Package ‘PFIM’

March 23, 2024

Type Package

Title Population Fisher Information Matrix

Version 6.0.3

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>,
Le Nagard H, Chao L, Tenaillon O (2011) <doi:10.1186/1471-2148-11-326>,

URL http://www.pfim.biostat.fr/

Depends R (>= 4.0.0)

License GPL (>= 2)

Encoding UTF-8

Imports inline, utils, methods, deSolve, Deriv, scales, devtools,
ggplot2, Matrix, pracma, stringr, Rcpp, knitr, rmarkdown,
kableExtra

RoxygenNote 7.2.3

VignetteBuilder knitr

Collate 'GenericMethods.R' 'Administration.R'
'AdministrationConstraints.R' 'Arm.R' 'Fim.R' 'BayesianFim.R'
'ModelError.R' 'Combined1.R' 'Constant.R' 'Design.R'
'Distribution.R' 'ModelParameter.R' 'LibraryOfPDModels.R'
'LibraryOfPKModels.R' 'LibraryOfModels.R'
'LibraryOfPKPDModels.R' 'Model.R' 'PFIMProject.R'
'Evaluation.R' 'OptimizationAlgorithm.R'
'FedorovWynnAlgorithm.R' 'IndividualFim.R' 'LogNormal.R'
'ModelODE.R' 'ModelAnalytic.R' 'ModelAnalyticBolus.R'
'ModelAnalyticSteadyState.R' 'ModelAnalyticBolusSteadyState.R'
'ModelInfusion.R' 'ModelAnalyticInfusion.R'
R topics documented:

- ModelAnalyticInfusionSteadyState.R
- ModelBolus.R
- ModelODEBolus.R
- ModelODEDoseInEquations.R
- ModelODEDoseNotInEquations.R
- ModelODEInfusion.DoseInEquations.R
- ModelODEInfusion.R
- MultiplicativeAlgorithm.R
- Normal.R
- Optimization.R
- PFIM-package.R
- PGBOAlgorithm.R
- PSOAlgorithm.R
- PlotEvaluation.R
- PopulationFim.R
- Proportional.R
- SamplingTimeConstraints.R
- SamplingTimes.R
- SimplexAlgorithm.R

Author  France Mentré [aut],
Romain Leroux [aut, cre],
Jérémy Seurat [aut],
Lucie Fayette [aut],
Emmanuelle Comets [ctb],
Sylvie Retout [ctb]

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

Repository  CRAN

Date/Publication  2024-03-23 14:30:02 UTC

R topics documented:

- PFIM-package ........................................ 7
- addModel ........................................... 20
- addModels ......................................... 20
- Administration-class .............................. 21
- AdministrationConstraints-class ............. 21
- Arm-class .......................................... 22
- BayesianFim-class ................................ 22
- checkSamplingTimeConstraintsForContinuousOptimization .......................... 22
- checkValiditySamplingConstraint ............ 23
- Combined1-class .................................. 24
- Constant-class ................................... 24
- convertPKModelAnalyticToPKModelODE ........ 25
- defineModel ....................................... 25
- defineModelFromLibraryOfModels ............. 26
- defineModelType .................................. 26
- defineModelUserDefined ......................... 27
- definePKModel ..................................... 27
- definePKPDMModel ................................ 28
- Design-class ...................................... 29
- Distribution-class ............................... 30
- EvaluateArm ....................................... 30
- EvaluateDesign ................................... 31
- EvaluateErrorModelDerivatives ............... 31
- EvaluateFisherMatrix ............................ 32
- EvaluateModel .................................... 33
- EvaluateVarianceFIM .............................. 34
R topics documented:

EvaluateVarianceModel .................................................. 34
Evaluation-class ............................................................ 35
FedorovWynnAlgorithm-class ............................................ 36
FedorovWynnAlgorithm_Rcpp ............................................. 36
Fim-class ................................................................. 37
fisher.simplex ............................................................. 38
fun.amoeba ................................................................. 38
generateFimsFromConstraints ........................................... 39
generateReportEvaluation .............................................. 39
generateReportOptimization ........................................... 41
generateSamplingsFromSamplingConstraints ....................... 42
generateTables ........................................................... 43
getAdjustedGradient ..................................................... 43
getAdministration ....................................................... 44
getAdministrationConstraint .......................................... 44
getAdministrations ...................................................... 45
getAdministrationsConstraints ......................................... 45
getArms ................................................................. 46
cgetError ................................................................. 46
cgetColumnAndParametersNamesFIM .................................. 47
cgetColumnAndParametersNamesFIMInLatex ......................... 48
cgetConditionNumberFixedEffects .................................... 48
cgetConditionNumberVarianceEffects ................................ 49
cgetContent ............................................................ 49
cgetCorrelationMatrix .................................................. 50
cgetDataFrameResults .................................................. 50
cgetDcriterion .......................................................... 51
cgetDelta ............................................................... 52
cgetDerivatives ........................................................ 52
cgetDescription ........................................................ 53
cgetDesigns ............................................................. 53
cgetDeterminant ........................................................ 54
cgetDistribution ......................................................... 54
cgetDose .............................................................. 55
cgetEigenValues ......................................................... 55
cgetElementaryProtocols .............................................. 56
cgetEquation ........................................................... 56
cgetEquations .......................................................... 57
cgetEquationsAfterInfusion .......................................... 57
cgetEquationsDuringInfusion ......................................... 58
cgetEvaluationFIMResults .............................................. 58
cgetEvaluationInitialDesignResults ................................ 59
cgetFim ................................................................. 59
cgetFisherMatrix ........................................................ 60
cgetFixedEffects ........................................................ 60
cgetFixedMu ............................................................ 61
cgetFixedOmega ........................................................ 61
cgetFixedParameters ................................................... 62
R topics documented:

- getFixedTimes
- getInitialConditions
- getIterationAndCriteria
- getLambda
- getLibraryPDModels
- getLibraryPKModels
- getMinSampling
- getModel
- getModelEquations
- getModelError
- getModelErrorParametersValues
- getModelFromLibrary
- getModelParameters
- getModelParametersValues
- getMu
- getName
- getNames
- getNumberOfArms
- getNumberOfIterations
- getNumberOfParameters
- getNumberOfsamplingsOptimisable
- getNumberOfTimesByWindows
- getOdeSolverParameters
- getOmega
- getOptimalDesign
- getOptimalFrequencies
- getOptimalWeights
- getOptimizationResults
- getOptimizer
- getOptimizerParameters
- getOutcome
- getOutcomes
- getOutcomesEvaluation
- getOutcomesForEvaluation
- getOutcomesGradient
- getParameters
- getPDModel
- getPKModel
- getPKPDModel
- getPlotOptions
- getProportionsOfSubjects
- getRSE
- getSamplings
- getSamplingsWindows
- getSamplingTime
- getSamplingTimeConstraint
- getSamplingTimes
- getSamplingTimesConstraints
### R topics documented:

- `getSE` .......................................................... 88
- `getShrinkage` .................................................. 88
- `getSigmaInter` ................................................. 89
- `getSigmaSlope` ................................................ 90
- `getSize` ........................................................ 90
- `getTau` .......................................................... 91
- `getTimeDose` ................................................... 91
- `getTinf` .......................................................... 91
- `getVarianceEffects` ........................................... 92
- `IndividualFim-class` ....................................... 93
- `isDoseInEquations` .......................................... 93
- `isModelAnalytic` .............................................. 93
- `isModelBolus` ................................................ 94
- `isModelInfusion` .............................................. 94
- `isModelODE` .................................................... 95
- `isModelSteadyState` ....................................... 95
- `LibraryOfModels-class` .................................... 96
- `LibraryOfPDModels` ......................................... 96
- `LibraryOfPKModels` ......................................... 96
- `LibraryOfPKPDModels-class` ............................... 97
- `LogNormal-class` ............................................. 97
- `Model-class` .................................................... 97
- `ModelAnalytic-class` ....................................... 98
- `ModelAnalyticBolus-class` ................................. 98
- `ModelAnalyticBolusSteadyState-class` .................. 98
- `ModelAnalyticInfusion-class` ............................. 98
- `ModelAnalyticInfusionSteadyState-class` ............... 99
- `ModelAnalyticSteadyState-class` ......................... 99
- `ModelBolus-class` ........................................... 99
- `ModelError-class` ............................................ 99
- `ModelInfusion-class` ....................................... 99
- `ModelODE-class` .............................................. 100
- `ModelODEBolus-class` ...................................... 100
- `ModelODEDoseInEquations-class` ......................... 100
- `ModelODEDoseNotInEquations-class` ..................... 100
- `ModelODEInfusion-class` .................................. 100
- `ModelODEInfusionDoseInEquations-class` ............... 101
- `ModelParameter-class` ..................................... 101
- `MultiplicativeAlgorithm-class` .......................... 101
- `MultiplicativeAlgorithm_Rcpp` ............................ 102
- `Normal-class` ................................................ 103
- `Optimization-class` ........................................ 103
- `OptimizationAlgorithm-class` ............................. 104
- `optimize` ...................................................... 104
- `parametersForComputingGradient` ....................... 105
- `PFIMProject-class` .......................................... 105
- `PGBOAlgorithm-class` ...................................... 106
- `plotEvaluation` .............................................. 107
R topics documented:

PlotEvaluation-class ................................................. 107
plotFrequencies ...................................................... 107
plotOutcomesEvaluation .............................................. 108
plotOutcomesGradient ................................................ 109
plotRSE ............................................................... 109
plotSE ................................................................. 110
plotSensitivityIndice .................................................. 110
plotShrinkage ........................................................... 111
plotWeights ............................................................. 111
PopulationFim-class ..................................................... 112
Proportional-class ...................................................... 112
PSOAlogrithm-class ..................................................... 113
Report ................................................................. 113
reportTablesAdministration ......................................... 114
reportTablesDesign .................................................... 114
reportTablesFIM ........................................................ 115
reportTablesModelError ............................................... 116
reportTablesModelParameters ....................................... 116
reportTablesPlot ........................................................ 117
reportTablesSamplingConstraints ................................... 117
resizeFisherMatrix ..................................................... 118
run ................................................................. 118
SamplingTimeConstraints-class ...................................... 119
SamplingTimes-class ................................................... 119
setAdministrations .................................................... 120
setArm ................................................................. 120
setArms ................................................................. 121
setcError .............................................................. 121
setContent ............................................................. 122
setDerivatives .......................................................... 122
setDescription ........................................................ 123
setDesigns ............................................................. 123
setDistribution ........................................................ 124
setDose ................................................................. 124
setEquation ............................................................ 125
setEquations ........................................................... 125
setEquationsAfterInfusion .......................................... 126
setEquationsDuringInfusion ......................................... 126
setEvaluationFIMResults ............................................. 127
setEvaluationInitialDesignResults .................................. 127
setFim ................................................................. 128
setFimTypeToString .................................................... 128
setFisherMatrix ........................................................ 129
setFixedEffects ........................................................ 129
setFixedMu ............................................................. 130
setFixedOmega ......................................................... 130
setInitialConditions ................................................... 131
setIterationAndCriteria .............................................. 131
**PFIM-package**

*Fisher Information matrix for design evaluation/optimization for nonlinear mixed effects models.*

**Description**

**PFIM-package**

**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 6.0 was released in 2023. The present version, **PFIM 6.0**, is an R package that uses the S4 object system for evaluating and/or optimizing population designs based on FIM in NLMEMs.

This version of **PFIM** now includes a library of models implemented also using the object oriented system S4 of R. This 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 6.0**, 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 6.0** 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].

**Documentation**

Documentation and user guide are available at [http://www.pfim.biostat.fr/](http://www.pfim.biostat.fr/)

**Validation**

**PFIM 6.0** 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 6.0** 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 files in the /R folder**

**PFIM 6.0** 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.
Content of the source code and files in the /R folder

Class Administration

- `getOutcome`
- `setOutcome`
- `getTimeDose`
- `setTimeDose`
- `getDose`
- `setDose`
- `getTinf`
- `setTinf`
- `getTau`
- `setTau`

Class AdministrationConstraints

- `getOutcome`
- `getDose`

Class Arm

- `getName`
- `setName`
- `getSize`
- `setSize`
- `getAdministrations`
- `setAdministrations`
- `getSamplingTimes`
- `setSamplingTimes`
- `getInitialConditions`
- `setInitialConditions`
- `getAdministrationsConstraints`
- `getSamplingTimesConstraints`
- `getSamplingTime`
- `getSamplingTimeConstraint`
- `setSamplingTimesConstraints`
- `setSamplingTime`
- `getAdministration`
- `getAdministrationConstraint`
- `EvaluateArm`
Class BayesianFim
  • EvaluateFisherMatrix
  • getRSE
  • getConditionNumberVarianceEffects
  • getShrinkage
  • setShrinkage
  • reportTablesFIM
  • generateReportEvaluation

Class Combined1
  • See class ModelError

Class Constant
  • See class ModelError

Class Design
  • getName
  • setName
  • getSize
  • setSize
  • setArms
  • getOutcomesEvaluation
  • setOutcomesEvaluation
  • getOutcomesGradient
  • setOutcomesGradient
  • getFim
  • setFim
  • getNumberOfArms
  • setNumberOfArms
  • setArm
  • EvaluateDesign
  • plotOutcomesEvaluation
  • plotOutcomesGradient
  • reportTablesAdministration
  • reportTablesDesign

Class Distribution
  • getParameters
• setParameters
• getMu
• setMu
• getOmega
• setOmega
• getAdjustedGradient

Class Evaluation
• run
• reportTablesPlot
• generateTables
• Report

Class FedorovWynnAlgorithm
• FedorovWynnAlgorithm_Rcpp
• resizeFisherMatrix
• setParameters
• optimize
• generateReportOptimization

Class FedorovWynnAlgorithm
• FedorovWynnAlgorithm_Rcpp
• resizeFisherMatrix
• setParameters
• optimize
• generateReportOptimization

Class Fim
• EvaluateFisherMatrix
• EvaluateVarianceFIM
• getFisherMatrix
• setFisherMatrix
• getFixedEffects
• setFixedEffects
• getVarianceEffects
• setVarianceEffects
• getDeterminant
• getDcriterion
• getCorrelationMatrix
• getSE
• getRSE
• getShrinkage
• getEigenValues
• getConditionNumberFixedEffects
• getConditionNumberVarianceEffects
• getColumnAndParametersNamesFIM
• getColumnAndParametersNamesFIMInLatex
• reportTablesFIM
• generateReportEvaluation
• setFimTypeToString

Class GenericMethods

• getName
• getNames
• getSize
• setSize
• getOutcome
• setOutcome
• getFim
• getOdeSolverParameters
• getMu
• setMu
• getOmega
• setOmega
• getParameters
• setParameters
• getModelError
• getSamplings
• getFim
• setName
• setArms
• getArms

Class IndividualFim

• EvaluateFisherMatrix
• EvaluateVarianceFIM
PFIM-package

- `getRSE`
- `getShrinkage`
- `setShrinkage`
- `reportTablesFIM`
- `generateReportEvaluation`

**Class LibraryOfModels**

- `getName`
- `getContent`
- `setContent`
- `addModel`
- `addModels`
- `getLibraryPKModels`
- `getLibraryPDModels`

**Class LibraryOfPKPDModels**

- `getPKModel`
- `getPDModel`
- `getPKPDModel`

**Class LogNormal**

- `getAdjustedGradient`

**Class Model**

- `getName`
- `setName`
- `getDescription`
- `setDescription`
- `getEquations`
- `setEquations`
- `setModelFromLibrary`
- `getOutcomes`
- `setOutcomes`
- `getOutcomesForEvaluation`
- `setOutcomesForEvaluation`
- `getParameters`
- `setParameters`
- `getModelError`
• setModelError
• getInitialConditions
• setInitialConditions
• getOdeSolverParameters
• setOdeSolverParameters
• getModelFromLibrary
• convertPKModelAnalyticToPKModelODE
• getNumberOfParameters
• isModelODE
• isModelAnalytic
• isDoseInEquations
• isModelInfusion
• isModelSteadyState
• isModelBolus
• definePKPDMModel
• definePKModel
• defineModel
• defineModelFromLibraryOfModels
• defineModelUserDefined
• defineModelType
• EvaluateModel
• parametersForComputingGradient
• EvaluateVarianceModel
• getFixedParameters
• getModelErrorParametersValues
• reportTablesModelParameters
• reportTablesModelError

Class ModelAnalytic

• EvaluateModel
• definePKModel
• definePKPDMModel
• convertPKModelAnalyticToPKModelODE

Class ModelAnalyticBolus

• See class ModelAnalytic

Class ModelAnalyticBolusSteadyState
• See class `ModelAnalyticBolus`

Class `ModelBolus`

• See class `Model`

Class `ModelError`

• `getOutcome`
• `getEquation`
• `setEquation`
• `getDerivatives`
• `setDerivatives`
• `getSigmaInter`
• `setSigmaInter`
• `getSigmaSlope`
• `setSigmaSlope`
• `getcError`
• `setcError`
• `getParameters`
• `EvaluateErrorModelDerivatives`

Class `ModelInfusion`

• `getEquationsDuringInfusion`
• `getEquationsAfterInfusion`
• `setEquationsAfterInfusion`
• `setEquationsDuringInfusion`

