

Package ‘PFIM’

September 28, 2026

Type Package

Title Population Fisher Information Matrix

Version 8.0

Date 2026-09-28

Maintainer France Mentré <pfim@inserm.fr>

NeedsCompilation yes

Description Evaluate or optimize designs for nonlinear mixed effects models using the Fisher Information matrix. Supports population, individual, and Bayesian 'FIMs', covariates, inter-occasion variability, and D-optimal search ('Fedorov-Wynn', multiplicative, simplex, 'PSO', 'PGB0').

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

Depends R (>= 4.5.0)

SystemRequirements C++17

License GPL-3 | file LICENSE

Encoding UTF-8

VignetteBuilder knitr

Config/build/clean-inst-doc FALSE

Imports utils, Deriv, methods, deSolve, purrr, stringr, S7 (>= 0.2.0), Matrix, ggplot2 (>= 3.5.0), Rcpp, kableExtra, tibble, scales, stats, rmarkdown, rlang, knitr, tools, grDevices

Suggests testthat (>= 3.2.3), withr

LinkingTo Rcpp, RcppArmadillo

Config/testthat/edition 3

Collate 'CovariateModelEquation.R' 'Additive.R' 'Administration.R' 'AdministrationConstraints.R' 'Fim.R' 'Model.R' 'Arm.R' 'model-type-dispatch.R' 'pfim-project-access.R' 'PFIMProject.R' 'pfim-fim-report-render.R' 'BayesianFim.R' 'Covariate.R' 'CategoricalCovariate.R' 'CategoricalCovariateWithIOV.R' 'ModelError.R' 'Combined1.R' 'Combined2.R' 'Constant.R' 'Evaluation.R' 'CovariateTest.R' 'Design.R' 'Distribution.R'

'Exponential.R' 'Optimization.R' 'FedorovWynnAlgorithm.R'
 'IndividualFim.R' 'LibraryOfModels.R' 'LibraryOfPDMModels.R'
 'LibraryOfPKModels.R' 'LogNormal.R' 'ModelODE.R'
 'ModelAnalytic.R' 'ModelInfusion.R' 'ModelAnalyticInfusion.R'
 'ModelAnalyticInfusionSteadyState.R'
 'ModelAnalyticSteadyState.R' 'ModelODEInfusion.R'
 'ModelODEInfusionDoseInEquation.R' 'ModelParameter.R'
 'MultiplicativeAlgorithm.R' 'Normal.R' 'PFIM-package.R'
 'PGBAlgorithm.R' 'PSOAlgorithm.R' 'PopulationFim.R'
 'Proportional.R' 'RcppExports.R' 'SamplingTimeConstraints.R'
 'SamplingTimes.R' 'SimplexAlgorithm.R' 'covariate-test-power.R'
 'covariates-fim-indbayes.R' 'covariates-fim.R'
 'evaluation-accessors.R' 'evaluation-plots.R'
 'evaluation-report-kable.R' 'evaluation-report.R'
 'model-analytic-eval.R' 'model-covariates.R'
 'model-error-variance.R' 'model-gradient.R'
 'model-library-remap.R' 'model-ode-bolus-simulate.R'
 'model-ode-bolus.R' 'model-variance.R' 'pfim-utils.R'
 'pfim-arm-constraints.R' 'pfim-constraint-grid.R'
 'pfim-constraints-helpers.R' 'pfim-continuous-opt.R'
 'pfim-errors.R' 'pfim-model-registry.R' 'pfim-registry.R'
 'pfim-extensions.R' 'pfim-fim-cache.R' 'pfim-fim-labels.R'
 'pfim-fim-optimal-arms.R' 'pfim-fim-optimizer-wiring.R'
 'pfim-flat-sampling-layout.R' 'pfim-gradient-perf.R'
 'pfim-linear-algebra.R' 'pfim-multi-design-opt.R'
 'pfim-plot-perf.R' 'pfim-rd-examples.R' 'pfim-session.R'
 'pfim-subject-fim.R' 'population-fim-variance.R'
 's7-reexports.R' 'zzz.R'

