

# Package: PFIM (via r-universe)

October 24, 2024

**Type** Package

**Title** Population Fisher Information Matrix

**Version** 6.1

**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](https://doi.org/10.1093/biomet/84.2.429)>, Retout S, Comets E, Samson A, Mentré F (2007) <[doi:10.1002/sim.2910](https://doi.org/10.1002/sim.2910)>, Bazzoli C, Retout S, Mentré F (2009) <[doi:10.1002/sim.3573](https://doi.org/10.1002/sim.3573)>, Le Nagard H, Chao L, Tenaillon O (2011) <[doi:10.1186/1471-2148-11-326](https://doi.org/10.1186/1471-2148-11-326)>, Combes FP, Retout S, Frey N, Mentré F (2013) <[doi:10.1007/s11095-013-1079-3](https://doi.org/10.1007/s11095-013-1079-3)> and Seurat J, Tang Y, Mentré F, Nguyen TT (2021) <[doi:10.1016/j.cmpb.2021.106126](https://doi.org/10.1016/j.cmpb.2021.106126)>.

**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, purrr, rmarkdown, kableExtra, stats

**RoxygenNote** 7.3.2

**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' 'LibraryOfPDMModels.R'  
'LibraryOfPKModels.R' 'LibraryOfModels.R'  
'LibraryOfPKPDMModels.R' 'Model.R' 'PFIMProject.R'  
'Evaluation.R' 'OptimizationAlgorithm.R'  
'FedorovWynnAlgorithm.R' 'IndividualFim.R' 'LogNormal.R'  
'ModelODE.R' 'ModelAnalytic.R' 'ModelAnalyticBulus.R'

'ModelAnalyticSteadyState.R' 'ModelAnalyticBolusSteadyState.R'  
 'ModelInfusion.R' 'ModelAnalyticInfusion.R'  
 'ModelAnalyticInfusionSteadyState.R' 'ModelBolus.R'  
 'ModelODEBolus.R' 'ModelODEDoseInEquations.R'  
 'ModelODEDoseNotInEquations.R' 'ModelODEInfusion.R'  
 'ModelODEInfusionDoseInEquations.R' 'MultiplicativeAlgorithm.R'  
 'Normal.R' 'Optimization.R' 'PFIM-package.R' 'PGBOAlgorithm.R'  
 'PSOAlgorithm.R' 'PlotEvaluation.R' 'PopulationFim.R'  
 'Proportional.R' 'SamplingTimeConstraints.R' 'SamplingTimes.R'  
 'SimplexAlgorithm.R'

**Author** France Mentré [aut], Romain Leroux [aut, cre], Jérémy Seurat  
 [aut], Lucie Fayette [aut]

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

**Repository** CRAN

**Date/Publication** 2024-10-23 17:00:06 UTC

## Contents

PFIM-package . . . . .	8
addModel . . . . .	21
addModels . . . . .	21
Administration-class . . . . .	22
AdministrationConstraints-class . . . . .	22
Arm-class . . . . .	23
BayesianFim-class . . . . .	23
checkSamplingTimeConstraintsForContinuousOptimization . . . . .	23
checkValiditySamplingConstraint . . . . .	24
Combined1-class . . . . .	25
computeVMat . . . . .	25
Constant-class . . . . .	26
convertPKModelAnalyticToPKModelODE . . . . .	26
dataForArmEvaluation . . . . .	27
defineModel . . . . .	27
defineModelEquationsFromStringToFunction . . . . .	28
defineModelFromLibraryOfModels . . . . .	30
defineModelType . . . . .	30
defineModelUserDefined . . . . .	31
definePKModel . . . . .	31
definePKPDMModel . . . . .	32
Design-class . . . . .	33
Distribution-class . . . . .	34
EvaluateArm . . . . .	34
EvaluateDesign . . . . .	35
EvaluateErrorModelDerivatives . . . . .	35
EvaluateFisherMatrix . . . . .	36
EvaluateModel . . . . .	37

EvaluateModelGradient . . . . .	38
EvaluateVarianceFIM . . . . .	39
EvaluateVarianceModel . . . . .	39
Evaluation-class . . . . .	40
FedorovWynnAlgorithm-class . . . . .	41
FedorovWynnAlgorithm_Rcpp . . . . .	41
Fim-class . . . . .	42
fisher.simplex . . . . .	43
fun.amoeba . . . . .	43
generateFimsFromConstraints . . . . .	44
generateReportEvaluation . . . . .	44
generateReportOptimization . . . . .	46
generateSamplingsFromSamplingConstraints . . . . .	47
generateTables . . . . .	48
getAdjustedGradient . . . . .	48
getAdministration . . . . .	49
getAdministrationConstraint . . . . .	49
getAdministrations . . . . .	50
getAdministrationsConstraints . . . . .	50
getArms . . . . .	51
getError . . . . .	51
getColumnAndParametersNamesFIM . . . . .	52
getColumnAndParametersNamesFIMInLatex . . . . .	53
getConditionNumberFixedEffects . . . . .	53
getConditionNumberVarianceEffects . . . . .	54
getContent . . . . .	54
getCorrelationMatrix . . . . .	55
getDataForArmEvaluation . . . . .	55
getDataFrameResults . . . . .	56
getDcriterion . . . . .	57
getDelta . . . . .	57
getDerivatives . . . . .	58
getDescription . . . . .	58
getDesigns . . . . .	59
getDeterminant . . . . .	59
getDistribution . . . . .	60
getDose . . . . .	60
getEigenValues . . . . .	61
getElementaryProtocols . . . . .	61
getEquation . . . . .	62
getEquations . . . . .	62
getEquationsAfterInfusion . . . . .	63
getEquationsDuringInfusion . . . . .	63
getEvaluationFIMResults . . . . .	64
getEvaluationInitialDesignResults . . . . .	64
getFim . . . . .	65
getFisherMatrix . . . . .	65
getFixedEffects . . . . .	66

getFixedMu . . . . .	66
getFixedOmega . . . . .	67
getFixedParameters . . . . .	67
getFixedTimes . . . . .	68
getInitialConditions . . . . .	68
getIterationAndCriteria . . . . .	69
getLambda . . . . .	69
getLibraryPDMODELS . . . . .	70
getLibraryPKMODELS . . . . .	70
getMinSampling . . . . .	71
getModel . . . . .	71
getModelEquations . . . . .	72
getModelError . . . . .	72
getModelErrorParametersValues . . . . .	73
getModelFromLibrary . . . . .	73
getModelParameters . . . . .	74
getModelParametersValues . . . . .	74
getMu . . . . .	75
getName . . . . .	75
getNames . . . . .	76
getNumberOfArms . . . . .	77
getNumberOfIterations . . . . .	77
getNumberOfParameters . . . . .	78
getNumberOfSamplingsOptimisable . . . . .	78
getNumberOfTimesByWindows . . . . .	79
getOdeSolverParameters . . . . .	79
getOmega . . . . .	80
getOptimalDesign . . . . .	80
getOptimalFrequencies . . . . .	81
getOptimalWeights . . . . .	81
getOptimizationResults . . . . .	82
getOptimizer . . . . .	82
getOptimizerParameters . . . . .	83
getOutcome . . . . .	83
getOutcomes . . . . .	84
getOutcomesEvaluation . . . . .	84
getOutcomesForEvaluation . . . . .	85
getOutcomesGradient . . . . .	85
getParameters . . . . .	86
getPDMODEL . . . . .	87
getPKMODEL . . . . .	87
getPKPDMODEL . . . . .	88
getPlotOptions . . . . .	88
getProportionsOfSubjects . . . . .	89
getRSE . . . . .	89
getSamplings . . . . .	90
getSamplingsWindows . . . . .	90
getSamplingTime . . . . .	91

getSamplingTimeConstraint . . . . .	91
getSamplingTimes . . . . .	92
getSamplingTimesConstraints . . . . .	92
getSE . . . . .	93
getShrinkage . . . . .	93
getSigmaInter . . . . .	94
getSigmaSlope . . . . .	95
getSize . . . . .	95
getTau . . . . .	96
getTimeDose . . . . .	96
getTinf . . . . .	97
getVariables . . . . .	97
getVarianceEffects . . . . .	98
getWeightThreshold . . . . .	98
IndividualFim-class . . . . .	99
initialize,Administration-method . . . . .	99
initialize,AdministrationConstraints-method . . . . .	100
initialize,Arm-method . . . . .	100
initialize,Combined1-method . . . . .	101
initialize,Constant-method . . . . .	102
initialize,Design-method . . . . .	103
initialize,Distribution-method . . . . .	104
initialize,Evaluation-method . . . . .	104
initialize,FedorovWynnAlgorithm-method . . . . .	105
initialize,Fim-method . . . . .	106
initialize,LibraryOfModels-method . . . . .	106
initialize,LibraryOfPKPDMODELS-method . . . . .	107
initialize,LogNormal-method . . . . .	107
initialize,Model-method . . . . .	108
initialize,ModelAnalytic-method . . . . .	109
initialize,ModelAnalyticBolus-method . . . . .	109
initialize,ModelAnalyticBolusSteadyState-method . . . . .	110
initialize,ModelAnalyticInfusion-method . . . . .	111
initialize,ModelAnalyticInfusionSteadyState-method . . . . .	112
initialize,ModelAnalyticSteadyState-method . . . . .	113
initialize,ModelBolus-method . . . . .	113
initialize,ModelError-method . . . . .	114
initialize,ModelInfusion-method . . . . .	115
initialize,ModelParameter-method . . . . .	116
initialize,MultiplicativeAlgorithm-method . . . . .	117
initialize,Normal-method . . . . .	118
initialize,Optimization-method . . . . .	118
initialize,OptimizationAlgorithm-method . . . . .	119
initialize,PFIMProject-method . . . . .	120
initialize,PGBOAlgorithm-method . . . . .	120
initialize,Proportional-method . . . . .	121
initialize,PSOAlgorithm-method . . . . .	122
initialize,SamplingTimeConstraints-method . . . . .	123

initialize,SamplingTimes-method . . . . .	124
initialize,SimplexAlgorithm-method . . . . .	125
isDoseInEquations . . . . .	126
isModelAnalytic . . . . .	126
isModelBolus . . . . .	127
isModelInfusion . . . . .	127
isModelODE . . . . .	128
isModelSteadyState . . . . .	128
LibraryOfModels-class . . . . .	129
LibraryOfPDMModels . . . . .	129
LibraryOfPKModels . . . . .	129
LibraryOfPKPDMModels-class . . . . .	130
LogNormal-class . . . . .	130
Model-class . . . . .	130
ModelAnalytic-class . . . . .	131
ModelAnalyticBolus-class . . . . .	131
ModelAnalyticBolusSteadyState-class . . . . .	131
ModelAnalyticInfusion-class . . . . .	131
ModelAnalyticInfusionSteadyState-class . . . . .	132
ModelAnalyticSteadyState-class . . . . .	132
ModelBolus-class . . . . .	132
ModelError-class . . . . .	132
ModelInfusion-class . . . . .	132
ModelODE-class . . . . .	133
ModelODEDoseInEquations-class . . . . .	133
ModelODEDoseNotInEquations-class . . . . .	133
ModelODEInfusion-class . . . . .	133
ModelODEInfusionDoseInEquations-class . . . . .	134
ModelParameter-class . . . . .	134
MultiplicativeAlgorithm-class . . . . .	134
MultiplicativeAlgorithm_Rcpp . . . . .	135
Normal-class . . . . .	136
Optimization-class . . . . .	136
OptimizationAlgorithm-class . . . . .	137
optimize . . . . .	137
parametersForComputingGradient . . . . .	138
PFIMProject-class . . . . .	139
PGBOAlgorithm-class . . . . .	139
plotEvaluation . . . . .	140
PlotEvaluation-class . . . . .	140
plotFrequencies . . . . .	140
plotOutcomesEvaluation . . . . .	141
plotOutcomesGradient . . . . .	142
plotRSE . . . . .	142
plotSE . . . . .	143
plotSensitivityIndice . . . . .	143
plotShrinkage . . . . .	144
plotWeights . . . . .	144

PopulationFim-class . . . . .	145
Proportional-class . . . . .	145
PSOAlgorithm-class . . . . .	146
Report . . . . .	146
reportTablesAdministration . . . . .	147
reportTablesDesign . . . . .	147
reportTablesFIM . . . . .	148
reportTablesModelError . . . . .	149
reportTablesModelParameters . . . . .	149
reportTablesPlot . . . . .	150
reportTablesSamplingConstraints . . . . .	150
resizeFisherMatrix . . . . .	151
run . . . . .	151
SamplingTimeConstraints-class . . . . .	152
SamplingTimes-class . . . . .	152
setAdministrations . . . . .	153
setArm . . . . .	153
setArms . . . . .	154
setcError . . . . .	154
setContent . . . . .	155
setDataForArmEvaluation . . . . .	155
setDataForModelEvaluation . . . . .	156
setDerivatives . . . . .	157
setDescription . . . . .	157
setDesigns . . . . .	158
setDistribution . . . . .	158
setDose . . . . .	159
setEquation . . . . .	159
setEquations . . . . .	160
setEquationsAfterInfusion . . . . .	160
setEquationsDuringInfusion . . . . .	161
setEvaluationFIMResults . . . . .	161
setEvaluationInitialDesignResults . . . . .	162
setFim . . . . .	162
setFimTypeToString . . . . .	163
setFisherMatrix . . . . .	163
setFixedEffects . . . . .	164
setFixedMu . . . . .	164
setFixedOmega . . . . .	165
setInitialConditions . . . . .	165
setIterationAndCriteria . . . . .	166
setModel . . . . .	166
setModelError . . . . .	167
setModelFromLibrary . . . . .	167
setMu . . . . .	168
setName . . . . .	168
setNumberOfArms . . . . .	169
setOdeSolverParameters . . . . .	169

setOmega . . . . .	170
setOptimalDesign . . . . .	171
setOptimalWeights . . . . .	171
setOptimizationResults . . . . .	172
setOutcome . . . . .	172
setOutcomes . . . . .	173
setOutcomesEvaluation . . . . .	173
setOutcomesForEvaluation . . . . .	174
setOutcomesGradient . . . . .	174
setParameters . . . . .	175
setSamplingConstraintForOptimization . . . . .	176
setSamplings . . . . .	176
setSamplingTime . . . . .	177
setSamplingTimes . . . . .	177
setSamplingTimesConstraints . . . . .	178
setShrinkage . . . . .	178
setSigmaInter . . . . .	179
setSigmaSlope . . . . .	179
setSize . . . . .	180
setTau . . . . .	181
setTimeDose . . . . .	181
setTinf . . . . .	182
setVarianceEffects . . . . .	182
show,Design-method . . . . .	183
SimplexAlgorithm-class . . . . .	184

**Index****185**


---

PFIM-package	<i>Fisher Information matrix for design evaluation/optimization for non-linear mixed effects models.</i>
--------------	--

---

**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](https://doi.org/10.1093/biomet/84.2.429), Retout S, Comets E, Samson A, Mentré F (2007) doi: [10.1002/sim.2910](https://doi.org/10.1002/sim.2910), Bazzoli C, Retout S, Mentré F (2009) doi: [10.1002/sim.3573](https://doi.org/10.1002/sim.3573), Le Nagard H, Chao L, Tenaillon O (2011) doi: [10.1186/1471214811326](https://doi.org/10.1186/1471214811326), Combes FP, Retout S, Frey N, Mentré F (2013) doi: [10.1007/s11095-01310793](https://doi.org/10.1007/s11095-01310793) and Seurat J, Tang Y, Mentré F, Nguyen TT (2021) doi: [10.1016/j.cmpb.2021.106126](https://doi.org/10.1016/j.cmpb.2021.106126).

