisher Information matrix.

PopulationFIMEvaluateVariance *Evaluate the Variance of a Population FIM*

Description

Evaluate the Variance of a Population FIM

Usage

```
PopulationFIMEvaluateVariance(
  object,
  equations,
  model_parameters,
  administrations,
  sampling_times,
  df_total,
  errorVariances,
  sigmaDerivatives
)
```

Arguments

object	A Response object.
equations	An object of class Response containing the name of the reponse and the equation of the model error.
model_parameters	An object of class ModelParameters containing the values and the distributions of the model parameters.
administrations	An object of class Administration containing the parametrization for the administration of the model.
sampling_times	An object of class SamplingTimes containing the parametrization for the sampling times of the model.
df_total	parameter df_total
errorVariances	parameter errorVariances
sigmaDerivatives	parameter sigmaDerivatives

Value

A list giving VDist and MF_var.

PrepareFIMs *Prepare the FIMs for the optimization.*

Description

Prepare the FIMs for the optimization.

Prepare the Fisher Informations matrices.

Prepare the Fisher Informations Matrices.

Usage

```
PrepareFIMs(object, statistical_model, cond_init, constraint, typeFim)

## S4 method for signature 'FedorovWynnAlgorithm'
PrepareFIMs(object, statistical_model, cond_init, constraint, typeFim)

## S4 method for signature 'MultiplicativeAlgorithm'
PrepareFIMs(object, statistical_model, cond_init, constraint, typeFim)
```

Arguments

object	A MultiplicativeAlgorithm object.
statistical_model	A StatisticalModel object.
cond_init	: cond_init
constraint	: A Constraint object.
typeFim	: A character string giving the r=type of FIM : Population, Individual or Bayesian.

Value

A list result of all the FIMs.

A list FIMs of the Fisher Informations matrices.

A list FIMs of the Fisher Informations Matrices.

Proportional-class *Class "Proportional"*

Description

The Class "Proportional" defines the the residual error variance according to the formula g(sigma_inter, sigma_slope, c_error, f(x, theta)) = sigma_slope*f(x,theta).

Objects from the Class Proportional

Objects are typically created by calls to `Proportional` and contain the following slots that are heritated from the class Combined1:

Slots for the `Proportional` objects

`.Object:` An object of the Class `Proportional`

`sigma_inter:` A numeric value giving the sigma inter of the error model

`sigma_slope:` A numeric value giving the sigma slope of the error model

---

`ProportionalC-class` *Class "ProportionalC"*

Description

The Class "ProportionalC" defines the the residual error variance according to the formula g(sigma_inter, sigma_slope, c_error, f(x, theta)) = sigma_slope*f(x,theta)^c_error.

Objects from the Class ProportionalC

objects are typically created by calls to `ProportionalC` and contain the following slots that are heritated from the class Combined1c:

Slots for the `ProportionalC` objects

`.Object:` An object of the Class `ProportionalC`

`sigma_inter:` A numeric value giving the sigma inter of the error model

`sigma_slope:` A numeric value giving the sigma slope of the error model

`c_error:` A numeric value giving the exponant c of the error model

`PSOAlgorithm-class` *Class "PSOAlgorithm"*

Description

The Class "PSOAlgorithm" implements the PSO algortihm : Particle Swarm Optimization

Objects from the class `PSOAlgorithm`

Objects form the xlass `PSOAlgorithm` can be created by calls of the form `PSOAlgorithm(...)` where (...) are the parameters for the `PSOAlgorithm` objects.

Slots for `PSOAlgorithm` **objects**

`maxIteration`**:** A numeric giving the maximum of iterations.

`populationSize`**:** A numeric giving the mpopulation size.

`intertiaWeight`**:** A numeric giving the inertial weight.

`personalLearningCoefficient`**:** A numeric giving the personal learning coefficient.

`globalLearningCoefficient`**:** A numeric giving the global learning coefficient.

`resultsPSO`**:** A list giving the iteration and the results when a new best criteria is found.

`remplaceDose` *Function to remplace a dose.*

Description

Function to remplace a dose.

Usage

```
remplaceDose(ex, progression, all)
```

Arguments

`ex`	parameter ex
`progression`	parameter progression
`all`	parameter all

Value

expression of the equation with dose expression.

`ReportAndPlots-class` *Class "ReportAndPlots"*

Description

The class `ReportAndPlots` defines the htmal reports for the evaluation and the optimization.