Class `ModelODE`

• See class `Model`

Class `ModelODEBolus`

• `EvaluateModel`
• `definePKPDMModel`

Class `ModelODEDoseInEquations`

• `EvaluateModel`
• `definePKPDMModel`
• `definePKPDMModel`

Class `ModelODEDoseNotInEquations`
- EvaluateModel
- definePKModel
- definePKPDModel

Class ModelODEInfusion
- See class ModelInfusion

Class ModelODEInfusionDoseInEquations
- EvaluateModel
- definePKModel
- definePKPDModel

Class ModelParameter
- getName
- getDistribution
- setDistribution
- getFixedMu
- setFixedMu
- getFixedOmega
- setFixedOmega
- getMu
- setMu
- getOmega
- setOmega

Class MultiplicativeAlgorithm
- MultiplicativeAlgorithm_Rcpp
- getLambda
- getDelta
- getNumberOfIterations
- getOptimalWeights
- setOptimalWeights
- setParameters
- optimize
- getDataFrameResults
- plotWeights
- generateReportOptimization

Class Normal
• getAdjustedGradient

Class Optimization

• getProportionsOfSubjects
• getOptimizationResults
• setOptimizationResults
• getEvaluationFIMResults
• setEvaluationFIMResults
• setEvaluationInitialDesignResults
• getEvaluationInitialDesignResults
• getElementaryProtocols
• generateFimsFromConstraints
• run
• plotWeights
• Report

Class PFIMProject

• getName
• setModel
• getModel
• getModelEquations
• getModelParameters
• getModelError
• getDesigns
• getFim
• getOdeSolverParameters
• getOutcomes
• getOptimizer
• getOptimizerParameters
• run
• generateTables
• Report

Class PGBOAlgorithm

• setParameters
• optimize
• generateReportOptimization

Class PlotEvaluation
• plot
• plotSE
• plotRSE
• plotShrinkage

Class PopulationFim
• EvaluateFisherMatrix
• EvaluateVarianceFIM
• getRSE
• getShrinkage
• setShrinkage
• reportTablesFIM
• generateReportEvaluation

Class Proportional
• See class ModelError

Class PSOAlgorithm
• setParameters
• optimize
• generateReportOptimization

Class SamplingTimeConstraints
• getOutcome
• getSamplings
• getFixedTimes
• getNumberOfTimesByWindows
• getMinSampling
• getSamplingsWindows
• getNumberOfSamplingsOptimisable
• checkSamplingTimeConstraintsForContinuousOptimization
• generateSamplingsFromSamplingConstraints

Class SamplingTimes
• getOutcome
• setOutcome
• getSamplings
• setSamplings
Class **SimplexAlgorithm**

- `setParameters`
- `fun.amoeba`
- `fisher.simplex`
- `optimize`
- `generateReportOptimization`

**Author(s)**

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

Authors:

- France Mentré <france.mentre@inserm.fr>
- Jérémy Seurat <jeremy.seurat@inserm.fr>
- Lucie Fayette <lucie.fayette@inserm.fr>

Other contributors:

- Emmanuelle Comets [contributor]
- Sylvie Retout [contributor]

**References**


addModels

Description
Add a models to a library of models.

Usage
addModels(object, models)

## S4 method for signature 'LibraryOfModels'
addModels(object, models)

Arguments
- object: An object from the class LibraryOfModels.
- models: A list of object from the class Model.

Value
The library of models with the added model.

See Also
Useful links:
- http://www.pfim.biostat.fr/
Value

The library of models with the added models.

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 response of the model.
dtimeDose: A numeric vector giving the times when doses are given.
dose: A numeric vector giving the amount of doses.
Tinf: A numeric vector giving the infusion duration Tinf (Tinf can be null).
tau: A numeric giving the frequency.

AdministrationConstraints-class Class "AdministrationConstraints"

Description

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

Objects from the class

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

Slots for AdministrationConstraints objects

outcome: A character string giving the name for the response of the model.
doses: A numeric vector giving the amount of doses.
Arm-class

Class "Arm"

Description
The class Arm combines the treatment and the sampling schedule.

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
- name: A string giving the name of the arm.
- size: An integer giving the number of subjects in the arm. By default set to 1.
- administrations: A list of the administrations.
- initialConditions: A list of the initial conditions.
- samplingTimes: A list of the sampling times.
- administrationsConstraints: A list of the administrations constraints.
- samplingTimesConstraints: A list of the sampling times constraints.

BayesianFim-class

Class "BayesianFim"

Description
The class BayesianFim represents the population Fisher information matrix. The class BayesianFim inherits from the class Fim.

checkSamplingTimeConstraintsForContinuousOptimization

Check for the samplingTime constraints for continuous optimization

Description
Check for the samplingTime constraints for continuous optimization
**Usage**

checkSamplingTimeConstraintsForContinuousOptimization(
  object,
  arm,
  newSamplings,
  outcome
)

## S4 method for signature 'SamplingTimeConstraints'
checkSamplingTimeConstraintsForContinuousOptimization(
  object,
  arm,
  newSamplings,
  outcome
)

**Arguments**

- **object** An object from the class `SamplingTimeConstraints`.
- **arm** An object from the class `Arm`.
- **newSamplings** A vector giving the new sampling.
- **outcome** The outcomes for the model.

**Value**

A list of Boolean giving true if the minimal sampling times is in the vector of sampling times & the number of sampling for each windows is respected false otherwise.

---

**Description**

Check the validity of he sampling times constraints

**Usage**

checkValiditySamplingConstraint(object)

## S4 method for signature 'Design'
checkValiditySamplingConstraint(object)

**Arguments**

- **object** An object from the class `Design`.
**Value**

An error message if a constraint is not valid.

---

**Combined1-class**  
Class "Combined1"

**Description**

The class Combined1 defines the residual error variance according to the formula $g(\sigma_{inter}, \sigma_{slope}, c_{error}, f(x, \theta)) = \sigma_{inter} + \sigma_{slope}f(x, \theta))$. The class Combined1 inherits from the class ModelError.

**Objects from the class**

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

- **outcome**: A string giving the name of the outcome.
- **equation**: An symbolic expression of the model error.
- **derivatives**: A list containing the derivatives of the model error expression.
- **sigmaInter**: A numeric value giving the sigma inter of the error model.
- **sigmaSlope**: A numeric value giving the sigma slope of the error model.
- **cError**: A numeric value giving the exponent c of the error model.

---

**Constant-class**  
Class "Constant"

**Description**

The class Constant defines the residual error variance according to the formula $g(\sigma_{inter}, \sigma_{slope}, c_{error}, f(x, \theta)) = \sigma_{inter}$. The class Constant inherits from the class ModelError.

**Objects from the class**

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

- **outcome**: A string giving the name of the outcome.
- **equation**: An symbolic expression of the model error.
- **derivatives**: A list containing the derivatives of the model error expression.
- **sigmaInter**: A numeric value giving the sigma inter of the error model.
- **sigmaSlope**: A numeric value giving the sigma slope of the error model.
- **cError**: A numeric value giving the exponent c of the error model.
**convertPKModelAnalyticToPKModelODE**

*Convert an analytic model to a ode model.*

**Description**

Convert an analytic model to a ode model.

**Usage**

```r
convertPKModelAnalyticToPKModelODE(object)
```

## S4 method for signature 'ModelAnalytic'

```r
close(convertPKModelAnalyticToPKModelODE(object))
```

## S4 method for signature 'ModelAnalyticSteadyState'

```r
close(convertPKModelAnalyticToPKModelODE(object))
```

## S4 method for signature 'ModelAnalyticInfusion'

```r
close(convertPKModelAnalyticToPKModelODE(object))
```

**Arguments**

- `object` An object from the class `Model`.

**Value**

A ode model.

---

**defineModel**

*Define a model.*

**Description**

Define a model.

**Usage**

```r
defineModel(object, designs)
```

## S4 method for signature 'Model'

```r
defineModel(object, designs)
```

**Arguments**

- `object` An object from the class `Model`.
- `designs` A list of objects from the class `Design`.  

---
**Value**

A model defined either from the library of models or user defined.

---

**defineModelFromLibraryOfModels**

*Define a model from the library of models.*

---

**Description**

Define a model from the library of models.

**Usage**

```r
defineModelFromLibraryOfModels(object, designs)
```

```r
## S4 method for signature 'Model'
defineModelFromLibraryOfModels(object, designs)
```

**Arguments**

- `object`: An object from the class `Model`.
- `designs`: A list of objects from the class `Design`.

**Value**

A model defined from the library of models.

---

**defineModelType**

*Define the type of a model.*

---

**Description**

Define the type of a model.

**Usage**

```r
defineModelType(object, designs)
```

```r
## S4 method for signature 'Model'
defineModelType(object, designs)
```

**Arguments**

- `object`: An object from the class `Model`.
- `designs`: A list of objects from the class `Design`.
**defineModelUserDefined**

*Define a user defined model.*

**Value**

Return a model defined as analytic, ode, etc.

## Arguments

- **object**: An object from the class `Model`.
- **designs**: A list of objects from the class `Design`.

## Value

A model giving a user defined model.

**definePKModel**

*Define a PK model.*

## Arguments

- **object**: An object from the class `ModelAnalytic`.
- **outcomes**: A list of outcomes.

## Usage

```r
definePKModel(object, outcomes)
```

**Value**

A model giving a PK model.

## Arguments

- **object**: An object from the class `ModelAnalyticSteadyState`.
- **outcomes**: A list of outcomes.

## Usage

```r
definePKModel(object, outcomes)
```

## Value

A model giving a PK model.

## Arguments

- **object**: An object from the class `ModelAnalyticInfusion`.
- **outcomes**: A list of outcomes.

## Usage

```r
definePKModel(object, outcomes)
```

## Value

A model giving a PK model.
definePKModel(object, outcomes)

## S4 method for signature 'ModelODEDoseInEquations'
definePKModel(object, outcomes)

## S4 method for signature 'ModelODE'
definePKModel(object, outcomes)

## S4 method for signature 'ModelODEInfusionDoseInEquations'
definePKModel(object, outcomes)

Arguments

object An object from the class Model.
outcomes A list giving the outcomes of the PK model.

Value

A model giving a PK model.

---

definePKPDModel Define a PKPD model.

Description

Define a PKPD model.

Usage

definePKPDModel(PKModel, PDModel, outcomes)

## S4 method for signature 'ModelAnalytic,ModelAnalytic'
definePKPDModel(PKModel, PDModel, outcomes)

## S4 method for signature 'ModelAnalytic,ModelODE'
definePKPDModel(PKModel, PDModel, outcomes)

## S4 method for signature 'ModelAnalyticSteadyState,ModelAnalyticSteadyState'
definePKPDModel(PKModel, PDModel, outcomes)

## S4 method for signature 'ModelAnalyticSteadyState,ModelODE'
definePKPDModel(PKModel, PDModel, outcomes)

## S4 method for signature 'ModelAnalyticInfusion,ModelAnalytic'
definePKPDModel(PKModel, PDModel, outcomes)

## S4 method for signature 'ModelAnalyticInfusion,ModelODE'
definePKPDMModel(PKModel, PDMModel, outcomes)

## S4 method for signature 'ModelODEBolus, ModelODE'
definePKPDMModel(PKModel, PDMModel, outcomes)

## S4 method for signature 'ModelODEDoseInEquations, ModelODE'
definePKPDMModel(PKModel, PDMModel, outcomes)

## S4 method for signature 'ModelODEDoseNotInEquations, ModelODE'
definePKPDMModel(PKModel, PDMModel, outcomes)

## S4 method for signature 'ModelODEInfusion, ModelODE'
definePKPDMModel(PKModel, PDMModel, outcomes)

## S4 method for signature 'ModelODEInfusionDoseInEquations, ModelODE'
definePKPDMModel(PKModel, PDMModel, outcomes)

### Arguments

PKModel  An object from the class Model.
PDMModel An object from the class Model.
outcomes  A list giving the outcomes of the PKPD model.

### Value

A model giving a PKPD model.

---

**Design-class**

*Class "Design"*

**Description**

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

**Objects from the class**

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**

- name: A string giving the name of the design.
- size: An integer giving the number of subjects in the design.
- arms: A list of the arms.
- outcomesEvaluation: A list of the results of the design evaluation for the outcomes.
- outcomesGradient: A list of the results of the design evaluation for the sensitivity indices.
numberOfArms: A numeric giving the number of arms in the design.
fim: An object of the class Fim containing the Fisher Information Matrix of the design.

Distribution-class  Class "Distribution"

Description
The class defines all the required methods for a distribution object.

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

Slots for Distribution objects
parameters: A list containing the distribution parameters.

EvaluateArm  EvaluateArm

Description
Evaluate an arm.

Usage
EvaluateArm(object, model, fim)

## S4 method for signature 'Arm'
EvaluateArm(object, model, fim)

Arguments
object An object arm from the class Arm.
model An object model from the class Model.
fim An object fim from the class Fim.

Value
The object fim containing the Fisher Information Matrix the two lists evaluationOutcomes, outcomesGradient containing the results of the evaluation of the outcome and the sensitivity indices.
EvaluateDesign

**Description**
Evaluate an design

**Usage**

EvaluateDesign(object, model, fim)

```r
## S4 method for signature 'Design'
EvaluateDesign(object, model, fim)
```

**Arguments**

- object: An object Design from the class Design.
- model: An object model from the class Model.
- fim: An object fim from the class Fim.

**Value**
The object Design with its slot fim, evaluationOutcomes, outcomesGradient updated.

---

EvaluateErrorModelDerivatives

**Evaluate the error model derivatives.**

**Description**
Evaluate the error model derivatives.

**Usage**

EvaluateErrorModelDerivatives(object, evaluationOutcome)

```r
## S4 method for signature 'ModelError'
EvaluateErrorModelDerivatives(object, evaluationOutcome)
```

**Arguments**

- object: An object from the class ModelError.
- evaluationOutcome: A list giving the results of the model evaluation.
EvaluateFisherMatrix

Value

A list giving the error variance and the Sigma derivatives.

Description

Evaluate the Fisher matrix (population, individual and Bayesian)

Usage

EvaluateFisherMatrix(object, model, arm, modelEvaluation, modelVariance)

## S4 method for signature 'BayesianFim'
EvaluateFisherMatrix(object, model, arm, modelEvaluation, modelVariance)

## S4 method for signature 'IndividualFim'
EvaluateFisherMatrix(object, model, arm, modelEvaluation, modelVariance)

## S4 method for signature 'PopulationFim'
EvaluateFisherMatrix(object, model, arm, modelEvaluation, modelVariance)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An object from the class Fim.</td>
</tr>
<tr>
<td>model</td>
<td>An object from the class Model.</td>
</tr>
<tr>
<td>arm</td>
<td>An object from the class Arm.</td>
</tr>
<tr>
<td>modelEvaluation</td>
<td>A list containing the evaluation results.</td>
</tr>
<tr>
<td>modelVariance</td>
<td>A list containing the model variance.</td>
</tr>
</tbody>
</table>

Value

An object from the class Fim containing the Fisher matrix.
EvaluateModel

Evaluate a model.

Description

Evaluate a model.

Usage

EvaluateModel(object, arm)

## S4 method for signature 'ModelAnalytic'
EvaluateModel(object, arm)

## S4 method for signature 'ModelAnalyticSteadyState'
EvaluateModel(object, arm)

## S4 method for signature 'ModelAnalyticInfusion'
EvaluateModel(object, arm)

## S4 method for signature 'ModelAnalyticInfusionSteadyState'
EvaluateModel(object, arm)

## S4 method for signature 'ModelODEBolus'
EvaluateModel(object, arm)

## S4 method for signature 'ModelODEDoseInEquations'
EvaluateModel(object, arm)

## S4 method for signature 'ModelODEDoseNotInEquations'
EvaluateModel(object, arm)

## S4 method for signature 'ModelODEInfusionDoseInEquations'
EvaluateModel(object, arm)

Arguments

object    An object from the class Model.
arm       An object from the class Arm.

Value

A list giving the results of the model evaluation.
EvaluateVarianceFIM

Evaluate the variance of the Fisher information matrix.

**Description**
Evaluate the variance of the Fisher information matrix.

**Usage**
EvaluateVarianceFIM(object, model, arm, modelEvaluation, modelVariance)

## S4 method for signature 'IndividualFim'
EvaluateVarianceFIM(object, model, arm, modelEvaluation, modelVariance)

## S4 method for signature 'PopulationFim'
EvaluateVarianceFIM(object, model, arm, modelEvaluation, modelVariance)

**Arguments**

- **object**
  An object from the class Fim.

- **model**
  An object from the class Model.

- **arm**
  An object from the class Arm.

- **modelEvaluation**
  A list containing the evaluation results.

- **modelVariance**
  A list containing the model variance.

**Value**
A list containing the matrices of the variance of the FIM.

---

EvaluateVarianceModel

Evaluate the variance of a model.

**Description**
Evaluate the variance of a model.

**Usage**
EvaluateVarianceModel(object, arm, evaluationModel)

## S4 method for signature 'Model'
EvaluateVarianceModel(object, arm, evaluationModel)
Evaluation-class

Arguments

  object  An object from the class Model.
  arm    An object from the class Arm.
  evaluationModel
   A list giving the outputs of the model evaluation.

Value

  Return a list giving the results of the evaluation of the model variance.

Description

  A class storing information concerning the evaluation of a design.

