Package ‘PFIM’

November 24, 2023

Type Package

Title Population Fisher Information Matrix

Version 6.0.2

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>,

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.R'
'ModelODEInfusionDoseInEquations.R' 'MultiplicativeAlgorithm.R'
'Normal.R' 'Optimization.R' 'PFIM-package.R' 'PGBAlgorithm.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  2023-11-24 11:50:02 UTC
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 ............................................................... 51
cGetDerivatives .......................................................... 52
cGetDescription ......................................................... 52
cGetDesigns .............................................................. 53
cGetDeterminant ........................................................ 53
cGetDistribution ........................................................ 54
cGetDose ............................................................... 54
cGetEigenValues ........................................................ 55
cGetElementaryProtocols ............................................. 55
cGetEquation ............................................................ 56
cGetEquations ........................................................... 56
cGetEquationsAfterInfusion ......................................... 57
cGetEquationsDuringInfusion ...................................... 57
cGetEvaluationFIMResults ............................................ 58
cGetEvaluationInitialDesignResults ................................. 58
cGetFim ................................................................. 59
cGetFisherMatrix ........................................................ 59
cGetFixedEffects ........................................................ 60
cGetFixedMu ............................................................. 60
cGetFixedOmega ........................................................ 61
cGetFixedParameters .................................................... 61
R topics documented:

getFixedTimes .............................................. 62
getInitialConditions ........................................ 62
getIterationAndCriteria ...................................... 63
getLambda ....................................................... 63
getLibraryPDModels .......................................... 64
getLibraryPKModels .......................................... 64
getMinSampling ................................................ 65
getModel ......................................................... 65
getModelEquations ........................................... 66
getModelError .................................................. 66
getModelErrorParametersValues .............................. 67
getModelFromLibrary .......................................... 67
getModelParameters .......................................... 68
getModelParametersValues .................................... 68
getMu ........................................................... 69
getName ........................................................ 69
getNames ......................................................... 70
getNumberOfArms ............................................ 71
getNumberOfIterations ....................................... 71
getNumberOfParameters ...................................... 72
getNumberOfSamplingsOptimisable ......................... 72
getNumberOfTimesByWindows ................................ 73
getOdeSolverParameters ..................................... 73
getOmega ......................................................... 74
getOptimalDesign ............................................. 74
getOptimalWeights ............................................ 75
getOptimizationResults ...................................... 75
getOptimizer .................................................... 76
getOptimizerParameters ...................................... 76
getOutcome ...................................................... 77
getOutcomes ................................................... 77
getOutcomesEvaluation ...................................... 78
getOutcomesForEvaluation ................................... 79
getOutcomesGradient ....................................... 79
getParameters .................................................. 80
getPDModel ...................................................... 80
getPKModel ...................................................... 80
getPKPDModel ................................................... 81
getPlotOptions .................................................. 81
getProportionsOfSubjects ................................... 82
getRSE .......................................................... 82
getSamplings .................................................... 83
getSamplingsWindows ........................................ 84
getSamplingTime ............................................... 84
getSamplingTimeConstraint ................................... 85
getSamplingTimes ............................................. 85
getSamplingTimesConstraints ................................ 86
getSE ............................................................ 86
getShrinkage .................................................. 87
getSigmaInter ................................................. 87
getSigmaSlope ................................................ 88
getSize ......................................................... 89
getTau .......................................................... 89
getTimeDose .................................................... 90
getTinf ......................................................... 90
getVarianceEffects ............................................ 91
IndividualFim-class ......................................... 91
isDoseInEquations .......................................... 91
isModelAnalytic .............................................. 92
isModelBolus ................................................ 92
isModelInfusion .............................................. 93
isModelODE ................................................... 93
isModelSteadyState ......................................... 94
LibraryOfModels-class .................................... 94
LibraryOfPDModels ......................................... 95
LibraryOfPKModels ......................................... 95
LibraryOfPKPDModels-class ................................. 95
LogNormal-class ............................................. 95
Model-class .................................................... 96
ModelAnalytic-class ........................................ 96
ModelAnalyticBolus-class .................................. 96
ModelAnalyticBolusSteadyState-class ..................... 97
ModelAnalyticInfusion-class ............................... 97
ModelAnalyticInfusionSteadyState-class .................. 97
ModelAnalyticSteadyState-class ......................... 97
ModelBolus-class ............................................ 98
ModelError-class .............................................. 98
ModelError-class ............................................ 98
ModelInfusion-class ......................................... 98
ModelODE-class .............................................. 98
ModelODEBolus-class ...................................... 98
ModelODEDoseInEquations-class ........................... 99
ModelODEDoseNotInEquations-class ....................... 99
ModelODEInfusion-class ................................... 99
ModelODEInfusionDoseInEquations-class ................ 99
ModelParameter-class ...................................... 100
MultiplicativeAlgorithm-class ............................ 100
MultiplicativeAlgorithm_Rcpp ............................. 101
Normal-class .................................................. 101
Optimization-class .......................................... 102
OptimizationAlgorithm-class ............................... 103
optimize ......................................................... 103
parametersForComputingGradient .......................... 104
PFIMProject-class ........................................... 104
PGBOAlgorithm-class ....................................... 105
plot .................................................................. 106
PlotEvaluation-class ........................................ 106
R topics documented:

plotOutcomesEvaluation ......................................................... 106
plotOutcomesGradient ............................................................. 107
plotRSE .......................................................... 108
plotSE .......................................................... 108
plotShrinkage .......................................................... 109
plotWeights .......................................................... 109
PopulationFim-class ......................................................... 110
Proportional-class ......................................................... 110
PSOAlgorithm-class ......................................................... 111
Report .......................................................... 111
reportTablesAdministration .................................................. 112
reportTablesDesign ......................................................... 112
reportTablesFIM .......................................................... 113
reportTablesModelError ..................................................... 114
reportTablesModelParameters .............................................. 114
reportTablesPlot .......................................................... 115
reportTablesSamplingConstraints ......................................... 115
resizeFisherMatrix ........................................................ 116
run .......................................................... 116
SamplingTimeConstraints-class ............................................. 117
SamplingTimes-class ....................................................... 117
setAdministrations ......................................................... 118
setArm .......................................................... 118
setArms .......................................................... 119
setcError .......................................................... 119
setContent .......................................................... 120
setDerivatives .......................................................... 120
setDescription ......................................................... 121
setDesigns .......................................................... 121
setDistribution .......................................................... 122
setDose .......................................................... 122
setEquation .......................................................... 123
setEquations ......................................................... 123
setEquationsAfterInfusion .............................................. 124
setEquationsDuringInfusion ............................................. 124
setEvaluationFIMResults .................................................. 125
setEvaluationInitialDesignResults ....................................... 125
setFim .......................................................... 126
setFimTypeToString ...................................................... 126
setFisherMatrix ........................................................ 127
setFixedEffects ......................................................... 127
setFixedMu .......................................................... 128
setFixedOmega .......................................................... 128
setInitialConditions ...................................................... 129
setIterationAndCriteria .................................................. 129
setModel .......................................................... 130
setModelError .......................................................... 130
setModelFromLibrary ...................................................... 131
PFIM-package

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

Description

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 2022. 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 Ima) 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/

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`
• 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
- definePKPMModel
- 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
PFIM-package

- 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
• get DataFrameResults
• 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**
PFIM-package

- 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

See Also

Useful links:

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

addModel

Add a model to a library of models.