Objects from the class

`ReportAndPlots` objects are typically created by calls to `{ReportAndPlots}` and contain the following slots:

`object`**:** An object from the class `ReportAndPlots`

reportPFIMProject *Generate a htmal report html.*

Description

Generate a htmal report html.

Usage

```
reportPFIMProject(object, ...)
```

Arguments

object	PFIMProject object.
...	A list giving options for the report.

Value

an html givin a report of the project for evaluation or optimization

resizeFisherMatrix *Resize the fisher Matrix from a vector to a matrix.*

Description

Resize the fisher Matrix from a vector to a matrix.

Usage

```
resizeFisherMatrix(nb_dimensions, fisherMatrix)
```

Arguments

nb_dimensions	: a numeric for the dimensions of the fisher matrix.
fisherMatrix	: a vector that contain the low triangular Fisher matrix + its main diagonal.

Value

The Fisher matrix of size nb_dimensions*nb_dimensions.

`Response-class` *Class "Response"*

Description

Class `Response` represents a structural model.

Objects from the class

`Response` objects are typically created by calls to `Response` and contain the following slots model_error = g(sigma_inter, sigma_slope , f(x, theta)), this part is considered in class ModelError. There are different possibilities to calculate g.

Slots for `Response` objects

`name`: A character string giving the name for model error.

`model_error`: An object `model_error` from the Class ModelError.

`SamplingConstraint-class`
Class "SamplingConstraint"

Description

Class "SamplingConstraint" storing information concerning sampling constraint.

Objects from the class

`SamplingConstraint` objects are typically created by calls to `SamplingConstraint` and contain the following slots:

Slots for `SamplingConstraint` objects

`response`: A character string for the name of the response of the model.

`numberOptimisability`: A boolean that gives TRUE for optimizing the number of times and FALSE for fixing the number of times.

`numberOfSamplingTimes`: A vector of the number of sampling times.

`fixedTimes`: A vector of the number of fixed times.

`continuousSamplingTimes`: A list of the continuous sampling times.

`discretSamplingTimes`: A list of the discrete sampling times.

`min_delay`: A numeric giving the minimal interval in the sampling times.

SamplingTimes-class *Class "SamplingTimes"*

Description

Class "SamplingTimes" stores information concerning sampling times.

Objects from the Class

Objects form the Class `SamplingTimes` can be created by calls of the form `SamplingTimes(...)` where (...) are the parameters for the `SamplingTimes` objects.

Slots for the `SamplingTimes` objects

`outcome`**:** A character string giving either a compartment name or number (character or integer, TBD with model) (nombre de reponses "1", "2").

`sample_time`**:** A list of discrete vectors giving the times when sampling design is performed.

`initialTime`**:** A numeric giving the initial time of the vecotr of sampling times.

scaleResponsesEvaluationODE *function to adjust Responses with variables values.*

Description

function to adjust Responses with variables values.

Usage

```
scaleResponsesEvaluationODE(
  out_variable,
  modelParameters,
  variablesNames,
  responseNames,
  inputsModel
)
```

Arguments

`out_variable`	parameter out_variable.
`modelParameters`	parameter modelParameters.
`variablesNames`	parameter variablesNames.
`responseNames`	parameter responseNames.
`inputsModel`	parameter inputsModel.

Value

A dataframe giving the evaluated responses adjusted with variables values.

`scaleResponsesEvaluationODEInfusion` *function to adjust Responses with variables values.*

Description

function to adjust Responses with variables values.

Usage

```
scaleResponsesEvaluationODEInfusion(
  out_variable,
  modelParameters,
  variablesNames,
  responseNames,
  inputsModel
)
```

Arguments

`out_variable`	parameter out_variable.
`modelParameters`	parameter modelParameters.
`variablesNames`	parameter variablesNames.
`responseNames`	parameter responseNames.
`inputsModel`	parameter inputsModel.

Value

A dataframe giving the evaluated responses adjusted with variables values.

setAllowedDose<- *Set the constraints on allowed dose.*

Description

Set the constraints on allowed dose.

Usage

```
setAllowedDose(object) <- value
```

Arguments

object	An object Administration from the class Administration.
value	A numeric value for the new dose value.

Value

The Administration object with the new constraints on the allowed dose.

setAllowedTime<- *Set the constraints on allowed times.*

Description

Set the constraints on allowed times.

Usage

```
setAllowedTime(object) <- value
```

Arguments

object	An object Administration from the class Administration.
value	A vector of the numeric values for the new constraints on allowed times.