Objects from the class

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

Slots for the Evaluation objects

  name: A string giving the name of the project.
  model: A object of class Model giving the model.
  modelEquations: A list giving the model equations.
  modelParameters: A list giving the model parameters.
  modelError: A list giving the model error for each outcome of the model.
  outcomes: A list giving the model outcomes.
  designs: A list giving the designs to be evaluated.
  fim: An object of the class Fim containing the Fisher Information Matrix of the design.
  odeSolverParameters:
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

- elementaryProtocols: A list of vector for the initial 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,
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 giving the results of the outputs of the Fedorov-Wynn algorithm.

Description

A class storing information regarding the Fisher matrix. Type of the Fisher information: population ("PopulationFIM"), individual ("IndividualFIM") or Bayesian ("BayesianFIM").

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

- fisherMatrix: A matrix giving the Fisher matrix.
- fixedEffects: A matrix giving the fixed effects of the Fisher matrix.
- varianceEffects: A matrix giving the variance effects of the Fisher matrix.
- shrinkage: A vector giving the shrinkage value of the parameters.
**fisher.simplex**

*Compute the fisher.simplex*

**Description**

Compute the fisher.simplex

**Usage**

```
fisher.simplex(simplex, optimizationObject, outcomes)
```

**Arguments**

- `simplex` A list giving the parameters of the simplex.
- `optimizationObject` An object from the class `Optimization`.
- `outcomes` A vector giving the outcomes of the arms.

**Value**

A list giving the results of the optimization.

---

**fun.amoeba**

*function fun.amoeba*

**Description**

function fun.amoeba

**Usage**

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

**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.
- `outcomes` A vector giving the outcomes.
- `data` a fixed set of data.
- `showProcess` A boolean for showing the process or not.
**generateFimsFromConstraints**

*Generate the fim from the constraints*

**Value**

A list containing the components of the optimized simplex. `getColumnAndParametersNamesFIMInLatex`.

**Description**

Generate the fim from the constraints

**Usage**

```r
generateFimsFromConstraints(object, fims)
```

```r
## S4 method for signature 'Optimization'
generateFimsFromConstraints(object)
```

**Arguments**

- `object` An object from the class `Optimization`.
- `fims` A list of object from the class `Fim`.

**Value**

A list giving the arms with their fims.

**generateReportEvaluation**

*Generate the report for the evaluation*

**Description**

Generate the report for the evaluation
Usage

generateReportEvaluation(
  object,
  evaluationObject,
  outputPath,
  outputFile,
  plotOptions
)

## S4 method for signature 'BayesianFim'

generateReportEvaluation(
  object,
  evaluationObject,
  outputPath,
  outputFile,
  plotOptions
)

## S4 method for signature 'IndividualFim'

generateReportEvaluation(
  object,
  evaluationObject,
  outputPath,
  outputFile,
  plotOptions
)

## S4 method for signature 'PopulationFim'

generateReportEvaluation(
  object,
  evaluationObject,
  outputPath,
  outputFile,
  plotOptions
)

Arguments

- `object`: An object from the class `Fim`.
- `evaluationObject`: A list giving the results of the evaluation of the model.
- `outputPath`: A string giving the output path.
- `outputFile`: A string giving the name of the output file.
- `plotOptions`: A list giving the plot options.

Value

Return the report for the evaluation in html.
generateReportOptimization

Generate report for the optimization.

Description

Generate report for the optimization.

Usage

generateReportOptimization(
  object,
  optimizationObject,
  outputPath,
  outputFile,
  plotOptions
)

## S4 method for signature 'FedorovWynnAlgorithm'
generateReportOptimization(
  object,
  optimizationObject,
  outputPath,
  outputFile,
  plotOptions
)

## S4 method for signature 'MultiplicativeAlgorithm'
generateReportOptimization(
  object,
  optimizationObject,
  outputPath,
  outputFile,
  plotOptions
)

## S4 method for signature 'PGBOAlgorithm'
generateReportOptimization(
  object,
  optimizationObject,
  outputPath,
  outputFile,
  plotOptions
)

## S4 method for signature 'PSOAlgorithm'
generateReportOptimization(
Arguments

object An object from the class OptimizationAlgorithm.
optimizationObject An object from the class Optimization.
outputPath A string giving the output path.
outputFile A string giving the name of the output file.
plotOptions A list giving the plot options.

Value

The report for the optimization in html.

Description

Generate samplings from sampling constraints

Usage

generateSamplingsFromSamplingConstraints(object)

## S4 method for signature 'SamplingTimeConstraints'
generateSamplingsFromSamplingConstraints(object)

Arguments

object An object from the class SamplingTimeConstraints.
Value

A list of sampling times generated from the sampling constraints.

---

**generateTables**  
*Generate the tables for the report.*

Description

Generate the tables for the report.

Usage

```r
generateTables(object, plotOptions)
```

## S4 method for signature 'Evaluation'

```r
generateTables(object, plotOptions)
```

## S4 method for signature 'Optimization'

```r
generateTables(object, plotOptions)
```

Arguments

- `object`: An object from the class `PFIMProject`.
- `plotOptions`: A list giving the plot options.

Value

A list giving the kable able for the report (evaluation and optimization).

---

**getAdjustedGradient**  
*getAdjustedGradient*

Description

Get the adjusted gradient.

Usage

```r
getAdjustedGradient(object, outcomesGradient)
```

## S4 method for signature 'LogNormal'

```r
getAdjustedGradient(object, outcomesGradient)
```

## S4 method for signature 'Normal'

```r
getAdjustedGradient(object, outcomesGradient)
```
getAdministrationConstraint

Arguments

object An object distribution from the class Distribution.
outcomesGradient A list containing the evaluation of the outcome gradients.

Value

A list giving the adjusted gradient.

getAdministration

getAdministration

Description

Get the administrations by outcome.

Usage

getAdministration(object, outcome)

## S4 method for signature 'Arm'
getAdministration(object, outcome)

Arguments

object An object Arm from the class Arm.
outcome A string giving the name of the outcome.

Value

The element of the list administrations containing the administration of the outcome outcome

getAdministrationConstraint

getAdministrationConstraint

Description

Get the administration constraints by outcome.

Usage

getAdministrationConstraint(object, outcome)

## S4 method for signature 'Arm'
getAdministrationConstraint(object, outcome)
getAdministrations

Arguments

object An object Arm from the class Arm.
outcome A string giving the name of the outcome.

Value

The element of the list getAdministrationConstraint containing the administration constraints of the outcome outcome

generateAdministrations getAdministrations

Description

Get all the administration for an arm.

Usage

generateAdministrations(object)

## S4 method for signature 'Arm'
generateAdministrations(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.

generateAdministrationsConstraints

Description

Get the administrations constraints.

Usage

generateAdministrationsConstraints(object)

## S4 method for signature 'Arm'
genenerateAdministrationsConstraints(object)
Arguments

  object              An object Arm from the class Arm.

Value

  The list administrationsConstraints.

getArms

  Get the arms of an object.

Description

  Get the arms of an object.

Usage

  getArms(object)

## S4 method for signature 'Design'
getArms(object)

## S4 method for signature 'OptimizationAlgorithm'
getArms(object)

Arguments

  object              An object defined form a class of PFIM.

Value

  A list containing the arms of the object.

cgetError

  Get the parameter c.

Description

  Get the parameter c.

Usage

  getCError(object)

## S4 method for signature 'ModelError'
cgetError(object)

## S4 method for signature 'ModelError'
cgetError(object)
**Arguments**

- **object**: An object from the class `ModelError`.

**Value**

A numeric giving the parameter c.

description

Get the names of the names of the parameters associated to each column of the fim.

**Usage**

getColumnAndParametersNamesFIM(object, model)

```r
## S4 method for signature 'BayesianFim'
ge getColumnAndParametersNamesFIM(object, model)
## S4 method for signature 'IndividualFim'
ge getColumnAndParametersNamesFIM(object, model)
## S4 method for signature 'PopulationFim'
ge getColumnAndParametersNamesFIM(object, model)
```

**Arguments**

- **object**: An object from the class `Fim`.
- **model**: An object from the class `Model`.

**Value**

A list giving the names of the parameters associated to each column of the fim.
getColumnAndParametersNamesFIMInLatex

Get the names of the names of the parameters associated to each column of the fim in Latex format.

Description

Get the names of the names of the parameters associated to each column of the fim in Latex format.

Usage

getColumnAndParametersNamesFIMInLatex(object, model)

## S4 method for signature 'BayesianFim'
getColumnAndParametersNamesFIMInLatex(object, model)

## S4 method for signature 'IndividualFim'
getColumnAndParametersNamesFIMInLatex(object, model)

## S4 method for signature 'PopulationFim'
getColumnAndParametersNamesFIMInLatex(object, model)

Arguments

- object: An object from the class Fim.
- model: An object from the class Model.

Value

A list giving the names of the parameters associated to each column of the fim in Latex format.

getConditionNumberFixedEffects

Get the condition number of the matrix of the fixed effects.

Description

Get the condition number of the matrix of the fixed effects.

Usage

getConditionNumberFixedEffects(object)

## S4 method for signature 'Fim'
getConditionNumberFixedEffects(object)
getConditionNumberVarianceEffects

Arguments

object
An object from the class Fim.

Value

A numeric giving the condition number of the matrix of the fixed effects.

description

getConditionNumberVarianceEffects

Get the condition number of the matrix of the variance effects.

Usage

getConditionNumberVarianceEffects(object)

## S4 method for signature 'Fim'
getConditionNumberVarianceEffects(object)

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

Arguments

object
An object from the class Fim.

Value

A numeric giving the condition number of the matrix of the variance effects.

description

getContent

Get content of a library of models.

Usage

getContent(object)

## S4 method for signature 'LibraryOfModels'
getContent(object)
Arguments

object  An object from the class LibraryOfModels.

Value

A list giving the content of the library of models.

dataFrameResults  Get the dataframe of the results.

Description

Get the dataframe of the results.

getCorrelationMatrix  Get the correlation matrix.

Arguments

object  An object from the class Fim.

Value

The correlation matrix of the fim.

dataFrameResults  Get the dataframe of the results.

Description

Get the dataframe of the results.
**getDcriterion**

Get the D criterion of the fim.

### Usage

```r
getDcriterion(object)
```

## S4 method for signature 'Fim'

```r
getcriterion(object)
```

## S4 method for signature 'Evaluation'

```r
getcriterion(object)
```

## S4 method for signature 'Optimization'

```r
getcriterion(object)
```

### Arguments

- **object**: An object from the class **Fim**.

### Description

Get the D criterion of the fim.

---

**getDataFrameResults**

Get the dataframe of the results.

### Usage

```r
getDataFrameResults(object, threshold)
```

## S4 method for signature 'FedorovWynnAlgorithm'

```r
getDataFrameResults(object, threshold)
```

## S4 method for signature 'MultiplicativeAlgorithm'

```r
getDataFrameResults(object, threshold)
```

## S4 method for signature 'Optimization'

```r
getDataFrameResults(object, threshold)
```

### Arguments

- **object**: An object from the class **OptimizationAlgorithm**.
- **threshold**: The threshold for the optimal weights / frequencies for the multiplicative and FW algorithms.

### Value

Return the dataframe of the results.
**Value**

A numeric giving the D criterion of the fim.

---

**getDelta**

*Get the parameter delta*

---

**Description**

Get the parameter delta

**Usage**

```r
getDelta(object)
```

```r
## S4 method for signature 'MultiplicativeAlgorithm'
getDelta(object)
```

**Arguments**

- `object` An object from the class `MultiplicativeAlgorithm`.

**Value**

A numeric giving the parameter delta.

---

**getDerivatives**

*Get the derivatives of the model error equation.*

---

**Description**

Get the derivatives of the model error equation.

**Usage**

```r
generateDerivatives(object)
```

```r
## S4 method for signature 'ModelError'
generateDerivatives(object)
```

**Arguments**

- `object` An object from the class `ModelError`.

**Value**

The derivatives of the model error equation.
getDescription

Get the description of a model.

**Description**

Get the description of a model.

**Usage**

getDescription(object)

```r
## S4 method for signature 'Model'
getDescription(object)
```

**Arguments**

- `object` An object from the class `Model`.

**Value**

A list giving the description of a model.

getDesigns

Get the designs.

**Description**

Get the designs.

**Usage**

getDesigns(object)

```r
## S4 method for signature 'PFIMProject'
getDesigns(object)
```

**Arguments**

- `object` An object from the class `PFIMProject`.

**Value**

A list giving the designs of the object.
getDeterminant

Get the determinant of the fim.

Description
Get the determinant of the fim.

Usage
getDeterminant(object)

## S4 method for signature 'Fim'
getDeterminant(object)

## S4 method for signature 'Evaluation'
getDeterminant(object)

## S4 method for signature 'Optimization'
getDeterminant(object)

Arguments

object An object from the class Fim.

Value
A numeric giving the determinant of the fim.

generate

getDistribution

Get the distribution.

Description
Get the distribution.

Usage
getDistribution(object)

## S4 method for signature 'ModelParameter'
getDistribution(object)

Arguments

object An object from the class ModelParameter.
**getDose**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the amount of doses.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getDose(object)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>object</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The numeric amount_dose giving the amount of doses.</td>
</tr>
</tbody>
</table>

---

**getEigenValues**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the eigenvalues of the fim.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getEigenValues(object)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>object</code></td>
</tr>
</tbody>
</table>
getEquation

Get the equation of a model error.

Description

Get the equation of a model error.

Usage

getEquation(object)

## S4 method for signature 'ModelError'
getCodeuation(object)

Arguments

- object: An object from the class ModelError.

Value

A list containing the results of the evaluation of the elementary protocols giving the numberOfTimes, nbOfDimensions, totalCost, samplingTimes and the fisherMatrices.

getElementaryProtocols

Get the elementary protocols.

Description

Get the elementary protocols.

Usage

getElementaryProtocols(object, fims)

## S4 method for signature 'Optimization'
getCodeelementaryProtocols(object, fims)

Arguments

- object: An object from the class Optimization.
- fims: A list of object from the class Fim.

Value

A vector giving the eigenvalues of the fim.
getEquations

Value

An expression giving the equation of a model error.

Description

Get the equations of a model.

Usage

getEquations(object)

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

Arguments

object An object from the class Model.

Value

The list giving the equations of the model.

getEquationsAfterInfusion

Get the equations after infusion.

Description

Get the equations after infusion.

Usage

getEquationsAfterInfusion(object)

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

Arguments

object An object from the class Model.

Value

A list giving the equations after the infusion.
getEquationsDuringInfusion

*Get the equations during infusion.*

**Description**
Get the equations during infusion.

**Usage**
```r
generateEquations(Infusion)
```

## S4 method for signature 'Model'
generateEquations(Infusion)

**Arguments**
- `object`: An object from the class `Model`.

**Value**
A list giving the equations during the infusion.

getEvaluationFIMResults

*Get the results of the evaluation.*

**Description**
Get the results of the evaluation.

**Usage**
```r
generateEvaluationFIMResults(Infusion)
```

## S4 method for signature 'Optimization'
generateEvaluationFIMResults(Infusion)

**Arguments**
- `object`: An object from the class `Optimization`.

**Value**
An object from the class `Evaluation` giving the evaluation results for the optimal design.
**getEvaluationInitialDesignResults**

Get the evaluation results of the initial design.

**Description**

Get the evaluation results of the initial design.

**Usage**

```r
getEvaluationInitialDesignResults(object)
## S4 method for signature 'Optimization'
getEvaluationInitialDesignResults(object)
```

**Arguments**

- `object`: An object from the class `Optimization`.

**Value**

The object from the class `Evaluation` giving the results of the evaluation of the initial design.

---

**getFim**

**Description**

Get the fim of an of an object.

**Usage**

```r
getFim(object)
## S4 method for signature 'Design'
getFim(object)

## S4 method for signature 'PFIMProject'
getFim(object)

## S4 method for signature 'OptimizationAlgorithm'
getFim(object)
```

**Arguments**

- `object`: An object defined form a class of PFIM.
getFisherMatrix

Get the FIM.

Description
Get the FIM.

Usage
getFisherMatrix(object)

## S4 method for signature 'Fim'
getFisherMatrix(object)

## S4 method for signature 'Evaluation'
getFisherMatrix(object)

## S4 method for signature 'Optimization'
getFisherMatrix(object)

Arguments

object An object from the class Fim.

Value
A matrix giving the FIM.

getFixedEffects

Get the matrix of fixed effects.

Description
Get the matrix of fixed effects.

Usage
getFixedEffects(object)

## S4 method for signature 'Fim'
getFixedEffects(object)
getFixedMu

Arguments

object An object from the class Fim.

Value

The matrix of the fixed effects.

Description

Get the fixed effect.

Usage

getFixedMu(object)

## S4 method for signature 'ModelParameter'
getFixedMu(object)

Arguments

object An object from the class ModelParameter.

Value

A boolean giving the fixed mu.

getFixedOmega

Description

Get the fixed variance.

Usage

getFixedOmega(object)

## S4 method for signature 'ModelParameter'
getFixedOmega(object)

Arguments

object An object from the class ModelParameter.
getFixedTimes

Value

A boolean giving the fixed omega.

getFixedParameters  Get the fixed parameters.

Description

Get the fixed parameters.

Usage

getFixedParameters(object)

## S4 method for signature 'Model'
generic

getFixedParameters(object)

Arguments

object  An object from the class Model.

Value

A list giving the fixed parameters of the model.

generic

getFixedTimes  Get the fixed sampling times.

Description

Get the fixed sampling times.