Description

Add a model to a library of models.

Usage

addModel(object, model)

## S4 method for signature 'LibraryOfModels'
addModel(object, model)

Arguments

object An object from the class LibraryOfModels.
model An object from the class Model.

Value

The library of models with the added model.

addModels

Add a models to a library of models.

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.
Administration-class

Value

The library of models with the added models.

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.
- timeDose: 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.

checkValiditySamplingConstraint

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 \( g(\text{sigmaInter, sigmaSlope, cError, } f(x, \theta)) = \text{sigmaInter} + \text{sigmaSlope} \cdot f(x, \theta). \) 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 \( g(\text{sigma_inter, sigma_slope, c_error, } f(x, \theta)) = \text{sigma_inter}. \) 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**

convertPKModelAnalyticToPKModelODE(object)

## S4 method for signature 'ModelAnalytic'
convertPKModelAnalyticToPKModelODE(object)

## S4 method for signature 'ModelAnalyticSteadyState'
convertPKModelAnalyticToPKModelODE(object)

## S4 method for signature 'ModelAnalyticInfusion'
convertPKModelAnalyticToPKModelODE(object)

**Arguments**

- object: An object from the class Model.

**Value**

A ode model.

defineModel

**Description**

Define a model.

**Usage**

defineModel(object, designs)

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

**Arguments**

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

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

defineModelFromLibraryOfModels

Description
Define a model from the library of models.

Usage
defineModelFromLibraryOfModels(object, designs)

## 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

Description
Define the type of a model.

Usage
defineModelType(object, designs)

## 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.

**definePKModel**

*Define a PK model.*

**Description**

Define a user defined model.

**Usage**

```r
defineModelUserDefined(object, designs)
```

## S4 method for signature 'Model'

```r
defineModelUserDefined(object, designs)
```

**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.*

**Description**

Define a PK model.

**Usage**

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

## S4 method for signature 'ModelAnalytic'

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

## S4 method for signature 'ModelAnalyticSteadyState'

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

## S4 method for signature 'ModelAnalyticInfusion'

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

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

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

## S4 method for signature 'ModelODEInfusionDoseInEquations'
definePKPDModel(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.

---

**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'
Design-class

Define PKPD Model

```r
definePKPDModel(PKModel, PDMModel, outcomes)
## S4 method for signature 'ModelODEBolus,ModelODE'
definePKPDModel(PKModel, PDMModel, outcomes)
## S4 method for signature 'ModelODEDoseInEquations,ModelODE'
definePKPDModel(PKModel, PDMModel, outcomes)
## S4 method for signature 'ModelODEDoseNotInEquations,ModelODE'
definePKPDModel(PKModel, PDMModel, outcomes)
## S4 method for signature 'ModelODEInfusion,ModelODE'
definePKPDModel(PKModel, PDMModel, outcomes)
## S4 method for signature 'ModelODEInfusionDoseInEquations,ModelODE'
definePKPDModel(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**

Evaluate an design

**Usage**

EvaluateDesign(object, model, fim)

## 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.

**Usage**

EvaluateErrorModelDerivatives(object, evaluationOutcome)

## 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.

EvaluateFisherMatrix  Evaluate the Fisher matrix ( population, individual and Bayesian )

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

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

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

```r
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**

```r
EvaluateVarianceModel(object, arm, evaluationModel)
```

## S4 method for signature 'Model'
EvaluateVarianceModel(object, arm, evaluationModel)
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 vectors 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,
Function

    fisher_input,  
nok_input,      
protdep_input, 
freqdep_input

Arguments

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

Value

    A list giving the results of the outputs of the Fedorov-Wynn algorithm.

---

Fim-class

Class "Fim"

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 param-
eters 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

_A generateFimsFromConstraints_

Generate the fim from the constraints

Description

Generate the fim from the constraints

Usage

`generateFimsFromConstraints(object, fims)`

## 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

_A generateReportEvaluation_

Generate the report for the evaluation

Description

Generate the report for the evaluation
Usage

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

## S4 method for signature 'BayesianFim'
```r
generateReportEvaluation(
  object,
  evaluationObject,
  outputPath,
  outputFile,
  plotOptions
)
```

## S4 method for signature 'IndividualFim'
```r
generateReportEvaluation(
  object,
  evaluationObject,
  outputPath,
  outputFile,
  plotOptions
)
```

## S4 method for signature 'PopulationFim'
```r
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**

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

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

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

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

```r
## S4 method for signature 'PSOAlgorithm'
generateReportOptimization(
```

generateSamplingsFromSamplingConstraints

Generate samplings from sampling constraints

Usage

generateSamplingsFromSamplingConstraints(object)

Arguments

object An object from the class SamplingTimeConstraints.

Arguments

object An object from the class SamplingTimeConstraints.

Value

The report for the optimization in html.
generateTables

**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)
```

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

```r
## S4 method for signature 'Optimization'
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)
```

```r
## S4 method for signature 'LogNormal'
getAdjustedGradient(object, outcomesGradient)
```

```r
## S4 method for signature 'Normal'
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

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

Description

Get the administration constraints by outcome.

Usage

getAdministrationConstraint(object, outcome)

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

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An object <code>Arm</code> from the class <code>Arm</code>.</td>
</tr>
<tr>
<td>outcome</td>
<td>A string giving the name of the outcome.</td>
</tr>
</tbody>
</table>

**Value**

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

---

**Description**

Get all the administration for an arm.

**Usage**

```r
getAdministrations(object)
```

```r
## S4 method for signature 'Arm'
getAdministrations(object)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An object <code>Arm</code> from the class <code>Arm</code>.</td>
</tr>
</tbody>
</table>

**Value**

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

---

**getAdministrationsConstraints**

**Description**

Get the administrations constraints.

**Usage**

```r
getAdministrationsConstraints(object)
```

```r
## S4 method for signature 'Arm'
getAdministrationsConstraints(object)
```
getArms

Get the arms of an object.

Arguments

object An object Arm from the class Arm.

Value

The list administrationsConstraints.

getcError

Get the parameter c.

Description

Get the parameter c.

Usage

getcError(object)

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

Arguments

object An object defined form a class of PFIM.