Value

The Administration object with the new constraints on allowed times.

setAllowedTinf<- *Set the constraints on Tinf.*

Description

Set the constraints on Tinf.

Usage

```
setAllowedTinf(object) <- value
```

Arguments

object	An object Administration from the class Administration.
value	A numeric value for the new constraints on Tinf.

Value

The Administration object with the new constraints on Tinf.

setAmountDose *Set the amount of dose*

Description

Set the amount of dose

Usage

```
setAmountDose(object, value)
```

Arguments

object	An object Administration from the class Administration.
value	A numeric value of the amount of dose.

Value

The numeric amount_dose giving the new value of the amount of dose.

setAmountOfArms *Set the amount of arms in a Design.*

Description

Set the amount of arms in a Design.

Usage

```
setAmountOfArms(object, value)
```

Arguments

object	A Design object.
value	A numeric giving the new value of the amount of arms in the design.

Value

The Design object with the new vamue of amount of arms.

setAmountOfArmsAim *Set amount of arms in a* DesignConstraint *object for the case we aim to obtain a fixed amount of arms as result.*

Description

Set amount of arms in a DesignConstraint object for the case we aim to obtain a fixed amount of arms as result.

Usage

```
setAmountOfArmsAim(object, value)
```

Arguments

object	A DesignConstraint object.
value	A numeric.

Value

A numeric amountOfArm giving the amount of arms for the case we aim to obtain a fixed amount of arms as result.

setArms *Set the arms of a design.*

Description

Set the arms of a design.

Usage

```
setArms(object, value)
```

Arguments

object	A Design object.
value	A Arm object.

Value

The design Design with the new arm.

setArmSize *Set the size of an arm.*

Description

Set the size of an arm.

Usage

```
setArmSize(object, value)
```

Arguments

object	An object Arm from the class Arm.
value	A numeric giving the new size of the object Arm.

Value

The object Arm object with its new size.

setCError<- *Set the CError of a* ModelError *object.*

Description

Set the CError of a ModelError object.

Usage

```
setCError(object) <- value
```

Arguments

object	An ModelError object.
value	The value for CError.

Value

The ModelError object with the new value of the CError.

setConstraint *Set the constraint to the PFIMProject projet.*

Description

Set the constraint to the PFIMProject projet.

Usage

```
setConstraint(object, constraint)
```

Arguments

object	A PFIMProject object.
constraint	The constraint to set

Value

The PFIMProject object with the constraint.

setDelta *Set the* delta *parameters for the Multiplicative algorithm.*

Description

Set the delta parameters for the Multiplicative algorithm.

Usage

```
setDelta(object, values)
```

Arguments

object	MultiplicativeAlgorithm object.
values	values

Value

The MultiplicativeAlgorithm object with the new value of delta.

setDesign *Set the design of* PFIMProject *object.*

Description

Set the design of PFIMProject object.

Usage

```
setDesign(object, value)
```

Arguments

object	A PFIMProject object.
value	A Design object.

Value

The PFIMProject object with the new Designs.

setDiscret<- *Set the possible values for a* DiscreteConstraint *object.*

Description

Set the possible values for a DiscreteConstraint object.

Usage

```
setDiscret(object) <- value
```

Arguments

object	A DiscreteConstraint object.
value	Value for the discrete constraint in the DiscreteConstraint object.

Value

The DiscreteConstraint object with the set of new values.

setInitialConditions *Set the initial conditions of an Arm for an ODE model.*

Description

Set the initial conditions of an Arm for an ODE model.

Usage

```
setInitialConditions(object, values)
```

Arguments

object	An object Arm from the class Arm.
values	A list of numeric giving the values of the initial conditions.

Value

The object Arm with the new initial conditions for an ODE model.

setIteration *Set the number of iterations for the multiplicative algorithm.*

Description

Set the number of iterations for the multiplicative algorithm.

Usage

```
setIteration(object, values)
```

Arguments

object	MultiplicativeAlgorithm object.
values	A numeric.

Value

The MultiplicativeAlgorithm object with the new values of the number of iterations.

setMfisher<- *Set a matrix value for the Fisher Information Matrix.*

Description

Set a matrix value for the Fisher Information Matrix.

Usage

```
setMfisher(object) <- value
```

Arguments

object	A Fim object.
value	A matrix of numerical values.

Value

The Fim object with the Fisher Information Matrix with the new values.

setModelError<- *Set the model error.*

Description

Set the model error.

Usage

```
setModelError(object) <- value
```

Arguments

object	A Response object.
value	The new value for the model error.