**Description**

Nonlinear mixed effects models (NLMEM) are widely used in model-based drug development and use to analyze longitudinal data. The use of the "population" Fisher Information Matrix (FIM) is a good alternative to clinical trial simulation to optimize the design of these studies. The present version, **PFIM** 6.1, 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 I<sub>max</sub>) 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 6.1**, 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 6.1** 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 6.1** 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 6.1** 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 6.1** 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`
- `getLibraryPDMModels`

#### Class `LibraryOfPKPDMModels`

- `getPKModel`
- `getPDMModel`
- `getPKPDMModel`

#### Class `LogNormal`

- `getAdjustedGradient`

#### Class `Model`

- `getName`
- `setName`
- `getDescription`
- `setDescription`
- `getEquations`
- `setEquations`
- `setModelFromLibrary`
- `getOutcomes`
- `setOutcomes`
- `getOutcomesForEvaluation`
- `setOutcomesForEvaluation`
- `getParameters`
- `setParameters`
- `getModelError`
- `setModelError`
- `getInitialConditions`
- `setInitialConditions`
- `getOdeSolverParameters`
- `setOdeSolverParameters`
- `getModelFromLibrary`
- `convertPKModelAnalyticToPKModelODE`
- `getNumberOfParameters`

- `isModelODE`
- `isModelAnalytic`
- `isDoseInEquations`
- `isModelInfusion`
- `isModelSteadyState`
- `isModelBolus`
- `definePKPDMModel`
- `definePKModel`
- `defineModel`
- `defineModelFromLibraryOfModels`
- `defineModelUserDefined`
- `defineModelType`
- `EvaluateModel`
- `parametersForComputingGradient`
- `EvaluateVarianceModel`
- `getFixedParameters`
- `getModelErrorParametersValues`
- `reportTablesModelParameters`
- `reportTablesModelError`

#### Class `ModelAnalytic`

- `EvaluateModel`
- `definePKModel`
- `definePKPDMModel`
- `convertPKModelAnalyticToPKModelODE`

#### Class `ModelAnalyticBolus`

- See class `ModelAnalytic`

#### Class `ModelAnalyticBolusSteadyState`

- See class `ModelAnalyticBolus`

#### Class `ModelBolus`

- See class `Model`

#### Class `ModelError`

- `getOutcome`
- `getEquation`

- `setEquation`
- `getDerivatives`
- `setDerivatives`
- `getSigmaInter`
- `setSigmaInter`
- `getSigmaSlope`
- `setSigmaSlope`
- `getcError`
- `setcError`
- `getParameters`
- `EvaluateErrorModelDerivatives`

#### Class `ModelInfusion`

- `getEquationsDuringInfusion`
- `getEquationsAfterInfusion`
- `setEquationsAfterInfusion`
- `setEquationsDuringInfusion`

#### Class `ModelODE`

- See class `Model`

#### Class `ModelODEBolus`

- `EvaluateModel`
- `definePKPDMModel`

#### Class `ModelODEDoseInEquations`

- `EvaluateModel`
- `definePKModel`
- `definePKPDMModel`

#### Class `ModelODEDoseNotInEquations`

- `EvaluateModel`
- `definePKModel`
- `definePKPDMModel`

#### Class `ModelODEInfusion`

- See class `ModelInfusion`

#### Class `ModelODEInfusionDoseInEquations`



- EvaluateModel
- definePKModel
- definePKPDMModel

#### Class ModelParameter

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

#### Class MultiplicativeAlgorithm

- MultiplicativeAlgorithm\_Rcpp
- getLambda
- getDelta
- getNumberOfIterations
- getOptimalWeights
- setOptimalWeights
- setParameters
- optimize
- getDataFrameResults
- plotWeights
- getWeightThreshold
- 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 `PGB0Algorithm`

- `setParameters`
- `optimize`
- `generateReportOptimization`

#### Class `PlotEvaluation`

- `plot`
- `plotSE`
- `plotRSE`
- `plotShrinkage`

#### Class `PopulationFim`

- EvaluateFisherMatrix
- EvaluateVarianceFIM
- getRSE
- getShrinkage
- setShrinkage
- reportTablesFIM
- computeVMat
- 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>

**References**

- [1] Dumont C, Lestini G, Le Nagard H, Mentré F, Comets E, Nguyen TT, et al. PFIM 4.0, an extended R program for design evaluation and optimization in nonlinear mixed-effect models. *Comput Methods Programs Biomed.* 2018;156:217-29.
- [2] Chambers JM. Object-Oriented Programming, Functional Programming and R. *Stat Sci.* 2014;29:167-80.
- [3] Mentré F, Mallet A, Baccar D. Optimal Design in Random-Effects Regression Models. *Biometrika.* 1997;84:429-42.
- [4] Combes FP, Retout S, Frey N, Mentré F. Prediction of shrinkage of individual parameters using the Bayesian information matrix in nonlinear mixed effect models with evaluation in pharmacokinetics. *Pharm Res.* 2013;30:2355-67.
- [5] Nelder JA, Mead R. A simplex method for function minimization. *Comput J.* 1965;7:308-13.
- [6] Seurat J, Tang Y, Mentré F, Nguyen, TT. Finding optimal design in nonlinear mixed effect models using multiplicative algorithms. *Computer Methods and Programs in Biomedicine*, 2021.
- [7] Fedorov VV. *Theory of Optimal Experiments.* Academic Press, New York, 1972.
- [8] Eberhart RC, Kennedy J. A new optimizer using particle swarm theory. *Proc. of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, 4-6 October 1995, 39-43.
- [9] Le Nagard H, Chao L, Tenaillon O. The emergence of complexity and restricted pleiotropy in adapting networks. *BMC Evol Biol.* 2011;11:326.
- [10] Wickham H. *ggplot2: Elegant Graphics for Data Analysis*, Springer-Verlag New York, 2016.

**See Also**

Useful links:

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

---

addModel	<i>Add a model to a library of models.</i>
----------	--

---

**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 <a href="#">LibraryOfModels</a> .
model	An object from the class <a href="#">Model</a> .

**Value**

The library of models with the added model.

---

addModels	<i>Add a models to a library of models.</i>
-----------	---

---

**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 <a href="#">LibraryOfModels</a> .
models	A list of object from the class <a href="#">Model</a> .

**Value**

The library of models with the added models.

---

Administration-class    *Class "Administration"*

---

### Description

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

### Objects from the class

Objects from 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 from 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 from 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 <a href="#">SamplingTimeConstraints</a> .
arm	An object from the class <a href="#">Arm</a> .
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
  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 <a href="#">Design</a> .
--------	---



**Value**

An error message if a constraint is not valid.

---

Combined1-class	Class "Combined1"
-----------------	-------------------

---

**Description**

The class Combined1 defines the the residual error variance according to the formula  $g(\text{sigmaInter}, \text{sigmaSlope}, \text{cError}, f(x, \text{theta})) = \text{sigmaInter} + \text{sigmaSlope} * f(x, \text{theta})$ . 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 exponant c of the error model.

---

computeVMat	<i>function computeVMat</i>
-------------	-----------------------------

---

**Description**

function computeVMat

**Usage**

```
computeVMat(varParam1, varParam2, invCholV)
```

**Arguments**

varParam1	varParam1
-----------	-----------

varParam2	varParam2
-----------	-----------

invCholV	invCholV
----------	----------

**Value**

VMat

---

Constant-class	<i>Class "Constant"</i>
----------------	-------------------------

---

### Description

The class Constant defines the the residual error variance according to the formula  $g(\text{sigma\_inter}, \text{sigma\_slope}, \text{c\_error}, f(x, \text{theta})) = \text{sigma\_inter}$ . 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	<i>Convert an analytic model to a ode model.</i>
------------------------------------	--

---

### 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.

---

dataForArmEvaluation    *dataForArmEvaluation*

---

**Description**

dataForArmEvaluation

**Usage**

```
dataForArmEvaluation(object, arm, model)
```

```
## S4 method for signature 'Design'
dataForArmEvaluation(object, arm, model)
```

**Arguments**

object	An object Design from the class <a href="#">Design</a> .
arm	...
model	An object Model from the class <a href="#">Model</a> .

**Value**

A list containing data for arm evaluation in the design.

---

defineModel            *Define a model.*

---

**Description**

Define a model.

**Usage**

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

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

**Arguments**

object	An object from the class <a href="#">Model</a> .
designs	A list of objects from the class <a href="#">Design</a> .

**Value**

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

---

```
defineModelEquationsFromStringToFunction  
    defineModelEquationsFromStringToFunction
```

---

**Description**

defineModelEquationsFromStringToFunction

**Usage**

```
defineModelEquationsFromStringToFunction(  
    object,  
    parametersNames,  
    outcomesWithAdministration,  
    outcomesWithNoAdministration  
)  
  
## S4 method for signature 'ModelAnalytic'  
defineModelEquationsFromStringToFunction(  
    object,  
    parametersNames,  
    outcomesWithAdministration,  
    outcomesWithNoAdministration  
)  
  
## S4 method for signature 'ModelAnalyticSteadyState'  
defineModelEquationsFromStringToFunction(  
    object,  
    parametersNames,  
    outcomesWithAdministration,  
    outcomesWithNoAdministration  
)  
  
## S4 method for signature 'ModelAnalyticInfusion'  
defineModelEquationsFromStringToFunction(  
    object,  
    parametersNames,  
    outcomesWithAdministration,  
    outcomesWithNoAdministration  
)  
  
## S4 method for signature 'ModelAnalyticInfusionSteadyState'  
defineModelEquationsFromStringToFunction(  
    object,  
    parametersNames,  
    outcomesWithAdministration,  
    outcomesWithNoAdministration  
)
```

```

    object,
    parametersNames,
    outcomesWithAdministration,
    outcomesWithNoAdministration
)

## S4 method for signature 'ModelODEBolus'
defineModelEquationsFromStringToFunction(
  object,
  parametersNames,
  outcomesWithAdministration,
  outcomesWithNoAdministration
)

## S4 method for signature 'ModelODEDoseInEquations'
defineModelEquationsFromStringToFunction(
  object,
  parametersNames,
  outcomesWithAdministration,
  outcomesWithNoAdministration
)

## S4 method for signature 'ModelODEDoseNotInEquations'
defineModelEquationsFromStringToFunction(
  object,
  parametersNames,
  outcomesWithAdministration,
  outcomesWithNoAdministration
)

## S4 method for signature 'ModelODEInfusionDoseInEquations'
defineModelEquationsFromStringToFunction(
  object,
  parametersNames,
  outcomesWithAdministration,
  outcomesWithNoAdministration
)

```

### Arguments

**object**            An object from the class [Model](#).  
**parametersNames**            Vector of parameter names.  
**outcomesWithAdministration**            Vector of the name of the outcome with administration.  
**outcomesWithNoAdministration**            Vector of the name of the outcome with no administration.

**Value**

....

---

```
defineModelFromLibraryOfModels
    Define a model from the library of models.
```

---

**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 <a href="#">Model</a> .
designs	A list of objects from the class <a href="#">Design</a> .

**Value**

A model defined from the library of models.

---

```
defineModelType    Define the type of a model.
```

---

**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 <a href="#">Model</a> .
designs	A list of objects from the class <a href="#">Design</a> .

**Value**

Return a model defined as analytic, ode, etc.

---

```
defineModelUserDefined
```

*Define a user defined model.*

---

**Description**

Define a user defined model.

**Usage**

```
defineModelUserDefined(object, designs)

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

**Arguments**

object	An object from the class <a href="#">Model</a> .
designs	A list of objects from the class <a href="#">Design</a> .

**Value**

A model giving a user defined model.

---

```
definePKModel
```

*Define a PK model.*

---

**Description**

Define a PK model.

**Usage**

```
definePKModel(object, outcomes)

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

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

## S4 method for signature 'ModelAnalyticInfusion'
```

```

definePKModel(object, outcomes)

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

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

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

```

### Arguments

object            An object from the class [Model](#).

outcomes         A list giving the outcomes of the PK model.

### Value

A model giving a PK model.

---

definePKPDMModel	<i>Define a PKPD model.</i>
------------------	-----------------------------

---

### Description

Define a PKPD model.

### Usage

```

definePKPDMModel(PKModel, PDModel, outcomes)

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

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

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

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

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

## S4 method for signature 'ModelAnalyticInfusion,ModelODE'

```



```

definePKPDMoDel(PKMoDel, PDMoDel, outcomes)

## S4 method for signature 'MoDelODEBolus,MoDelODE'
definePKPDMoDel(PKMoDel, PDMoDel, outcomes)

## S4 method for signature 'MoDelODEDoseInEquations,MoDelODE'
definePKPDMoDel(PKMoDel, PDMoDel, outcomes)

## S4 method for signature 'MoDelODEDoseNotInEquations,MoDelODE'
definePKPDMoDel(PKMoDel, PDMoDel, outcomes)

## S4 method for signature 'MoDelODEInfusion,MoDelODE'
definePKPDMoDel(PKMoDel, PDMoDel, outcomes)

## S4 method for signature 'MoDelODEInfusionDoseInEquations,MoDelODE'
definePKPDMoDel(PKMoDel, PDMoDel, outcomes)

```

### Arguments

PKMoDel	An object from the class <a href="#">MoDel</a> .
PDMoDel	An object from the class <a href="#">MoDel</a> .
outcomes	A list giving the outcomes of the PKPD model.

### Value

A model giving a PKPD model.

---

Design-class	<i>Class "Design"</i>
--------------	-----------------------

---

### Description

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

### Objects from the class

Objects from 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 from 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, dataForModelEvaluation, fim)
```

```
## S4 method for signature 'Arm'
```

```
EvaluateArm(object, model, dataForModelEvaluation, fim)
```

### Arguments

object	An object arm from the class <a href="#">Arm</a> .
model	An object model from the class <a href="#">Model</a> .
dataForModelEvaluation	
	....
fim	An object fim from the class <a href="#">Fim</a> .

### 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	<i>EvaluateDesign</i>
----------------	-----------------------

---

**Description**

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 <a href="#">Design</a> .
model	An object model from the class <a href="#">Model</a> .
fim	An object fim from the class <a href="#">Fim</a> .

**Value**

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

---

EvaluateErrorModelDerivatives	<i>Evaluate the error model derivatives.</i>
-------------------------------	--

---

**Description**

Evaluate the error model derivatives.

**Usage**

```
EvaluateErrorModelDerivatives(object, evaluationOutcome)
```

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

**Arguments**

object	An object from the class <a href="#">ModelError</a> .
evaluationOutcome	A list giving the results of the model evaluation.

**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 <a href="#">Fim</a> .
model	An object from the class <a href="#">Model</a> .
arm	An object from the class <a href="#">Arm</a> .
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.

---

EvaluateModel	<i>Evaluate a model.</i>
---------------	--------------------------

---

### Description

Evaluate a model.

### Usage

```
EvaluateModel(object, dataForModelEvaluation, arm)

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

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

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

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

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

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

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

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

### Arguments

object	An object from the class <a href="#">Model</a> .
dataForModelEvaluation	...
arm	An object from the class <a href="#">Arm</a> .

### Value

A list giving the results of the model evaluation.

---

EvaluateModelGradient *Evaluate model gradient.*

---

### Description

Evaluate model gradient.

### Usage

```
EvaluateModelGradient(object, dataForModelEvaluation, arm)
```

```
## S4 method for signature 'ModelAnalytic'
```

```
EvaluateModelGradient(object, dataForModelEvaluation, arm)
```

```
## S4 method for signature 'ModelAnalyticSteadyState'
```

```
EvaluateModelGradient(object, dataForModelEvaluation, arm)
```

```
## S4 method for signature 'ModelAnalyticInfusion'
```

```
EvaluateModelGradient(object, dataForModelEvaluation, arm)
```

```
## S4 method for signature 'ModelAnalyticInfusionSteadyState'
```

```
EvaluateModelGradient(object, dataForModelEvaluation, arm)
```

```
## S4 method for signature 'ModelODEBolus'
```

```
EvaluateModelGradient(object, dataForModelEvaluation, arm)
```

```
## S4 method for signature 'ModelODEDoseInEquations'
```

```
EvaluateModelGradient(object, dataForModelEvaluation, arm)
```

```
## S4 method for signature 'ModelODEDoseNotInEquations'
```

```
EvaluateModelGradient(object, dataForModelEvaluation, arm)
```

```
## S4 method for signature 'ModelODEInfusionDoseInEquations'
```

```
EvaluateModelGradient(object, dataForModelEvaluation, arm)
```

### Arguments

object            An object from the class [Model](#).

dataForModelEvaluation

...

arm                An object from the class [Arm](#).

### Value

A list giving the results of the model evaluation.

---

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

---

**Description**

Evaluate the variance of the Fisher information matrix.

**Usage**

```
EvaluateVarianceFIM(object, model, arm, modelEvaluation, modelVariance)
```

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

```
EvaluateVarianceFIM(object, model, arm, modelEvaluation, modelVariance)
```

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

```
EvaluateVarianceFIM(object, model, arm, modelEvaluation, modelVariance)
```

**Arguments**

object            An object from the class [Fim](#).  
model            An object from the class [Model](#).  
arm               An object from the class [Arm](#).  
modelEvaluation    A list containing the evaluation results.  
modelVariance    A list containing the model variance.

**Value**

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

---

EvaluateVarianceModel *Evaluate the variance of a model.*

---

**Description**

Evaluate the variance of a model.

**Usage**

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

```
## S4 method for signature 'Model'
```

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

**Arguments**

object	An object from the class <a href="#">Model</a> .
arm	An object from the class <a href="#">Arm</a> .
evaluationModel	A list giving the outputs of the model evaluation.
data	...

**Value**

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

---

Evaluation-class	<i>Class "Evaluation"</i>
------------------	---------------------------

---

**Description**

A class storing information concerning the evaluation of a design.

**Objects from the class**

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

### Slots for FedorovWynnAlgorithm objects

**elementaryProtocols:** A list of vector for the initial elementary protocols.

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

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

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

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

**FisherMatrix:** A vector giving the Fisher Information

**optimalFrequencies:** A vector of the optimal frequencies.

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

**optimalDoses:** A vector for the optimal doses.

---

FedorovWynnAlgorithm\_Rcpp

*Fedorov-Wynn algorithm in Rcpp.*

---

### Description

Run the FedorovWynnAlgorithm in Rcpp

### Usage

```
FedorovWynnAlgorithm_Rcpp(  
  protocols_input,  
  ndimen_input,  
  nbprot_input,  
  numprot_input,  
  freq_input,  
  nbdata_input,  
  vectps_input,
```

```

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

### Slots for Fim objects

**fisherMatrix:** A matrix giving the Fisher matrix.

**fixedEffects:** A matrix giving the fixed effects of the Fisher matrix.

**varianceEffects:** A matrix giving the variance effects of the Fisher matrix.

**shrinkage:** A vector giving the shrinkage value of the parameters.

---

fisher.simplex	<i>Compute the fisher.simplex</i>
----------------	-----------------------------------

---

**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 <a href="#">Optimization</a> .
outcomes	A vector giving the outcomes of the arms.

**Value**

A list giving the results of the optimization.

---

fun.amoeba	<i>function fun.amoeba</i>
------------	----------------------------

---

**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.

**Value**

A list containing the components of the optimized simplex. `'getColumnAndParametersNames-FIMInLatex`.

---

`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`

*Generate the report for the evaluation*

---

**Description**

Generate the report for the evaluation

**Usage**

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

**Arguments**

object	An object from the class <a href="#">Fim</a> .
evaluationObject	A list giving the results of the evaluation of the model.
outputPath	A string giving the output path.
outputFile	A string giving the name of the output file.
plotOptions	A list giving the plot options.