Value

A list containing the arms of the object.
**getColumnAndParametersNamesFIM**

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

**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**

```r
getColumnAndParametersNamesFIM(object, model)
```

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

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

```r
## S4 method for signature 'PopulationFim'
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.

getConditionNumberVarianceEffects

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

Description

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.

ggetContent

Get content of a library of models.

Description

Get content of a library of models.

Usage

ggetContent(object)

# S4 method for signature 'LibraryOfModels'
ggetContent(object)
getDataFrameResults

Arguments

object An object from the class LibraryOfModels.

Value

A list giving the content of the library of models.

generateCorrelationMatrix Get the correlation matrix.

Description

Get the correlation matrix.

Usage

generateCorrelationMatrix(object)

## S4 method for signature 'Fim'

generateCorrelationMatrix(object)

Arguments

object An object from the class Fim.

Value

The correlation matrix of the fim.

generateDataFrameResults Get the dataframe of the results.

Description

Get the dataframe of the results.

Usage

generateDataFrameResults(object, threshold)

## S4 method for signature 'MultiplicativeAlgorithm'

generateDataFrameResults(object, threshold)
**getDcriterion**

**Arguments**
- object: An object from the class `MultiplicativeAlgorithm`.
- threshold: The threshold for the optimal weights.

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

---

**getDescription**

**Get the D criterion of the fim.**

**Description**
Get the D criterion of the fim.

**Usage**

```r
getcriterion(object)
```

```r
## S4 method for signature 'Fim'
getcriterion(object)
```

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

**Value**
A numeric giving the D criterion of the fim.

---

**getDelta**

**Get the parameter delta**

**Description**
Get the parameter delta.

**Usage**

```r
gedelta(object)
```

```r
## S4 method for signature 'MultiplicativeAlgorithm'
gedelta(object)
```
**getDerivatives**  
*Get the derivatives of the model error equation.*

**Description**
Get the derivatives of the model error equation.

**Usage**
```r
generateDerivatives(object)
```

## 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**
```r
generateDescription(object)
```

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

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

Value
A list giving the description of a model.

getDesigns  Get the designs.

Description
Get the designs.

Usage
getDesigns(object)

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

Arguments
object An object from the class PFIMProject.

Value
A list giving the designs of the object.

generateDeterminant  Get the determinant of the fim.

Description
Get the determinant of the fim.

Usage
getDeterminant(object)

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

Arguments
object An object from the class Fim.

Value
A numeric giving the determinant of the fim.
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.

Value
The parameter distribution.

gDose  getDose

Description
Get the amount of doses.

Usage
getDose(object)

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

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

Arguments

object  An object Administration from the class Administration.

Value
The numeric amount_dose giving the amount of doses.
getEigenValues

Get the eigenvalues of the fim.

Description
Get the eigenvalues of the fim.

Usage
getEigenValues(object)

Arguments
object An object from the class Fim.

Value
A vector giving the eigenvalues of the fim.

gElementaryProtocols

Get the elementary protocols.

Description
Get the elementary protocols.

Usage
gElementaryProtocols(object, fims)

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

Value
A list containing the results of the evaluation of the elementary protocols giving the numberofTimes, nbOfDimensions, totalCost, samplingTimes and the fisherMatrices.
getEquation    Get the equation of a model error.

Description
Get the equation of a model error.

Usage
getEquation(object)

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

Arguments
object    An object from the class ModelError.

Value
An expression giving the equation of a model error.

getEquations    Get the equations of a model.

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**

```r
getEquationsAfterInfusion(object)
```

## S4 method for signature 'Model'

```r
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
getEquationsDuringInfusion(object)
```

## S4 method for signature 'Model'

```r
getEquationsDuringInfusion(object)
```

**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
getEvaluationFIMResults(object)
```

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

**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
generateInitialDesignResults(object)
```

```r
## S4 method for signature 'Optimization'
generateInitialDesignResults(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.
### Description

Get the fim of an object.

### Usage

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 from a class of PFIM.

### Value

The FIM of the object.

The fim of the object.

---

### Description

Get the FIM.

### Usage

getFisherMatrix(object)

### S4 method for signature 'Fim'

getFisherMatrix(object)
getFixedMu

Arguments

object An object from the class Fim.

Value

A matrix giving the FIM.

getFixedEffects

Get the matrix of fixed effects.

Arguments

object An object from the class Fim.

Value

The matrix of the fixed effects.

getFixedMu

Get the fixed effect.

Arguments

object An object from the class ModelParameter.
getFixedOmega

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.

Value
A boolean giving the fixed omega.

getFixedParameters

Description
Get the fixed parameters.

Usage
genericFixedParameters(object)

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

Arguments
object An object from the class Model.

Value
A list giving the fixed parameters of the model.
**getFixedTimes**

*Get the fixed sampling times.*

**Description**

Get the fixed sampling times.

**Usage**

```r
getFixedTimes(object)
```

## S4 method for signature 'SamplingTimeConstraints'

```r
getFixedTimes(object)
```

**Arguments**

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

**Value**

A vector giving the fixed sampling times.

---

**getInitialConditions**

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

**Description**

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

**Usage**

```r
getInitialConditions(object)
```

## S4 method for signature 'Arm'

```r
getInitialConditions(object)
```

## S4 method for signature 'Model'

```r
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

*Get the parameter lambda.*

**Description**
Get the parameter lambda.

**Usage**
```
getLambda(object)
```

### S4 method for signature 'MultiplicativeAlgorithm'
```
getLambda(object)
```

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

**Value**
A numeric giving the parameter lambda.
getLibraryPDModels

Get the library of PD models.

Description
Get the library of PD models.

Usage
getLibraryPDModels(object)

## S4 method for signature 'LibraryOfModels'
getLibraryPDModels(object)

Arguments
object An object from the class LibraryOfModels.

Value
A list giving the PD models.

getLibraryPKModels

Get the library of PK models.

Description
Get the library of PK models.

Usage
getLibraryPKModels(object)

## 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)

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

Arguments
object An object from the class SamplingTimeConstraints.

Value
A numeric giving the minimal sampling times.

getModel

Get the model.

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  Get the model equations.

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 Values

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.

---

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.
**Description**

Get the fixed effect of an object.

**Usage**

\[
\text{getMu}(\text{object})
\]

\[
\text{## S4 method for signature 'Distribution'}
\text{getMu}(\text{object})
\]

\[
\text{## S4 method for signature 'ModelParameter'}
\text{getMu}(\text{object})
\]

**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**

\[
\text{getName}(\text{object})
\]

\[
\text{## S4 method for signature 'Arm'}
\text{getName}(\text{object})
\]

\[
\text{## S4 method for signature 'Design'}
\text{getName}(\text{object})
\]

\[
\text{## S4 method for signature 'ModelParameter'}
\text{getName}(\text{object})
\]

\[
\text{## S4 method for signature 'LibraryOfModels'}
\]
getNames(object)

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

## S4 method for signature 'PFIMProject'
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.

**Usage**

getNames(object)

## S4 method for signature 'list'
getName(object)

**Arguments**

object An object defined form a class of PFIM.

**Value**

A vector giving the names of the object.
**getDescription**

Get the number of arms in a design.

**Usage**

getDescription(object)

### S4 method for signature 'Design'

description(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.

**Usage**

getNumberOfIterations(object)

### 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)
```

## S4 method for signature 'Model'

```r
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**

```r
getNumberOfSamplingsOptimisable(object)
```

## S4 method for signature 'SamplingTimeConstraints'

```r
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An object from the class <code>SamplingTimeConstraints</code>.</td>
</tr>
</tbody>
</table>

Value
A vector giving the number of sampling times by windows.

getOdeSolverParameters

getOdeSolverParameters

Description
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An object defined form a class of PFIM.</td>
</tr>
</tbody>
</table>
getOmega

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.

Value
The matrix omega of an object.

getOptimalDesign

Description
Get the optimal design.

Usage
getOptimalDesign(object)

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

Arguments

object An object from the class OptimizationAlgorithm.
**getOptimalWeights**

Value

The optimal design.

---

**getOptimalWeights**  
*Get the optimal weights.*

---

Description

Get the optimal weights.

Usage

getOptimalWeights(object)

## S4 method for signature 'MultiplicativeAlgorithm'
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)

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

Arguments

object An object from the class Optimization.

Value

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

*Get the optimization algorithm.*

**Description**

Get the optimization algorithm.