Value

The Response object with the new value for the model error.

setMu *Set the mu vector.*

Description

Set the mu vector.

Usage

```
setMu(object, mu)
```

Arguments

object	A Fim object.
mu	A vector mu of the new values of mu.

Value

The Fim object with the mu vector with the new values.

setNameDesign *Set the name of the design.*

Description

Set the name of the design.

Usage

```
setNameDesign(object, name)
```

Arguments

object	Design object.
name	A character string name giving the new name of design.

Value

The Design object with its new name.

setNamePFIMProject *Set the name of a PFIMProject projet.*

Description

Set the name of a PFIMProject projet.

Usage

```
setNamePFIMProject(object, value)
```

Arguments

object	A PFIMProject object.
value	A character string giving the new name of the PFIMProject project.

Value

The PFIMProject object with a new name.

setNumberSamples<- *Set the number of Sample in a Design.*

Description

Set the number of Sample in a Design.

Usage

```
setNumberSamples(object) <- value
```

Arguments

object	A Design object.
value	A numeric giving the new value of samples.

Value

The Design object with the new number of samples.

setOmega *Set the Omega matrix.*

Description

Set the Omega matrix.

Usage

```
setOmega(object, omega)
```

Arguments

object	A Fim object.
omega	A matrix omega giving the new values of the variances.

Value

The Fim object with the Omega matrix with the new values.

setOptimalDesign<- *Set an optimal design.*

Description

Set an optimal design.

Usage

```
setOptimalDesign(object) <- value
```

Arguments

object	A PFIM object.
value	A Design object.

Value

The PFIM object with the optimal design.

setParametersForEvaluateModel *Set the parameters for the evaluation of the model.*

Description

Set the parameters for the evaluation of the model.

Usage

```
setParametersForEvaluateModel(
  object,
  administrations,
  sampling_times,
  cond_init
)
```

Arguments

object	A StatisticalModel object.
administrations	An Administration object.
sampling_times	A SamplingTimes object.
cond_init	A list for the initial conditions of the StatisticalModel object.

Value

A list containing the parameters used for the model evaluation.

`setParametersModel` *Set the parameters of the* `Model` *object.*

Description

Set the parameters of the `Model` object.

Usage

```
setParametersModel(object, parameters)
```

Arguments

`object`	A `Model` object.
`parameters`	The vector of character string giving the parameters names.

Value

The `Model` object with the new parameters.

`setParametersOdeSolver`
Set parameters for the ode solver

Description

Set parameters for the ode solver

Usage

```
setParametersOdeSolver(object, value)
```

Arguments

`object`	A `StatisticalModel` object.
`value`	A list giving the values of the parameters.

Value

The `StatisticalModel` object with the new parameters for the ode solver.

setPossibleArms *Set the possible arms in a Design or the case when lots of arms are defined and aim to optimise amoung several of them.*

Description

Set the possible arms in a Design or the case when lots of arms are defined and aim to optimise amoung several of them.

Usage

```
setPossibleArms(object, design, choice)
```

Arguments

object	A DesignConstraint object.
design	A Design object.
choice	A vector of arm's serial number, to form an arm-space

Value

The DesignConstraint object with all the possible arms

setRange<- *Set the range of a* ContinuousConstraint *object.*

Description

Set the range of a ContinuousConstraint object.

Usage

```
setRange(object) <- value
```

Arguments

object	A ContinuousConstraint object.
value	A numeric.

Value

The ContinuousConstraint object with the new range.

setSampleTime *Set the sample time of the response of the* SamplingTimes *object.*

Description

Set the sample time of the response of the SamplingTimes object.

Usage

```
setSampleTime(object, values)
```

Arguments

object	A SamplingTimes object.
values	A vector giving the new values of the sampling times.

Value

The SamplingTimes object with the new sample times.

setSamplings<- *Set the sampling times for an arm.*

Description

Set the sampling times for an arm.

Usage

```
setSamplings(object) <- value
```

Arguments

object	An object Arm from the class Arm.
value	The sampling times given by the objects from the class SamplingTimes.

Value

The object Arm with its new sampling times.

setShowProcess *Show the process for the optimization.*

Description

Show the process for the optimization.

Shows the process for FIM computing.

Show the process of optimization.

Usage

```
setShowProcess(object, ifShow)

## S4 method for signature 'MultiplicativeAlgorithm'
setShowProcess(object, ifShow)

## S4 method for signature 'SimplexAlgorithm'
setShowProcess(object, ifShow)
```

Arguments

object	A SimplexAlgorithm object.
ifShow	A boolean.