Usage

generic

getFixedTimes(object)

## S4 method for signature 'SamplingTimeConstraints'
generic

getFixedTimes(object)

Arguments

object  An object from the class SamplingTimeConstraints.

Value

A vector giving the fixed sampling times.
getInitialConditions

Description
Get the initial condition for the evaluation of an ode model.

Usage
getInitialConditions(object)

## S4 method for signature 'Arm'
getInitialConditions(object)

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

Arguments

object An object Arm from the class Arm.

Value
The list initialConditions for the object Arm.

getIterationAndCriteria

Get the iteration with the convergence criteria.

Description
Get the iteration with the convergence criteria.

Usage
getIterationAndCriteria(object)

## S4 method for signature 'OptimizationAlgorithm'
getIterationAndCriteria(object)

Arguments

object An object from the class OptimizationAlgorithm.

Value
A dataframe giving the iteration with the convergence criteria.
getLambda

Description
Get the parameter lambda.

Usage
getLambda(object)

ARGS
object

Value
A numeric giving the parameter lambda.

getLibraryPDModels

Description
Get the library of PD models.

Usage
getLibraryPDModels(object)

ARGS
object

Value
A list giving the PD models.
**getLibraryPKModels**  
*Get the library of PK models.*

**Description**  
Get the library of PK models.

**Usage**  
getLibraryPKModels(object)

```r
## S4 method for signature 'LibraryOfModels'
getLibraryPKModels(object)
```

**Arguments**  
- `object`  
  An object from the class `LibraryOfModels`.

**Value**  
A list giving the PK models.

**getMinSampling**  
*Get the minimal sampling times.*

**Description**  
Get the minimal sampling times.

**Usage**  
getMinSampling(object)

```r
## S4 method for signature 'SamplingTimeConstraints'
getMinSampling(object)
```

**Arguments**  
- `object`  
  An object from the class `SamplingTimeConstraints`.

**Value**  
A numeric giving the minimal sampling times.
getModel

Description
Get the model.

Usage
getModel(object)

## S4 method for signature 'PFIMProject'
getModel(object)

Arguments
object An object from the class PFIMProject.

Value
The model of the object.

getModelEquations

Description
Get the model equations.

Usage
getModelEquations(object)

## S4 method for signature 'PFIMProject'
getModelEquations(object)

Arguments
object An object from the class PFIMProject.

Value
A list giving the model equations.
getModelError

Get the model error.

Description
Get the model error.

Usage
getModelError(object)

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

## S4 method for signature 'PFIMProject'
getModelError(object)

Arguments
object An object defined from a class of PFIM.

Value
The model error of the object.

getModelErrorParametersValues

Get the values of the model error parameters.

Description
Get the values of the model error parameters.

Usage
getModelErrorParametersValues(object)

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

Arguments
object An object from the class Model.

Value
A list giving the values of the model error parameters.
getModelFromLibrary  Get a model from the library of models.

Description
Get a model from the library of models.

Usage
getModelFromLibrary(object)

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

Arguments
object An object from the class Model.

Value
Return a model from the library of models.

getModelParameters  Get the model parameters.

Description
Get the model parameters.

Usage
getModelParameters(object)

## S4 method for signature 'PFIMProject'
getModelParameters(object)

Arguments
object An object from the class PFIMProject.

Value
A list giving the model parameters.
getModelParametersValues

Get the values of the model parameters.

Description
Get the values of the model parameters.

Usage
getModelParametersValues(object)

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

Arguments
object An object from the class Model.

Value
A list giving the values of the model parameters.

getMu

describe

Description
Get the fixed effect of an object.

Usage
getMu(object)

# S4 method for signature 'Distribution'
getMu(object)

# S4 method for signature 'ModelParameter'
getMu(object)

Arguments
object An object defined form a class of PFIM.

Value
The object with the updated fixed effect.
**Description**

Get the name of an object.

**Usage**

```r
getName(object)
```

## S4 method for signature 'Arm'

```r
getName(object)
```

## S4 method for signature 'Design'

```r
getName(object)
```

## S4 method for signature 'ModelParameter'

```r
getName(object)
```

## S4 method for signature 'LibraryOfModels'

```r
getName(object)
```

## S4 method for signature 'Model'

```r
getName(object)
```

## S4 method for signature 'PFIMProject'

```r
getName(object)
```

**Arguments**

- `object`  
  An object defined form a class of PFIM.

**Value**

A character string name giving the name of the object.

**Description**

Get the names of an object.
**getNumberOfArms**

Usage

```r
getNames(object)
## S4 method for signature 'list'
getNames(object)
```

Arguments

- `object` An object defined form a class of PFIM.

Value

A vector giving the names of the object.

---

**getNumberOfArms**

Description

Get the number of arms in a design.

Usage

```r
getNumberOfArms(object)
## S4 method for signature 'Design'
getNumberOfArms(object)
```

Arguments

- `object` An object Design from the class `Design`.

Value

A numeric `numberOfArms` giving the number of arms in the design.
### `getNumberOfIterations`  
Get the number of iterations.

**Description**
Get the number of iterations.

**Usage**
```r
getNumberOfIterations(object)
```

```r
## S4 method for signature 'MultiplicativeAlgorithm'
getNumberOfIterations(object)
```

**Arguments**
- `object`: An object from the class `MultiplicativeAlgorithm`.

**Value**
- A numeric giving the number of iterations.

### `getNumberOfParameters`  
Get the number of parameters.

**Description**
Get the number of parameters.

**Usage**
```r
getNumberOfParameters(object)
```

```r
## S4 method for signature 'Model'
getNumberOfParameters(object)
```

**Arguments**
- `object`: An object from the class `Model`.

**Value**
- A numeric giving the number of parameters of the model.
getNumberOfsamplingsOptimisable

Get the number of sampling times that are optimisable.

Description

Get the number of sampling times that are optimisable.

Usage

getNumberOfsamplingsOptimisable(object)

## S4 method for signature 'SamplingTimeConstraints'
getNumberOfsamplingsOptimisable(object)

Arguments

object An object from the class SamplingTimeConstraints.

Value

A vector giving the number of sampling times that are optimisable.

getNumberOfTimesByWindows

Get the number of sampling times by windows.

Description

Get the number of sampling times by windows.

Usage

getNumberOfTimesByWindows(object)

## S4 method for signature 'SamplingTimeConstraints'
getNumberOfTimesByWindows(object)

Arguments

object An object from the class SamplingTimeConstraints.

Value

A vector giving the number of sampling times by windows.
**getOdeSolverParameters**

Get the parameters for the ode solvers of an object.

**Usage**

getOdeSolverParameters(object)

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

## S4 method for signature 'PFIMProject'
getOdeSolverParameters(object)

**Arguments**

object An object defined form a class of PFIM.

**Value**

The list giving the parameters for the ode solvers.

---

**getOmega**

*Get the matrix omega of an object.*

**Description**

Get the matrix omega of an object.

**Usage**

getOmega(object)

## S4 method for signature 'Distribution'
getOmega(object)

## S4 method for signature 'ModelParameter'
getOmega(object)

**Arguments**

object An object defined form a class of PFIM.
getOptimalDesign

Value
The matrix omega of an object.

getOptimalDesign Get the optimal design.

Description
Get the optimal design.

Usage
getOptimalDesign(object)

## S4 method for signature 'OptimizationAlgorithm'
getOptimalDesign(object)

Arguments
object An object from the class OptimizationAlgorithm.

Value
The optimal design.

gOptimalFrequencies Get the optimal frequencies

Description
Get the optimal frequencies.

Usage
gOptimalFrequencies(object)

## S4 method for signature 'FedorovWynnAlgorithm'
gOptimalFrequencies(object)

Arguments
object An object from the class FedorovWynnAlgorithm.

Value
A vector giving the optimal frequencies
getOptimalWeights

Get the optimal weights.

Description
Get the optimal weights.

Usage
getOptimalWeights(object)

Arguments
object An object from the class MultiplicativeAlgorithm.

Value
A vector giving the optimal weights.

getOptimizationResults

Get the optimization results.

Description
Get the optimization results.

Usage
getOptimizationResults(object)

Arguments
object An object from the class Optimization.

Value
An object from the class OptimizationAlgorithm giving the optimization results.
getOptimizer

Description
Get the optimization algorithm.

Usage
getOptimizer(object)

## S4 method for signature 'PFIMProject'
getOptimizer(object)

Arguments
object An object from the class PFIMProject.

Value
A string giving the name of the optimization algorithm.

getOptimizerParameters

Description
Get the optimization parameters.

Usage
getOptimizerParameters(object)

## S4 method for signature 'PFIMProject'
getOptimizerParameters(object)

Arguments
object An object from the class PFIMProject.

Value
A list giving the optimization parameters.
getOutcome

Description
Get the outcome of an object.

Usage
getOutcome(object)

## S4 method for signature 'Administration'
getOutcome(object)

## S4 method for signature 'AdministrationConstraints'
getOutcome(object)

## S4 method for signature 'ModelError'
getOutcome(object)

## S4 method for signature 'SamplingTimeConstraints'
getOutcome(object)

## S4 method for signature 'SamplingTimes'
getOutcome(object)

Arguments

object An object defined from a class of PFIM.

Value
A string giving the outcome of the object.

getOutcomes

Get the outcomes of a model.

Description
Get the outcomes of a model.
getOutcomesEvaluation

Usage

getOutcomes(object)

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

## S4 method for signature 'PFIMProject'
getOutcomes(object)

Arguments

object An object from the class Model.

Value

A list giving the outcomes of the model.

getOutcomesEvaluation getOutcomesEvaluation

Description

Get the results of the evaluation of the outcomes.

Usage

getOutcomesEvaluation(object)

## S4 method for signature 'Design'
getOutcomesEvaluation(object)

Arguments

object An object Design from the class Design.

Value

The list outcomesEvaluation containing the results of the design evaluation for the outcomes.
getOutcomesForEvaluation

Get the outcomes of a model used for the evaluation (is scales outcomes).

Description
Get the outcomes of a model used for the evaluation (is scales outcomes).

Usage
getOutcomesForEvaluation(object)

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

Arguments

object An object from the class Model.

Value
A list giving the outcomes of a model used for the evaluation (is scales outcomes).

getOutcomesGradient

Description
Get the results of the evaluation of the outcome gradients.

Usage
getOutcomesGradient(object)

## S4 method for signature 'Design'
getOutcomesGradient(object)

Arguments

object An object Design from the class Design.

Value
The list outcomesGradient containing the results of the design evaluation for the outcome gradients.
getParameters

Get the parameters of an object.

Description
Get the parameters of an object.

Usage
getParameters(object)

## S4 method for signature 'ModelError'
getParameters(object)

## S4 method for signature 'Distribution'
getParameters(object)

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

Arguments
object An object defined form a class of PFIM.

Value
Return the list of the parameters of the object.

getPDModel

Get a PD model.

Description
Get a PD model.

Usage
getPDModel(object, PDModelName)

## S4 method for signature 'LibraryOfPKPDM'odels'
getPDModel(object, PDModelName)

Arguments
object An object from the class LibraryOfPKPDMModels.
PDModelName A string giving the name of the PD model.
getPKModel

Description
Get a PK model.

Usage
getPKModel(object, PKModelName)

Arguments
object An object from the class LibraryOfPKPDModels.
PKModelName A string giving the name of the PK model.

Value
Return a PK model.

getPKPDModel

Description
Get a PKPD model.

Usage
getPKPDModel(object, namesModel)

Arguments
object An object from the class LibraryOfPKPDModels.
namesModel A vector of strings giving the names of the PK and PD models.

Value
Return a PKPD model.
getPlotOptions

Get the plot options for graphs responses and SI

Description
Get the plot options for graphs responses and SI

Usage
getPlotOptions(plotOptions, outcomesNames)

Arguments
plotOptions A list giving the plots options.
outcomesNames A list giving the output names.

Value
The list containing the plot options.

getProportionsOfSubjects

Get the proportion of subjects.

Description
Get the proportion of subjects.

Usage
getProportionsOfSubjects(object)

Arguments
object An object from the class Optimization.

Value
A vector giving the proportion of subjects.
getRSE

Get the RSE

Description

Get the RSE

Usage

getRSE(object, model)

## S4 method for signature 'BayesianFim'
getRSE(object, model)

## S4 method for signature 'Evaluation'
getRSE(object, model)

## S4 method for signature 'IndividualFim'
getRSE(object, model)

## S4 method for signature 'Optimization'
getRSE(object, model)

## S4 method for signature 'PopulationFim'
getRSE(object, model)

Arguments

object
An object from the class Fim.

model
An object from the class Model.

Value

A vector giving the RSE.

getSamplings

Get the sampling of an object.

Description

Get the sampling of an object.
getSamplingsWindows

Usage

getSamplings(object)

## S4 method for signature 'SamplingTimeConstraints'
getSamplings(object)

## S4 method for signature 'SamplingTimes'
getSamplings(object)

Arguments

object An object defined form a class of PFIM.

Value

A list of the samplings of the object.

getSamplingsWindows Get the windows for the sampling times.

Description

Get the windows for the sampling times.

Usage

getSamplingsWindows(object)

## S4 method for signature 'SamplingTimeConstraints'
getSamplingsWindows(object)

Arguments

object An object from the class SamplingTimeConstraints.

Value

A list giving the vector of the windows for the sampling times.
### getSamplingTime

**Description**

Get the sampling times by outcome.

**Usage**

```r
getSamplingTime(object, outcome)
```

#### S4 method for signature 'Arm'

```r
getsamplingTime(object, outcome)
```

**Arguments**

- `object`: An object `Arm` from the class `Arm`.
- `outcome`: A string giving the name of the outcome.

**Value**

The element of the list `samplingTimes` containing the sampling times of the outcome `outcome`.

### getSamplingTimeConstraint

**Description**

Get the sampling times constraints by outcome.

**Usage**

```r
getsamplingTimeConstraint(object, outcome)
```

#### S4 method for signature 'Arm'

```r
getsamplingTimeConstraint(object, outcome)
```

**Arguments**

- `object`: An object `Arm` from the class `Arm`.
- `outcome`: A string giving the name of the outcome.

**Value**

The element of the list `samplingTimesConstraints` containing the sampling times constraints of the outcome `outcome`.
getSamplingTimes

Description
Get the vectors of sampling times for an arm.

Usage
getSamplingTimes(object)

## S4 method for signature 'Arm'
getSamplingTimes(object)

Arguments
object An object Arm from the class Arm.

Value
The list samplingTimes for the object Arm.

getSamplingTimesConstraints

Description
Get the sampling times constraints.

Usage
getSamplingTimesConstraints(object)

## S4 method for signature 'Arm'
getSamplingTimesConstraints(object)

Arguments
object An object Arm from the class Arm.

Value
The list getSamplingTimesConstraints.
getSE

\textit{Get the SE.}

\section*{Description}
Get the SE.

\section*{Usage}
\begin{verbatim}
getSE(object)
\end{verbatim}

\section*{Arguments}
\begin{itemize}
  \item \texttt{object} \hspace{1cm} An object from the class \texttt{Fim}.
\end{itemize}

\section*{Value}
A vector giving the SE.

getShrinkage

\textit{Get the shrinkage.}

\section*{Description}
Get the shrinkage.

\section*{Usage}
\begin{verbatim}
getShrinkage(object)
\end{verbatim}

\section*{Arguments}
\begin{itemize}
  \item \texttt{object} \hspace{1cm} An object from the class \texttt{BayesianFim}, \texttt{Evaluation}, or \texttt{IndividualFim}.
\end{itemize}

\section*{Value}
A vector giving the shrinkage.
getShrinkage(object)

## S4 method for signature 'Optimization'
getShrinkage(object)

## S4 method for signature 'PopulationFim'
getShrinkage(object)

**Arguments**

- `object` An object from the class `Fim`.

**Value**

A vector giving the shrinkage of the Bayesian fim.

---

getSigmaInter Get the parameter sigma inter.

**Description**

Get the parameter sigma inter.

**Usage**

getSigmaInter(object)

## S4 method for signature 'ModelError'
getSigmaInter(object)

**Arguments**

- `object` An object from the class `ModelError`.

**Value**

A numeric giving the parameter sigma inter.
**getDescription**

*GetSize*

**Description**

Get the parameter sigma slope.

**Usage**

```r
getSigmaSlope(object)
```

## S4 method for signature 'ModelError'

gSigmaSlope(object)

**Arguments**

- **object**: An object from the class `ModelError`.

**Value**

A numeric giving the parameter sigma slope.

---

**getSize**

*GetSize*

**Description**

Get the size of an object.

**Usage**

```r
getSize(object)
```

## S4 method for signature 'Arm'

gSize(object)

## S4 method for signature 'Design'

gSize(object)

**Arguments**

- **object**: An object defined form a class of PFIM.

**Value**

A numeric giving the size of the object.
getTau

Description
Get the frequency \( \tau \).

Usage
getTau(object)

## S4 method for signature 'Administration'
getTau(object)

Arguments
object An object Administration from the class Administration.

Value
The numeric \( \tau \) giving the frequency \( \tau \).

gTimeDose

Description
Get the times vector when doses are given.

Usage
gTimeDose(object)

## S4 method for signature 'Administration'
gTimeDose(object)

Arguments
object An object Administration from the class Administration.

Value
The vector timeDose giving the times when the doses are given.
getTinf  

Get the infusion duration.

Description
Get the infusion duration.

Usage
getTinf(object)

## S4 method for signature 'Administration'
getTinf(object)

Arguments
object An object Administration from the class Administration.