**Value**

Return the report for the evaluation in html.

---

`generateReportOptimization`*Generate report for the optimization.*

---

**Description**

Generate report for the optimization.

**Usage**

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

```

    object,
    optimizationObject,
    outputPath,
    outputFile,
    plotOptions
)

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

```

### Arguments

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

### Value

The report for the optimization in html.

---

```

generateSamplingsFromSamplingConstraints
Generate samplings from sampling constraints

```

---

### Description

Generate samplings from sampling constraints

### Usage

```

generateSamplingsFromSamplingConstraints(object)

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

```

### Arguments

object            An object from the class [SamplingTimeConstraints](#).

**Value**

A list of sampling times generated from the sampling constraints.

---

<code>generateTables</code>	<i>Generate the tables for the report.</i>
-----------------------------	--

---

**Description**

Generate the tables for the report.

**Usage**

```
generateTables(object, plotOptions)

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

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

**Arguments**

<code>object</code>	An object from the class <a href="#">PFIMProject</a> .
<code>plotOptions</code>	A list giving the plot options.

**Value**

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

---

<code>getAdjustedGradient</code>	<i>getAdjustedGradient</i>
----------------------------------	----------------------------

---

**Description**

Get the adjusted gradient.

**Usage**

```
getAdjustedGradient(object, outcomesGradient)

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

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



**Arguments**

object            An object distribution from the class [Distribution](#).  
outcomesGradient            A list containing the evaluation of the outcome gradients.

**Value**

A list giving the adjusted gradient.

---

`getAdministration`        *getAdministration*

---

**Description**

Get the administrations by outcome.

**Usage**

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

**Arguments**

object            An object Arm from the class [Arm](#).  
outcome           A string giving the name of the outcome.

**Value**

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

---

`getAdministrationConstraint`  
                          *getAdministrationConstraint*

---

**Description**

Get the administration constraints by outcome.

**Usage**

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

**Arguments**

object            An object Arm from the class [Arm](#).  
 outcome         A string giving the name of the outcome.

**Value**

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

---

`getAdministrations`     *getAdministrations*

---

**Description**

Get all the administration for an arm.

**Usage**

```
getAdministrations(object)

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

**Arguments**

object            An object Arm from the class [Arm](#).

**Value**

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

---

`getAdministrationsConstraints`  
                                   *getAdministrationsConstraints*

---

**Description**

Get the administrations constraints.

**Usage**

```
getAdministrationsConstraints(object)

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

**Arguments**

object            An object Arm from the class [Arm](#).

**Value**

The list administrationsConstraints.

---

getArms            *Get the arms of an object.*

---

**Description**

Get the arms of an object.

**Usage**

```
getArms(object)
```

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

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

**Arguments**

object            An object defined form a class of PFIM.

**Value**

A list containing the arms of the object.

---

getError            *Get the parameter c.*

---

**Description**

Get the parameter c.

**Usage**

```
getError(object)
```

```
## S4 method for signature 'ModelError'  
getError(object)
```

**Arguments**

object            An object from the class [ModelError](#).

**Value**

A numeric giving the parameter c.

---

getColumnAndParametersNamesFIM

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

---

**Description**

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

**Usage**

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

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

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

```
## 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 <a href="#">Fim</a> .
model	An object from the class <a href="#">Model</a> .

### 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)
```

**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.

---

getContent

*Get content of a library of models.*

---

**Description**

Get content of a library of models.

**Usage**

```
getContent(object)
```

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

**Arguments**

object            An object from the class [LibraryOfModels](#).

**Value**

A list giving the content of the library of models.

---

`getCorrelationMatrix`    *Get the correlation matrix.*

---

**Description**

Get the correlation matrix.

**Usage**

```
getCorrelationMatrix(object)

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

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

## S4 method for signature 'Optimization'
getCorrelationMatrix(object)
```

**Arguments**

object            An object from the class [Fim](#).

**Value**

The correlation matrix of the fim.

---

`getDataForArmEvaluation`  
*getDataForArmEvaluation*

---

**Description**

`getDataForArmEvaluation`

**Usage**

```
getDataForArmEvaluation(object)

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

**Arguments**

object            An object Arm from the class [Arm](#).

**Value**

A list containing the data for arm evaluation.

---

getDataFrameResults    *Get the dataframe of the results.*

---

**Description**

Get the dataframe of the results.

**Usage**

```
getDataFrameResults(object)

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

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

## S4 method for signature 'Optimization'
getDataFrameResults(object)
```

**Arguments**

object            An object from the class [OptimizationAlgorithm](#).

**Value**

Return the dataframe of the results.



---

getDcriterion	<i>Get the D criterion of the fim.</i>
---------------	--

---

**Description**

Get the D criterion of the fim.

**Usage**

```
getDcriterion(object)

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

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

## S4 method for signature 'Optimization'
getDcriterion(object)
```

**Arguments**

object            An object from the class [Fim](#).

**Value**

A numeric giving the D criterion of the fim.

---

getDelta	<i>Get the parameter delta</i>
----------	--------------------------------

---

**Description**

Get the parameter delta

**Usage**

```
getDelta(object)

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

**Arguments**

object            An object from the class [MultiplicativeAlgorithm](#).

**Value**

A numeric giving the parameter delta.

---

getDerivatives	<i>Get the derivatives of the model error equation.</i>
----------------	---

---

**Description**

Get the derivatives of the model error equation.

**Usage**

```
getDerivatives(object)

## S4 method for signature 'ModelError'
getDerivatives(object)
```

**Arguments**

object            An object from the class [ModelError](#).

**Value**

The derivatives of the model error equation.

---

getDescription	<i>Get the description of a model.</i>
----------------	--

---

**Description**

Get the description of a model.

**Usage**

```
getDescription(object)

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

**Arguments**

object            An object from the class [Model](#).

**Value**

A list giving the description of a model.

---

getDesigns	<i>Get the designs.</i>
------------	-------------------------

---

**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.

---

getDeterminant	<i>Get the determinant of the fim.</i>
----------------	--

---

**Description**

Get the determinant of the fim.

**Usage**

```
getDeterminant(object)
```

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

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

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

**Arguments**

object            An object from the class [Fim](#).

**Value**

A numeric giving the determinant of the fim.

---

getDistribution	<i>Get the distribution.</i>
-----------------	------------------------------

---

**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.

---

getDose	<i>getDose</i>
---------	----------------

---

**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	<i>Get the eigenvalues of the fim.</i>
----------------	--

---

**Description**

Get the eigenvalues of the fim.

**Usage**

```
getEigenValues(object)

## S4 method for signature 'Fim'
getEigenValues(object)
```

**Arguments**

object            An object from the class [Fim](#).

**Value**

A vector giving the eigenvalues of the fim.

---

getElementaryProtocols	<i>Get the elementary protocols.</i>
------------------------	--------------------------------------

---

**Description**

Get the elementary protocols.

**Usage**

```
getElementaryProtocols(object, fims)

## S4 method for signature 'Optimization'
getElementaryProtocols(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	<i>Get the equation of a model error.</i>
-------------	---

---

**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	<i>Get the equations of a model.</i>
--------------	--------------------------------------

---

**Description**

Get the equations of a model.

**Usage**

```
getEquations(object)
```

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

**Arguments**

object            An object from the class [Model](#).

**Value**

The list giving the equations of the model.

---

```
getEquationsAfterInfusion  
    Get the equations after infusion.
```

---

**Description**

Get the equations after infusion.

**Usage**

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

**Arguments**

object            An object from the class [Model](#).

**Value**

A list giving the equations after the infusion.

---

```
getEquationsDuringInfusion  
    Get the equations during infusion.
```

---

**Description**

Get the equations during infusion.

**Usage**

```
getEquationsDuringInfusion(object)  
  
## S4 method for signature 'Model'  
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

```
getEvaluationFIMResults(object)
```

```
## 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

```
getEvaluationInitialDesignResults(object)
```

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

### Arguments

object            An object from the class [Optimization](#).

### Value

The object from the class [Evaluation](#) giving the results of the evaluation of the initial design.



---

getFim	<i>getFim</i>
--------	---------------

---

**Description**

Get the fim of an 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 form a class of PFIM.

**Value**

The FIM of the object.

---

getFisherMatrix	<i>Get the FIM.</i>
-----------------	---------------------

---

**Description**

Get the FIM.

**Usage**

```
getFisherMatrix(object)
```

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

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

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

**Arguments**

object            An object from the class [Fim](#).

**Value**

A matrix giving the FIM.

---

getFixedEffects            *Get the matrix of fixed effects.*

---

**Description**

Get the matrix of fixed effects.

**Usage**

```
getFixedEffects(object)

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

**Arguments**

object            An object from the class [Fim](#).

**Value**

The matrix of the fixed effects.

---

getFixedMu                *Get the fixed effect.*

---

**Description**

Get the fixed effect.

**Usage**

```
getFixedMu(object)

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

**Arguments**

object            An object from the class [ModelParameter](#).

**Value**

A boolean giving the fixed mu.

---

getFixedOmega	<i>Get the fixed variance.</i>
---------------	--------------------------------

---

**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	<i>Get the fixed parameters.</i>
--------------------	----------------------------------

---

**Description**

Get the fixed parameters.

**Usage**

```
getFixedParameters(object)

## S4 method for signature 'Model'
getFixedParameters(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**

```
getFixedTimes(object)
```

```
## S4 method for signature 'SamplingTimeConstraints'  
getFixedTimes(object)
```

**Arguments**

`object`      An object from the class [SamplingTimeConstraints](#).

**Value**

A vector giving the fixed sampling times.

---

`getInitialConditions`      *getInitialConditions*

---

**Description**

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

**Usage**

```
getInitialConditions(object)
```

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

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

**Arguments**

`object`      An object Arm from the class [Arm](#).

**Value**

The list initialConditions for the object Arm.

---

```
getIterationAndCriteria
```

*Get the iteration with the convergence criteria.*

---

**Description**

Get the iteration with the convergence criteria.

**Usage**

```
getIterationAndCriteria(object)
```

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

**Arguments**

object            An object from the class [OptimizationAlgorithm](#).

**Value**

A dataframe giving the iteration with the convergence criteria.

---

```
getLambda
```

*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	<i>Get the minimal sampling times.</i>
----------------	--

---

**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	<i>Get the model.</i>
----------	-----------------------

---

**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 form 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 the library of models.

getModelParameters     *Get the model parameters.*

---

**Description**

Get the model parameters.

**Usage**

```
getModelParameters(object)

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

**Arguments**

object             An object from the class [PFIMProject](#).

**Value**

A list giving the model parameters.

---

getModelParametersValues  
                          *Get the values of the model parameters.*

---

**Description**

Get the values of the model parameters.

**Usage**

```
getModelParametersValues(object)

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

**Arguments**

object             An object from the class [Model](#).

**Value**

A list giving the values of the model parameters.

---

getMu	<i>getMu</i>
-------	--------------

---

**Description**

Get the fixed effect of an object.

**Usage**

```
getMu(object)

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

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

**Arguments**

object            An object defined form a class of PFIM.

**Value**

The object with the updated fixed effect.

---

getName	<i>getName</i>
---------	----------------

---

**Description**

Get the name of an object

**Usage**

```
getName(object)

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

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

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

## S4 method for signature 'LibraryOfModels'
```

```
getName(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.

---

getNames	<i>getNames</i>
----------	-----------------

---

**Description**

Get the names of an object.

**Usage**

```
getNames(object)

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

**Arguments**

object            An object defined form a class of PFIM.

**Value**

A vector giving the names of the object.

---

`getNumberOfArms`      *getNumberOfArms*

---

**Description**

Get the number of arms in a design.

**Usage**

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

**Arguments**

`object`              An object `Design` from the class [Design](#).

**Value**

A numeric `numberOfArms` giving the number of arms in the design.

---

`getNumberOfIterations`    *Get the number of iterations.*

---

**Description**

Get the number of iterations.

**Usage**

```
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

```
getNumberOfParameters(object)
```

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

### Arguments

object            An object from the class [Model](#).

### Value

A numeric giving the number of parameters of the model.

---

getNumberOfSamplingsOptimisable  
*Get the number of sampling times that are optimisable.*

---

### Description

Get the number of sampling times that are optimisable.

### Usage

```
getNumberOfSamplingsOptimisable(object)
```

```
## S4 method for signature 'SamplingTimeConstraints'
getNumberOfSamplingsOptimisable(object)
```

### Arguments

object            An object from the class [SamplingTimeConstraints](#).

### Value

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

---

`getNumberOfTimesByWindows`*Get the number of sampling times by windows.*

---

**Description**

Get the number of sampling times by windows.

**Usage**

```
getNumberOfTimesByWindows(object)
```

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

**Arguments**

`object`            An object from the class [SamplingTimeConstraints](#).

**Value**

A vector giving the number of sampling times by windows.

---

`getOdeSolverParameters`*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**

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

**Value**

The list giving the parameters for the ode solvers.

---

getOmega	<i>Get the matrix omega of an object.</i>
----------	---

---

**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	<i>Get the optimal design.</i>
------------------	--------------------------------

---

**Description**

Get the optimal design.

**Usage**

```
getOptimalDesign(object)

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

**Arguments**

object            An object from the class [OptimizationAlgorithm](#).



**Value**

The optimal design.

---

`getOptimalFrequencies` *Get the optimal frequencies*

---

**Description**

Get the optimal frequencies

**Usage**

```
getOptimalFrequencies(object)
```

```
## S4 method for signature 'FedorovWynnAlgorithm'  
getOptimalFrequencies(object)
```

**Arguments**

`object` An object from the class [FedorovWynnAlgorithm](#).

**Value**

A vector giving the optimal frequencies

---

`getOptimalWeights` *Get the optimal weights.*

---

**Description**

Get the optimal weights.

**Usage**

```
getOptimalWeights(object)
```

```
## 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

```
getOptimizer(object)
```

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

### Arguments

object            An object from the class [PFIMProject](#).

### Value

A string giving the name of the optimization algorithm.

---

getOptimizerParameters  
*Get the optimization parameters.*

---

**Description**

Get the optimization parameters.

**Usage**

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

**Arguments**

object            An object from the class [PFIMProject](#).

**Value**

A list giving the optimization parameters.

---

getOutcome            *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.

---

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 out-comes).*

---

**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            *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	<i>Get the parameters of an object.</i>
---------------	---

---

**Description**

Get the parameters of an object.

**Usage**

```
getParameters(object)

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

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

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

**Arguments**

object            An object defined form a class of PFIM.

**Value**

Return the list of the parameters of the object.

---

getPDMModel	<i>Get a PD model.</i>
-------------	------------------------

---

**Description**

Get a PD model.

**Usage**

```
getPDMModel(object, PDModelName)
```

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

**Arguments**

object	An object from the class <a href="#">LibraryOfPKPDMModels</a> .
PDModelName	A string giving the name of the PD model.

**Value**

Return a PD model.

---

getPKModel	<i>Get a PK model.</i>
------------	------------------------

---

**Description**

Get a PK model.

**Usage**

```
getPKModel(object, PKModelName)
```

```
## S4 method for signature 'LibraryOfPKPDMModels'  
getPKModel(object, PKModelName)
```

**Arguments**

object	An object from the class <a href="#">LibraryOfPKPDMModels</a> .
PKModelName	A string giving the name of the PK model.

**Value**

Return a PK model.

---

getPKPDModel	<i>Get a PKPD model.</i>
--------------	--------------------------

---

**Description**

Get a PKPD model.

**Usage**

```
getPKPDModel(object, namesModel)
```

```
## S4 method for signature 'LibraryOfPKPDModels'  
getPKPDModel(object, namesModel)
```

**Arguments**

object	An object from the class <a href="#">LibraryOfPKPDModels</a> .
namesModel	A vector of strings giving the names of the PK and PD models.

**Value**

Return a PKPD model.

---

getPlotOptions	<i>Get the plot options for graphs responses and SI</i>
----------------	---

---

**Description**

Get the plot options for graphs responses and SI

**Usage**

```
getPlotOptions(plotOptions, outcomesNames)
```

**Arguments**

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

**Value**

The list containing the plot options.



---

getProportionsOfSubjects  
*Get the proportion of subjects.*

---

**Description**

Get the proportion of subjects.

**Usage**

```
getProportionsOfSubjects(object)

## S4 method for signature 'Optimization'
getProportionsOfSubjects(object)
```

**Arguments**

object            An object from the class [Optimization](#).

**Value**

A vector giving the proportion of subjects.

---

getRSE            *Get the RSE*

---

**Description**

Get the RSE

**Usage**

```
getRSE(object, model)

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

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

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

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

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

**Arguments**

object            An object from the class [Fim](#).  
 model            An object from the class [Model](#).

**Value**

A vector giving the RSE.

---

getSamplings            *Get the sampling of an object.*

---

**Description**

Get the sampling of an object.

**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)
```

**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            *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`            *getSamplingTimes*

---

**Description**

Get the vectors of sampling times for an arm.

**Usage**

```
getSamplingTimes(object)

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

**Arguments**

object            An object Arm from the class [Arm](#).

**Value**

The list `samplingTimes` for the object Arm.

---

`getSamplingTimesConstraints`  
                                   *getSamplingTimesConstraints*

---

**Description**

Get the sampling times constraints.

**Usage**

```
getSamplingTimesConstraints(object)

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

**Arguments**

object            An object Arm from the class [Arm](#).

**Value**

The list getSamplingTimesConstraints.

---

getSE            *Get the SE.*

---

**Description**

Get the SE.