Config/roxygen2/version 8.1.0

Author Romain Leroux [aut] (ORCID: <<https://orcid.org/0009-0009-5779-5303>>),
 France Mentré [cre] (ORCID: <<https://orcid.org/0000-0002-7045-1275>>),
 Antoine Croxo [ctb] (ORCID: <<https://orcid.org/0009-0007-4726-5259>>),
 Jérémy Seurat [ctb]

Repository CRAN

Date/Publication 2026-09-28 14:50:21 UTC

Contents

PFIM-package	5
Additive	6
Administration	7
AdministrationConstraints	8
Arm	8
BayesianFim	10
CategoricalCovariate	11
CategoricalCovariateWithIOV	12
Combined1	13

Combined2	14
Constant	15
Covariate	16
CovariateModelEquation	17
CovariateTest-class	17
Design	18
Distribution	19
Evaluation	20
Exponential	22
FedorovWynnAlgorithm	22
Fim	23
getCorrelationMatrix	24
getCovariateTestTables	25
getDcriterion	25
getDeterminant	26
getEvaluationDesign	27
getFisherMatrix	27
getMixtureDcriterion	28
getModelParametersData	28
getRealisedDcriterion	29
getRSE	29
getSE	30
getShrinkage	31
IndividualFim	31
LibraryOfModels	33
LibraryOfPDMModels	33
LibraryOfPKModels	34
Linear2BolusSingleDose_CIQV1V2	35
Linear2BolusSingleDose_kk12k21V	35
Linear2BolusSteadyState_CIQV1V2tau	36
Linear2BolusSteadyState_kk12k21Vtau	36
Linear2FirstOrderSingleDose_kaCIQV1V2	37
Linear2FirstOrderSingleDose_kakk12k21V	37
Linear2FirstOrderSteadyState_kaCIQV1V2tau	38
Linear2FirstOrderSteadyState_kakk12k21Vtau	38
Linear2InfusionSingleDose_CIQV1V2	39
Linear2InfusionSingleDose_kk12k21V	39
Linear2InfusionSteadyState_CIQV1V2tau	40
Linear2InfusionSteadyState_kk12k21Vtau	40
LogNormal	41
Model	41
ModelAnalytic	43
ModelAnalyticInfusion	45
ModelAnalyticInfusionSteadyState	47
ModelAnalyticSteadyState	49
ModelError	52
ModelInfusion	53
ModelODE	55

ModelODEBolus	57
ModelODEDoseInEquations	59
ModelODEDoseNotInEquations	61
ModelODEInfusion	63
ModelODEInfusionDoseInEquation	65
ModelParameter	67
MultiplicativeAlgorithm	68
Normal	68
Optimization	69
pdModelLibrary	71
pfim-constraints-helpers	71
PFIMProject	72
pfim_get_option	73
pfim_list_extensions	74
pfim_registered_model_classes	75
pfim_register_fim_type	75
pfim_register_model_class	76
pfim_register_optimizer	77
pfim_reset_session	77
pfim_resolve_model_class	78
pfim_set_option	78
PGBOAlgorithm	79
pkModelLibrary	80
plotEvaluation	81
plotFrequencies	81
plotRSE	82
plotSE	83
plotSensitivityIndices	83
plotShrinkage	84
plotWeights	85
PopulationFim	85
projectOf	86
projectProp	87
prop	88
Proportional	88
PSOAlgorithm	89
Report	90
run	91
SamplingTimeConstraints	92
SamplingTimes	93
saveCovariateTest	94
showFIM	94
SimplexAlgorithm	95

PFIM-package

Population Fisher Information for Design Evaluation and Optimization in NLME Models

Description

Nonlinear mixed-effects models (NLMEM) are widely used in model-based drug development. The population Fisher information matrix (FIM) is an efficient alternative to clinical trial simulation for optimizing study designs. **PFIM 8.0** is an R package using the **S7** object system to evaluate and optimize population designs from the FIM.