Value
The numeric Tinf giving the infusion duration Tinf.

getVarianceEffects  

Get the matrix of the variance effects.

Description
Get the matrix of the variance effects.

Usage
getVarianceEffects(object)

## S4 method for signature 'Fim'
getVarianceEffects(object)

Arguments
object An object from the class Fim.

Value
The matrix of the variance effects.
Description
A class storing information regarding the individual Fisher matrix. The class IndividualFim inherits from the class Fim.

isDoseInEquations Test if the dose is in the equations of the model.

Description
Test if the dose is in the equations of the model.

Usage
isDoseInEquations(object)

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

Arguments

object An object from the class Model.

Value
Return a Boolean giving if the dose is in the equations of the model.

isModelAnalytic Test if a mode is analytic.

Description
Test if a mode is analytic.

Usage
isModelAnalytic(object)

## S4 method for signature 'Model'
isModelAnalytic(object)
Arguments

object An object from the class Model.

Value

Return a Boolean giving if the mode is analytic or not.

Description

Test if a mode is bolus.

Usage

isModelBolus(object, designs)

## S4 method for signature 'Model'
isModelBolus(object, designs)

Arguments

object An object from the class Model.
designs A list of objects from the class Design.

Value

Return a Boolean giving if the mode is bolus or not.

Description

Test if a mode is infusion

Usage

isModelInfusion(object)

## S4 method for signature 'Model'
isModelInfusion(object)
### isModelODE

**Description**
Test if a mode is ode.

**Usage**
```r
isModelODE(object)
```

#### Arguments
- **object** An object from the class Model.

**Value**
Return a Boolean giving if the mode is infusion or not.

### isModelSteadyState

**Description**
Test if a mode is steady state.

**Usage**
```r
isModelSteadyState(object)
```

#### Arguments
- **object** An object from the class Model.

**Value**
Return a Boolean giving if the mode is ode or not.
Value

Return a Boolean giving if the mode is steady state or not.

LibraryOfModels-class

Class "LibraryOfModels"

Description

The class LibraryOfModels represents the library of models.

Objects from the class

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

Slots for LibraryOfModels objects

name: A string giving the name of the library of models.
content: A list giving the content of the library of model.

LibraryOfPDModels

Library of the PK models

Description

Library of the PK models

Usage

LibraryOfPDModels()

LibraryOfPKModels

Library of the PK models

Description

Library of the PK models

Usage

LibraryOfPKModels()
LibraryOfPKPDModels-class  

Class "LibraryOfPKPDModels"

Description

The class LibraryOfPKPDModels represents the library of PKPD models. The class LibraryOfPKPDModels inherits from the class LibraryOfModels.

LogNormal-class  

Class "LogNormal"

Description

The class defines all the required methods for a LogNormal distribution object. The class LogNormal inherits from the class Distribution.

Model-class  

Class "Model"

Description

The class Model defines information concerning the construction of a model.

Objects from the class

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

Slots for Administration objects

name: A string giving the name of the model.
description: A list of string giving the description of the model.
equations: A list giving the equations of the model.
outcomes: A list giving the outcomes of the model.
outcomesForEvaluation: A list giving the outcomes used for the evaluation of the model.
parameters: A list giving the parameters of the model.
modelError: A list giving the model error of the model.
initialConditions: A list giving the initial conditions of the model.
odeSolverParameters: A list giving the parameters for the solver of the model.
modelFromLibrary: A list giving the model equations when the model is constructed from the library of model.
**ModelAnalytic-class**

*Class "ModelAnalytic"*

**Description**

The class Model defines information concerning the construction of an analytical model. The class ModelAnalytic inherits from the class Model.

**ModelAnalyticBolus-class**

*Class "ModelAnalyticBolus"*

**Description**

The class Model defines information concerning the construction of an analytical bolus model. The class ModelAnalyticBolus inherits from the class ModelAnalytic.

**ModelAnalyticBolusSteadyState-class**

*Class "ModelAnalyticBolusSteadyState"*

**Description**

The class Model defines information concerning the construction of an analytical model in steady state. The class ModelAnalyticBolusSteadyState inherits from the class ModelAnalyticSteadyState.

**ModelAnalyticInfusion-class**

*Class "ModelAnalyticInfusion"*

**Description**

The class Model defines information concerning the construction of an analytical model in infusion. The class ModelAnalyticInfusion inherits from the class ModelInfusion.
ModelAnalyticInfusionSteadyState-class

Class "ModelAnalyticInfusionSteadyState"

Description
The class Model defines information concerning the construction of an analytical model in infusion in steady state. The class ModelAnalyticInfusionSteadyState inherits from the class ModelAnalyticInfusion.

ModelAnalyticSteadyState-class

Class "ModelAnalyticSteadyState"

Description
The class ModelAnalyticSteadyState defines information concerning the construction of an analytical model steady state. The class ModelAnalyticSteadyState inherits from the class ModelAnalytic.

ModelBolus-class

Class "ModelBolus"

Description
...

ModelError-class

Class "ModelError" representing a Model error.

Description
...

ModelInfusion-class

Class "ModelInfusion"

Description
...
The class `ModelODE` defines information concerning the construction of an ode model. The class `ModelODE` inherits from the class `Model`.

The class `ModelODEBolus` defines information concerning the construction of an ode model bolus. The class `ModelODEBolus` inherits from the class `ModelBolus`.

The class `ModelODEDoseInEquations` defines information concerning the construction of an ode model where the dose is in the model equations. The class `ModelODEDoseInEquations` inherits from the class `ModelODE`.

The class `ModelODEDoseNotInEquations` defines information concerning the construction of an ode model where the dose is not in the model equations. The class `ModelODEDoseNotInEquations` inherits from the class `ModelODE`.

The class `ModelODEInfusion` defines information concerning the construction of an ode model in infusion. The class `ModelODEInfusion` inherits from the class `ModelInfusion`. 
ModelODEInfusionDoseInEquations-class
Class "ModelODEInfusionDoseInEquations"

Description
The class ModelODEInfusionDoseInEquations defines information concerning the construction of an ode model in infusion where the dose is in the model equations. The class ModelODEInfusionDoseInEquations inherits from the class ModelODEInfusion.

ModelParameter-class
Class "ModelParameter"

Description
The class ModelParameter defines information concerning the model parameters.

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 string giving the name of the parameter.
distribution: An object from the class Distribution giving the distribution of the parameter.
fixedMu: A boolean giving if mu is fixed or not.
fixedOmega: A boolean giving if omega is fixed or not.

MultiplicativeAlgorithm-class
Class "MultiplicativeAlgorithm"

Description
The 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

- **arms**: A list giving the arms.
- **lambda**: A numeric giving the lambda parameter of the multiplicative algorithm.
- **delta**: A numeric giving the delta parameter of the multiplicative algorithm.
- **numberOfIterations**: A numeric giving the maximal number iteration of the optimization process.
- **optimalWeights**: A vector giving the optimal weights.
- **optimalDesign**: An object of the class `Design` giving the optimal design.
- **showProcess**: A boolean for showing or not the process of optimization.

---

**MultiplicativeAlgorithm_Rcpp**

*Function `MultiplicativeAlgorithm_Rcpp`*

---

**Description**

Run the `MultiplicativeAlgorithm_Rcpp` in Rcpp

**Usage**

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

**Arguments**

- `fisherMatrices_input`
- `numberOfFisherMatrices_input`
- `weights_input`
- `numberOfParameters_input`
- `dim_input`
- `lambda_input`
- `delta_input`
- `iterationInit_input`
Normal-class

Description

The class defines all the required methods for a Normal distribution object. The class Normal inherits from the class Distribution.

Optimization-class

Description

A class storing information concerning the design optimization.

Objects from the class

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

Slots for Administration objects

name: A character string giving the name of the optimization process.
model: A object of class Model giving the model.
modelEquations: A list giving the model equations.
modelParameters: A list giving the model parameters.
ModelError: A list giving the model error.
optimizer: A object of class OptimizationAlgorithm giving the optimization algorithm.
optimizerParameters: A list giving the parameters of the optimization algorithm.
outcomes: A list giving the outcomes of the model.
designs: A list giving the designs to be optimized.
fim: A object of class FIM giving the Fisher information matrix.
odeSolverParameters: A list giving the parameters for the ode solver.
optimizationResults: A object of class OptimizationAlgorithm giving the results of the optimization.
evaluationFIMResults: A object of class Evaluation giving the results of the evaluation of the optimal design.
evaluationInitialDesignResults: A object of class Evaluation giving the results of the evaluation of the initial design.
OptimizationAlgorithm-class

Class "OptimizationAlgorithm"

Description

A class storing information concerning the optimization algorithm.

Objects from the class

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

Slots for Administration objects

name: A character string giving the name of the optimization algorithm.

parameters: A list giving the parameters of the optimization algorithm.

optimize

Optimize a design.

Description

Optimize a design.

Usage

optimize(object, optimizerParameters, optimizationObject)

## S4 method for signature 'FedorovWynnAlgorithm'
optimize(object, optimizerParameters, optimizationObject)

## S4 method for signature 'MultiplicativeAlgorithm'
optimize(object, optimizerParameters, optimizationObject)

## S4 method for signature 'PGBOAlgorithm'
optimize(object, optimizationObject)

## S4 method for signature 'PSOAlgorithm'
optimize(object, optimizationObject)

## S4 method for signature 'SimplexAlgorithm'
optimize(object, optimizationObject)
parametersForComputingGradient

Arguments

- object: An object from the class OptimizationAlgorithm.
- optimizerParameters: A list giving the optimization parameters.
- optimizationObject: An object giving the optimization algorithm.

Value

A list giving the results if the optimization.

Description

Define the parameters for computing the gradients of a model.

Usage

parametersForComputingGradient(object)

PFIMProject-class

Class "PFIMProject"

Description

A class storing information concerning a PFIM project.

Objects from the class

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

Slots for **PFIMProject** objects

- **name**: A character string giving the name of the PFIM project.
- **description**: A list giving the description of the PFIM project.

Description

The class "PGBOAlgorithm" implements the PGBO algorithm: Population Genetics Based Optimizer, developed 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.
- **maxIteration**: A numeric giving the maximum number of iterations.
- **seed**: A numeric giving the seed.
- **showProcess**: A boolean to show or not the process.
- **optimalDesign**: A Design object giving the optimal design.
- **iterationAndCriteria**: A list giving the optimal criteria at each iteration.

References

**plotEvaluation**

*Graphs of the results of the evaluation.*

### Description

Graphs of the results of the evaluation.

### Usage

```r
plotEvaluation(object, plotOptions)
```

#### Arguments

- `object` An object from the class `Evaluation`.
- `plotOptions` A list giving the plot options.

### Value

A list giving the graphs for the evaluation of the responses and sensitivity indices.

---

**PlotEvaluation-class**

*Class "PlotEvaluation"*

### Description

A class storing information concerning the design evaluation. The class `PlotEvaluation` inherits from the class `Evaluation`.

---

**plotFrequencies**

*Graph of the frequencies for the FW algorithm.*

### Description

Graph of the frequencies for the FW algorithm.
### plotOutcomesEvaluation

#### Usage

```r
plotFrequencies(object, threshold)
```

```r
## S4 method for signature 'FedorovWynnAlgorithm'
plotFrequencies(object, threshold)
```

```r
## S4 method for signature 'Optimization'
plotFrequencies(object, threshold)
```

#### Arguments

- **object**: An object from the class `OptimizationAlgorithm`.
- **threshold**: A numeric giving the threshold for the frequencies for the FW algorithm.

#### Value

The graphs of the frequencies for the FW algorithm.

---

### plotOutcomesEvaluation

#### Description

Plot the evaluation of the outcomes.

#### Usage

```r
plotOutcomesEvaluation(object, initialDesign, model, plotOptions)
```

```r
## S4 method for signature 'Design'
plotOutcomesEvaluation(object, initialDesign, model, plotOptions)
```

#### Arguments

- **object**: An object Design from the class `Design`.
- **initialDesign**: An object design from the class `Design`.
- **model**: An object model from the class `Model`.
- **plotOptions**: A list containing the plot options.

#### Value

A list containing the plots the evaluation of the outcomes.
plotOutcomesGradient

Description
Plot the evaluation of the outcome gradients.

Usage
plotOutcomesGradient(object, initialDesign, model, plotOptions)

## S4 method for signature 'Design'
plotOutcomesGradient(object, initialDesign, model, plotOptions)

Arguments
- object: An object design from the class Design.
- initialDesign: An object design from the class Design.
- model: An object model from the class Model.
- plotOptions: A list containing the plot options.

Value
A list containing the plots the evaluation of the outcome gradients.

plotRSE
Graph of the RSE.

Description
Graph of the RSE.

Usage
plotRSE(object, plotOptions)

## S4 method for signature 'PFIMProject'
plotRSE(object, plotOptions)

Arguments
- object: An object from the class Evaluation.
- plotOptions: A list giving the plot options.

Value
A graph of the RSE.
plotSE  

*Graph the SE.*

**Description**

Graph the SE.

**Usage**

```r
plotSE(object, plotOptions)
```

```r
## S4 method for signature 'PFIMProject'
plotSE(object, plotOptions)
```

**Arguments**

- `object`  
  An object from the class `Evaluation`.

- `plotOptions`  
  A list giving the plot options.

**Value**

A graph of the SE.

---

plotSensitivityIndice  

*Graphs of the results of the evaluation.*

**Description**

Graphs of the results of the evaluation.

**Usage**

```r
plotSensitivityIndice(object, plotOptions)
```

```r
## S4 method for signature 'Evaluation'
plotSensitivityIndice(object, plotOptions)
```

**Arguments**

- `object`  
  An object from the class `Evaluation`.

- `plotOptions`  
  A list giving the plot options.

**Value**

A list giving the graphs for the evaluation of the responses and sensitivity indices.
**plotShrinkage**

*Graph of the shrinkage.*

**Description**

Graph of the shrinkage.

**Usage**

```r
plotShrinkage(object, plotOptions)
```

## S4 method for signature 'PFIMProject'

```r
plotShrinkage(object, plotOptions)
```

**Arguments**

- **object** An object from the class `Evaluation`.
- **plotOptions** A list giving the plot options.

**Value**

A graph of the shrinkage.

---

**plotWeights**

*Graph of the weights for the multiplicative algorithm.*

**Description**

Graph of the weights for the multiplicative algorithm.

**Usage**

```r
plotWeights(object, threshold)
```

## S4 method for signature 'MultiplicativeAlgorithm'

```r
plotWeights(object, threshold)
```

## S4 method for signature 'Optimization'

```r
plotWeights(object, threshold)
```

**Arguments**

- **object** An object from the class `OptimizationAlgorithm`.
- **threshold** A numeric giving the threshold for the optimal weights in the multiplicative algorithm.
Value

The graphs of the weights for the multiplicative algorithm.

PopulationFim-class  

Class "PopulationFim"

Description

A class storing information regarding the population Fisher matrix. The class PopulationFim inherits from the class Fim.

Proportional-class  

Class "Proportional"

Description

The Class "Proportional" defines the the residual error variance according to the formula $g(\sigma_{\text{inter}}, \sigma_{\text{slope}}, c_{\text{error}}, f(x, \theta)) = \sigma_{\text{slope}} \cdot f(x, \theta)$.

Objects from the Class Proportional

Objects are typically created by calls to Proportional and contain the following slots that are inherited 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
Description

The class "PSOAlgorithm" implements the PSO algorithm.

Objects from the class PSOAlgorithm

Objects form the class 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 population size.
seed: A numeric giving the seed.
personalLearningCoefficient: A numeric giving the personal learning coefficient.
globalLearningCoefficient: A numeric giving the global learning coefficient.
showProcess: A boolean to show or not the process.
optimalDesign: A Design object giving the optimal design.
iterationAndCriteria: A list giving the optimal criteria at each iteration.

Report

Report

Description

Report

Usage

Report(object, outputPath, outputFile, plotOptions)

## S4 method for signature 'Evaluation'
Report(object, outputPath, outputFile, plotOptions)

## S4 method for signature 'Optimization'
Report(object, outputPath, outputFile, plotOptions)
Arguments

object An object from the class PFIMProject.
outputPath A string giving the output path.
outputFile A string giving the name of the output file.
plotOptions A list giving the plot options.

Value

The report in html.

Description

Generate table for the report.

Usage

reportTablesAdministration(object)

## S4 method for signature 'Design'
reportTablesAdministration(object)

Arguments

object An object design from the class Design.

Value

A table of the administration parameters for the report.

Description

Generate table for the report.

Usage

reportTablesDesign(object)

## S4 method for signature 'Design'
reportTablesDesign(object)
**reportTablesFIM**

**Arguments**

- **object**
  An object design from the class `Design`.

**Value**

A table of the design parameters for the report.

---

Generate the tables for the report.

**Description**

Generate the tables for the report.

**Usage**

```r
reportTablesFIM(object, evaluationObject)
```

```r
## S4 method for signature 'BayesianFim'
reportTablesFIM(object, evaluationObject)
```

```r
## S4 method for signature 'IndividualFim'
reportTablesFIM(object, evaluationObject)
```

```r
## S4 method for signature 'PopulationFim'
reportTablesFIM(object, evaluationObject)
```

**Arguments**

- **object**
  An object from the class `Fim`.
- **evaluationObject**
  A list giving the results of the evaluation of the model.