**Usage**

```
getSE(object)

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

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

## S4 method for signature 'Optimization'
getSE(object)
```

**Arguments**

object            An object from the class [Fim](#).

**Value**

A vector giving the SE.

---

getShrinkage    *Get the shrinkage.*

---

**Description**

Get the shrinkage.

**Usage**

```

getShrinkage(object)

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

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

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

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

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

```

**Arguments**

object            An object from the class [Fim](#).

**Value**

A vector giving the shrinkage of the Bayesian fim.

---

<code>getSigmaInter</code>	<i>Get the parameter sigma inter.</i>
----------------------------	---------------------------------------

---

**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	<i>Get the parameter sigma slope.</i>
---------------	---------------------------------------

---

**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.

---

getSize	<i>getSize</i>
---------	----------------

---

**Description**

Get the size of an object.

**Usage**

```
getSize(object)

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

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

**Arguments**

object            An object defined form a class of PFIM.

**Value**

A numeric giving the size of the object.

---

getTau	<i>getTau</i>
--------	---------------

---

**Description**

Get the frequency tau.

**Usage**

```
getTau(object)
```

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

**Arguments**

object            An object Administration from the class [Administration](#).

**Value**

The numeric tau giving the frequency tau.

---

getTimeDose	<i>getTimeDose</i>
-------------	--------------------

---

**Description**

Get the times vector when doses are given.

**Usage**

```
getTimeDose(object)
```

```
## 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	<i>Get the infusion duration.</i>
---------	-----------------------------------

---

**Description**

Get the infusion duration.

**Usage**

```
getTinf(object)

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

**Arguments**

object            An object Administration from the class [Administration](#).

**Value**

The numeric Tinf giving the infusion duration Tinf.

---

getVariables	<i>Return the variable of an ode model</i>
--------------	--

---

**Description**

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

**Usage**

```
getVariables(object)

## S4 method for signature 'ModelODE'
getVariables(object)

## S4 method for signature 'ModelODEBolus'
getVariables(object)

## S4 method for signature 'ModelInfusion'
getVariables(object)
```

**Arguments**

object            An object from the class [Model](#).

**Value**

Return the variable of an ode model

---

`getVarianceEffects`      *Get the matrix of the variance effects.*

---

**Description**

Get the matrix of the variance effects.

**Usage**

```
getVarianceEffects(object)

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

**Arguments**

`object`              An object from the class [Fim](#).

**Value**

The matrix of the variance effects.

---

`getWeightThreshold`      *Get the parameter weightThreshold*

---

**Description**

Get the parameter `weightThreshold`

**Usage**

```
getWeightThreshold(object)

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

**Arguments**

`object`              An object from the class [MultiplicativeAlgorithm](#).

**Value**

A numeric giving the `WeightThreshold`.

---

IndividualFim-class    *Class "Fim"*

---

### Description

A class storing information regarding the individual Fisher matrix. The class IndividualFim inherits from the class Fim.

---

initialize,Administration-method  
*initialize*

---

### Description

initialize

### Usage

```
## S4 method for signature 'Administration'
initialize(.Object, outcome, timeDose, dose, Tinf, tau)
```

### Arguments

.Object	.Object
outcome	outcome
timeDose	timeDose
dose	dose
Tinf	Tinf
tau	tau

### Value

Administration

---

initialize,AdministrationConstraints-method  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'AdministrationConstraints'
initialize(.Object, outcome, doses)
```

**Arguments**

.Object	.Object
outcome	outcome
doses	doses

---

initialize,Arm-method *initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'Arm'
initialize(
  .Object,
  name,
  size,
  administrations,
  initialConditions,
  samplingTimes,
  administrationsConstraints,
  samplingTimesConstraints,
  dataForArmEvaluation
)
```

**Arguments**

.Object	.Object
name	name
size	size
administrations	administrations
initialConditions	initialConditions
samplingTimes	samplingTimes
administrationsConstraints	administrationsConstraints
samplingTimesConstraints	samplingTimesConstraints
dataForArmEvaluation	dataForArmEvaluation

**Value**

Arm

---

*initialize,Combined1-method*  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'Combined1'  
initialize(  
  .Object,  
  outcome,  
  equation,  
  derivatives,  
  sigmaInter,  
  sigmaSlope,  
  cError  
)
```

**Arguments**

.Object	.Object
outcome	outcome
equation	equation
derivatives	derivatives
sigmaInter	sigmaInter
sigmaSlope	sigmaSlope
cError	cError

**Value**

Combined1

---

initialize,Constant-method  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'Constant'
initialize(
  .Object,
  outcome,
  equation,
  derivatives,
  sigmaInter,
  sigmaSlope,
  cError
)
```

**Arguments**

.Object	.Object
outcome	outcome
equation	equation
derivatives	derivatives
sigmaInter	sigmaInter
sigmaSlope	sigmaSlope
cError	cError

**Value**

Constant

---

initialize,Design-method  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'Design'  
initialize(  
  .Object,  
  name,  
  size,  
  arms,  
  outcomesEvaluation,  
  outcomesGradient,  
  numberOfArms,  
  fim  
)
```

**Arguments**

.Object	.Object
name	name
size	size
arms	arms
outcomesEvaluation	outcomesEvaluation
outcomesGradient	outcomesGradient
numberOfArms	numberOfArms
fim	fim

**Value**

Design

---

```
initialize,Distribution-method
      initialize
```

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'Distribution'
initialize(.Object, parameters)
```

**Arguments**

.Object	.Object
parameters	parameters

**Value**

Distribution

---

```
initialize,Evaluation-method
      initialize
```

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'Evaluation'
initialize(
  .Object,
  name,
  model,
  modelEquations,
  modelParameters,
  modelError,
  outcomes,
  designs,
  fim,
  odeSolverParameters
)
```



**Arguments**

.Object	.Object
name	name
model	model
modelEquations	modelEquations
modelParameters	modelParameters
modelError	modelError
outcomes	outcomes
designs	designs
fim	fim
odeSolverParameters	odeSolverParameters

**Value**

Evaluation

---

initialize,FedorovWynnAlgorithm-method  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'FedorovWynnAlgorithm'
initialize(
  .Object,
  elementaryProtocols,
  numberOfSubjects,
  proportionsOfSubjects,
  showProcess
)
```

**Arguments**

.Object	.Object
elementaryProtocols	elementaryProtocols
numberOfSubjects	numberOfSubjects
proportionsOfSubjects	proportionsOfSubjects
showProcess	showProcess

**Value**

FedorovWynnAlgorithm

---

*initialize,Fim-method initialize*


---

**Description**

initialize

**Usage**

```
## S4 method for signature 'Fim'
initialize(.Object, fisherMatrix, fixedEffects, varianceEffects, shrinkage)
```

**Arguments**

.Object	.Object
fisherMatrix	fisherMatrix
fixedEffects	fixedEffects
varianceEffects	varianceEffects
shrinkage	shrinkage

**Value**

Fim

---

*initialize,LibraryOfModels-method  
initialize*


---

**Description**

initialize

**Usage**

```
## S4 method for signature 'LibraryOfModels'
initialize(.Object, name, content)
```

**Arguments**

.Object	.Object
name	fisherMatrix
content	fixedEffects

**Value**

LibraryOfModels

---

initialize,LibraryOfPKPDMModels-method  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'LibraryOfPKPDMModels'  
initialize(.Object)
```

**Arguments**

.Object            .Object

**Value**

LibraryOfPKPDMModels

---

initialize,LogNormal-method  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'LogNormal'  
initialize(.Object, ...)
```

**Arguments**

.Object            .Object  
...                args

**Value**

LogNormal

---

```
initialize,Model-method
      initialize
```

---

### Description

initialize

### Usage

```
## S4 method for signature 'Model'
initialize(
  .Object,
  name,
  description,
  equations,
  outcomes,
  outcomesForEvaluation,
  parameters,
  modelError,
  initialConditions,
  odeSolverParameters,
  modelFromLibrary
)
```

### Arguments

.Object	.Object
name	name
description	description
equations	equations
outcomes	outcomes
outcomesForEvaluation	outcomesForEvaluation
parameters	parameters
modelError	modelError
initialConditions	initialConditions
odeSolverParameters	odeSolverParameters
modelFromLibrary	modelFromLibrary

### Value

Model

---

initialize,ModelAnalytic-method  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'ModelAnalytic'  
initialize(  
  .Object,  
  name,  
  description,  
  equations,  
  outcomes,  
  parameters,  
  modelError  
)
```

**Arguments**

.Object	.Object
name	name
description	description
equations	equations
outcomes	outcomes
parameters	parameters
modelError	modelError

**Value**

ModelAnalytic

---

initialize,ModelAnalyticBolus-method  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'ModelAnalyticBolus'
initialize(
  .Object,
  name,
  description,
  equations,
  outcomes,
  parameters,
  modelError
)
```

**Arguments**

.Object	.Object
name	name
description	description
equations	equations
outcomes	outcomes
parameters	parameters
modelError	modelError

**Value**

ModelAnalyticBolus

---

*initialize,ModelAnalyticBolusSteadyState-method*  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'ModelAnalyticBolusSteadyState'
initialize(
  .Object,
  name,
  description,
  equations,
  outcomes,
  parameters,
  modelError
)
```

**Arguments**

.Object	.Object
name	name
description	description
equations	equations
outcomes	outcomes
parameters	parameters
modelError	modelError

**Value**

ModelAnalyticBolusSteadyState

---

initialize,ModelAnalyticInfusion-method  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'ModelAnalyticInfusion'  
initialize(  
  .Object,  
  name,  
  description,  
  equations,  
  outcomes,  
  parameters,  
  modelError  
)
```

**Arguments**

.Object	.Object
name	name
description	description
equations	equations
outcomes	outcomes
parameters	parameters
modelError	modelError

**Value**

ModelAnalyticInfusion

---

*initialize,ModelAnalyticInfusionSteadyState-method*  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'ModelAnalyticInfusionSteadyState'
initialize(
  .Object,
  name,
  description,
  equations,
  outcomes,
  parameters,
  modelError
)
```

**Arguments**

<code>.Object</code>	<code>.Object</code>
<code>name</code>	<code>name</code>
<code>description</code>	<code>description</code>
<code>equations</code>	<code>equations</code>
<code>outcomes</code>	<code>outcomes</code>
<code>parameters</code>	<code>parameters</code>
<code>modelError</code>	<code>modelError</code>

**Value**

ModelAnalyticInfusionSteadyState



---

initialize,ModelAnalyticSteadyState-method  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'ModelAnalyticSteadyState'  
initialize(  
  .Object,  
  name,  
  description,  
  equations,  
  outcomes,  
  parameters,  
  modelError  
)
```

**Arguments**

.Object	.Object
name	name
description	description
equations	equations
outcomes	outcomes
parameters	parameters
modelError	modelError

**Value**

ModelAnalyticSteadyState

---

initialize,ModelBolus-method  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'ModelBolus'  
initialize(  
  .Object,  
  name,  
  description,  
  equations,  
  outcomes,  
  parameters,  
  modelError,  
  initialConditions,  
  odeSolverParameters  
)
```

**Arguments**

<code>.Object</code>	<code>.Object</code>
<code>name</code>	<code>name</code>
<code>description</code>	<code>description</code>
<code>equations</code>	<code>equations</code>
<code>outcomes</code>	<code>outcomes</code>
<code>parameters</code>	<code>parameters</code>
<code>modelError</code>	<code>modelError</code>
<code>initialConditions</code>	<code>initialConditions</code>
<code>odeSolverParameters</code>	<code>odeSolverParameters</code>

**Value**

ModelBolus

---

`initialize,ModelError-method`  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'ModelError'
initialize(
  .Object,
  outcome,
  equation,
  derivatives,
  sigmaInter,
  sigmaSlope,
  cError
)
```

**Arguments**

.Object	.Object
outcome	outcome
equation	equation
derivatives	derivatives
sigmaInter	sigmaInter
sigmaSlope	sigmaSlope
cError	cError

**Value**

ModelError

---

initialize,ModelInfusion-method  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'ModelInfusion'
initialize(
  .Object,
  name,
  description,
  equations,
  outcomes,
  parameters,
  modelError,
```

```

    initialConditions,
    odeSolverParameters
)

```

**Arguments**

.Object	.Object
name	name
description	description
equations	equations
outcomes	outcomes
parameters	parameters
modelError	modelError
initialConditions	initialConditions
odeSolverParameters	odeSolverParameters

**Value**

ModelInfusion

---

*initialize,ModelParameter-method*  
*initialize*

---

**Description**

initialize

**Usage**

```

## S4 method for signature 'ModelParameter'
initialize(.Object, name, distribution, fixedMu, fixedOmega)

```

**Arguments**

.Object	.Object
name	name
distribution	distribution
fixedMu	fixedMu
fixedOmega	fixedOmega

**Value**

ModelParameter

---

`initialize, MultiplicativeAlgorithm-method`  
*initialize*

---

### **Description**

`initialize`

### **Usage**

```
## S4 method for signature 'MultiplicativeAlgorithm'  
initialize(  
  .Object,  
  arms,  
  lambda,  
  delta,  
  numberOfIterations,  
  weightThreshold,  
  optimalWeights,  
  optimalDesign,  
  showProcess  
)
```

### **Arguments**

<code>.Object</code>	<code>.Object</code>
<code>arms</code>	<code>arms</code>
<code>lambda</code>	<code>lambda</code>
<code>delta</code>	<code>delta</code>
<code>numberOfIterations</code>	<code>numberOfIterations</code>
<code>weightThreshold</code>	<code>weightThreshold</code>
<code>optimalWeights</code>	<code>optimalWeights</code>
<code>optimalDesign</code>	<code>optimalDesign</code>
<code>showProcess</code>	<code>showProcess</code>

### **Value**

`MultiplicativeAlgorithm`

---

```
initialize, Normal-method
      initialize
```

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'Normal'
initialize(.Object, ...)
```

**Arguments**

.Object	.Object
...	args

**Value**

Normal

---

```
initialize, Optimization-method
      initialize
```

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'Optimization'
initialize(
  .Object,
  name,
  model,
  modelEquations,
  modelParameters,
  modelError,
  optimizer,
  optimizerParameters,
  outcomes,
  designs,
  fim,
```

```

    odeSolverParameters,
    optimizationResults,
    evaluationFIMResults,
    evaluationInitialDesignResults
)

```

### Arguments

.Object	.Object
name	name
model	model
modelEquations	modelEquations
modelParameters	modelParameters
modelError	modelError
optimizer	optimizer
optimizerParameters	optimizerParameters
outcomes	outcomes
designs	designs
fim	fim
odeSolverParameters	odeSolverParameters
optimizationResults	optimizationResults
evaluationFIMResults	evaluationFIMResults
evaluationInitialDesignResults	evaluationInitialDesignResults

### Value

Optimization

---

```

initialize, OptimizationAlgorithm-method
      initialize

```

---

### Description

initialize

### Usage

```

## S4 method for signature 'OptimizationAlgorithm'
initialize(.Object, name, parameters)

```

**Arguments**

.Object	.Object
name	name
parameters	parameters

**Value**

OptimizationAlgorithm

---

*initialize,PFIMProject-method*  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'PFIMProject'
initialize(.Object, name, description)
```

**Arguments**

.Object	.Object
name	name
description	description

**Value**

PFIMProject

---

*initialize,PGBOAlgorithm-method*  
*initialize*

---

**Description**

initialize



**Usage**

```
## S4 method for signature 'PGBAlgorithm'  
initialize(  
  .Object,  
  N,  
  muteEffect,  
  maxIteration,  
  purgeIteration,  
  seed,  
  showProcess,  
  optimalDesign,  
  iterationAndCriteria  
)
```

**Arguments**

.Object	.Object
N	N
muteEffect	muteEffect
maxIteration	maxIteration
purgeIteration	purgeIteration
seed	seed
showProcess	showProcess
optimalDesign	optimalDesign
iterationAndCriteria	iterationAndCriteria

**Value**

PGBAlgorithm

---

initialize,Proportional-method  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'Proportional'
initialize(
  .Object,
  outcome,
  equation,
  derivatives,
  sigmaInter,
  sigmaSlope,
  cError
)
```

**Arguments**

.Object	.Object
outcome	outcome
equation	equation
derivatives	derivatives
sigmaInter	sigmaInter
sigmaSlope	sigmaSlope
cError	cError

**Value**

Proportional

---

*initialize,PSOAlgorithm-method*  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'PSOAlgorithm'
initialize(
  .Object,
  maxIteration,
  populationSize,
  personalLearningCoefficient,
  globalLearningCoefficient,
  seed,
  showProcess,
```

```

    optimalDesign,
    iterationAndCriteria
)

```

### Arguments

.Object	.Object
maxIteration	maxIteration
populationSize	populationSize
personalLearningCoefficient	personalLearningCoefficient
globalLearningCoefficient	globalLearningCoefficient
seed	seed
showProcess	showProcess
optimalDesign	optimalDesign
iterationAndCriteria	iterationAndCriteria

### Value

PSOAlgorithm

---

```

initialize, SamplingTimeConstraints-method
      initialize

```

---

### Description

initialize

### Usage

```

## S4 method for signature 'SamplingTimeConstraints'
initialize(
  .Object,
  outcome,
  initialSamplings,
  fixedTimes,
  numberOfSamplingsOptimisable,
  samplingsWindows,
  numberOfTimesByWindows,
  minSampling
)

```

**Arguments**

.Object	.Object
outcome	outcome
initialSamplings	initialSamplings
fixedTimes	fixedTimes
numberOfSamplingsOptimisable	numberOfSamplingsOptimisable
samplingsWindows	samplingsWindows
numberOfTimesByWindows	numberOfTimesByWindows
minSampling	minSampling

**Value**

SamplingTimeConstraints

---

*initialize, SamplingTimes-method*  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'SamplingTimes'
initialize(.Object, outcome, samplings)
```

**Arguments**

.Object	.Object
outcome	outcome
samplings	samplings

**Value**

SamplingTimes

---

initialize, SimplexAlgorithm-method  
*initialize*

---

**Description**

initialize

**Usage**

```
## S4 method for signature 'SimplexAlgorithm'  
initialize(  
  .Object,  
  pctInitialSimplexBuilding,  
  maxIteration,  
  tolerance,  
  optimalDesigns,  
  iterationAndCriteria,  
  showProcess  
)
```

**Arguments**

.Object	.Object
pctInitialSimplexBuilding	pctInitialSimplexBuilding
maxIteration	maxIteration
tolerance	tolerance
optimalDesigns	optimalDesigns
iterationAndCriteria	iterationAndCriteria
showProcess	showProcess

**Value**

SimplexAlgorithm

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	<i>Test if a mode is bolus.</i>
--------------	---------------------------------

---

**Description**

Test if a mode is bolus.