Value

Show process for the optimization.

Shows the process for FIM computing.

Show SimplexAlgorithm object.

setSigmaInter<- *Set the value for* sigma_inter *of a* ModelError *object.*

Description

Set the value for sigma_inter of a ModelError object.

Usage

```
setSigmaInter(object) <- value
```

Arguments

object	An ModelError object.
value	The value for sigma_inter

Value

The ModelError object with the new value for the sigma_inter.

---

setSigmaSlope<- *Set the value for* sigma_slope *of a* ModelError *object.*

Description

Set the value for sigma_slope of a ModelError object.

Usage

```
setSigmaSlope(object) <- value
```

Arguments

object	An ModelError object.
value	The value for sigma_slope.

Value

The ModelError object with the new value for the sigma_slope.

---

setTau *Set the infusion lag tau.*

Description

Set the infusion lag tau.

Usage

```
setTau(object, value)
```

Arguments

object	An object Administration from the class Administration.
value	A numeric value for the infusion lag tau.

Value

The object Administration object with its new value of the infusion lag tau.

setTimeDose<- *Set the times vector when doses are given.*

Description

Set the times vector when doses are given.

Usage

```
setTimeDose(object) <- value
```

Arguments

object	An object Administration from the class Administration.
value	A numeric value of the time dose.

Value

The object Administration with its new times vector for doses.

setTinf *Set the infusion duration.*

Description

Set the infusion duration.

Usage

```
setTinf(object, value)
```

Arguments

object	An object Administration from the class Administration.
value	A numeric value for the infusion duration Tinf.

Value

The object Administration with its new value of the infusion duration Tinf.

setTotalNumberOfIndividuals

Set the total number of individuals in a `DesignConstraint` *object.*

Description

Set the total number of individuals in a `DesignConstraint` object.

Usage

```
setTotalNumberOfIndividuals(object, totalNumberOfIndividual)
```

Arguments

`object`	`DesignConstraint` object.
`totalNumberOfIndividual`	Total number of individual to be set.

Value

The `DesignConstraint` object with the total number of individual.

setTotalSize<-

Set the total size of a Design.

Description

Set the total size of a Design.

Usage

```
setTotalSize(object) <- value
```

Arguments

`object`	A `Design` object.
`value`	A numeric giving the new value of the size of the design.

Value

The `Design` object with the new size.

show,Fim-method *Show the Fisher Information Matrix for a* `Fim` *object and its information: Determinant, D-criterion, SE, Eigenvalues, Correlation.*

Description

Show the Fisher Information Matrix for a `Fim` object and its information: Determinant, D-criterion, SE, Eigenvalues, Correlation.

Show the Individual Fim.

Show the values of `sigma_inter`, `sigma_slope`, and `c_error`.

Show the model errors

Show the model errors.

Show the model errors.

Show the model errors.

Show the model errors.

Show for an `Optimization` object.

Show a design.

Show the content of a design.

Show the end of the process for the multiplicative algorithm.

Show the content of a `StatisticalModel` object.

show the content of the `PFIMProject` object.

Usage

```
## S4 method for signature 'Fim'
show(object)

## S4 method for signature 'IndividualFim'
show(object)

## S4 method for signature 'ModelError'
show(object)

## S4 method for signature 'Combined1'
show(object)

## S4 method for signature 'Combined1c'
show(object)

## S4 method for signature 'Combined2c'
show(object)
```

```
## S4 method for signature 'Combined2'
show(object)

## S4 method for signature 'Constant'
show(object)

## S4 method for signature 'Optimization'
show(object)

## S4 method for signature 'Design'
show(object)

## S4 method for signature 'DesignConstraint'
show(object)

## S4 method for signature 'MultiplicativeAlgorithm'
show(object)

## S4 method for signature 'StatisticalModel'
show(object)

## S4 method for signature 'PFIMProject'
show(object)
```

Arguments

object	PFIMProject object.

Value

Print the Fisher Information Matrix and its informations: Determinant, D-criterion, SE, Eigenvalues, Correlation.

Show the Individual Fim.

Show the values of sigma_inter, sigma_slope, and c_error.

Display the model errors

Display the model errors.

Display the model errors.

Display the model errors.

The model errors.

The content of an Optimization object.

Return the FIM of the design and the data summary of the arm in the design.

Show the content of a design.

Print the end of the process for the multiplicative algorithm.