**Value**

A list giving the table in kable format for the report.
reportTablesModelError

*Generate the tables for model errors for the evaluation report.*

**Description**

Generate the tables for model errors for the evaluation report.

**Usage**

```r
reportTablesModelError(object)
```

**Arguments**

- `object` An object from the class `Model`.

**Value**

A `kable` table for the evaluation report.

---

reportTablesModelParameters

*Generate the tables for model parameters for the evaluation report.*

**Description**

Generate the tables for model parameters for the evaluation report.

**Usage**

```r
reportTablesModelParameters(object)
```

**Arguments**

- `object` An object from the class `Model`.

**Value**

A `kable` table for the evaluation report.
**reportTablesPlot**

**Description**
Generate all the table for the evaluation report

**Usage**

```
reportTablesPlot(object, plotOptions)
```

**Arguments**
- **object**: An object evaluation from the class `Evaluation`.
- **plotOptions**: A list containing the options for the plots.

**Value**
A list tables containing the tables for the evaluation report.

----

**reportTablesSamplingConstraints**

**Description**
Generate table for the report.

**Usage**

```
reportTablesSamplingConstraints(object)
```

**Arguments**
- **object**: An object design from the class `Design`.

**Value**
A table of the sampling constraints parameters for the report.
resizeFisherMatrix

Resizes the fisher matrix from a vector to a matrix.

Description

Resize the fisher matrix from a vector to a matrix.

Usage

resizeFisherMatrix(nbOfDimensions, fisherMatrix)

Arguments

- nbOfDimensions: 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 nbOfDimensions*nbOfDimensions

run

Description

Run

Usage

run(object)

Arguments

- object: An object from the class PFIMProject.

Value

A list giving the results of evaluation or optimization.
**SamplingTimeConstraints-class**

*Class "SamplingTimeConstraints"*

**Description**

The class "SamplingTimeConstraints" implements the constraints for the sampling times.

**Objects from the class** SamplingTimeConstraints

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

**Slots for SamplingTimeConstraints objects**

- **outcome**: A string giving the outcome.
- **initialSamplings**: A vector giving the sampling times.
- **fixedTimes**: A vector giving the fixed sampling times.
- **numberOfSamplingsOptimisable**: A vector giving the sampling times to be optimized.
- **samplingsWindows**: A list giving the windows for the sampling times.
- **numberOfTimesByWindows**: A vector giving the number of sampling times by windows.
- **minSampling**: A numeric giving the minimal sampling times.

---

**SamplingTimes-class**

*Class "SamplingTimes"*

**Description**

The class "SamplingTimes" implements the sampling times.

**Objects from the class** SamplingTimes

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

**Slots for SamplingTimes objects**

- **outcome**: A string giving the outcome.
- **samplings**: A vector giving the sampling times.
setAdministrations

Description
Set all the administration for an arm.

Usage
setAdministrations(object, administrations)

## S4 method for signature 'Arm'
setAdministrations(object, administrations)

Arguments

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

Value
The object Arm with the list administrations of objects from the class Administration class giving the parameters of the administration for the object Arm.

setArm

Description
Set the arms in a design.

Usage
setArm(object, arm)

## S4 method for signature 'Design'
setArm(object, arm)

Arguments

- **object**: An object Design from the class Design.
- **arm**: A list of object Arm giving the arms of the design.

Value
An object Design with the list Arm updated.
setArms

`setArms` is a method for setting the arms of an object. It is used in the context of the `PFIM` package.

**Description**

Set the arms of an object.

**Usage**

```r
setArms(object, arms)
```

**Arguments**

- `object`: An object defined from a class of `PFIM`.
- `arms`: A list of arms.

**Value**

The object with the updated arms.

---

setcError

`setcError` is a method for setting the parameter `c`.

**Description**

Set the parameter `c`.

**Usage**

```r
setcError(object, cError)
```

**Arguments**

- `object`: An object from the class `ModelError`.
- `cError`: A numeric giving the parameter `c`. 

---
setContent

Set content of a library of models.

Description
Set content of a library of models.

Usage
setContent(object, content)

## S4 method for signature 'LibraryOfModels'
setContent(object, content)

Arguments

object An object from the class LibraryOfModels.
content A list giving the content of the library of models.

Value
The library of models with the updated content.

setDerivatives

Set the derivatives of the model error equation.

Description
Set the derivatives of the model error equation.

Usage
setDerivatives(object, derivatives)

## S4 method for signature 'ModelError'
setDerivatives(object, derivatives)

Arguments

object An object from the class ModelError.
derivatives The derivatives of the model error equation.

Value
The model error with the updated model error equation.
**setDescription**

Set the description of a model.

**Usage**

```r
setDescription(object, description)
```

```r
## S4 method for signature 'Model'
setDescription(object, description)
```

**Arguments**

- `object` An object from the class `Model`.
- `description` A list giving the description of a model.

**Value**

The model with the updated description.

**setDesigns**

Set the designs.

**Description**

Set the designs.

**Usage**

```r
setDesigns(object, designs)
```

```r
## S4 method for signature 'Optimization'
setDesigns(object, designs)
```

**Arguments**

- `object` An object from the class `Optimization`.
- `designs` A list of objects from the class `Design`.

**Value**

The object with the new designs.
setDistribution

Set the distribution.

Description
Set the distribution.

Usage

setDistribution(object, distribution)

## S4 method for signature 'ModelParameter'
setDistribution(object, distribution)

Arguments

object An object from the class ModelParameter.
distribution An object from the class Distribution.

Value
The model parameter with the updated distribution.

setDose

Set the amount of dose

Description
Set the amount of dose

Usage

setDose(object, dose)

## S4 method for signature 'Administration'
setDose(object, dose)

Arguments

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

Value
The numeric amount_dose giving the new value of the amount of dose.
**setEquation**

Set the equation of a model error.

**Description**

Set the equation of a model error.

**Usage**

```r
setEquation(object, equation)
```

## S4 method for signature 'ModelError'

```r
setEquation(object, equation)
```

**Arguments**

- `object`: An object from the class `ModelError`.
- `equation`: An expression giving the equation of a model error.

**Value**

The model error with the updated equation.

---

**setEquations**

Set the equations of a model.

**Description**

Set the equations of a model.

**Usage**

```r
setEquations(object, equations)
```

## S4 method for signature 'Model'

```r
setEquations(object, equations)
```

**Arguments**

- `object`: An object from the class `Model`.
- `equations`: A list giving the equations of the model.

**Value**

The model with the updated equations.
setEquationsAfterInfusion

*Set the equations after infusion.*

### Description

Set the equations after infusion.

### Usage

```r
setEquationsAfterInfusion(object, equations)
```

```r
## S4 method for signature 'Model'
setEquationsAfterInfusion(object, equations)
```

### Arguments

- **object**: An object from the class `Model`.
- **equations**: A list giving the equations after the infusion.

### Value

The model with the updated equations after the infusion.

---

setEquationsDuringInfusion

*Set the equations during infusion.*

### Description

Set the equations during infusion.

### Usage

```r
setEquationsDuringInfusion(object, equations)
```

```r
## S4 method for signature 'Model'
setEquationsDuringInfusion(object, equations)
```

### Arguments

- **object**: An object from the class `Model`.
- **equations**: A list giving the equations during the infusion.

### Value

The model with the updated equations during the infusion.
**setEvaluationFIMResults**

*Set the evaluation results.*

**Description**

Set the evaluation results.

**Usage**

```
setEvaluationFIMResults(object, value)
```

```
# S4 method for signature 'Optimization'
setEvaluationFIMResults(object, value)
```

**Arguments**

- `object`  An object from the class `Optimization`.
- `value`   An object from the class `Evaluation` giving the evaluation results.

**Value**

The object with the updated object from the class `Evaluation`.

---

**setEvaluationInitialDesignResults**

*Set the evaluation results of the initial design.*

**Description**

Set the evaluation results of the initial design.

**Usage**

```
setEvaluationInitialDesignResults(object, value)
```

```
# S4 method for signature 'Optimization'
setEvaluationInitialDesignResults(object, value)
```

**Arguments**

- `object`  An object from the class `Optimization`.
- `value`   An object from the class `Evaluation` giving the evaluation results of the initial design.
Value

The object with the updated object from the class Evaluation.

```r
setFim

Description

Set the fim of the design.

Usage

setFim(object, fim)

## S4 method for signature 'Design'
setFim(object, fim)

Arguments

object An object Design from the class Design.

fim An object fim from the class Fim.

Value

An object Design with the fim updated.

setFimTypeToString

Description

Convert the type of the object fim to a string.

Usage

setFimTypeToString(object)

## S4 method for signature 'Fim'
setFimTypeToString(object)

Arguments

object An object from the class Fim.

Value

The type of the object fim convert as a string.
setFisherMatrix  

Set the FIM.

Description

Set the FIM.

Usage

setFisherMatrix(object, value)

## S4 method for signature 'Fim'
setFisherMatrix(object, value)

Arguments

object  
value  

An object from the class Fim.

A matrix giving the FIM.

Value

The object from the class Fim with the FIM updated.

setFixedEffects  

Set the fixed effects.

Description

Set the fixed effects.

Usage

setFixedEffects(object)

## S4 method for signature 'Fim'
setFixedEffects(object)

Arguments

object  

An object from the class Fim.

Value

Update the matrix of the fixed effects.
**setFixedMu**  
*Set the mu as fixed or not.*

**Description**
Set the mu as fixed or not.

**Usage**
```
setFixedMu(object, value)
```

```r
## S4 method for signature 'ModelParameter'
setFixedMu(object, value)
```

**Arguments**
- `object`: An object from the class `ModelParameter`.
- `value`: A Boolean if fixed or not.

**Value**
The model parameter with the mu updated as fixed or not.

---

**setFixedOmega**  
*Set the omega as fixed or not.*

**Description**
Set the omega as fixed or not.

**Usage**
```
setFixedOmega(object, value)
```

```r
## S4 method for signature 'ModelParameter'
setFixedOmega(object, value)
```

**Arguments**
- `object`: An object from the class `ModelParameter`.
- `value`: A Boolean fixed or not.

**Value**
The model parameter with the omega updated as fixed or not.
setInitialConditions

Description
Set the initial conditions of a ode model.

Usage
setInitialConditions(object, initialConditions)

## S4 method for signature 'Arm'
setInitialConditions(object, initialConditions)

## S4 method for signature 'Model'
setInitialConditions(object, initialConditions)

Arguments
- object: An object from the class Model.
- initialConditions: A list giving the initial conditions.

Value
The model with the updated initial conditions.

setIterationAndCriteria

Set the iteration with the convergence criteria.

Description
Set the iteration with the convergence criteria.

Usage
setIterationAndCriteria(object, value)

## S4 method for signature 'OptimizationAlgorithm'
setIterationAndCriteria(object, value)

Arguments
- object: An object from the class OptimizationAlgorithm.
- value: A dataframe giving the iteration with the convergence criteria.
setModelError

Set the model error.

Description

Set the model error.

Usage

setModelError(object, modelError)

## S4 method for signature 'Model'
setModelError(object, modelError)

Arguments

object An object from the class Model.
modelError An object from the class ModelError.

Value

The model with the updated model error.
**setModelFromLibrary**  
*Set a model from the library of model*

**Description**

Set a model from the library of model

**Usage**

```r
setModelFromLibrary(object, modelFromLibrary)
```

## S4 method for signature 'Model'

```r
setModelFromLibrary(object, modelFromLibrary)
```

**Arguments**

- **object**
  - An object from the class `Model`.
- **modelFromLibrary**
  - An object from the class `Model`.

**Value**

The model with the updated model from library of models.

---

**setMu**  
*Set the value of the fixed effect mu of an object.*

**Description**

Set the value of the fixed effect mu of an object.

**Usage**

```r
setMu(object, value)
```

## S4 method for signature 'Distribution'

```r
setMu(object, value)
```

## S4 method for signature 'ModelParameter'

```r
setMu(object, value)
```

**Arguments**

- **object**
  - An object defined form a class of PFIM.
- **value**
  - The value of the fixed effect mu.
### setName

*Set the name of an object.*

**Description**

Set the name of an object.

**Usage**

```r
def.setName(object, name)
```

**Arguments**

- `object` (An object defined from a class of PFIM.)
- `name` (A string giving the name of the object.)

**Value**

The object with the updated name.

### setNumberOfArms

**Description**

Set the number of arms in a design.

**Usage**

```r
def.setNumberOfArms(object, numberOfArms)
```

## R Code

```r
## S4 method for signature 'Arm'
def.setNumberOfArms(object, numberOfArms)

## S4 method for signature 'Design'
def.setNumberOfArms(object, numberOfArms)

## S4 method for signature 'Model'
def.setNumberOfArms(object, numberOfArms)
```
setOdeSolverParameters

Arguments

object An object Design from the class Design.
numberOfArms A numeric numberOfArms giving the new number of arms in the design.

Value

An object Design with the numberOfArms updated.

setOdeSolverParameters

Set the parameters of the ode solver.

Description

Set the parameters of the ode solver.

Usage

setOdeSolverParameters(object, odeSolverParameters)

## S4 method for signature 'Model'
setOdeSolverParameters(object, odeSolverParameters)

Arguments

object An object from the class Model.
odeSolverParameters A list giving the parameters of the ode solver.

Value

The model with the updated parameters of the ode solver.

setOmega

Set the matrix omega of an object.

Description

Set the matrix omega of an object.
Usage

setOmega(object, value)

```r
## S4 method for signature 'Distribution'
setOmega(object, value)

## S4 method for signature 'ModelParameter'
setOmega(object, value)
```

Arguments

- **object**: An object defined from a class of PFIM.
- **value**: The matrix omega.

Value

The object with the updated matrix omega.

---

setOptimalDesign  

*Set the optimal design.*

Description

Set the optimal design.

Usage

setOptimalDesign(object, optimalDesign)

```r
## S4 method for signature 'OptimizationAlgorithm'
setOptimalDesign(object, optimalDesign)
```

Arguments

- **object**: An object from the class `OptimizationAlgorithm`.
- **optimalDesign**: An object from the class `Design`.

Value

The object with the updated optimal design.
setOptimalWeights

Set the optimal weights.

Description
Set the optimal weights.

Usage
setOptimalWeights(object, optimalWeights)

## S4 method for signature 'MultiplicativeAlgorithm'
setOptimalWeights(object, optimalWeights)

Arguments
- object: An object from the class `MultiplicativeAlgorithm`.
- optimalWeights: A vector giving the optimal weights.

Value
The object with the updated optimal weights.

setOptimizationResults

Set the optimization results.

Description
Set the optimization results.

Usage
setOptimizationResults(object, value)

## S4 method for signature 'Optimization'
setOptimizationResults(object, value)

Arguments
- object: An object from the class `Optimization`.
- value: An object from the class `OptimizationAlgorithm` giving the optimization results.

Value
The object with the updated object from the class `OptimizationAlgorithm`. 
**setOutcome**

Set the outcome of an object.

**Usage**

```
setOutcome(object, outcome)
```

```
## S4 method for signature 'Administration'
setOutcome(object, outcome)
```

```
## S4 method for signature 'SamplingTimes'
setOutcome(object, outcome)
```

**Arguments**

- **object**: An object defined form a class of PFIM.
- **outcome**: A string defined the outcome.

**Value**

A string giving the updated outcome of the object.

---

**setOutcomes**

Set the outcomes of a model.

**Description**

Set the outcomes of a model.

**Usage**

```
setOutcomes(object, outcomes)
```

```
## S4 method for signature 'Model'
setOutcomes(object, outcomes)
```

**Arguments**

- **object**: An object from the class `Model`.
- **outcomes**: A list giving the outcomes of the model.
setOutcomesEvaluation

Value

The model with the updated outcomes.

Description

Set the results of the evaluation of the outcomes.

Usage

setOutcomesEvaluation(object, outcomesEvaluation)

## S4 method for signature 'Design'
setOutcomesEvaluation(object, outcomesEvaluation)

Arguments

- **object**: An object Design from the class Design.
- **outcomesEvaluation**: A list containing the evaluation of the outcomes.

Value

An object Design with the list outcomesEvaluation updated.

setOutcomesForEvaluation

Description

Set the outcomes of a model used for the evaluation (is scales outcomes).

Usage

setOutcomesForEvaluation(object, outcomes)

## S4 method for signature 'Model'
setOutcomesForEvaluation(object, outcomes)
Arguments

object  An object from the class Model.
outcomes  A list giving the outcomes of a model used for the evaluation (is scales outcomes).

Value

The model with the updated outcomes for the evaluation.

Description

Set the results of the evaluation of the outcomes.

Usage

setOutcomesGradient(object, outcomesGradient)

# S4 method for signature 'Design'
setOutcomesGradient(object, outcomesGradient)

Arguments

object  An object Design from the class Design.
outcomesGradient  A list containing the evaluation of the outcome gradients.

Value

An object Design with the list outcomesGradient updated.

Description

Set the parameters of an object.
setSamplingConstraintForOptimization

Usage

setParameters(object, parameters)

## S4 method for signature 'Distribution'
setParameters(object, parameters)

## S4 method for signature 'Model'
setParameters(object, parameters)

## S4 method for signature 'FedorovWynnAlgorithm'
setParameters(object, parameters)

## S4 method for signature 'MultiplicativeAlgorithm'
setParameters(object, parameters)

## S4 method for signature 'PGBOAlgorithm'
setParameters(object, parameters)

## S4 method for signature 'PSOAlgorithm'
setParameters(object, parameters)

## S4 method for signature 'SimplexAlgorithm'
setParameters(object, parameters)

Arguments

object An object defined from a class of PFIM.
parameters A list of parameters.

Value

The object with the updated list of parameters.

Description

Set the sampling times constraint for optimization with PSO, PGBO and Simplex

Usage

setSamplingConstraintForOptimization(object)

## S4 method for signature 'Design'
setSamplingConstraintForOptimization(object)
**Arguments**

object An object from the class Design.

**Value**

The arms with the sampling times constraints.

---

**Description**

Set the sampling times.

**Usage**