**Usage**

```
isModelBolus(object, designs)
```

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

**Arguments**

object	An object from the class <a href="#">Model</a> .
designs	A list of objects from the class <a href="#">Design</a> .

**Value**

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

---

isModelInfusion	<i>Test if a mode is infusion</i>
-----------------	-----------------------------------

---

**Description**

Test if a mode is infusion

**Usage**

```
isModelInfusion(object)
```

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

**Arguments**

object	An object from the class <a href="#">Model</a> .
--------	--

**Value**

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

---

isModelODE	<i>Test if a mode is ode.</i>
------------	-------------------------------

---

**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	<i>Test if a mode is steady state.</i>
--------------------	--

---

**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 from 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 from 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.

---

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 from 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 from 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.
- weightThreshold: A numeric giving the threshold of the weights.
- optimalWeights: A vector giving the optimal weights.
- optimalDesign: An object of the class Design giving the optimal design.
- showProcess: A boolean for showing or not the process of optimization.

MultiplicativeAlgorithm\_Rcpp

*Function MultiplicativeAlgorithm\_Rcpp*

**Description**

Run the MultiplicativeAlgorithm\_Rcpp in Rcpp

**Usage**

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

**Arguments**

```
fisherMatrices_input      fisherMatrices_input
numberOfFisherMatrices_input  numberOfFisherMatrices_input
weights_input             weights_input
numberOfParameters_input  numberOfParameters_input
dim_input                 dim_input
lambda_input              lambda_input
```

```

delta_input    delta_input
iterationInit_input
              iterationInit_input

```

---

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

### Slots for Administration objects

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



---

OptimizationAlgorithm-class  
*Class "OptimizationAlgorithm"*

---

**Description**

A class storing information concerning the optimization algorithm.

**Objects from the class**

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

**Slots for Administration objects**

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

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

---

optimize	<i>Optimize a design.</i>
----------	---------------------------

---

**Description**

Optimize a design.

**Usage**

```
optimize(object, optimizerParameters, optimizationObject)
```

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

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

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

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

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

**Arguments**

object	An object from the class <a href="#">OptimizationAlgorithm</a> .
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, valuePars)
```

```
## S4 method for signature 'Model'
parametersForComputingGradient(object, valuePars)
```

**Arguments**

object	An object from the class <a href="#">Model</a> .
valuePars	Vector of parameter values

**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 from 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.

---

PGBAlgorithm-class      *Class "PGBAlgorithm"*

---

**Description**

The class "PGBAlgorithm" implements the PGB algorithm: Population Genetics Based Optimizer, developed by Hervé Le Nagard [1].

**Objects from the Class PGBAlgorithm**

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

**Slots for PGBAlgorithm 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**

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

---

plotEvaluation      *Graphs of the results of the evaluation.*

---

### Description

Graphs of the results of the evaluation.

### Usage

```
plotEvaluation(object, plotOptions)

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

### Arguments

object            An object from the class [Evaluation](#).  
plotOptions      A list giving the plot options.

### Value

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

---

PlotEvaluation-class    *Class "PlotEvaluation"*

---

### Description

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

---

plotFrequencies      *Graph of the frequencies for the FW algorithm.*

---

### Description

Graph of the frequencies for the FW algorithm.

**Usage**

```
plotFrequencies(object)

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

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

**Arguments**

object            An object from the class [OptimizationAlgorithm](#).

**Value**

The graphs of the frequencies for the FW algorithm.

---

```
plotOutcomesEvaluation
                          plotOutcomesEvaluation
```

---

**Description**

Plot the evaluation of the outcomes.

**Usage**

```
plotOutcomesEvaluation(
  object,
  outcomesEvaluationInitialDesign,
  model,
  plotOptions
)

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

**Arguments**

object            An object Design from the class [Design](#).  
outcomesEvaluationInitialDesign  
                  A list containing the evaluation of the initial design.

model            An object model from the class [Model](#).  
 plotOptions    A list containing the plot options.

**Value**

A list containing the plots the evaluation of the outcomes.

---

plotOutcomesGradient    *plotOutcomesGradient*

---

**Description**

Plot the evaluation of the outcome gradients.

**Usage**

```
plotOutcomesGradient(object, outcomesGradientInitialDesign, model, plotOptions)

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

**Arguments**

object            An object design from the class [Design](#).  
 outcomesGradientInitialDesign  
                   A list with the evaluation of the gradient for the initial design.  
 model            An object model from the class [Model](#).  
 plotOptions    A list containing the plot options.

**Value**

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

---

plotRSE                    *Graph of the RSE.*

---

**Description**

Graph of the RSE.

**Usage**

```
plotRSE(object, plotOptions)

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

**Arguments**

object            An object from the class [Evaluation](#).  
plotOptions      A list giving the plot options.

**Value**

A graph of the RSE.

---

plotSE            *Graph the SE.*

---

**Description**

Graph the SE.

**Usage**

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

**Arguments**

object            An object from the class [Evaluation](#).  
plotOptions      A list giving the plot options.

**Value**

A graph of the SE.

---

plotSensitivityIndice    *Graphs of the results of the evaluation.*

---

**Description**

Graphs of the results of the evaluation.

**Usage**

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

**Arguments**

object            An object from the class [Evaluation](#).  
 plotOptions     A list giving the plot options.

**Value**

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

---

plotShrinkage     *Graph of the shrinkage.*

---

**Description**

Graph of the shrinkage.

**Usage**

```
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)

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

## S4 method for signature 'Optimization'
plotWeights(object)
```



**Arguments**

object            An object from the class [OptimizationAlgorithm](#).

**Value**

The graphs of the weights for the multiplicative algorithm.

---

PopulationFim-class    *Class "PopulationFim"*

---

**Description**

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

---

Proportional-class    *Class "Proportional"*

---

**Description**

The Class "Proportional" defines the the residual error variance according to the formula  $g(\text{sigma\_inter}, \text{sigma\_slope}, c\_error, f(x, \text{theta})) = \text{sigma\_slope} * f(x, \text{theta})$ .

**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

---

PSOAlgorithm-class      *Class "PSOAlgorithm"*

---

### Description

The class "PSOAlgorithm" implements the PSO algorithm.

### Objects from the class PSOAlgorithm

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

### Slots for PSOAlgorithm objects

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

---

Report

*Report*

---

### Description

Report

### Usage

```
Report(object, outputPath, outputFile, plotOptions)

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

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

**Arguments**

object	An object from the class <a href="#">PFIMProject</a> .
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
      reportTablesAdministration
```

---

**Description**

Generate table for the report.

**Usage**

```
reportTablesAdministration(object)

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

**Arguments**

object	An object design from the class <a href="#">Design</a> .
--------	--

**Value**

A table of the administration parameters for the report.

---

```
reportTablesDesign      reportTablesDesign
```

---

**Description**

Generate table for the report.

**Usage**

```
reportTablesDesign(object)

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

**Arguments**

object            An object design from the class [Design](#).

**Value**

A table of the design parameters for the report.

---

reportTablesFIM            *Generate the tables for the report.*

---

**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.

---

reportTablesPlot	<i>reportTablesPlot</i>
------------------	-------------------------

---

**Description**

Generate all the table for the evaluation report

**Usage**

```
reportTablesPlot(object, plotOptions)

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

**Arguments**

object	An object evaluation from the class <a href="#">Evaluation</a> .
plotOptions	A list containing the options for the plots.

**Value**

The list tables containing the tables for the evaluation report.

---

reportTablesSamplingConstraints	<i>reportTablesSamplingConstraints</i>
---------------------------------	--

---

**Description**

Generate table for the report.

**Usage**

```
reportTablesSamplingConstraints(object)

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

**Arguments**

object	An object design from the class <a href="#">Design</a> .
--------	--

**Value**

A table of the sampling constraints parameters for the report.

---

resizeFisherMatrix	<i>Resize the fisher Matrix from a vector to a matrix.</i>
--------------------	--

---

**Description**

Resize the fisher Matrix from a vector to a matrix.

**Usage**

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

```
## 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

---

run	<i>run</i>
-----	------------

---

**Description**

run

**Usage**

```
run(object)
```

```
## S4 method for signature 'Evaluation'
```

```
run(object)
```

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

```
run(object)
```

**Arguments**

object            An object from the class [PFIMProject](#).

**Value**

A list giving the results of evaluation or optimization.

---

SamplingTimeConstraints-class  
*Class "SamplingTimeConstraints"*

---

**Description**

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

**Objects from the class SamplingTimeConstraints**

Objects from 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 from 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	<i>setAdministrations</i>
--------------------	---------------------------

---

**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 <a href="#">Arm</a> .
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	<i>setArm</i>
--------	---------------

---

**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 <a href="#">Design</a> .
arm	A list of object Arm giving the arms of the design.

**Value**

An object Design with the list Arm updated.

---

setArms	<i>Set the arms of an object.</i>
---------	-----------------------------------

---

**Description**

Set the arms of an object.

**Usage**

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

```
## S4 method for signature 'Design'  
setArms(object, arms)
```

```
## S4 method for signature 'OptimizationAlgorithm'  
setArms(object, arms)
```

**Arguments**

object	An object defined form a class of PFIM.
arms	A list of arms.

**Value**

The object with the updated arms.

---

setcError	<i>Set the parameter c.</i>
-----------	-----------------------------

---

**Description**

Set the parameter c.

**Usage**

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

```
## S4 method for signature 'ModelError'  
setcError(object, cError)
```

**Arguments**

object	An object from the class <a href="#">ModelError</a> .
cError	A numeric giving the parameter c.

**Value**

The model error with the parameter c.

---

setContent	<i>Set content of a library of models.</i>
------------	--

---

**Description**

Set content of a library of models.

**Usage**

```
setContent(object, content)
```

```
## S4 method for signature 'LibraryOfModels'
```

```
setContent(object, content)
```

**Arguments**

object            An object from the class [LibraryOfModels](#).

content           A list giving the content of the library of models.

**Value**

The library of models with the updated content.

---

setDataForArmEvaluation	<i>setDataForArmEvaluation</i>
-------------------------	--------------------------------

---

**Description**

setDataForArmEvaluation

**Usage**

```
setDataForArmEvaluation(object, data)
```

```
## S4 method for signature 'Arm'
```

```
setDataForArmEvaluation(object, data)
```

**Arguments**

object            An object Arm from the class [Arm](#).

data              A list containing the data for arm evaluation

**Value**

Set the list containing the data for arm evaluation.

---

```
setDataForModelEvaluation
```

*Generate the table of dose, time dose etc. for model evaluation*

---

**Description**

Generate the table of dose, time dose etc. for model evaluation

**Usage**

```
setDataForModelEvaluation(object, arm)

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

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

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

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

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

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

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

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

**Arguments**

object	An object from the class <a href="#">Model</a> .
arm	An object from the class <a href="#">Arm</a> .

**Value**

Return a dataframe with all the data for model evaluation

---

setDerivatives	<i>Set the derivatives of the model error equation.</i>
----------------	---

---

**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 <a href="#">ModelError</a> .
derivatives	The derivatives of the model error equation.

**Value**

The model error with the updated model error equation.

---

setDescription	<i>Set the description of a model.</i>
----------------	--

---

**Description**

Set the description of a model.

**Usage**

```
setDescription(object, description)

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

**Arguments**

object	An object from the class <a href="#">Model</a> .
description	A list giving the description of a model.

**Value**

The model with the updated description.

---

setDesigns	<i>Set the designs.</i>
------------	-------------------------

---

**Description**

Set the designs.

**Usage**

```
setDesigns(object, designs)

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

**Arguments**

object	An object from the class <a href="#">Optimization</a> .
designs	A list of objects from the class <a href="#">Design</a> .

**Value**

The object with the new designs.

---

setDistribution	<i>Set the distribution.</i>
-----------------	------------------------------

---

**Description**

Set the distribution.

**Usage**

```
setDistribution(object, distribution)

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

**Arguments**

object	An object from the class <a href="#">ModelParameter</a> .
distribution	An object from the class <a href="#">Distribution</a> .

**Value**

The model parameter with the updated distribution.

---

setDose	<i>Set the amount of dose</i>
---------	-------------------------------

---

**Description**

Set the amount of dose

**Usage**

```
setDose(object, dose)

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

**Arguments**

object	An object Administration from the class <a href="#">Administration</a> .
dose	A numeric value of the amount of dose.

**Value**

The numeric amount\_dose giving the new value of the amount of dose.

---

setEquation	<i>Set the equation of a model error.</i>
-------------	---

---

**Description**

Set the equation of a model error.

**Usage**

```
setEquation(object, equation)

## S4 method for signature 'ModelError'
setEquation(object, equation)
```

**Arguments**

object	An object from the class <a href="#">ModelError</a> .
equation	An expression giving the equation of a model error.

**Value**

The model error with the updated equation.

---

setEquations	<i>Set the equations of a model.</i>
--------------	--------------------------------------

---

**Description**

Set the equations of a model.

**Usage**

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

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

**Arguments**

object	An object from the class <a href="#">Model</a> .
equations	A list giving the equations of the model.

**Value**

The model with the updated equations.

---

setEquationsAfterInfusion	<i>Set the equations after infusion.</i>
---------------------------	--

---

**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 <a href="#">Model</a> .
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 <a href="#">Model</a> .
equations	A list giving the equations during the infusion.

**Value**

The model with the updated equations during the infusion.

---

```
setEvaluationFIMResults
    Set the evaluation results.
```

---

**Description**

Set the evaluation results.

**Usage**

```
setEvaluationFIMResults(object, value)

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

**Arguments**

object	An object from the class <a href="#">Optimization</a> .
value	An object from the class <a href="#">Evaluation</a> giving the evaluation results.

**Value**

The object with the updated object from the class [Evaluation](#).

---

setEvaluationInitialDesignResults

*Set the evaluation results of the initial design.*

---

### Description

Set the evaluation results of the initial design.

### Usage

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

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

### Arguments

object	An object from the class <a href="#">Optimization</a> .
value	An object from the class <a href="#">Evaluation</a> giving the evaluation results of the initial design.

### Value

The object with the updated object from the class [Evaluation](#).

---

setFim

*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 <a href="#">Design</a> .
fim	An object fim from the class <a href="#">Fim</a> .

### Value

An object Design with the fim updated.

---

setFimTypeToString	<i>Convert the type of the object fim to a string.</i>
--------------------	--

---

**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 <a href="#">Fim</a> .
--------	--

**Value**

The type of the object fim convert as a string.

---

setFisherMatrix	<i>Set the FIM.</i>
-----------------	---------------------

---

**Description**

Set the FIM.

**Usage**

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

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

**Arguments**

object	An object from the class <a href="#">Fim</a> .
value	A matrix giving the FIM.

**Value**

The object from the class [Fim](#) with the FIM updated.

---

setFixedEffects	<i>Set the fixed effects.</i>
-----------------	-------------------------------

---

**Description**

Set the fixed effects.

**Usage**

```
setFixedEffects(object)

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

**Arguments**

object            An object from the class [Fim](#).

**Value**

Update the matrix of the fixed effects.

---

setFixedMu	<i>Set the mu as fixed or not.</i>
------------	------------------------------------

---

**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 mode parameter with the the mu updated as fixed or not.

---

setFixedOmega	<i>Set the omega as fixed of not.</i>
---------------	---------------------------------------

---

**Description**

Set the omega as fixed of not.

**Usage**

```
setFixedOmega(object, value)
```

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

**Arguments**

object	An object from the class <a href="#">ModelParameter</a> .
value	A Boolean fixed or not.

**Value**

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

---

setInitialConditions	<i>setInitialConditions</i>
----------------------	-----------------------------

---

**Description**

Set the initial conditions of a ode model.

**Usage**

```
setInitialConditions(object, initialConditions)
```

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

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

**Arguments**

object	An object from the class <a href="#">Model</a> .
initialConditions	A list giving the initial conditions.

**Value**

The model with the updated initial conditions.

---

```
setIterationAndCriteria
```

*Set the iteration with the convergence criteria.*

---

**Description**

Set the iteration with the convergence criteria.

**Usage**

```
setIterationAndCriteria(object, value)
```

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

**Arguments**

object	An object from the class <a href="#">OptimizationAlgorithm</a> .
value	A dataframe giving the iteration with the convergence criteria.

**Value**

A dataframe giving the iteration with the convergence criteria.

---

```
setModel
```

*Set the model.*

---

**Description**

Set the model.

**Usage**

```
setModel(object, model)
```

```
## S4 method for signature 'PFIMProject'
setModel(object, model)
```

**Arguments**

object	An object from the class <a href="#">PFIMProject</a> .
model	An object from the class <a href="#">Model</a> .