Display the responses name of the model equations, the ordinary derivatives of the model equations (for an ODE model), the parameters of the model.

Summary of the PFIMProject PFIMProject object given by the: PopulationFim, IndividualFim, Fim, Determinant, Dcriterion, Eigenvalues, SEfixedEffects, SEstandardDeviationOfRandomEffect, SEerrorModel, Designs.

---

showArmData *Show the data of an arm for a design.*

Description

Show the data of an arm for a design.

Usage

```
showArmData(object)
```

Arguments

object	A Design object.

Value

Return a character string giving the the data summary of an arm for a design.

---

showConstraints *Show all the constraints of the* PFIMProject *object.*

Description

Show all the constraints of the PFIMProject object.

Usage

```
showConstraints(object)
```

Arguments

object	A PFIMProject object.

Value

Show the all the objects Constraints in the PFIMProject object.

showDesigns *Show all the Designs.*

Description

Show all the Designs.

Usage

```
showDesigns(object)
```

Arguments

object	A PFIMProject object.

Value

Show all the design designs in the PFIMProject object.

showFims *Show the Fisher Information Matrix for all the designs.*

Description

Show the Fisher Information Matrix for all the designs.

Usage

```
showFims(object)
```

Arguments

object	A PFIMProject object.

Value

Show the Fisher Information Matrix fimOfDesign for all the designs.

showStatisticalModelStandardErrors

Show expected standard error data frame.

Description

Show expected standard error data frame.

Show the statistical model standard errors

Show the statistical model standards errors.

Usage

```
showStatisticalModelStandardErrors(object, modelParameters)

## S4 method for signature 'IndividualFim'
showStatisticalModelStandardErrors(object, modelParameters)

## S4 method for signature 'PopulationFim'
showStatisticalModelStandardErrors(object, modelParameters)
```

Arguments

object	An IndividualFim object.
modelParameters	A modelParameters. object.

Value

A data frame giving the standard error.

A dataframe giving the standard errors.

A dataframe giving the model standards errors.

SimplexAlgorithm-class

Class "SimplexAlgorithm"

Description

The Class "SimplexAlgorithm" implements the Nelder-Mead method (also downhill simplex method, amoeba method) [1].

Objects from the class

Objects form the class SimplexAlgorithm can be created by calls of the form SimplexAlgorithm(...) where (...) are the parameters for the SimplexAlgorithm objects.

Slots for `SimplexAlgorithm` objects

`pct_initial_simplex_building`: A numeric giving the percentage of initial vertices for the simplex algorithm.

`max_iteration`: A numeric giving the maximum of iterations.

`tolerance`: A numeric giving the tolerance criteria for stopping the algorithm.

`showProcess`: A boolean to show or not the process.

`OptimalDesign`: A `Design` object giving the optimal design.

References

[1] Nelder JA, Mead R (1965) A Simplex Method for Function Minimization. Comput J 7: 308-313.

`StandardDistribution-class` *Class "StandardDistribution"*

Description

Class `StandardDistribution` represents class for standard distributions

`StatisticalModel-class` *Class "StatisticalModel"*

Description

Class `StatisticalModel` represents a statistical model.

Mathematical description of the statistical model

-

$$y = f(x, \theta) + g * \epsilon$$

, this part is considered in class Response

-

$$f(x, \theta)$$

, this part is considered in class Response

-

$$\theta = (\mu, \omega)$$

, this part is considered in class ModelParameter

-

$$\epsilon$$

, this is a slot of the object `NormalDistribution` with mean = 0 and covariate_matrix = I

Objects from the class

`StatisticalModel` objects are typically created by calls to `StatisticalModel` and contain the following slots:

Slots for `StatisticalModel` objects

`modelEquations`**:** An object from the class `ModelEquations`

`responses`**:** A list of objects of type Responses -> f(x, theta)

`correlations`**:** A list giving all the covariables.

`model_parameters`**:** A list giving all the parameters of the models.

`summary,Design-method` *summary*

Description

summary

Usage

```
## S4 method for signature 'Design'
summary(object)
```

Arguments

`object`	A `Design` object.

Value

Return a list giving the name, the number of individuals, the total size of the design, the the summary of all the parameters of the arms for a design and the amount of arm in the design.

`summaryArmData` *Gives a summary of all the parameters of an arm for a design.*

Description

Gives a summary of all the parameters of an arm for a design.

Usage

```
summaryArmData(object)
```

Arguments

object	A Design object.

Value

Display a summary of all the parameters of the arms for a design.

Index