**Usage**

```r
getOptimizer(object)
```

```r
## 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

*Get the optimization parameters.*

**Description**

Get the optimization parameters.

**Usage**

```r
getOptimizerParameters(object)
```

```r
## 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.
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.

desc

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**
```r
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.

---

**getPDMModel**  Get a PD model.

**Description**
Get a PD model.

**Usage**
```r
getPDMModel(object, PDModelName)

## S4 method for signature 'LibraryOfPKPDModels'
getPDMModel(object, PDModelName)
```

**Arguments**
- `object` An object from the class `LibraryOfPKPDModels`.
- `PDModelName` A string giving the name of the PD model.
getPKModel

Value
Return a PD model.

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.
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.

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.
getRSE

Get the RSE

Description
Get the RSE

Usage
getRSE(object, model)

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

## S4 method for signature 'IndividualFim'
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.

gSamplings

Get the sampling of an object.

Description
Get the sampling of an object.

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  
getSamplingTime

Description

Get the sampling times by outcome.

Usage

getsamplingTime(object, outcome)

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

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

getSamplingTimeConstraint(object, outcome)

## S4 method for signature 'Arm'
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)
getSE

Arguments

object An object Arm from the class Arm.

Value

The list samplingTimes for the object Arm.

getsSamplingTimesConstraints

Description

Get the sampling times constraints.

Usage

getsSamplingTimesConstraints(object)

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

Arguments

object An object Arm from the class Arm.

Value

The list getsSamplingTimesConstraints.

gse

Description

Get the SE.

Usage

gse(object)

## S4 method for signature 'Fim'
gse(object)
getShrinkage

**Arguments**

object  
An object from the class Fim.

**Value**

A vector giving the SE.

---

getShrinkage  
*Get the shrinkage.*

---

**Description**

Get the shrinkage.

**Usage**

```r
getShrinkage(object)
```

## S4 method for signature 'BayesianFim'
```
getShrinkage(object)
```

## S4 method for signature 'IndividualFim'
```
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.

---

getSigmaSlope Get the parameter sigma slope.

Description

Get the parameter sigma slope.

Usage

getSigmaSlope(object)

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

Arguments

object An object from the class ModelError.

Value

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

Get the size of an object.

**Usage**

ggetSize(object)

`## S4 method for signature 'Arm'
ggetSize(object)`

`## S4 method for signature 'Design'
ggetSize(object)`

**Arguments**

object  
An object defined form a class of PFIM.

**Value**

A numeric giving the size of the object.

---

**getDescription**

Get the frequency tau.

**Usage**

ggetTau(object)

`## S4 method for signature 'Administration'
ggetTau(object)`

**Arguments**

object  
An object Administration from the class Administration.

**Value**

The numeric tau giving the frequency tau.
### getTimeDose

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

**Usage**

```r
getTimeDose(object)
```

```r
## S4 method for signature 'Administration'
getTimeDose(object)
```

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

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

### getTinf

**Description**
Get the infusion duration.

**Usage**

```r
getTinf(object)
```

```r
## 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)

Arguments
object
An object from the class Fim.

Value
The matrix of the variance effects.

IndividualFim-class
Class "Fim"

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.

---

isModelBolus Test if a mode is bolus.

---

Description

Test if a mode is bolus.

Usage

isModelBolus(object, designs)

## S4 method for signature 'Model'

isModelBolus(object, designs)
isModelInfusion

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.

isModelODE Test if a mode is ode

Description

Test if a mode is infusion

Usage

isModelInfusion(object)

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

Arguments

object An object from the class Model.

Value

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

isModelODE Test if a mode is ode

Description

Test if a mode is ode.

Usage

isModelODE(object)

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

object  An object from the class Model.

Value

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

isModelSteadyState  Test if a mode is steady state.

Description

Test if a mode is steady state.

Usage

isModelSteadyState(object)

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

Arguments

object  An object from the class Model.

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

...

ModelODE-class  

Class "ModelODE"

Description

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

ModelODEBolus-class  

Class "ModelODEBolus"

Description

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

Class "ModelODEDoseInEquations"

Description

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.

ModelODEDoseNotInEquations-class

Class "ModelODEDoseNotInEquations"

Description

...

ModelODEInfusion-class

Class "ModelODEInfusion"

Description

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.
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**

**Class "Normal"**

**Description**

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

Class "Optimization"

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.
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.

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)
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.

---

`parametersForComputingGradient`

*Define the parameters for computing the gradients of a model.*

---

Description

Define the parameters for computing the gradients of a model.

Usage

`parametersForComputingGradient(object)`

### S4 method for signature 'Model'

`parametersForComputingGradient(object)`

Arguments

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

Value

A list giving the parameters for computing the gradients of a model.

---

`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.
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

plot

*Graphs of the results of the evaluation.*

**Description**

Graphs of the results of the evaluation.

**Usage**

```r
plot(object, plotOptions)
```

```r
## S4 method for signature 'Evaluation'
plot(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`.

---

### plotOutcomesEvaluation

*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.

---

**Description**

Plot the evaluation of the outcome gradients.

**Usage**

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

### S4 method for signature 'Design'

```r
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**

```r
plotRSE(object, plotOptions)
```

```r
## 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.
plotShrinkage

Graph of the shrinkage.

Description
Graph of the shrinkage.

Usage
plotShrinkage(object, plotOptions)

## S4 method for signature 'PFIMProject'
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
plotWeights(object, threshold)

## S4 method for signature 'MultiplicativeAlgorithm'
plotWeights(object, threshold)

## S4 method for signature 'Optimization'
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.
Proportional-class

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

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

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)
reportTablesDesign

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.

---

reportTablesAdministration

Description
Generate table for the report.

Usage
```r
reportTablesAdministration(object)
```

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

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

Value
A table of the administration parameters for the report.

---

reportTablesDesign

Description
Generate table for the report.

Usage
```r
reportTablesDesign(object)
```

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

object An object design from the class Design.

description

Generate the tables for the report.

Usage

reportTablesFIM(object, evaluationObject)

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

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

## 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

reportTablesModelError(object)

## S4 method for signature 'Model'
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

reportTablesModelParameters(object)

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

Arguments

object An object from the class Model.

Value

A kable table for the evaluation report.
**Description**
Generate all the tables for the evaluation report.

**Usage**

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

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

**Arguments**

- `object`: An object of class `Evaluation`.
- `plotOptions`: A list containing options for the plots.

**Value**

The list `tables` containing the tables for the evaluation report.

---

**Description**
Generate table for the report.

**Usage**

```r
reportTablesSamplingConstraints(object)
```

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

**Arguments**

- `object`: An object of 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**

```r
resizeFisherMatrix(nbOfDimensions, fisherMatrix)
```

```r
## S4 method for signature 'ANY'
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`
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

*Set the arms of an object.*

### Description
Set the arms of an object.

### Usage
```r
guaranteedMin(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

*Set the parameter c.*

### Description
Set the parameter c.

### Usage
```r
guaranteedMin(object, cError)
```

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

The model error with 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.
collection 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
description(object, description)
```

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

```
# S4 method for signature 'Optimization'
designs(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

**Description**
Set the distribution.

**Usage**
setAddression(object, distribution)

## S4 method for signature 'ModelParameter'
setAddression(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

**Description**
Set the amount of dose

**Usage**
setAddression(object, dose)

## S4 method for signature 'Administration'
setAddression(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'
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'
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

setEquationsAfterInfusion(object, equations)

## 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

setEquationsDuringInfusion(object, equations)

## 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**

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

### S4 method for signature 'Optimization'

```r
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**

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

### S4 method for signature 'Optimization'

```r
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.