**Value**

The object with the updated model.

---

setModelError	<i>Set the model error.</i>
---------------	-----------------------------

---

**Description**

Set the model error.

**Usage**

```
setModelError(object, modelError)

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

**Arguments**

object	An object from the class <a href="#">Model</a> .
modelError	An object from the class <a href="#">ModelError</a> .

**Value**

The model with the updated model error.

---

setModelFromLibrary	<i>Set a model from the library of model</i>
---------------------	--

---

**Description**

Set a model from the library of model

**Usage**

```
setModelFromLibrary(object, modelFromLibrary)

## S4 method for signature 'Model'
setModelFromLibrary(object, modelFromLibrary)
```

**Arguments**

object	An object from the class <a href="#">Model</a> .
modelFromLibrary	An object from the class <a href="#">Model</a> .

**Value**

The model with the updated model from library of models.

---

setMu	<i>Set the value of the fixed effect mu of an object.</i>
-------	---

---

**Description**

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

**Usage**

```
setMu(object, value)

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

## S4 method for signature 'ModelParameter'
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	<i>Set the name of an object.</i>
---------	-----------------------------------

---

**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	<i>setNumberOfArms</i>
-----------------	------------------------

---

**Description**

Set the number of arms in a design.

**Usage**

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

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

**Arguments**

object	An object Design from the class <a href="#">Design</a> .
numberOfArms	A numeric numberOfArms giving the new number of arms in the design.

**Value**

An object Design with the numberOfArms updated.

---

setOdeSolverParameters	<i>Set the parameters of the ode solver.</i>
------------------------	--

---

**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	<i>Set the matrix omega of an object.</i>
----------	---

---

**Description**

Set the matrix omega of an object.

**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	<i>Set the optimal design.</i>
------------------	--------------------------------

---

**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	<i>Set the optimal weights.</i>
-------------------	---------------------------------

---

**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
```

*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	<i>Set the outcomes of a model.</i>
-------------	-------------------------------------

---

**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 <a href="#">Model</a> .
outcomes	A list giving the outcomes of the model.

**Value**

The model with the updated outcomes.

---

setOutcomesEvaluation	<i>setOutcomesEvaluation</i>
-----------------------	------------------------------

---

**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 <a href="#">Design</a> .
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)
```

**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    setOutcomesGradient
```

---

**Description**

Set the results of the evaluation of the outcomes.

**Usage**

```
setOutcomesGradient(object, outcomesGradient)
```

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

```
setOutcomesGradient(object, outcomesGradient)
```

**Arguments**

object            An object Design from the class [Design](#).  
 outcomesGradient            A list containing the evaluation of the outcome gradients.

**Value**

An object Design with the list outcomesGradient updated.

---

setParameters	<i>Set the parameters of an object.</i>
---------------	---

---

**Description**

Set the parameters of an object.

**Usage**

```
setParameters(object, parameters)

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

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

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

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

## S4 method for signature 'PGBAlgorithm'
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.

---

```
setSamplingConstraintForOptimization
    setSamplingConstraintForOptimization
```

---

**Description**

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

**Usage**

```
setSamplingConstraintForOptimization(object)

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

**Arguments**

object            An object from the class [Design](#).

**Value**

The arms with the sampling times constraints.

---

```
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	<i>setSamplingTime</i>
-----------------	------------------------

---

**Description**

Set the sampling time of an arm.

**Usage**

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

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

**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.

---

setSamplingTimes	<i>setSamplingTimes</i>
------------------	-------------------------

---

**Description**

Set the vectors of sampling times for an arm.

**Usage**

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

```
## S4 method for signature 'Arm'  
setSamplingTimes(object, samplingTimes)
```

**Arguments**

object            An object Arm from the class [Arm](#).  
samplingTimes    The list containing the new sampling times.

**Value**

An object Arm from the class [Arm](#) with the new sampling times samplingTimes.

---

```
setSamplingTimesConstraints
      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`            An object `SamplingTimeConstraints` from the class [SamplingTimeConstraints](#).

**Value**

The arm with the new sampling time constraints.

---

```
setShrinkage            Set the shrinkage.
```

---

**Description**

Set the shrinkage.

**Usage**

```
setShrinkage(object, value)

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

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

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

**Arguments**

object            An object from the class [Fim](#).  
 value            A vector giving the shrinkage of the Bayesian fim.

**Value**

The object with the updated shrinkage.

---

setSigmaInter            *Set the parameter sigma inter.*

---

**Description**

Set the parameter sigma inter.

**Usage**

```
setSigmaInter(object, sigmaInter)

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

**Arguments**

object            An object from the class [ModelError](#).  
 sigmaInter        A numeric giving the parameter sigma inter.

**Value**

The model error with the updated sigma inter.

---

setSigmaSlope            *Set the parameter sigma slope.*

---

**Description**

Set the parameter sigma slope.

**Usage**

```
setSigmaSlope(object, sigmaSlope)

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

**Arguments**

object	An object from the class <a href="#">ModelError</a> .
sigmaSlope	A numeric giving the parameter sigma slope.

**Value**

The model error with the updated sigma slope.

---

setSize	<i>setSize</i>
---------	----------------

---

**Description**

Set the size of an object.

Set the size of an arm.

**Usage**

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

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

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

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

**Arguments**

object	An object Arm from the class <a href="#">Arm</a> .
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	<i>setTau</i>
--------	---------------

---

**Description**

Set the frequency tau.

**Usage**

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

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

**Arguments**

object	An object Administration from the class <a href="#">Administration</a> .
tau	A numeric value for the infusion lag tau.

**Value**

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

---

setTimeDose	<i>setTimeDose</i>
-------------	--------------------

---

**Description**

Set the times vector when doses are given.

**Usage**

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

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

**Arguments**

object	An object Administration from the class <a href="#">Administration</a> .
timeDose	A numeric value of the time dose.

**Value**

The object Administration with its new times vector for doses.

---

setTinf	<i>Set the infusion duration.</i>
---------	-----------------------------------

---

**Description**

Set the infusion duration.

**Usage**

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

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

**Arguments**

object	An object Administration from the class <a href="#">Administration</a> .
Tinf	A numeric value for the infusion duration Tinf.

**Value**

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

---

setVarianceEffects	<i>Set the matrix of the variance effects.</i>
--------------------	--

---

**Description**

Set the matrix of the variance effects.

**Usage**

```
setVarianceEffects(object)
```

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

**Arguments**

object	An object from the class <a href="#">Fim</a> .
--------	--

**Value**

Update the matrix of the variance effects.

---

show,Design-method     *show*

---

**Description**

show  
show  
show  
show  
show  
show  
show  
show

**Usage**

```
## S4 method for signature 'Design'  
show(object)  
  
## S4 method for signature 'Evaluation'  
show(object)  
  
## S4 method for signature 'FedorovWynnAlgorithm'  
show(object)  
  
## S4 method for signature 'MultiplicativeAlgorithm'  
show(object)  
  
## S4 method for signature 'Optimization'  
show(object)  
  
## S4 method for signature 'PGBAlgorithm'  
show(object)  
  
## S4 method for signature 'PSOAlgorithm'  
show(object)  
  
## S4 method for signature 'SimplexAlgorithm'  
show(object)
```

**Arguments**

object             object

---

SimplexAlgorithm-class

*Class "SimplexAlgorithm"*

---

### **Description**

Class "SimplexAlgorithm" implements the Multiplicative algorithm.

### **Objects from the class SimplexAlgorithm**

Objects from 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

- addModel, [14, 21](#)
- addModel, LibraryOfModels-method  
(addModel), [21](#)
- addModels, [14, 21](#)
- addModels, LibraryOfModels-method  
(addModels), [21](#)
- Administration, [9, 60, 96, 97, 159, 181, 182](#)
- Administration (Administration-class),  
[22](#)
- Administration-class, [22](#)
- AdministrationConstraints, [10](#)
- AdministrationConstraints  
(AdministrationConstraints-class),  
[22](#)
- AdministrationConstraints-class, [22](#)
- Arm, [10, 24, 34, 36–40, 49–51, 56, 68, 91–93,](#)  
[153, 155, 156, 177, 178, 180](#)
- Arm (Arm-class), [23](#)
- Arm-class, [23](#)
- BayesianFim, [10](#)
- BayesianFim (BayesianFim-class), [23](#)
- BayesianFim-class, [23](#)
- checkSamplingTimeConstraintsForContinuousOptimization, [19, 23](#)
- checkSamplingTimeConstraintsForContinuousOptimization, Design-method  
(checkSamplingTimeConstraintsForContinuousOptimization),  
[23](#)
- checkValiditySamplingConstraint, [24](#)
- checkValiditySamplingConstraint, Design-method  
(checkValiditySamplingConstraint),  
[24](#)
- Combined1, [11, 145](#)
- Combined1 (Combined1-class), [25](#)
- Combined1-class, [25](#)
- computeVMat, [19, 25](#)
- Constant, [11](#)
- Constant (Constant-class), [26](#)
- Constant-class, [26](#)
- convertPKModelAnalyticToPKModelODE, [14,](#)  
[15, 26](#)
- convertPKModelAnalyticToPKModelODE, ModelAnalytic-method  
(convertPKModelAnalyticToPKModelODE),  
[26](#)
- convertPKModelAnalyticToPKModelODE, ModelAnalyticInfusion-method  
(convertPKModelAnalyticToPKModelODE),  
[26](#)
- convertPKModelAnalyticToPKModelODE, ModelAnalyticSteadyState-method  
(convertPKModelAnalyticToPKModelODE),  
[26](#)
- dataForArmEvaluation, [27](#)
- dataForArmEvaluation, Design-method  
(dataForArmEvaluation), [27](#)
- defineModel, [15, 27](#)
- defineModel, Model-method (defineModel),  
[27](#)
- defineModelEquationsFromStringToFunction,  
[28](#)
- defineModelEquationsFromStringToFunction, ModelAnalytic-method  
(defineModelEquationsFromStringToFunction),  
[28](#)
- defineModelEquationsFromStringToFunction, ModelAnalyticInfusion-method  
(defineModelEquationsFromStringToFunction),  
[28](#)
- defineModelEquationsFromStringToFunction, ModelAnalyticSteadyState-method  
(defineModelEquationsFromStringToFunction),  
[28](#)
- defineModelEquationsFromStringToFunction, ModelODEBolus-method  
(defineModelEquationsFromStringToFunction),  
[28](#)
- defineModelEquationsFromStringToFunction, ModelODEDoseInEquation-method  
(defineModelEquationsFromStringToFunction),  
[28](#)
- defineModelEquationsFromStringToFunction, ModelODEDoseNotInEquation-method  
(defineModelEquationsFromStringToFunction),  
[28](#)

- 28
- defineModelEquationsFromStringToFunction, ModelODEInfusionDosePKModel-method  
(defineModelEquationsFromStringToFunction), 32  
Design, 11, 24, 27, 30, 31, 35, 77, 85, 86, 127,  
141, 142, 147, 148, 150, 153, 158,  
162, 169, 171, 173, 175, 176
- defineModelFromLibraryOfModels, 15, 30
- defineModelFromLibraryOfModels, Model-method  
(defineModelFromLibraryOfModels),  
30
- defineModelType, 15, 30
- defineModelType, Model-method  
(defineModelType), 30
- defineModelUserDefined, 15, 31
- defineModelUserDefined, Model-method  
(defineModelUserDefined), 31
- definePKModel, 15–17, 31
- definePKModel, ModelAnalytic-method  
(definePKModel), 31
- definePKModel, ModelAnalyticInfusion-method  
(definePKModel), 31
- definePKModel, ModelAnalyticSteadyState-method  
(definePKModel), 31
- definePKModel, ModelODE-method  
(definePKModel), 31
- definePKModel, ModelODEDoseInEquations-method  
(definePKModel), 31
- definePKModel, ModelODEInfusionDoseInEquations-method  
(definePKModel), 31
- definePKPDMModel, 15–17, 32
- definePKPDMModel, ModelAnalytic, ModelAnalytic-method  
(definePKPDMModel), 32
- definePKPDMModel, ModelAnalytic, ModelODE-method  
(definePKPDMModel), 32
- definePKPDMModel, ModelAnalyticInfusion, ModelAnalytic-method  
(definePKPDMModel), 32
- definePKPDMModel, ModelAnalyticInfusion, ModelODE-method  
(definePKPDMModel), 32
- definePKPDMModel, ModelAnalyticSteadyState, ModelAnalyticSteadyState-method  
(definePKPDMModel), 32
- definePKPDMModel, ModelAnalyticSteadyState, ModelODE-method  
(definePKPDMModel), 32
- definePKPDMModel, ModelODEBolus, ModelODE-method  
(definePKPDMModel), 32
- definePKPDMModel, ModelODEDoseInEquations, ModelODE-method  
(definePKPDMModel), 32
- definePKPDMModel, ModelODEDoseNotInEquations, ModelODE-method  
(definePKPDMModel), 32
- definePKPDMModel, ModelODEInfusion, ModelODE-method  
(definePKPDMModel), 32
- definePKPDMModel, ModelODEInfusionDoseInEquations, ModelODE-method  
(definePKPDMModel), 32
- Design (Design-class), 33
- Design-class, 33
- Distribution, 11, 49, 158
- Distribution (Distribution-class), 34
- Distribution-class, 34
- EvaluateArm, 10, 34
- EvaluateArm, Arm-method (EvaluateArm), 34
- EvaluateDesign, 11, 35
- EvaluateDesign, Design-method  
(EvaluateDesign), 35
- EvaluateErrorModelDerivatives, 16, 35
- EvaluateErrorModelDerivatives, ModelError-method  
(EvaluateErrorModelDerivatives),  
35
- EvaluateFisherMatrix, 10, 12, 13, 19, 36
- EvaluateFisherMatrix, BayesianFim-method  
(EvaluateFisherMatrix), 36
- EvaluateFisherMatrix, IndividualFim-method  
(EvaluateFisherMatrix), 36
- EvaluateFisherMatrix, PopulationFim-method  
(EvaluateFisherMatrix), 36
- EvaluateModel, 15–17, 37
- EvaluateModel, ModelAnalytic-method  
(EvaluateModel), 37
- EvaluateModel, ModelAnalyticInfusion-method  
(EvaluateModel), 37
- EvaluateModel, ModelAnalyticInfusionSteadyState-method  
(EvaluateModel), 37
- EvaluateModel, ModelAnalyticSteadyState-method  
(EvaluateModel), 37
- EvaluateModel, ModelODEBolus-method  
(EvaluateModel), 37
- EvaluateModel, ModelODEDoseInEquations-method  
(EvaluateModel), 37
- EvaluateModel, ModelODEDoseNotInEquations-method  
(EvaluateModel), 37
- EvaluateModel, ModelODEInfusionDoseInEquations-method  
(EvaluateModel), 37
- EvaluateModelGradient, 38
- EvaluateModelGradient, ModelAnalytic-method  
(EvaluateModelGradient), 38
- EvaluateModelGradient, ModelAnalyticInfusion-method  
(EvaluateModelGradient), 38

- EvaluateModelGradient, ModelAnalyticInfusionSteadyStateReportEvaluation, PopulationFim-method (EvaluateModelGradient), 38 (generateReportEvaluation), 44
- EvaluateModelGradient, ModelAnalyticSteadyStateReportOptimization, 12, 17–19, (EvaluateModelGradient), 38 46
- EvaluateModelGradient, ModelODEBolus-method generateReportOptimization, FedorovWynnAlgorithm-method (EvaluateModelGradient), 38 (generateReportOptimization), 46
- EvaluateModelGradient, ModelODEDoseInEquations-method generateReportOptimization, MultiplicativeAlgorithm-method (EvaluateModelGradient), 38 (generateReportOptimization), 46
- EvaluateModelGradient, ModelODEDoseNotInEquations-method generateReportOptimization, (EvaluateModelGradient), 38 46
- EvaluateModelGradient, ModelODEInfusionDoseInEquationsReportOptimization, PGBAlgorithm-method (EvaluateModelGradient), 38 (generateReportOptimization), 46
- EvaluateVarianceFIM, 12, 13, 19, 39
- EvaluateVarianceFIM, IndividualFim-method generateReportOptimization, PSOAlgorithm-method (EvaluateVarianceFIM), 39 (generateReportOptimization), 46
- EvaluateVarianceFIM, PopulationFim-method generateReportOptimization, SimplexAlgorithm-method (EvaluateVarianceFIM), 39 (generateReportOptimization), 46
- EvaluateVarianceModel, 15, 39
- EvaluateVarianceModel, Model-method generateSamplingsFromSamplingConstraints, (EvaluateVarianceModel), 39 19, 47
- Evaluation, 11, 64, 140, 143, 144, 150, 161, generateSamplingsFromSamplingConstraints, SamplingTimeConst 162 (generateSamplingsFromSamplingConstraints), 47
- Evaluation (Evaluation-class), 40
- Evaluation-class, 40
  
- FedorovWynnAlgorithm, 12, 81
- FedorovWynnAlgorithm (FedorovWynnAlgorithm-class), 41
- FedorovWynnAlgorithm-class, 41
- FedorovWynnAlgorithm\_Rcpp, 12, 41
- Fim, 12, 34–36, 39, 44, 45, 52–55, 57, 59, 61, 66, 90, 93, 94, 98, 148, 162–164, 179, 182
- Fim (Fim-class), 42
- Fim-class, 42
- fisher.simplex, 19, 43
- fun.amoeba, 19, 43
  