```
setSamplings(object, samplings)
```

## S4 method for signature 'SamplingTimes'

```
setSamplings(object, samplings)
```

**Arguments**

object An object from the class SamplingTimes.

samplings A vector giving the sampling times.

**Value**

The updated sampling times.

---

**Description**

Set the sampling time of an arm.

**Usage**

```
setSamplingTime(object, samplingTime)
```

## S4 method for signature 'Arm'

```
setSamplingTime(object, samplingTime)
```
**setSamplingTimes**

**Arguments**

- **object**: An object of type `Arm` from the class `Arm`.
- **samplingTime**: An object of type `SamplingTimes` from the class `SamplingTimes`.

**Value**

An object of type `Arm` from the class `Arm` with the new sampling time `samplingTime`.

---

**setSamplingTimesConstraints**

**Description**

Set the sampling times constraints.

**Usage**

```
setSamplingTimesConstraints(object, samplingTimesConstraints)
```

**Arguments**

- **object**: An object of type `Arm` from the class `Arm`.
- **samplingTimesConstraints**: The list containing the new sampling times.

**Value**

An object of type `Arm` from the class `Arm` with the new sampling times `samplingTimes`.

---

**setSamplingTimesConstraints**

**Description**

Set the sampling times constraints.

**Usage**

```
setSamplingTimesConstraints(object, samplingTimesConstraints)
```

**Arguments**

- **object**: An object of type `Arm` from the class `Arm`.
- **samplingTimesConstraints**: The list containing the new sampling times.

**Value**

An object of type `Arm` from the class `Arm` with the new sampling times `samplingTimes`.

---

**setSamplingTimes**

**Description**

Set the vectors of sampling times for an arm.

**Usage**

```
setSamplingTimes(object, samplingTimes)
```

## S4 method for signature 'Arm'

```
setSamplingTimes(object, samplingTimes)
```

**Arguments**

- **object**: An object of type `Arm` from the class `Arm`.
- **samplingTimes**: The list containing the new sampling times.

**Value**

An object of type `Arm` from the class `Arm` with the new sampling times `samplingTimes`.
Arguments

- **object**: An object of class `Arm`.
- **samplingTimesConstraints**: An object of class `SamplingTimeConstraints`.

Value

The arm with the new sampling time constraints.

---

**setShrinkage**

*Set the shrinkage.*

Description

Set the shrinkage.

Usage

```
setShrinkage(object, value)

## S4 method for signature 'BayesianFim'
setShrinkage(object, value)

## S4 method for signature 'IndividualFim'
setShrinkage(object, value)

## S4 method for signature 'PopulationFim'
setShrinkage(object, value)
```

Arguments

- **object**: An object from the class `Fim`.
- **value**: A vector giving the shrinkage of the Bayesian fim.

Value

The object with the updated shrinkage.
setSigmaInter

Set the parameter sigma inter.

Description
Set the parameter sigma inter.

Usage
setSigmaInter(object, sigmaInter)

## S4 method for signature 'ModelError'
setSigmaInter(object, sigmaInter)

Arguments

- **object**: An object from the class `ModelError`.
- **sigmaInter**: A numeric giving the parameter sigma inter.

Value
The model error with the updated sigma inter.

setSigmaSlope

Set the parameter sigma slope.

Description
Set the parameter sigma slope.

Usage
setSigmaSlope(object, sigmaSlope)

## S4 method for signature 'ModelError'
setSigmaSlope(object, sigmaSlope)

Arguments

- **object**: An object from the class `ModelError`.
- **sigmaSlope**: A numeric giving the parameter sigma slope.

Value
The model error with the updated sigma slope.
### setSize

**Description**
Set the size of an object.
Set the size of an arm.

**Usage**

```r
setSize(object, size)
```

```r
## S4 method for signature 'Arm'
setSize(object, size)
```

```r
## S4 method for signature 'Design'
setSize(object, size)
```

**Arguments**

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

**Value**

The object with its size updated.
The object Arm object with its new size.

### setTau

**Description**
Set the frequency $\tau$.

**Usage**

```r
setTau(object, tau)
```

```r
## S4 method for signature 'Administration'
setTau(object, tau)
```
**setTimeDose**

**Arguments**

- **object**: An object of class `Administration`.
- **tau**: A numeric value for the infusion lag tau.

**Value**

The object `Administration` object with its new value of the infusion lag tau.

---

**setDescription**

**Description**

Set the times vector when doses are given.

**Usage**

```r
setTimeDose(object, timeDose)
```

## S4 method for signature 'Administration'

```r
setTimeDose(object, timeDose)
```

**Arguments**

- **object**: An object of class `Administration`.
- **timeDose**: 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**

```r
setTinf(object, Tinf)
```

## S4 method for signature 'Administration'