---

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  

Convert the type of the object fim to a string.

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

Sets the Fisher Information Matrix (FIM).

**Description**

Set the FIM.

**Usage**

```r
setFisherMatrix(object, value)
```

**Arguments**

- `object`: An object from the class `Fim`.
- `value`: A matrix giving the FIM.

**Value**

The object from the class `Fim` with the FIM updated.

### setFixedEffects

Sets the fixed effects.

**Description**

Set the fixed effects.

**Usage**

```r
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)

## 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 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)

## 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)

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

Value
The model with the updated initial conditions.

setIterationAndCriteria

Description
Set the iteration with the convergence criteria.

Usage
setIterationAndCriteria(object, value)

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

**Description**
Set the model error.

**Usage**

```r
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.
Value

The object with the updated fixed effect mu.

```
setName

Set the name of an object.
```

Description

Set the name of an object.

Usage

```
setName(object, name)
```

```
## S4 method for signature 'Arm'
setName(object, name)
```

```
## S4 method for signature 'Design'
setName(object, name)
```

```
## S4 method for signature 'Model'
setName(object, name)
```

Arguments

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

Value

The object with the updated name.

```
setNumberOfArms

setNumberOfArms
```

Description

Set the number of arms in a design.

Usage

```
setNumberOfArms(object, numberOfArms)
```

```
## S4 method for signature 'Design'
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.

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.
setOptimalDesign

Usage

setOmega(object, value)

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

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

Arguments

object An object defined form 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)

## 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

Description
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

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

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)
setParameters

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.

---

setOutcomesGradient

Description

Set the results of the evaluation of the outcomes.

Usage

```r
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.

---

setParameters

Description

Set the parameters of an object.
Usage

```r
setParameters(object, parameters)
```

```r
## S4 method for signature 'Design'
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 form 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

```r
setSamplingConstraintForOptimization(object)
```

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

object An object from the class Design.

Value

The arms with the sampling times constraints.

setSamplings

Set the sampling times.

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.

setSamplingTime

Set the sampling time of an arm.

Usage

setSamplingTime(object, samplingTime)

## S4 method for signature 'Arm'
setSamplingTime(object, samplingTime)
setSamplingTimes

**Arguments**

- object: An object Arm from the class Arm.
- samplingTime: An object samplingTime from the class SamplingTimes.

**Value**

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

---

setSamplingTimesConstraints

**Description**

Set the sampling times constraints.

**Usage**

setSamplingTimesConstraints(object, samplingTimesConstraints)

## S4 method for signature 'Arm'
setSamplingTimesConstraints(object, samplingTimesConstraints)

**Arguments**

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

**Value**

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

---
Arguments

object  An object Arm from the class Arm.
samplingTimesConstraints  An object SamplingTimeConstraints from the class SamplingTimeConstraints.

Value

The arm with the new sampling time constraints.

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 Administration from the 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.

**Description**

Set the times vector when doses are given.

**Usage**

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

```r
## S4 method for signature 'Administration'
setTimeDose(object, timeDose)
```

**Arguments**

- **object**: An object Administration from the 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)
```

```r
## S4 method for signature 'Administration'
setTinf(object, Tinf)
```
**setVarianceEffects**

*Set the matrix of the variance effects.*

**Description**

Set the matrix of the variance effects.

**Usage**

```
setVarianceEffects(object)
```

**Arguments**

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

**Value**

Update the matrix of the variance effects.

---

**show,Design-method**

**Description**

```
show
```

**Arguments**

- `object` An object `Administration`.

**Value**

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

---

**Arguments**

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

**Value**

A numeric value for the infusion duration `Tinf`.

---
Usage

```r
## 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 'PGBAlgorithm'
show(object)

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

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

Arguments