- generateFimsFromConstraints, 18, 44
- generateFimsFromConstraints, Optimization-method (generateFimsFromConstraints), 44
- generateReportEvaluation, 11, 13, 19, 44
- generateReportEvaluation, BayesianFim-method (generateReportEvaluation), 44
- generateReportEvaluation, IndividualFim-method (generateReportEvaluation), 44
- generateReportEvaluation, PopulationFim-method (generateReportEvaluation), 44
- generateReportOptimization, FedorovWynnAlgorithm-method (generateReportOptimization), 46
- generateReportOptimization, MultiplicativeAlgorithm-method (generateReportOptimization), 46
- generateReportOptimization, PGBAlgorithm-method (generateReportOptimization), 46
- generateReportOptimization, PSOAlgorithm-method (generateReportOptimization), 46
- generateReportOptimization, SimplexAlgorithm-method (generateReportOptimization), 46
- generateSamplingsFromSamplingConstraints, 19, 47
- generateSamplingsFromSamplingConstraints, SamplingTimeConst (generateSamplingsFromSamplingConstraints), 47
- generateTables, 12, 18, 48
- generateTables, Evaluation-method (generateTables), 48
- generateTables, Optimization-method (generateTables), 48
- getAdjustedGradient, 11, 14, 17, 48
- getAdjustedGradient, LogNormal-method (getAdjustedGradient), 48
- getAdjustedGradient, Normal-method (getAdjustedGradient), 48
- getAdministration, 10, 49
- getAdministration, Arm-method (getAdministration), 49
- getAdministrationConstraint, 10, 49
- getAdministrationConstraint, Arm-method (getAdministrationConstraint), 49
- getAdministrations, 10, 50
- getAdministrations, Arm-method (getAdministrations), 50
- getAdministrationsConstraints, 10, 50
- getAdministrationsConstraints, Arm-method (getAdministrationsConstraints), 50

- getArms, [13](#), [51](#)
- getArms, Design-method (getArms), [51](#)
- getArms, OptimizationAlgorithm-method (getArms), [51](#)
- getcError, [16](#), [51](#)
- getcError, ModelError-method (getcError), [51](#)
- getColumnAndParametersNamesFIM, [12](#), [52](#)
- getColumnAndParametersNamesFIM, BayesianFim-method (getColumnAndParametersNamesFIM), [52](#)
- getColumnAndParametersNamesFIM, IndividualFim-method (getColumnAndParametersNamesFIM), [52](#)
- getColumnAndParametersNamesFIM, PopulationFim-method (getColumnAndParametersNamesFIM), [52](#)
- getColumnAndParametersNamesFIMInLatex, [13](#), [53](#)
- getColumnAndParametersNamesFIMInLatex, BayesianFim-method (getColumnAndParametersNamesFIMInLatex), [53](#)
- getColumnAndParametersNamesFIMInLatex, IndividualFim-method (getColumnAndParametersNamesFIMInLatex), [53](#)
- getColumnAndParametersNamesFIMInLatex, PopulationFim-method (getColumnAndParametersNamesFIMInLatex), [53](#)
- getConditionNumberFixedEffects, [12](#), [53](#)
- getConditionNumberFixedEffects, Fim-method (getConditionNumberFixedEffects), [53](#)
- getConditionNumberVarianceEffects, [10](#), [12](#), [54](#)
- getConditionNumberVarianceEffects, BayesianFim-method (getConditionNumberVarianceEffects), [54](#)
- getConditionNumberVarianceEffects, Fim-method (getConditionNumberVarianceEffects), [54](#)
- getContent, [14](#), [54](#)
- getContent, LibraryOfModels-method (getContent), [54](#)
- getCorrelationMatrix, [12](#), [55](#)
- getCorrelationMatrix, Evaluation-method (getCorrelationMatrix), [55](#)
- getCorrelationMatrix, Fim-method (getCorrelationMatrix), [55](#)
- getCorrelationMatrix, Optimization-method (getCorrelationMatrix), [55](#)
- getDataForArmEvaluation, [55](#)
- getDataForArmEvaluation, Arm-method (getDataForArmEvaluation), [55](#)
- getDataFrameResults, [17](#), [56](#)
- getDataFrameResults, FedorovWynnAlgorithm-method (getDataFrameResults), [56](#)
- getDataFrameResults, MultiplicativeAlgorithm-method (getDataFrameResults), [56](#)
- getDataFrameResults, Optimization-method (getDataFrameResults), [56](#)
- getDcriterion, [12](#), [57](#)
- getDcriterion, Evaluation-method (getDcriterion), [57](#)
- getDcriterion, Fim-method (getDcriterion), [57](#)
- getDcriterion, Optimization-method (getDcriterion), [57](#)
- getDelta, [14](#), [57](#)
- getDelta, MultiplicativeAlgorithm-method (getDelta), [57](#)
- getDerivatives, [16](#), [58](#)
- getDerivatives, ModelError-method (getDerivatives), [58](#)
- getDescription, [14](#), [58](#)
- getDescription, Model-method (getDescription), [58](#)
- getDesigns, [18](#), [59](#)
- getDesigns, PFIMProject-method (getDesigns), [59](#)
- getDeterminant, [12](#), [59](#)
- getDeterminant, Evaluation-method (getDeterminant), [59](#)
- getDeterminant, Fim-method (getDeterminant), [59](#)
- getDeterminant, Optimization-method (getDeterminant), [59](#)
- getDistribution, [17](#), [60](#)
- getDistribution, ModelParameter-method (getDistribution), [60](#)
- getDose, [9](#), [10](#), [60](#)
- getDose, Administration-method (getDose), [60](#)
- getDose, AdministrationConstraints-method (getDose), [60](#)
- getEigenValues, [12](#), [61](#)
- getEigenValues, Fim-method

- (getEigenValues), 61
- getElementaryProtocols, 18, 61
- getElementaryProtocols, Optimization-method (getElementaryProtocols), 61
- getEquation, 15, 62
- getEquation, ModelError-method (getEquation), 62
- getEquations, 14, 62
- getEquations, Model-method (getEquations), 62
- getEquationsAfterInfusion, 16, 63
- getEquationsAfterInfusion, Model-method (getEquationsAfterInfusion), 63
- getEquationsDuringInfusion, 16, 63
- getEquationsDuringInfusion, Model-method (getEquationsDuringInfusion), 63
- getEvaluationFIMResults, 18, 64
- getEvaluationFIMResults, Optimization-method (getEvaluationFIMResults), 64
- getEvaluationInitialDesignResults, 18, 64
- getEvaluationInitialDesignResults, Optimization-method (getEvaluationInitialDesignResults), 64
- getFim, 11, 13, 18, 65
- getFim, Design-method (getFim), 65
- getFim, OptimizationAlgorithm-method (getFim), 65
- getFim, PFIMProject-method (getFim), 65
- getFisherMatrix, 12, 65
- getFisherMatrix, Evaluation-method (getFisherMatrix), 65
- getFisherMatrix, Fim-method (getFisherMatrix), 65
- getFisherMatrix, Optimization-method (getFisherMatrix), 65
- getFixedEffects, 12, 66
- getFixedEffects, Fim-method (getFixedEffects), 66
- getFixedMu, 17, 66
- getFixedMu, ModelParameter-method (getFixedMu), 66
- getFixedOmega, 17, 67
- getFixedOmega, ModelParameter-method (getFixedOmega), 67
- getFixedParameters, 15, 67
- getFixedParameters, Model-method (getFixedParameters), 67
- getFixedTimes, 19, 68
- getFixedTimes, SamplingTimeConstraints-method (getFixedTimes), 68
- getInitialConditions, 10, 14, 68
- getInitialConditions, Arm-method (getInitialConditions), 68
- getInitialConditions, Model-method (getInitialConditions), 68
- getIterationAndCriteria, 69
- getIterationAndCriteria, OptimizationAlgorithm-method (getIterationAndCriteria), 69
- getLambda, 17, 69
- getLambda, MultiplicativeAlgorithm-method (getLambda), 69
- getLibraryPDMModels, 14, 70
- getLibraryPDMModels, LibraryOfModels-method (getLibraryPDMModels), 70
- getLibraryPKModels, 14, 70
- getLibraryPKModels, LibraryOfModels-method (getLibraryPKModels), 70
- getMinSampling, 19, 71
- getMinSampling, SamplingTimeConstraints-method (getMinSampling), 71
- getModel, 18, 71
- getModel, PFIMProject-method (getModel), 71
- getModelEquations, 18, 72
- getModelEquations, PFIMProject-method (getModelEquations), 72
- getModelError, 13, 14, 18, 72
- getModelError, Model-method (getModelError), 72
- getModelError, PFIMProject-method (getModelError), 72
- getModelErrorParametersValues, 15, 73
- getModelErrorParametersValues, Model-method (getModelErrorParametersValues), 73
- getModelFromLibrary, 14, 73
- getModelFromLibrary, Model-method (getModelFromLibrary), 73
- getModelParameters, 18, 74
- getModelParameters, PFIMProject-method (getModelParameters), 74
- getModelParametersValues, 74
- getModelParametersValues, Model-method (getModelParametersValues), 74

- getMu, [11, 13, 17, 75](#)
- getMu, Distribution-method (getMu), [75](#)
- getMu, ModelParameter-method (getMu), [75](#)
- getName, [10, 11, 13, 14, 17, 18, 75](#)
- getName, Arm-method (getName), [75](#)
- getName, Design-method (getName), [75](#)
- getName, LibraryOfModels-method (getName), [75](#)
- getName, Model-method (getName), [75](#)
- getName, ModelParameter-method (getName), [75](#)
- getName, PFIMProject-method (getName), [75](#)
- getNames, [13, 76](#)
- getNames, list-method (getNames), [76](#)
- getNumberOfArms, [11, 77](#)
- getNumberOfArms, Design-method (getNumberOfArms), [77](#)
- getNumberOfIterations, [17, 77](#)
- getNumberOfIterations, MultiplicativeAlgorithm-method (getNumberOfIterations), [77](#)
- getNumberOfParameters, [14, 78](#)
- getNumberOfParameters, Model-method (getNumberOfParameters), [78](#)
- getNumberOfSamplingsOptimisable, [19, 78](#)
- getNumberOfSamplingsOptimisable, SamplingTimeConstraints-method (getNumberOfSamplingsOptimisable), [78](#)
- getNumberOfTimesByWindows, [19, 79](#)
- getNumberOfTimesByWindows, SamplingTimeConstraints-method (getNumberOfTimesByWindows), [79](#)
- getOdeSolverParameters, [13, 14, 18, 79](#)
- getOdeSolverParameters, Model-method (getOdeSolverParameters), [79](#)
- getOdeSolverParameters, PFIMProject-method (getOdeSolverParameters), [79](#)
- getOmega, [11, 13, 17, 80](#)
- getOmega, Distribution-method (getOmega), [80](#)
- getOmega, ModelParameter-method (getOmega), [80](#)
- getOptimalDesign, [80](#)
- getOptimalDesign, OptimizationAlgorithm-method (getOptimalDesign), [80](#)
- getOptimalFrequencies, [81](#)
- getOptimalFrequencies, FedorovWynnAlgorithm-method (getOptimalFrequencies), [81](#)
- getOptimalWeights, [17, 81](#)
- getOptimalWeights, MultiplicativeAlgorithm-method (getOptimalWeights), [81](#)
- getOptimizationResults, [17, 82](#)
- getOptimizationResults, Optimization-method (getOptimizationResults), [82](#)
- getOptimizer, [18, 82](#)
- getOptimizer, PFIMProject-method (getOptimizer), [82](#)
- getOptimizerParameters, [18, 83](#)
- getOptimizerParameters, PFIMProject-method (getOptimizerParameters), [83](#)
- getOutcome, [9, 10, 13, 15, 19, 83](#)
- getOutcome, Administration-method (getOutcome), [83](#)
- getOutcome, AdministrationConstraints-method (getOutcome), [83](#)
- getOutcome, ModelError-method (getOutcome), [83](#)
- getOutcome, SamplingTimeConstraints-method (getOutcome), [83](#)
- getOutcome, SamplingTimes-method (getOutcome), [83](#)
- getOutcomes, [14, 18, 84](#)
- getOutcomes, Model-method (getOutcomes), [84](#)
- getOutcomes, PFIMProject-method (getOutcomes), [84](#)
- getOutcomesEvaluation, [11, 84](#)
- getOutcomesEvaluation, Design-method (getOutcomesEvaluation), [84](#)
- getOutcomesForEvaluation, [14, 85](#)
- getOutcomesForEvaluation, Model-method (getOutcomesForEvaluation), [85](#)
- getOutcomesGradient, [11, 85](#)
- getOutcomesGradient, Design-method (getOutcomesGradient), [85](#)
- getParameters, [11, 13, 14, 16, 86](#)
- getParameters, Distribution-method (getParameters), [86](#)
- getParameters, Model-method (getParameters), [86](#)
- getParameters, ModelError-method (getParameters), [86](#)
- getPDModel, [14, 87](#)
- getPDModel, LibraryOfPKPDModels-method (getPDModel), [87](#)
- getPKModel, [14, 87](#)
- getPKModel, LibraryOfPKPDModels-method (getPKModel), [87](#)

- getPKPDMoDel, [14, 88](#)
- getPKPDMoDel, LibraryOfPKPDMoDels-method (getPKPDMoDel), [88](#)
- getPlotOptions, [88](#)
- getProportionsOfSubjects, [17, 89](#)
- getProportionsOfSubjects, Optimization-method (getProportionsOfSubjects), [89](#)
- getRSE, [10, 12, 13, 19, 89](#)
- getRSE, BayesianFim-method (getRSE), [89](#)
- getRSE, Evaluation-method (getRSE), [89](#)
- getRSE, IndividualFim-method (getRSE), [89](#)
- getRSE, Optimization-method (getRSE), [89](#)
- getRSE, PopulationFim-method (getRSE), [89](#)
- getSamplings, [13, 19, 90](#)
- getSamplings, SamplingTimeConstraints-method (getSamplings), [90](#)
- getSamplings, SamplingTimes-method (getSamplings), [90](#)
- getSamplingsWindows, [19, 90](#)
- getSamplingsWindows, SamplingTimeConstraints-method (getSamplingsWindows), [90](#)
- getSamplingTime, [10, 91](#)
- getSamplingTime, Arm-method (getSamplingTime), [91](#)
- getSamplingTimeConstraint, [10, 91](#)
- getSamplingTimeConstraint, Arm-method (getSamplingTimeConstraint), [91](#)
- getSamplingTimes, [10, 92](#)
- getSamplingTimes, Arm-method (getSamplingTimes), [92](#)
- getSamplingTimesConstraints, [10, 92](#)
- getSamplingTimesConstraints, Arm-method (getSamplingTimesConstraints), [92](#)
- getSE, [12, 93](#)
- getSE, Evaluation-method (getSE), [93](#)
- getSE, Fim-method (getSE), [93](#)
- getSE, Optimization-method (getSE), [93](#)
- getShrinkage, [10, 12, 13, 19, 93](#)
- getShrinkage, BayesianFim-method (getShrinkage), [93](#)
- getShrinkage, Evaluation-method (getShrinkage), [93](#)
- getShrinkage, IndividualFim-method (getShrinkage), [93](#)
- getShrinkage, Optimization-method (getShrinkage), [93](#)
- getShrinkage, PopulationFim-method (getShrinkage), [93](#)
- getSigmaInter, [16, 94](#)
- getSigmaInter, ModelError-method (getSigmaInter), [94](#)
- getSigmaSlope, [16, 95](#)
- getSigmaSlope, ModelError-method (getSigmaSlope), [95](#)
- getSize, [10, 11, 13, 95](#)
- getSize, Arm-method (getSize), [95](#)
- getSize, Design-method (getSize), [95](#)
- getTau, [10, 96](#)
- getTau, Administration-method (getTau), [96](#)
- getTimeDose, [9, 96](#)
- getTimeDose, Administration-method (getTimeDose), [96](#)
- getTinf, [10, 97](#)
- getTinf, Administration-method (getTinf), [97](#)
- getVariables, [97](#)
- getVariables, ModelInfusion-method (getVariables), [97](#)
- getVariables, ModelODE-method (getVariables), [97](#)
- getVariables, ModelODEBolus-method (getVariables), [97](#)
- getVarianceEffects, [12, 98](#)
- getVarianceEffects, Fim-method (getVarianceEffects), [98](#)
- getWeightThreshold, [17, 98](#)
- getWeightThreshold, MultiplicativeAlgorithm-method (getWeightThreshold), [98](#)
- IndividualFim, [13](#)
- IndividualFim (IndividualFim-class), [99](#)
- IndividualFim-class, [99](#)
- initialize, Administration-method, [99](#)
- initialize, AdministrationConstraints-method, [100](#)
- initialize, Arm-method, [100](#)
- initialize, Combined1-method, [101](#)
- initialize, Constant-method, [102](#)
- initialize, Design-method, [103](#)
- initialize, Distribution-method, [104](#)
- initialize, Evaluation-method, [104](#)
- initialize, FedorovWynnAlgorithm-method, [105](#)
- initialize, Fim-method, [106](#)
- initialize, LibraryOfModels-method, [106](#)