```r
setTinf(object, Tinf)
```

**Arguments**

- **object**: An object of class `Administration`.
- **Tinf**: A numeric value of the time dose.

**Value**

The object `Administration` with its new infusion duration.
Arguments

object       An object Administration from the class Administration.
Tinf         A numeric value for the infusion duration Tinf.

Value

The object Administration with its new value of the infusion duration Tinf.

setVarianceEffects

Set the matrix of the variance effects.

Description

Set the matrix of the variance effects.

Usage

setVarianceEffects(object)

## S4 method for signature 'Fim'
setVarianceEffects(object)

Arguments

object       An object from the class Fim.

Value

Update the matrix of the variance effects.

show,Design-method

show
Usage

## S4 method for signature 'Design'
show(object)

## S4 method for signature 'Evaluation'
show(object)

## S4 method for signature 'FedorovWynnAlgorithm'
show(object)

## S4 method for signature 'MultiplicativeAlgorithm'
show(object)

## S4 method for signature 'Optimization'
show(object)

## S4 method for signature 'PGBOAlgorithm'
show(object)

## S4 method for signature 'PSOAlgorithm'
show(object)

## S4 method for signature 'SimplexAlgorithm'
show(object)

Arguments

object object

Description

Class "SimplexAlgorithm" implements the Multiplicative algorithm.

Objects from the class SimplexAlgorithm

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

Slots for SamplingTimes objects

pctInitialSimplexBuilding: A numeric giving the percentage of the initial simplex.
maxIteration: A numeric giving the number of maximum iteration.
tolerance: A numeric giving the tolerance threshold.
showProcess: A boolean to show or not the process.
optimalDesign: A Design object giving the optimal design.
iterationAndCriteria: A list giving the optimal criteria at each iteration.
Index

_PACKAGE (PFIM-package), 7

addModel, 13, 20
addModel, LibraryOfModels-method (addModel), 20
addModels, 13, 20
addModels, LibraryOfModels-method (addModels), 20
Administration, 9, 55, 91, 124, 147, 148
Administration (Administration-class), 21
Administration-class, 21
AdministrationConstraints, 9
AdministrationConstraints (AdministrationConstraints-class), 21
AdministrationConstraints-class, 21
Arm, 9, 23, 30, 32–35, 44–46, 63, 86, 87, 120, 143, 144, 146
Arm (Arm-class), 22
Arm-class, 22
BayesianFim, 10
BayesianFim (BayesianFim-class), 22
BayesianFim-class, 22
checkSamplingTimeConstraintsForContinuousOptimization (checkSamplingTimeConstraintsForContinuousOptimization), 18, 22
checkSamplingTimeConstraintsForContinuousOptimization, SampleTimeConstraints-method (checkSamplingTimeConstraintsForContinuousOptimization), 22
checkValiditySamplingConstraint, 23
checkValiditySamplingConstraint, Design-method (checkValiditySamplingConstraint), 23
Combined1, 10, 112
Combined1 (Combined1-class), 24
Combined1-class, 24
Constant, 10
Constant (Constant-class), 24
Constant-class, 24
convertPKModelAnalyticToPKModelODE, 14, 25
convertPKModelAnalyticToPKModelODE, ModelAnalytic-method (convertPKModelAnalyticToPKModelODE), 25
convertPKModelAnalyticToPKModelODE, ModelAnalyticInfusion-method (convertPKModelAnalyticToPKModelODE), 25
convertPKModelAnalyticToPKModelODE, ModelAnalyticSteadyState-method (convertPKModelAnalyticToPKModelODE), 25
defineModel, 14, 25
defineModel, Model-method (defineModel), 25
defineModelFromLibraryOfModels, 14, 26
defineModelFromLibraryOfModels, Model-method (defineModelFromLibraryOfModels), 26
defineModelType, 14, 26
defineModelType, Model-method (defineModelType), 26
defineModelUserDefined, 14, 27
defineModelUserDefined, Model-method (defineModelUserDefined), 27
definePKModel, 14–16, 27
definePKModel, ModelAnalytic-method (definePKModel), 14–16, 27
definePKModel, ModelAnalyticInfusion-method (definePKModel), 27
definePKModel, ModelAnalyticSteadyState-method (definePKModel), 27
definePKModel, ModelODE-method (definePKModel), 27
definePKModel, ModelODEDoseInEquations-method (definePKModel), 27
definePKModel, ModelODEInfusionDoseInEquations-method (definePKModel), 27
definePKPDModel, 14–16, 28
definePKPDModel, ModelAnalytic, ModelAnalytic-method
(EvaluateModel), 33
definePKPDModel, ModelAnalyticInfusion, ModelODE-method
(EvaluateModel), 33
definePKPDModel, ModelAnalyticInfusionSteadyState-method
(EvaluateModel), 33
definePKPDModel, ModelAnalyticSteadyState-method
(EvaluateModel), 33
definePKPDModel, ModelODEBolus-method
(EvaluateModel), 33
definePKPDModel, ModelODEDoseInEquations-method
(EvaluateModel), 33
definePKPDModel, ModelODEDoseNotInEquations-method
(EvaluateModel), 33
definePKPDModel, ModelODEInfusionDoseInEquations-method
(EvaluateModel), 33
definePKPDModel, ModelODEInfusionDoseNotInEquations-method
(EvaluateModel), 33
evaluateFisherMatrix, Model-method
(evaluateFisherMatrix), 34
FedorovWynnAlgorithm-class
(FedorovWynnAlgorithm), 36
FedorovWynnAlgorithm-class
(FedorovWynnAlgorithm-class), 36
FedorovWynnAlgorithm.Rcpp-class
(FedorovWynnAlgorithm.Rcpp), 36
Fim-class
(Fim), 37
fisher.simplex
(fisher.simplex), 38
fun.amoeba
(fun.amoeba), 38
generateFimsFromConstraints
(generateFimsFromConstraints), 39
generateReportEvaluation
(generateReportEvaluation), 39
generateReportEvaluation, BayesianFim-method
  (generateReportEvaluation), 39
getAdministrationsConstraints, 9, 45
getAdministrationsConstraints, Arm-method
  (getAdministrationsConstraints), 45
getArms, Design-method (getArms), 46
getArms, OptimizationAlgorithm-method
  (getArms), 46
getColumnAndParametersNamesFIM, IndividualFim-method
  (getColumnAndParametersNamesFIM), 47
getColumnAndParametersNamesFIM, PopulationFim-method
  (getColumnAndParametersNamesFIM), 47
getContent, LibraryOfModels-method
  (getContent), 49
getCorrelationMatrix, 12, 50
getFixedParameters, Model-method
  (getFixedParameters), 62
getFixedTimes, 18, 62
getFixedTimes, SamplingTimeConstraints-method
  (getFixedTimes), 62
getInitialConditions, 9, 14, 63
getInitialConditions, Arm-method
  (getInitialConditions), 63
getInitialConditions, Model-method
  (getInitialConditions), 63
getIterationAndCriteria, 63
getIterationAndCriteria, OptimizationAlgorithm-method
  (getIterationAndCriteria), 63
getLambda, 16, 64
getLambda, MultiplicativeAlgorithm-method
  (getLambda), 64
getLibraryPDModes, 13, 64
getLibraryPDModes, LibraryOfModels-method
  (getLibraryPDModes), 64
getLibraryPKModels, 13, 65
getLibraryPKModels, LibraryOfModels-method
  (getLibraryPKModels), 65
getMinSampling, 18, 65
getMinSampling, SamplingTimeConstraints-method
  (getMinSampling), 65
getModel, 17, 66
getModel, PFIMProject-method (getModel), 66
getModelEquations, 17, 66
getModelEquations, PFIMProject-method
  (getModelEquations), 66
getModelError, 12, 13, 17, 67
getModelError, Model-method
  (getModelError), 67
getModelError, PFIMProject-method
  (getModelError), 67
getModelErrorParametersValues, 14, 67
getModelErrorParametersValues, Model-method
  (getModelErrorParametersValues), 67
getModelFromLibrary, 14, 68
getModelFromLibrary, Model-method
  (getModelFromLibrary), 68
getModelParameters, 17, 68
getModelParameters, PFIMProject-method
  (getModelParameters), 68
getModelParametersValues, 69
getModelParametersValues, Model-method
  (getModelParametersValues), 69
getMu, 11, 12, 16, 69
getMu, Distribution-method (getMu), 69
getMu, ModelParameter-method (getMu), 69
getName, 9, 10, 12, 13, 16, 17, 70
getName, Arm-method (getName), 70
getName, Design-method (getName), 70
getName, LibraryOfModels-method
  (getName), 70
getName, Model-method (getName), 70
getName, ModelParameter-method
  (getName), 70
getNumberOfArms, 10, 71
getNumberOfArms, Design-method
  (getNumberOfArms), 71
getNumberOfIterations, 16, 72
getNumberOfIterations, MultiplicativeAlgorithm-method
  (getNumberOfIterations), 72
getNumberOfParameters, 14, 72
getNumberOfParameters, Model-method
  (getNumberOfParameters), 72
getNumberOfSamplesOptimisable, 18, 73
getNumberOfSamplesOptimisable, SamplingTimeConstraints-method
  (getNumberOfSamplesOptimisable), 73
getNumberOfTimesByWindows, 18, 73
getNumberOfTimesByWindows, SamplingTimeConstraints-method
  (getNumberOfTimesByWindows), 73
getOdeSolverParameters, 12, 14, 17, 74
getOdeSolverParameters, Model-method
  (getOdeSolverParameters), 74
getOdeSolverParameters, PFIMProject-method
  (getOdeSolverParameters), 74
getOmega, 11, 12, 16, 74
getOmega, Distribution-method
  (getOmega), 74
getOmega, ModelParameter-method
  (getOmega), 74
getOptimalDesign, 75
getOptimalDesign, OptimizationAlgorithm-method
  (getOptimalDesign), 75
getOptimalFrequencies, 75
getOptimalFrequencies, FedorovWynnAlgorithm-method
  (getOptimalFrequencies), 75
getOptimalWeights, 16, 76
getOptimalWeights, MultiplicativeAlgorithm-method  (getOptimalWeights), 76
getOptimizationResults, 17, 76
getOptimizationResults, Optimization-method  (getOptimizationResults), 76
getOptimizer, 17, 77
getOptimizer, PFIMProject-method  (getOptimizer), 77
getOptimizerParameters, 17, 77
getOptimizerParameters, PFIMProject-method  (getOptimizerParameters), 77
getOutcome, 9, 12, 15, 18, 78
getOutcome, Administration-method  (getOutcome), 78
getOutcome, AdministrationConstraints-method  (getOutcome), 78
getOutcome, ModelError-method  (getOutcome), 78
getOutcome, SamplingTimeConstraints-method  (getOutcome), 78
getOutcome, SamplingTimes-method  (getOutcome), 78
getOutcomes, 13, 17, 78
getOutcomes, Model-method  (getOutcomes), 78
getOutcomesEvaluation, 10, 79
getOutcomesEvaluation, Design-method  (getOutcomesEvaluation), 79
getOutcomesForEvaluation, 13, 80
getOutcomesForEvaluation, Model-method  (getOutcomesForEvaluation), 80
getOutcomesGradient, 10, 80
getOutcomesGradient, Design-method  (getOutcomesGradient), 80
getParameters, 10, 12, 13, 15, 81
getParameters, Distribution-method  (getParameters), 81
getParameters, Model-method  (getParameters), 81
getParameters, ModelError-method  (getParameters), 81
getPDModel, 13, 81
getPDModel, LibraryOfPKPDModes-method  (getPDModel), 81
getPKModel, 13, 82
getPKModel, LibraryOfPKPDModes-method  (getPKModel), 82
getPKPDModel, 13, 82
getPKPDModel, LibraryOfPKPDModes-method  (getPKPDModel), 82
getPlotOptions, 83
getProportionsOfSubjects, 17, 83
getProportionsOfSubjects, Optimization-method  (getProportionsOfSubjects), 83
gertRSE, 10, 12, 13, 18, 84
gertRSE, BayesianFim-method  (getRSE), 84
gertRSE, Evaluation-method  (getRSE), 84
gertRSE, IndividualFim-method  (getRSE), 84
gertRSE, Optimization-method  (getRSE), 84
gertRSE, PopulationFim-method  (getRSE), 84
getsamplings, 12, 18, 84
getsamplings, SamplingTimeConstraints-method  (getsamplings), 84
getsamplings, SamplingTimes-method  (getsamplings), 84
getsamplingsWindows, 18, 85
getsamplingsWindows, SamplingTimeConstraints-method  (getsamplingsWindows), 85
getsamplingtime, 9, 86
getsamplingtime, Arm-method  (getsamplingtime), 86
getsamplingtimeConstraint, 9, 86
getsamplingtimeConstraint, Arm-method  (getsamplingtimeConstraint), 86
getsamplingtimes, 9, 87
getsamplingtimes, Arm-method  (getsamplingtimes), 87
getsamplingtimesConstraints, 9, 87
getsamplingtimesConstraints, Arm-method  (getsamplingtimesConstraints), 87
gese, 12, 88
gese, Evaluation-method  (gese), 88
gese, Fim-method  (gese), 88
gese, Optimization-method  (gese), 88
geshrinkage, 10, 12, 13, 18, 88
geshrinkage, BayesianFim-method  (geshrinkage), 88
geshrinkage, Evaluation-method  (geshrinkage), 88
geshrinkage, IndividualFim-method  (geshrinkage), 88
geshrinkage, Optimization-method  (geshrinkage), 88
getShrinkage, PopulationFim-method
(getShrinkage), 88
getSigmaInter, 15, 89
getSigmaInter, ModelError-method
(getSigmaInter), 89
getSigmaSlope, 15, 90
getSigmaSlope, ModelError-method
(getSigmaSlope), 90
getSize, 9, 10, 12, 90
getSize, Arm-method (getSize), 90
getSize, Design-method (getSize), 90
getTau, 9, 91
getTau, Administration-method (getTau), 91
timeDose, 9, 91
timeDose, Administration-method
(timeDose), 91
tInf, 9, 92
tInf, Administration-method
(tInf), 92
getVarianceEffects, 11, 92
gammaVarianceEffects, Fim-method
(getVarianceEffects), 92
IndividualFim, 12
IndividualFim (IndividualFim-class), 93
IndividualFim-class, 93
isDoseInEquations, 14, 93
isDoseInEquations, Model-method
(isDoseInEquations), 93
isModelAnalytic, 14, 93
isModelAnalytic, Model-method
(isModelAnalytic), 93
isModelBolus, 14, 94
isModelBolus, Model-method
(isModelBolus), 94
isModelInfusion, 14, 94
isModelInfusion, Model-method
(isModelInfusion), 94
isModelODE, 14, 95
isModelODE, Model-method (isModelODE), 95
isModelSteadyState, 14, 95
isModelSteadyState, Model-method
(isModelSteadyState), 95
LibraryOfModels, 13, 20, 50, 64, 65, 122
LibraryOfModels (LibraryOfModels-class), 96
LibraryOfModels-class, 96

LibraryOfPDModels, 96
LibraryOfPKModels, 96
LibraryOfPKPDModels, 13, 81, 82
LibraryOfPKPDModels
(LibraryOfPKPDModels-class), 97
LogNormal, 13
LogNormal (LogNormal-class), 97
LogNormal-class, 97
Model, 13, 15, 20, 25–35, 47, 48, 53, 57, 58, 62, 67–69, 72, 79, 80, 84, 93–95,
105, 108, 109, 116, 123, 125, 126, 131–133, 135, 138, 140
Model (Model-class), 97
Model-class, 97
ModelAnalytic, 14
ModelAnalytic (ModelAnalytic-class), 98
ModelAnalytic-class, 98
ModelAnalyticBolus, 14, 15
ModelAnalyticBolus
(ModelAnalyticBolus-class), 98
ModelAnalyticBolus-class, 98
ModelAnalyticBolusSteadyState, 14
ModelAnalyticBolusSteadyState
(ModelAnalyticBolusSteadyState-class), 98
ModelAnalyticBolusSteadyState-class, 98
ModelAnalyticInfusion
(ModelAnalyticInfusion-class), 98
ModelAnalyticInfusion-class, 98
ModelAnalyticInfusionSteadyState
(ModelAnalyticInfusionSteadyState-class), 99
ModelAnalyticInfusionSteadyState-class, 99
ModelAnalyticSteadyState
(ModelAnalyticSteadyState-class), 99
ModelAnalyticSteadyState-class, 99
ModelBolus, 15
ModelBolus (ModelBolus-class), 99
ModelBolus-class, 99
ModelError, 10, 15, 18, 24, 31, 47, 52, 56, 89,
90, 121, 122, 125, 132, 145
ModelError (ModelError-class), 99
ModelError-class, 99
ModelInfusion, 15, 16
ModelInfusion (ModelInfusion-class), 99
ModelInfusion-class, 99
ModelODE, 15
ModelODE (ModelODE-class), 100
ModelODE-class, 100
ModelODEBolus, 15
ModelODEBolus (ModelODEBolus-class), 100
ModelODEBolus-class, 100
ModelODEDoseInEquations, 15
ModelODEDoseInEquations
(ModelODEDoseInEquations-class), 100
ModelODEDoseInEquations-class, 100
ModelODEDoseNotInEquations, 15
ModelODEDoseNotInEquations
(ModelODEDoseNotInEquations-class), 100
ModelODEDoseNotInEquations-class, 100
ModelODEInfusion, 16
ModelODEInfusion
(ModelODEInfusion-class), 100
ModelODEInfusion-class, 100
ModelODEInfusionDoseInEquations, 16
ModelODEInfusionDoseInEquations
(ModelODEInfusionDoseInEquations-class), 101
ModelODEInfusionDoseInEquations-class, 101
ModelParameter, 16, 54, 61, 124, 130
ModelParameter (ModelParameter-class), 101
ModelParameter-class, 101
MultiplicativeAlgorithm, 16, 52, 64, 72, 76, 137
MultiplicativeAlgorithm
(MultiplicativeAlgorithm-class), 101
MultiplicativeAlgorithm-class, 101
MultiplicativeAlgorithm_Rcpp, 16, 102
Normal, 16
Normal (Normal-class), 103
Normal-class, 103
Optimization, 17, 38, 39, 42, 56, 58, 59, 76, 83, 123, 127, 137
Optimization (Optimization-class), 103
Optimization-class, 103
OptimizationAlgorithm, 42, 51, 63, 75, 76, 105, 108, 111, 131, 136, 137
OptimizationAlgorithm
(OptimizationAlgorithm-class), 104
OptimizationAlgorithm-class, 104
optimize, 11, 16–19, 104
optimize, FedorovWynnAlgorithm-method
(optimize), 104
optimize, MultiplicativeAlgorithm-method
(optimize), 104
optimize, PGBOAlgorithm-method
(optimize), 104
optimize, PSOAlgorithm-method
(optimize), 104
optimize, SimplexAlgorithm-method
(optimize), 104
package-PFIM (PFIM-package), 7
parametersForComputingGradient, 14, 105
parametersForComputingGradient, Model-method
(parametersForComputingGradient), 105
PFIM, (PFIM-package), 7
PFIM-package, 7
PFIMProject, 17, 43, 53, 66, 68, 77, 114, 118, 132
PFIMProject (PFIMProject-class), 105
PFIMProject-class, 105
PGBOAlgorithm, 17
PGBOAlgorithm (PGBOAlgorithm-class), 106
PGBOAlgorithm-class, 106
plot, 18
PlotEvaluation, 17
PlotEvaluation (PlotEvaluation-class), 107
plotEvaluation, 107
plotEvaluation, Evaluation-method
(plotEvaluation), 107
PlotEvaluation-class, 107
plotFrequencies, 107
plotFrequencies, FedorovWynnAlgorithm-method
(plotFrequencies), 107
plotFrequencies, Optimization-method
(plotFrequencies), 107
plotOutcomesEvaluation, 10, 108
plotOutcomesEvaluation, Design-method
(plotOutcomesEvaluation), 108
plotOutcomesGradient, 10, 109
plotOutcomesGradient, Design-method
(plotOutcomesGradient), 109
plotRSE, 18, 109
plotRSE, PFIMProject-method (plotRSE), 109
plotSE, 18, 110
plotSE, PFIMProject-method (plotSE), 110
plotSensitivityIndice, 110
plotSensitivityIndice, Evaluation-method (plotSensitivityIndice), 110
plotShrinkage, 18, 111
plotShrinkage, PFIMProject-method (plotShrinkage), 111
plotWeights, 16, 17, 111
plotWeights, MultiplicativeAlgorithm-method (plotWeights), 111
plotWeights, Optimization-method (plotWeights), 111
PopulationFim, 18
PopulationFim (PopulationFim-class), 112
PopulationFim-class, 112
Proportional, 18, 112
Proportional (Proportional-class), 112
Proportional-class, 112
PSOAlgorithm, 18
PSOAlgorithm (PSOAlgorithm-class), 113
PSOAlgorithm-class, 113
Report, 11, 17, 113
Report, Evaluation-method (Report), 113
Report, Optimization-method (Report), 113
reportTablesAdministration, 10, 114
reportTablesAdministration, Design-method (reportTablesAdministration), 114
reportTablesDesign, 10, 114
reportTablesDesign, Design-method (reportTablesDesign), 114
reportTablesFIM, 10, 12, 13, 18, 115
reportTablesFIM, BayesianFim-method (reportTablesFIM), 115
reportTablesFIM, IndividualFim-method (reportTablesFIM), 115
reportTablesFIM, PopulationFim-method (reportTablesFIM), 115
reportTablesModelError, 14, 116
reportTablesModelError, Model-method (reportTablesModelError), 116
reportTablesModelParameters, 14, 116
reportTablesModelParameters, Model-method (reportTablesModelParameters), 116
reportTablesPlot, 11, 117
reportTablesPlot, Evaluation-method (reportTablesPlot), 117
reportTablesSamplingConstraints, 117
reportTablesSamplingConstraints, Design-method (reportTablesSamplingConstraints), 117
resizeFisherMatrix, 11, 118
resizeFisherMatrix, ANY-method (resizeFisherMatrix), 118
run, 11, 17, 118
run, Evaluation-method (run), 118
run, Optimization-method (run), 118
SamplingTimeConstraints, 18, 23, 42, 62, 65, 73, 85, 144
SamplingTimeConstraints (SamplingTimeConstraints-class), 119
SamplingTimeConstraints-class, 119
SamplingTimes, 18, 142, 143
SamplingTimes (SamplingTimes-class), 119
SamplingTimes-class, 119
setAdministrations, 9, 120
setAdministrations, Arm-method (setAdministrations), 120
setArm, 10, 120
setArm, Design-method (setArm), 120
setArms, 10, 12, 121
setArms, Design-method (setArms), 121
setArms, OptimizationAlgorithm-method (setArms), 121
setError, 15, 121
setError, ModelError-method (setError), 121
setContent, 13, 122
setContent, LibraryOfModels-method (setContent), 122
setDerivatives, 15, 122
setDerivatives, ModelError-method (setDerivatives), 122
setDescription, 13, 123
setDescription, Model-method (setDescription), 123
setDesigns, 123
setDesigns, Optimization-method
(setDesigns), 123
setDistribution, 16, 124
setDistribution, ModelParameter-method
(setDistribution), 124
setDose, 9, 124
setDose, Administration-method
(setDose), 124
setEquation, 15, 125
setEquation, ModelError-method
(setEquation), 125
setEquations, 13, 125
setEquations, Model-method
(setEquations), 125
setEquationsAfterInfusion, 15, 126
setEquationsAfterInfusion, Model-method
(setEquationsAfterInfusion), 126
setEquationsDuringInfusion, 15, 126
setEquationsDuringInfusion, Model-method
(setEquationsDuringInfusion), 126
setEvaluationFIMResults, 17, 127
setEvaluationFIMResults, Optimization-method
(setEvaluationFIMResults), 127
setEvaluationInitialDesignResults, 17, 127
setEvaluationInitialDesignResults, Optimization-method
(setEvaluationInitialDesignResults), 127
setFim, 10, 128
setFim, Design-method (setFim), 128
setFimTypeToString, 12, 128
setFimTypeToString, Fim-method
(setFimTypeToString), 128
setFisherMatrix, 11, 129
setFisherMatrix, Fim-method
(setFisherMatrix), 129
setFixedEffects, 11, 129
setFixedEffects, Fim-method
(setFixedEffects), 129
setFixedMu, 16, 130
setFixedMu, ModelParameter-method
(setFixedMu), 130
setFixedOmega, 16, 130
setFixedOmega, ModelParameter-method
(setFixedOmega), 130
setInitialConditions, 9, 14, 131
setInitialConditions, Arm-method
(setInitialConditions), 131
setInitialConditions, Model-method
(setInitialConditions), 131
setIterationAndCriteria, 131
setIterationAndCriteria, OptimizationAlgorithm-method
(setIterationAndCriteria), 131
setModel, 17, 132
setModel, PFIMProject-method (setModel), 132
setModelError, 14, 132
setModelError, Model-method
(setModelError), 132
setModelFromLibrary, 13, 133
setModelFromLibrary, Model-method
(setModelFromLibrary), 133
setMu, 11, 12, 16, 133
setMu, Distribution-method (setMu), 133
setMu, ModelParameter-method (setMu), 133
setName, 9, 10, 12, 13, 134
setName, Arm-method (setName), 134
setName, Design-method (setName), 134
setName, Model-method (setName), 134
setNumberOfArms, 10, 134
setNumberOfArms, Design-method
(setNumberOfArms), 134
setOdeSolverParameters, 14, 135
setOdeSolverParameters, Model-method
(setOdeSolverParameters), 135
setOmega, 11, 12, 16, 135
setOmega, Distribution-method
(setOmega), 135
setOmega, ModelParameter-method
(setOmega), 135
setOptimalDesign, 136
setOptimalDesign, OptimizationAlgorithm-method
(setOptimalDesign), 136
setOptimalWeights, 16, 137
setOptimalWeights, MultiplicativeAlgorithm-method
(setOptimalWeights), 137
setOptimizationResults, 17, 137
setOptimizationResults, Optimization-method
(setOptimizationResults), 137
setOutcome, 9, 12, 18, 138
setOutcome, Administration-method
(setOutcome), 138
setOutcome, SamplingTimes-method
(setOutcome), 138
setOutcomes, 13, 18
setOutcomes, Model-method (setOutcomes), 13
setOutcomesEvaluation, 10, 139
setOutcomesEvaluation, Design-method (setOutcomesEvaluation), 139
setOutcomesForEvaluation, 13, 139
setOutcomesForEvaluation, Model-method (setOutcomesForEvaluation), 139
setOutcomesGradient, 10, 140
setOutcomesGradient, Design-method (setOutcomesGradient), 140
setParameters, 11–13, 16–19, 140
setParameters, Distribution-method (setParameters), 140
setParameters, FedorovWynnAlgorithm-method (setParameters), 140
setParameters, Model-method (setParameters), 140
setParameters, MultiplicativeAlgorithm-method (setParameters), 140
setParameters, PGBOAlgorithm-method (setParameters), 140
setParameters, PSOAlgorithm-method (setParameters), 140
setParameters, SimplexAlgorithm-method (setParameters), 140
setSamplingConstraintForOptimization, 141
setSamplingConstraintForOptimization, Design-method (setSamplingConstraintForOptimization), 141
setSamplingConstraintForOptimization, SamplingTimes-method (setSamplingConstraintForOptimization), 141
setSamplingConstraintForOptimization, SamplingTimes-method (setSamplingConstraintForOptimization), 141
setSamplingTimes, 18, 142
setSamplingTimes, SamplingTimes-method (setSamplingTimes), 142
setSamplingTime, 9, 142
setSamplingTime, Arm-method (setSamplingTime), 142
setSamplingTimes, 9, 143
setSamplingTimes, Arm-method (setSamplingTimes), 143
setSamplingTimesConstraints, 9, 143
setSamplingTimesConstraints, Arm-method (setSamplingTimesConstraints), 143
setShrinkage, 10, 13, 18, 144
setShrinkage, BayesianFim-method (setShrinkage), 144
setShrinkage, IndividualFim-method (setShrinkage), 144
setShrinkage, PopulationFim-method (setShrinkage), 144
setSigmaInter, 15, 145
setSigmaInter, ModelError-method (setSigmaInter), 145
setSigmaSlope, 15, 145
setSigmaSlope, ModelError-method (setSigmaSlope), 145
setSize, 9, 10, 12, 146
setSize, Arm-method (setSize), 146
setSize, Design-method (setSize), 146
setTau, 9, 146
setTau, Administration-method (setTau), 146
setTimeDose, 9, 147
setTimeDose, Administration-method (setTimeDose), 147
setTinf, 9, 147
setTinf, Administration-method (setTinf), 147
setVarianceEffects, 11, 148
setVarianceEffects, Fim-method (setVarianceEffects), 148
show, Design-method, 148
show, Evaluation-method (show, Design-method), 148
show, FedorovWynnAlgorithm-method (show, Design-method), 148
show, MultiplicativeAlgorithm-method (show, Design-method), 148
show, Optimization-method (show, Design-method), 148
show, PGBOAlgorithm-method (show, Design-method), 148
show, PSOAlgorithm-method (show, Design-method), 148
show, SimplexAlgorithm-method (show, Design-method), 148
SimplexAlgorithm, 19
SimplexAlgorithm (SimplexAlgorithm-class), 149
SimplexAlgorithm (SimplexAlgorithm-class), 149