```
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, 54, 89, 122, 145, 146
Administration (Administration-class), 21
Administration-class, 21
AdministrationConstraints, 9
AdministrationConstraints
(AdministrationConstraints-class), 21
AdministrationConstraints-class, 21
Arm, 9, 23, 30, 32–35, 44–46, 62, 85, 86, 118,
141, 142, 144
Arm (Arm-class), 22
Arm-class, 22
BayesianFim, 10
BayesianFim (BayesianFim-class), 22
BayesianFim-class, 22
checkSamplingTimeConstraintsForContinuousOptimization,
18, 22
checkSamplingTimeConstraintsForContinuousOptimization,
(checkSamplingTimeConstraintsForContinuousOptimization-class),
22
checkValiditySamplingConstraint, 23
checkValiditySamplingConstraint,Design-method
(checkValiditySamplingConstraint), 23
Combined1, 10, 110
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), 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, ModelAnalytic, ModelODE-method
(EvaluateModel), 33
definePKPDModel, ModelAnalyticInfusion, ModelAnalyticInfusionSteadyState-method
(EvaluateModel), 33
definePKPDModel, ModelAnalyticInfusion, ModelODE-method
(EvaluateModel), 33
definePKPDModel, ModelAnalyticSteadyState, ModelAnalyticInfusion-method
(EvaluateModel), 33
definePKPDModel, ModelAnalyticInfusion, ModelODE-method
(EvaluateModel), 33
definePKPDModel, ModelODEBolus, ModelODE-method
(EvaluateModel), 33
definePKPDModel, ModelODEDoseInEquations, ModelODE-method
(EvaluateModel), 33
definePKPDModel, ModelODEDoseNotInEquations, ModelODE-method
(EvaluateModel), 33
definePKPDModel, ModelODEInfusion, ModelODE-method
(EvaluateModel), 33
definePKPDModel, ModelODEInfusionDoseInEquations, ModelODE-method
(EvaluateModel), 33
Design (Design-class), 29
Design-class, 29
Distribution, 10, 44, 122
Distribution (Distribution-class), 30
Distribution-class, 30
EvaluateArm, 9, 30
EvaluateArm, Arm-method (EvaluateArm), 30
EvaluateDesign, 10, 31
EvaluateDesign, Design-method (EvaluateDesign), 31
EvaluateErrorModelDerivatives, 15, 31
EvaluateErrorModelDerivatives, ModelError-method (EvaluateErrorModelDerivatives), 31
EvaluateFisherMatrix, 10–12, 18, 32
EvaluateFisherMatrix, BayesianFim-method (EvaluateFisherMatrix), 32
EvaluateFisherMatrix, IndividualFim-method (EvaluateFisherMatrix), 32
EvaluateFisherMatrix, PopulationFim-method (EvaluateFisherMatrix), 32
EvaluateModel, 14–16, 33
EvaluateDesign, Design-method
(EvaluateDesign), 31
FedorovWynnAlgorithm, 11
FedorovWynnAlgorithm (FedorovWynnAlgorithm-class), 36
FedorovWynnAlgorithm-class, 36
FedorovWynnAlgorithm_Rcpp, 11, 36
Fim, 11, 30–32, 34, 39, 40, 47–51, 53, 55, 60, 83, 87, 91, 113, 126, 127, 142, 146
Fim (Fim-class), 37
Fim-class, 37
fisher.simplex, 19, 38
fun.amoeba, 19, 38
generateFimsFromConstraints, 17, 39
generateFimsFromConstraints, Optimization-method (generateFimsFromConstraints), 39
generateReportEvaluation, 10, 12, 13, 18, 39
getCorrelationMatrix, Fim-method
  (getCorrelationMatrix), 50
getDataFrameResults, 16, 50
dataFrameResults, MultiplicativeAlgorithm-method
  (getDataFrameResults), 50
dcriterion, 11, 51
dcriterion, Fim-method
  (dcriterion), 51
delta, 16, 51
delta, MultiplicativeAlgorithm-method
  (delta), 51
getDerivatives, 15, 52
derivatives, ModelError-method
  (getDerivatives), 52
description, 13, 52
description, Model-method
  (description), 52
designs, 17, 53
designs, PFIMProject-method
  (designs), 53
determinant, 11, 53
determinant, Fim-method
  (determinant), 53
distribution, 16, 54
distribution, ModelParameter-method
  (distribution), 54
dose, 9, 54
dose, Administration-method
  (dose), 54
dose, AdministrationConstraints-method
  (dose), 54
eigenvalues, 12, 55
eigenvalues, Fim-method
  (eigenvalues), 55
getLocalProtocols, 17, 55
globalProtocols, Optimization-method
  (getLocalProtocols), 55
getEquation, 15, 56
equation, ModelError-method
  (getEquation), 56
equations, 13, 56
equations, Model-method
  (equations), 56
equationsAfterInfusion, 15, 57
equationsAfterInfusion, Model-method
  (equationsAfterInfusion), 57
equationsDuringInfusion, 13, 57
equationsDuringInfusion, Model-method
  (equationsDuringInfusion), 57
evaluationFIMResults, 17, 58
evaluationFIMResults, Optimization-method
  (evaluationFIMResults), 58
evaluationInitialDesignResults, 17, 58
evaluationInitialDesignResults, Optimization-method
  (evaluationInitialDesignResults), 58
getFim, 10, 12, 17, 59
getFim, Design-method (getFim), 59
getFim, OptimizationAlgorithm-method
  (getfim), 59
getFisherMatrix, 11, 59
getFisherMatrix, Fim-method
  (getFisherMatrix), 59
getFixedEffects, 11, 60
getFixedEffects, Fim-method
  (getFixedEffects), 60
getFixedMu, 16, 60
getFixedMu, ModelParameter-method
  (getFixedMu), 60
getFixedOmega, 16, 61
getFixedOmega, ModelParameter-method
  (getFixedOmega), 61
getFixedParameters, 14, 61
getFixedParameters, Model-method
  (getFixedParameters), 61
getFixedTimes, 18, 62
getFixedTimes, SamplingTimeConstraints-method
  (getFixedTimes), 62
getInitialConditions, 9, 14, 62
getInitialConditions, Arm-method
  (getInitialConditions), 62
getInitialConditions, Model-method
  (getInitialConditions), 62
getIterationAndCriteria, 63
getIterationAndCriteria, OptimizationAlgorithm-method
  (getIterationAndCriteria), 63
getLambda, 16, 63
getLambda, MultiplicativeAlgorithm-method
  (getLambda), 63
getLibraryPDModels, 13, 64
getLibraryPDModels, LibraryOfModels-method
  (getLibraryPDModels), 64
getLibraryPKModels, 13, 64
getLibraryPKModels, LibraryOfModels-method (getLibraryPKModels), 64
getMinSampling, 18, 65
getMinSampling, SamplingTimeConstraints-method (getMinSampling), 65
getModel, 17, 65
getModel, PFIMProject-method (getModel), 65
getModelEquations, 17, 66
getModelEquations, PFIMProject-method (getModelEquations), 66
getModelError, 12, 13, 17, 66
getModelError, Model-method (getModelError), 66
getModelError, PFIMProject-method (getModelError), 66
getModelErrorParametersValues, 14, 67
getModelErrorParametersValues, Model-method (getModelErrorParametersValues), 67
getModelFromLibrary, 14, 67
getModelFromLibrary, Model-method (getModelFromLibrary), 67
getModelParameters, 17, 68
getModelParameters, PFIMProject-method (getModelParameters), 68
getModelParametersValues, 68
getModelParametersValues, Model-method (getModelParametersValues), 68
getMu, 11, 12, 16, 69
getMu, Distribution-method (getMu), 69
getMu, ModelParameter-method (getMu), 69
getName, 9, 10, 12, 13, 16, 17, 69
getName, Arm-method (getName), 69
getName, Design-method (getName), 69
getName, LibraryOfModels-method (getName), 69
generateModel, 17, 67
getName, Model-method (getName), 69
getName, ModelParameter-method (getName), 69
getName, PFIMProject-method (getName), 69
getNames, 12, 70
getNames, list-method (getNames), 70
getNumberOfArms, 10, 71
getNumberOfArms, Design-method (getNumberOfArms), 71