- initialize,LibraryOfPKPDMODELS-method, 107
- initialize,LogNormal-method, 107
- initialize,Model-method, 108
- initialize,ModelAnalytic-method, 109
- initialize,ModelAnalyticBolus-method, 109
- initialize,ModelAnalyticBolusSteadyState-method, 110
- initialize,ModelAnalyticInfusion-method, 111
- initialize,ModelAnalyticInfusionSteadyState-method, 112
- initialize,ModelAnalyticSteadyState-method, 113
- initialize,ModelBolus-method, 113
- initialize,ModelError-method, 114
- initialize,ModelInfusion-method, 115
- initialize,ModelParameter-method, 116
- initialize,MultiplicativeAlgorithm-method, 117
- initialize,Normal-method, 118
- initialize,Optimization-method, 118
- initialize,OptimizationAlgorithm-method, 119
- initialize,PFIMProject-method, 120
- initialize,PGBOAlgorithm-method, 120
- initialize,Proportional-method, 121
- initialize,PSOAlgorithm-method, 122
- initialize,SamplingTimeConstraints-method, 123
- initialize,SamplingTimes-method, 124
- initialize,SimplexAlgorithm-method, 125
- isDoseInEquations, 15, 126
- isDoseInEquations,Model-method (isDoseInEquations), 126
- isModelAnalytic, 15, 126
- isModelAnalytic,Model-method (isModelAnalytic), 126
- isModelBolus, 15, 127
- isModelBolus,Model-method (isModelBolus), 127
- isModelInfusion, 15, 127
- isModelInfusion,Model-method (isModelInfusion), 127
- isModelODE, 15, 128
- isModelODE,Model-method (isModelODE), 128
- isModelSteadyState, 15, 128
- isModelSteadyState,Model-method (isModelSteadyState), 128
- LibraryOfModels, 13, 21, 55, 70, 155
- LibraryOfModels (LibraryOfModels-class), 129
- LibraryOfModels-class, 129
- LibraryOfPDMODELS, 129
- LibraryOfPKMODELS, 129
- LibraryOfPKPDMODELS, 14, 87, 88
- LibraryOfPKPDMODELS (LibraryOfPKPDMODELS-class), 130
- LibraryOfPKPDMODELS-class, 130
- LogNormal, 14
- LogNormal (LogNormal-class), 130
- LogNormal-class, 130
- Model, 14–16, 21, 26, 27, 29–40, 52, 53, 58, 62, 63, 67, 73, 74, 78, 84, 85, 90, 97, 126–128, 138, 142, 149, 156, 157, 160, 161, 165–167, 170, 173, 174
- Model (Model-class), 130
- Model-class, 130
- ModelAnalytic, 15
- ModelAnalytic (ModelAnalytic-class), 131
- ModelAnalytic-class, 131
- ModelAnalyticBolus, 15
- ModelAnalyticBolus (ModelAnalyticBolus-class), 131
- ModelAnalyticBolus-class, 131
- ModelAnalyticBolusSteadyState, 15
- ModelAnalyticBolusSteadyState (ModelAnalyticBolusSteadyState-class), 131
- ModelAnalyticBolusSteadyState-class, 131
- ModelAnalyticInfusion (ModelAnalyticInfusion-class), 131
- ModelAnalyticInfusion-class, 131
- ModelAnalyticInfusionSteadyState (ModelAnalyticInfusionSteadyState-class), 132
- ModelAnalyticInfusionSteadyState-class, 132



- ModelAnalyticSteadyState
  - (ModelAnalyticSteadyState-class), 132
- ModelAnalyticSteadyState-class, 132
- ModelBolus, 15
- ModelBolus (ModelBolus-class), 132
- ModelBolus-class, 132
- ModelError, 11, 15, 19, 25, 26, 35, 52, 58, 62, 94, 95, 154, 157, 159, 167, 179, 180
- ModelError (ModelError-class), 132
- ModelError-class, 132
- ModelInfusion, 16
- ModelInfusion (ModelInfusion-class), 132
- ModelInfusion-class, 132
- ModelODE, 16
- ModelODE (ModelODE-class), 133
- ModelODE-class, 133
- ModelODEBolus, 16
- ModelODEBolus (getVariables), 97
- ModelODEBolus-class (getVariables), 97
- ModelODEDoseInEquations, 16
- ModelODEDoseInEquations
  - (ModelODEDoseInEquations-class), 133
- ModelODEDoseInEquations-class, 133
- ModelODEDoseNotInEquations, 16
- ModelODEDoseNotInEquations
  - (ModelODEDoseNotInEquations-class), 133
- ModelODEDoseNotInEquations-class, 133
- ModelODEInfusion, 16
- ModelODEInfusion
  - (ModelODEInfusion-class), 133
- ModelODEInfusion-class, 133
- ModelODEInfusionDoseInEquations, 16
- ModelODEInfusionDoseInEquations
  - (ModelODEInfusionDoseInEquations-class), 134
- ModelODEInfusionDoseInEquations-class, 134
- ModelParameter, 17, 60, 66, 67, 158, 164, 165
- ModelParameter (ModelParameter-class), 134
- ModelParameter-class, 134
- MultiplicativeAlgorithm, 17, 57, 69, 77, 81, 98, 171
- MultiplicativeAlgorithm
  - (MultiplicativeAlgorithm-class), 134
- MultiplicativeAlgorithm-class, 134
- MultiplicativeAlgorithm\_Rcpp, 17, 135
- Normal, 17
- Normal (Normal-class), 136
- Normal-class, 136
- Optimization, 17, 43, 44, 47, 61, 64, 82, 89, 158, 161, 162, 172
- Optimization (Optimization-class), 136
- Optimization-class, 136
- OptimizationAlgorithm, 47, 56, 69, 80, 82, 138, 141, 145, 166, 171, 172
- OptimizationAlgorithm
  - (OptimizationAlgorithm-class), 137
- OptimizationAlgorithm-class, 137
- optimize, 12, 17–19, 137
- optimize, FedorovWynnAlgorithm-method
  - (optimize), 137
- optimize, MultiplicativeAlgorithm-method
  - (optimize), 137
- optimize, PGBAlgorithm-method
  - (optimize), 137
- optimize, PSOAlgorithm-method
  - (optimize), 137
- optimize, SimplexAlgorithm-method
  - (optimize), 137
- package-PFIM (PFIM-package), 8
- parametersForComputingGradient, 15, 138
- parametersForComputingGradient, Model-method
  - (parametersForComputingGradient), 138
- PFIM (PFIM-package), 8
- PFIM, (PFIM-package), 8
- PFIM-package, 8
- PFIMProject, 18, 48, 59, 71, 72, 74, 82, 83, 147, 151, 166
- PFIMProject (PFIMProject-class), 139
- PFIMProject-class, 139
- PGBAlgorithm, 18
- PGBAlgorithm (PGBAlgorithm-class), 139
- PGBAlgorithm-class, 139
- plot, 18
- PlotEvaluation, 18
- PlotEvaluation (PlotEvaluation-class), 140

- plotEvaluation, 140
- plotEvaluation, Evaluation-method (plotEvaluation), 140
- PlotEvaluation-class, 140
- plotFrequencies, 140
- plotFrequencies, FedorovWynnAlgorithm-method (plotFrequencies), 140
- plotFrequencies, Optimization-method (plotFrequencies), 140
- plotOutcomesEvaluation, 11, 141
- plotOutcomesEvaluation, Design-method (plotOutcomesEvaluation), 141
- plotOutcomesGradient, 11, 142
- plotOutcomesGradient, Design-method (plotOutcomesGradient), 142
- plotRSE, 18, 142
- plotRSE, PFIMProject-method (plotRSE), 142
- plotSE, 18, 143
- plotSE, PFIMProject-method (plotSE), 143
- plotSensitivityIndice, 143
- plotSensitivityIndice, Evaluation-method (plotSensitivityIndice), 143
- plotShrinkage, 18, 144
- plotShrinkage, PFIMProject-method (plotShrinkage), 144
- plotWeights, 17, 18, 144
- plotWeights, MultiplicativeAlgorithm-method (plotWeights), 144
- plotWeights, Optimization-method (plotWeights), 144
- PopulationFim, 18
- PopulationFim (PopulationFim-class), 145
- PopulationFim-class, 145
- Proportional, 19, 145
- Proportional (Proportional-class), 145
- Proportional-class, 145
- PSOAlgorithm, 19
- PSOAlgorithm (PSOAlgorithm-class), 146
- PSOAlgorithm-class, 146
- Report, 12, 18, 146
- Report, Evaluation-method (Report), 146
- Report, Optimization-method (Report), 146
- reportTablesAdministration, 11, 147
- reportTablesAdministration, Design-method (reportTablesAdministration), 147
- reportTablesDesign, 11, 147
- reportTablesDesign, Design-method (reportTablesDesign), 147
- reportTablesFIM, 10, 13, 19, 148
- reportTablesFIM, BayesianFim-method (reportTablesFIM), 148
- reportTablesFIM, IndividualFim-method (reportTablesFIM), 148
- reportTablesFIM, PopulationFim-method (reportTablesFIM), 148
- reportTablesModelError, 15, 149
- reportTablesModelError, Model-method (reportTablesModelError), 149
- reportTablesModelParameters, 15, 149
- reportTablesModelParameters, Model-method (reportTablesModelParameters), 149
- reportTablesPlot, 12, 150
- reportTablesPlot, Evaluation-method (reportTablesPlot), 150
- reportTablesSamplingConstraints, 150
- reportTablesSamplingConstraints, Design-method (reportTablesSamplingConstraints), 150
- resizeFisherMatrix, 12, 151
- resizeFisherMatrix, ANY-method (resizeFisherMatrix), 151
- run, 12, 18, 151
- run, Evaluation-method (run), 151
- run, Optimization-method (run), 151
- SamplingTimeConstraints, 19, 24, 47, 68, 71, 78, 79, 91, 178
- SamplingTimeConstraints (SamplingTimeConstraints-class), 152
- SamplingTimeConstraints-class, 152
- SamplingTimes, 19, 176, 177
- SamplingTimes (SamplingTimes-class), 152
- SamplingTimes-class, 152
- setAdministrations, 10, 153
- setAdministrations, Arm-method (setAdministrations), 153
- setArm, 11, 153
- setArm, Design-method (setArm), 153
- setArms, 11, 13, 154
- setArms, Design-method (setArms), 154
- setArms, OptimizationAlgorithm-method (setArms), 154
- setcError, 16, 154

- setcError, ModelError-method  
(setcError), 154
- setContent, 14, 155
- setContent, LibraryOfModels-method  
(setContent), 155
- setDataForArmEvaluation, 155
- setDataForArmEvaluation, Arm-method  
(setDataForArmEvaluation), 155
- setDataForModelEvaluation, 156
- setDataForModelEvaluation, ModelAnalytic-method  
(setDataForModelEvaluation), 156
- setDataForModelEvaluation, ModelAnalyticInfusion-method  
(setDataForModelEvaluation), 156
- setDataForModelEvaluation, ModelAnalyticInfusionSteadyStateFIMResults, Optimization-method  
(setDataForModelEvaluation), 156
- setDataForModelEvaluation, ModelAnalyticSteadyState-method  
(setDataForModelEvaluation), 156
- setDataForModelEvaluation, ModelODEBolus-method  
(setDataForModelEvaluation), 156
- setDataForModelEvaluation, ModelODEDoseInEquation-method  
(setDataForModelEvaluation), 156
- setDataForModelEvaluation, ModelODEDoseNotInEquation-method  
(setDataForModelEvaluation), 156
- setDataForModelEvaluation, ModelODEInfusionDoseInEquation-method  
(setDataForModelEvaluation), 156
- setDerivatives, 16, 157
- setDerivatives, ModelError-method  
(setDerivatives), 157
- setDescription, 14, 157
- setDescription, Model-method  
(setDescription), 157
- setDesigns, 158
- setDesigns, Optimization-method  
(setDesigns), 158
- setDistribution, 17, 158
- setDistribution, ModelParameter-method  
(setDistribution), 158
- setDose, 10, 159
- setDose, Administration-method  
(setDose), 159
- setEquation, 16, 159
- setEquation, ModelError-method  
(setEquation), 159
- setEquations, 14, 160
- setEquations, Model-method  
(setEquations), 160
- setEquationsAfterInfusion, 16, 160
- setEquationsAfterInfusion, Model-method  
(setEquationsAfterInfusion), 160
- setEquationsDuringInfusion, 16, 161
- setEquationsDuringInfusion, Model-method  
(setEquationsDuringInfusion), 161
- setEvaluationFIMResults, 18, 161
- setEvaluationFIMResults, Optimization-method  
(setEvaluationFIMResults), 161
- setEvaluationInitialDesignResults, 18, 162
- setEvaluationInitialDesignResults, Optimization-method  
(setEvaluationInitialDesignResults), 162
- setFim, 11, 162
- setFim, Design-method (setFim), 162
- setFimTypeToString, 13, 163
- setFimTypeToString, Fim-method  
(setFimTypeToString), 163
- setFisherMatrix, 12, 163
- setFisherMatrix, Fim-method  
(setFisherMatrix), 163
- setFixedEffects, 12, 164
- setFixedEffects, Fim-method  
(setFixedEffects), 164
- setFixedMu, 17, 164
- setFixedMu, ModelParameter-method  
(setFixedMu), 164
- setFixedOmega, 17, 165
- setFixedOmega, ModelParameter-method  
(setFixedOmega), 165
- setInitialConditions, 10, 14, 165
- setInitialConditions, Arm-method  
(setInitialConditions), 165
- setInitialConditions, Model-method  
(setInitialConditions), 165
- setIterationAndCriteria, 166
- setIterationAndCriteria, OptimizationAlgorithm-method  
(setIterationAndCriteria), 166
- setModel, 18, 166

- setModel, PFIMProject-method (setModel), 166
- setModelError, 14, 167
- setModelError, Model-method (setModelError), 167
- setModelFromLibrary, 14, 167
- setModelFromLibrary, Model-method (setModelFromLibrary), 167
- setMu, 11, 13, 17, 168
- setMu, Distribution-method (setMu), 168
- setMu, ModelParameter-method (setMu), 168
- setName, 10, 11, 13, 14, 168
- setName, Arm-method (setName), 168
- setName, Design-method (setName), 168
- setName, Model-method (setName), 168
- setNumberOfArms, 11, 169
- setNumberOfArms, Design-method (setNumberOfArms), 169
- setOdeSolverParameters, 14, 169
- setOdeSolverParameters, Model-method (setOdeSolverParameters), 169
- setOmega, 11, 13, 17, 170
- setOmega, Distribution-method (setOmega), 170
- setOmega, ModelParameter-method (setOmega), 170
- setOptimalDesign, 171
- setOptimalDesign, OptimizationAlgorithm-method (setOptimalDesign), 171
- setOptimalWeights, 17, 171
- setOptimalWeights, MultiplicativeAlgorithm-method (setOptimalWeights), 171
- setOptimizationResults, 17, 172
- setOptimizationResults, Optimization-method (setOptimizationResults), 172
- setOutcome, 9, 13, 19, 172
- setOutcome, Administration-method (setOutcome), 172
- setOutcome, SamplingTimes-method (setOutcome), 172
- setOutcomes, 14, 173
- setOutcomes, Model-method (setOutcomes), 173
- setOutcomesEvaluation, 11, 173
- setOutcomesEvaluation, Design-method (setOutcomesEvaluation), 173
- setOutcomesForEvaluation, 14, 174
- setOutcomesForEvaluation, Model-method (setOutcomesForEvaluation), 174
- setOutcomesGradient, 11, 174
- setOutcomesGradient, Design-method (setOutcomesGradient), 174
- setParameters, 11–14, 17–19, 175
- setParameters, Distribution-method (setParameters), 175
- setParameters, FedorovWynnAlgorithm-method (setParameters), 175
- setParameters, Model-method (setParameters), 175
- setParameters, MultiplicativeAlgorithm-method (setParameters), 175
- setParameters, PGB0Algorithm-method (setParameters), 175
- setParameters, PSOAlgorithm-method (setParameters), 175
- setParameters, SimplexAlgorithm-method (setParameters), 175
- setSamplingConstraintForOptimization, 176
- setSamplingConstraintForOptimization, Design-method (setSamplingConstraintForOptimization), 176
- setSamplings, 19, 176
- setSamplings, SamplingTimes-method (setSamplings), 176
- setSamplingTime, 10, 177
- setSamplingTime, Arm-method (setSamplingTime), 177
- setSamplingTimes, 10, 177
- setSamplingTimes, Arm-method (setSamplingTimes), 177
- setSamplingTimesConstraints, 10, 178
- setSamplingTimesConstraints, Arm-method (setSamplingTimesConstraints), 178
- setShrinkage, 10, 13, 19, 178
- setShrinkage, BayesianFim-method (setShrinkage), 178
- setShrinkage, IndividualFim-method (setShrinkage), 178
- setShrinkage, PopulationFim-method (setShrinkage), 178
- setSigmaInter, 16, 179
- setSigmaInter, ModelError-method (setSigmaInter), 179
- setSigmaSlope, 16, 179

setSigmaSlope, ModelError-method  
(setSigmaSlope), 179

setSize, 10, 11, 13, 180

setSize, Arm-method (setSize), 180

setSize, Design-method (setSize), 180

setTau, 10, 181

setTau, Administration-method (setTau),  
181

setTimeDose, 9, 181

setTimeDose, Administration-method  
(setTimeDose), 181

setTinf, 10, 182

setTinf, Administration-method  
(setTinf), 182

setVarianceEffects, 12, 182

setVarianceEffects, Fim-method  
(setVarianceEffects), 182

show, Design-method, 183

show, Evaluation-method  
(show, Design-method), 183

show, FedorovWynnAlgorithm-method  
(show, Design-method), 183

show, MultiplicativeAlgorithm-method  
(show, Design-method), 183

show, Optimization-method  
(show, Design-method), 183

show, PGBOAlgorithm-method  
(show, Design-method), 183

show, PSOAlgorithm-method  
(show, Design-method), 183

show, SimplexAlgorithm-method  
(show, Design-method), 183

SimplexAlgorithm, 19

SimplexAlgorithm  
(SimplexAlgorithm-class), 184

SimplexAlgorithm-class, 184