getNumberOfIterations, 16, 71
getNumberOfIterations, OptimizationAlgorithm-method (getNumberOfIterations), 71
getNumberOfParameters, 14, 72
getNumberOfParameters, Model-method (getNumberOfParameters), 72
getNumberOfSamplingOptimisable, 18, 72
getNumberOfSamplingOptimisable, SamplingTimeConstraints-method (getNumberOfSamplingOptimisable), 72
getNumberOfTimesByWindows, 18, 73
getNumberOfTimesByWindows, SamplingTimeConstraints-method (getNumberOfTimesByWindows), 73
generateOdeSolverParameters, 12, 14, 17, 73
generateOdeSolverParameters, Model-method (generateOdeSolverParameters), 73
generateOdeSolverParameters, PFIMProject-method (generateOdeSolverParameters), 73
generateOmega, 11, 12, 16, 74
generateOmega, Distribution-method (generateOmega), 74
generateOmega, ModelParameter-method (generateOmega), 74
generateOptimalDesign, 74
generateOptimalDesign, OptimizationAlgorithm-method (generateOptimalDesign), 74
generateOptimalWeights, 16, 75
generateOptimalWeights, MultiplicativeAlgorithm-method (generateOptimalWeights), 75
generateOptimizationResults, 17, 75
generateOptimizationResults, Optimization-method (generateOptimizationResults), 75
generateOptimizer, 17, 76
generateOptimizer, PFIMProject-method (generateOptimizer), 76
generateOptimizerParameters, 17, 76
generateOptimizerParameters, PFIMProject-method (generateOptimizerParameters), 76
generateOutcome, 9, 12, 15, 18, 77
generateOutcome, Administration-method (generateOutcome), 77
generateOutcome, AdministrationConstraints-method (generateOutcome), 77
generateOutcome, ModelError-method (generateOutcome), 77
generateOutcome, SamplingTimeConstraints-method (generateOutcome), 77
generateOutcome, SamplingTimes-method (generateOutcome), 77
generateOutcome, MultiplicativeAlgorithm-method (generateOutcome), 13, 17, 77
getOutcomes, Model-method (getOutcomes), 77
getOutcomes, PFIMProject-method (getOutcomes), 77
getOutcomesEvaluation, 10, 78
getOutcomesEvaluation, Design-method (getOutcomesEvaluation), 78
getOutcomesForEvaluation, 13, 79
getOutcomesForEvaluation, Model-method (getOutcomesForEvaluation), 79
getOutcomesGradient, 10, 79
getOutcomesGradient, Design-method (getOutcomesGradient), 79
getParameters, 10, 12, 13, 15, 80
getParameters, Model-method (getParameters), 80
getParameters, ModelError-method (getParameters), 80
getPDModel, 13, 80
getPDModel, LibraryOfPKPDModels-method (getPDModel), 80
getPKModel, 13, 81
getPKModel, LibraryOfPKPDModels-method (getPKModel), 81
getPKPDModel, 13, 81
getPKPDModel, LibraryOfPKPDModels-method (getPKPDModel), 81
getPlotOptions, 82
getProportionsOfSubjects, 17, 82
getProportionsOfSubjects, Optimization-method (getProportionsOfSubjects), 82
getRSE, 10, 12, 13, 18, 83
getRSE, BayesianFim-method (getRSE), 83
getRSE, IndividualFim-method (getRSE), 83
getRSE, PopulationFim-method (getRSE), 83
getSamplings, 12, 18, 83
getSamplings, SamplingTimeConstraints-method (getSamplings), 83
getSamplings, SamplingTimes-method (getSamplings), 83
getSamplingsWindows, 18, 84
getSamplingsWindows, SamplingTimeConstraints-method (getSamplingsWindows), 84
getSamplingTime, 9, 84
getSamplingTime, Arm-method (getSamplingTime), 84
getSamplingTimeConstraint, 9, 85
getSamplingTimeConstraint, Arm-method (getSamplingTimeConstraint), 85
getSamplingTimes, 9, 85
getSamplingTimes, Arm-method (getSamplingTimes), 85
getSamplingTimesConstraints, 9, 86
getSamplingTimesConstraints, Arm-method (getSamplingTimesConstraints), 86
getSE, 12, 86
getSE, Fim-method (getSE), 86
getShrinkage, 10, 12, 13, 18, 87
getShrinkage, BayesianFim-method (getShrinkage), 87
getShrinkage, IndividualFim-method (getShrinkage), 87
getShrinkage, PopulationFim-method (getShrinkage), 87
getSigmaInter, 15, 87
getSigmaInter, ModelError-method (getSigmaInter), 87
getSigmaSlope, 15, 88
getSigmaSlope, ModelError-method (getSigmaSlope), 88
getSize, 9, 10, 12, 89
getSize, Arm-method (getSize), 89
getSize, Design-method (getSize), 89
getTau, 9, 89
getTau, Administration-method (getTau), 89
gTimeDose, 9, 90
gTimeDose, Administration-method (gTimeDose), 90
gTinf, 9, 90
gTinf, Administration-method (gTinf), 90
gVarianceEffects, 11, 91
gVarianceEffects, Fim-method (gVarianceEffects), 91
IndividualFim, 12
IndividualFim (IndividualFim-class), 91
IndividualFim-class, 91
isDoseInEquations, 14, 91
isDoseInEquations, Model-method (isDoseInEquations), 91
isModelAnalytic, 14, 92
isModelAnalytic, Model-method
(isModelAnalytic), 92
isModelBolus, 14, 92
isModelBolus, Model-method
(isModelBolus), 92
isModelInfusion, 14, 93
isModelInfusion, Model-method
(isModelInfusion), 93
isModelODE, 14, 93
isModelODE, Model-method (isModelODE), 93
isModelSteadyState, 14, 94
isModelSteadyState, Model-method
(isModelSteadyState), 94
LibraryOfModels, 13, 20, 50, 64, 120
LibraryOfModels (LibraryOfModels-class), 94
LibraryOfModels-class, 94
LibraryOfPKPDModels, 13, 80, 81
LibraryOfPKPDModels (LibraryOfPKPDModels-class), 95
LibraryOfPKPDModels-class, 95
Model (Model-class), 96
Model-class, 96
ModelAnalytic, 14
ModelAnalytic (ModelAnalytic-class), 96
ModelAnalytic-class, 96
ModelAnalyticBolus, 14, 15
ModelAnalyticBolus (ModelAnalyticBolus-class), 96
ModelAnalyticBolus-class, 96
ModelAnalyticBolusSteadyState, 14
ModelAnalyticBolusSteadyState (ModelAnalyticBolusSteadyState-class), 97
ModelAnalyticBolusSteadyState-class, 97
ModelAnalyticInfusion
(ModelAnalyticInfusion-class), 97
ModelAnalyticInfusionSteadyState (ModelAnalyticInfusionSteadyState-class), 97
ModelAnalyticInfusionSteadyState-class, 97
ModelAnalyticSteadyState (ModelAnalyticSteadyState-class), 97
ModelAnalyticSteadyState-class, 97
ModelError, 10, 15, 18, 24, 31, 47, 52, 56, 88, 119, 120, 123, 130, 143
ModelError (ModelError-class), 98
ModelError-class, 98
ModelInfusion, 15, 16
ModelInfusion (ModelInfusion-class), 98
ModelInfusion-class, 98
ModelODE, 15
ModelODE (ModelODE-class), 98
ModelODE-class, 98
ModelODEBolus, 15
ModelODEBolus (ModelODEBolus-class), 98
ModelODEBolus-class, 98
ModelODEDoseInEquations, 15
ModelODEDoseInEquations (ModelODEDoseInEquations-class), 99
ModelODEDoseInEquations-class, 99
ModelODEDoseNotInEquations, 15
ModelODEDoseNotInEquations (ModelODEDoseNotInEquations-class), 99
ModelODEDoseNotInEquations-class, 99
ModelODEInfusion, 16
ModelODEInfusion (ModelODEInfusion-class), 99
ModelODEInfusion-class, 99
ModelODEInfusionDoseInEquations, 16
ModelODEInfusionDoseInEquations (ModelODEInfusionDoseInEquations-class), 99
ModelODEInfusionDoseInEquations-class, 99
ModelParameter, 16, 54, 60, 61, 122, 128
ModelParameter (ModelParameter-class), 100
ModelParameter-class, 100
MultiplicativeAlgorithm, 16, 51, 52, 63, 71, 75, 135
MultiplicativeAlgorithm (MultiplicativeAlgorithm-class), 100
MultiplicativeAlgorithm-class, 100
MultiplicativeAlgorithm_Rcpp, 16, 51, 52, 63, 71, 75, 135
Normal, 16
Normal (Normal-class), 101
Normal-class, 101
Optimization, 17, 38, 39, 42, 55, 58, 75, 82, 121, 125, 135
Optimization (Optimization-class), 102
Optimization-class, 102
OptimizationAlgorithm, 42, 63, 74, 75, 104, 109, 129, 134, 135
OptimizationAlgorithm (OptimizationAlgorithm-class), 103
OptimizationAlgorithm-class, 103
optimize, 11, 16–19, 103
optimize, MultiplicativeAlgorithm-method (optimize), 103
optimize, MultiplicativeAlgorithm-class, 103
optimize, PGOBAAlgorithm-method (optimize), 103
optimize, PSOAlgorithm-method (optimize), 103
optimize, SimplexAlgorithm-method (optimize), 103
package-PFIM (PFIM-package), 7
parametersForComputingGradient, 14, 104
parametersForComputingGradient, Model-method (parametersForComputingGradient), 104
PFIM, (PFIM-package), 7
PFIM-package, 7
PFIMProject, 17, 43, 53, 65, 66, 68, 76, 112, 116, 130
PFIMProject (PFIMProject-class), 104
PFIMProject-class, 104
PGBOAlgorithm, 17
PGBOAlgorithm (PGBOAlgorithm-class), 105
PGBOAlgorithm-class, 105
plot, 18, 106
plot, Evaluation-method (plot), 106
PlotEvaluation, 17
PlotEvaluation (PlotEvaluation-class), 106
PlotEvaluation-class, 106
plotOutcomesEvaluation, 10, 106
plotOutcomesEvaluation, Design-method (plotOutcomesEvaluation), 106
plotOutcomesGradient, 10, 107
plotOutcomesGradient, Design-method (plotOutcomesGradient), 107
plotRSE, 18, 108
plotRSE, PFIMProject-method (plotRSE), 108
plotSE, 18, 108
plotSE, PFIMProject-method (plotSE), 108
plotShrinkage, 18, 109
plotShrinkage, PFIMProject-method (plotShrinkage), 109
plotWeights, 16, 17, 109
plotWeights, MultiplicativeAlgorithm-method (plotWeights), 109
plotWeights, Optimization-method (plotWeights), 109
PopulationFim, 18
PopulationFim (PopulationFim-class), 110
PopulationFim-class, 110
Proportional, 18, 110
Proportional (Proportional-class), 110
Proportional-class, 110
PSOAlgorithm, 18
PSOAlgorithm (PSOAlgorithm-class), 111
PSOAlgorithm-class, 111
Report, 11, 17, 111
Report, Evaluation-method (Report), 111
Report, Optimization-method (Report), 111
reportTablesAdministration, 10, 112
reportTablesAdministration, Design-method (reportTablesAdministration), 112
reportTablesDesign, 10, 112
reportTablesDesign, Design-method (reportTablesDesign), 112
reportTablesFIM, 10, 12, 13, 18, 113
setFixedEffects, Fim-method
  (setFixedEffects), 127
setFixedMu, 16, 128
setFixedMu, ModelParameter-method
  (setFixedMu), 128
setFixedOmega, 16, 128
setFixedOmega, ModelParameter-method
  (setFixedOmega), 128
setInitialConditions, 9, 14, 129
setInitialConditions, Arm-method
  (setInitialConditions), 129
setInitialConditions, Model-method
  (setInitialConditions), 129
setIterationAndCriteria, 129
setIterationAndCriteria, OptimizationAlgorithm-method
  (setIterationAndCriteria), 129
setModel, 17, 130
setModel, PFIMProject-method
  (setModel), 130
setModelError, 14, 130
setModelError, Model-method
  (setModelError), 130
setModelFromLibrary, 13, 131
setModelFromLibrary, Model-method
  (setModelFromLibrary), 131
setMu, 11, 12, 16, 131
setMu, Distribution-method
  (setMu), 131
setMu, ModelParameter-method
  (setMu), 131
setName, 9, 10, 12, 13, 132
setName, Arm-method
  (setName), 132
setName, Design-method
  (setName), 132
setName, Model-method
  (setName), 132
setNumberOfArms, 10, 132
setNumberOfArms, Design-method
  (setNumberOfArms), 132
setOdeSolverParameters, 14, 133
setOdeSolverParameters, Model-method
  (setOdeSolverParameters), 133
setOmega, 11, 12, 16, 133
setOmega, Distribution-method
  (setOmega), 133
setOmega, ModelParameter-method
  (setOmega), 133
setOptimalDesign, 134
setOptimalDesign, OptimizationAlgorithm-method
  (setOptimalDesign), 134
setOptimalWeights, 16, 135
setOptimalWeights, Model-method
  (setOptimalWeights), 135
setOptimizationResults, 17, 135
setOptimizationResults, Optimization-method
  (setOptimizationResults), 135
setOutcome, 9, 12, 18, 136
setOutcome, Administration-method
  (setOutcome), 136
setOutcome, SamplingTimes-method
  (setOutcome), 136
setOutcomes, 13, 136
setOutcomes, Model-method
  (setOutcomes), 136
setOutcomesEvaluation, 10, 137
setOutcomesEvaluation, Design-method
  (setOutcomesEvaluation), 137
setOutcomesForEvaluation, 13, 137
setOutcomesForEvaluation, Model-method
  (setOutcomesForEvaluation), 137
setOutcomesGradient, 10, 138
setOutcomesGradient, Design-method
  (setOutcomesGradient), 138
setParameters, 11–13, 16–19, 138
setParameters, Distribution-method
  (setParameters), 138
setParameters, FedorovWynnAlgorithm-method
  (setParameters), 138
setParameters, Model-method
  (setParameters), 138
setParameters, MultiplicativeAlgorithm-method
  (setParameters), 138
setParameters, PGBOAlgorithm-method
  (setParameters), 138
setParameters, PSOAlgorithm-method
  (setParameters), 138
setParameters, SimplexAlgorithm-method
  (setParameters), 138
setSamplingConstraintForOptimization, 139
setSamplingConstraintForOptimization, Design-method
  (setSamplingConstraintForOptimization), 139
setSamplings, 18, 140
setSamplings, SamplingTimes-method
  (setSamplings), 140
setSamplingTime, Arm-method
  (setSamplingTime), 140
setOptimizationResults, SamplingTimes-method
  (setOptimizationResults), 141
setSamplingTimes, Arm-method
  (setSamplingTimes), 141
setSamplingTimesConstraints, 9, 141
setSamplingTimesConstraints, Arm-method
  (setSamplingTimesConstraints), 141
setShrinkage, 10, 13, 18, 142
setShrinkage, BayesianFim-method
  (setShrinkage), 142
setShrinkage, IndividualFim-method
  (setShrinkage), 142
setShrinkage, PopulationFim-method
  (setShrinkage), 142
setSigmaInter, 15, 143
setSigmaInter, ModelError-method
  (setSigmaInter), 143
setSigmaSlope, 15, 143
setSigmaSlope, ModelError-method
  (setSigmaSlope), 143
setSize, 9, 10, 12, 144
setSize, Arm-method (setSize), 144
setSize, Design-method (setSize), 144
setTau, 9, 144
setTau, Administration-method (setTau), 144
setTimeDose, 9, 145
setTimeDose, Administration-method
  (setTimeDose), 145
setTinf, 9, 145
setTinf, Administration-method
  (setTinf), 145
setVarianceEffects, 11, 146
setVarianceEffects, Fim-method
  (setVarianceEffects), 146
show, Design-method, 146
show, Evaluation-method
  (show, Design-method), 146
show, FedorovWynnAlgorithm-method
  (show, Design-method), 146
show, MultiplicativeAlgorithm-method
  (show, Design-method), 146
show, Optimization-method
  (show, Design-method), 146
show, PGBAAlgorithm-method
  (show, Design-method), 146
show, PSOAlgorithm-method
  (show, Design-method), 146
show, SimplexAlgorithm-method
  (show, Design-method), 146
SimplexAlgorithm, 19
SimplexAlgorithm
  (SimplexAlgorithm-class), 147
SimplexAlgorithm-class, 147
SimplexAlgorithm, 19
SimplexAlgorithm-class, 147
SimplexAlgorithm-class, 147