PFIM includes libraries of PK and PD models (S7 classes): bolus, infusion, first-order absorption, one- and two-compartment structures, linear or Michaelis-Menten elimination, direct and turnover PD models, and combined PK/PD models. Users may also supply custom analytical or ODE models; set `modelClass` explicitly or use `pfim_resolve_model_class`. The FIM is computed by first-order linearization with a block-diagonal structure. Population, individual and Bayesian FIMs are available; the Bayesian FIM provides shrinkage predictions. Covariate effects can be tested with `covariateTest`. Design optimization under the D-criterion uses the simplex algorithm (Nelder-Mead), the multiplicative algorithm, Fedorov-Wynn, PSO (Particle Swarm Optimization) and PGBO (Population Genetics Based Optimizer).

User API

Constructors: `Evaluation`, `Optimization`, `Arm`, `Design`, `ModelParameter`, residual-error and distribution classes, FIM types, and algorithm classes. Verbs: `run`, `Report`, `defineFim`, `definePKModel`, `definePKPDModel`, `getSE`/`getRSE`/`getFisherMatrix`/`getDcriterion`/`getShrinkage`, `plotEvaluation`/`plotSE`/`plotRSE`/`showFIM`, `covariateTest`. Session: `pfim_get_option`, `pfim_set_option`, `pfim_reset_session`. Pipeline helpers (`evaluateDesign`, `optimizeDesign`, ...) are internal.

Documentation

Package source, issues, and vignettes are available at <http://www.pfim.biostat.fr/>.

Session options

`pfim_set_option`/`pfim_get_option`; `pfim_reset_session` clears caches. See `?pfim_set_option` for `fim.cache`, `constraints.maxTasks`, and performance options.

Covariates and inter-occasion variability

Categorical covariates and inter-occasion variability (IOV) are supported for population, individual and Bayesian FIMs; covariate effect tests use the population FIM. The number of occasions is inferred automatically from the model. Results are shown via `show()`, plots (`ggplot2`), and HTML reports.

Author(s)

Author of PFIM 8.0: Romain Leroux <romain.leroux@inserm.fr> ([ORCID](#)).

Contributors: J  r  my Seurat <jeremy.seurat@inserm.fr>, Antoine Croxo <antoine.croxox@inserm.fr>.

Maintainer: France Mentr   <pfim@inserm.fr> ([ORCID](#)).

References

- Dumont C, Lestini G, Le Nagard H, 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.
- Chambers JM. Object-Oriented Programming, Functional Programming and R. *Stat Sci.* 2014;29:167-80.
- Nelder JA, Mead R. A simplex method for function minimization. *Comput J.* 1965;7:308-13.
- Seurat J, Tang Y, Mentr   F, Nguyen TT. Finding optimal design in nonlinear mixed effect models using multiplicative algorithms. *Comput Methods Programs Biomed.* 2021;207:106126.
- Fedorov VV. *Theory of Optimal Experiments.* Academic Press, New York, 1972.
- 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.
- Le Nagard H, Chao L, Tenaillon O. The emergence of complexity and restricted pleiotropy in adapting networks. *BMC Evol Biol.* 2011;11:326.
- Wickham H. *ggplot2: Elegant Graphics for Data Analysis*, Springer-Verlag New York, 2016.

See Also

Useful links:

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

Additive

Additive

Description

Additive covariate link: $\text{theta}(\text{cov}) = \mu * (1 + \text{beta} * \text{cov})$.

Usage

```
Additive(beta = 0, combinedEffect = 0, value = 0)
```

Arguments

beta	Fixed-effect coefficient (default 0).
combinedEffect	Combined covariate effect term (default 0).
value	Computed adjusted value (default 0).

Value

An S7 object of class Additive.

Examples

```
Additive(beta = 1, combinedEffect = 0.2)
```

Administration

Administration

Description

Dosing regimen for one outcome: dose amounts, times, infusion duration, and optional inter-dose interval (τ) for repeated dosing or steady state.

When `timeDose` has length 1 and `dose` is longer, solvers treat that as a shared administration time (see `.alignAdministrationDosing`).

Usage

```
Administration(
  outcome = character(0),
  timeDose = numeric(0),
  dose = numeric(0),
  Tinf = numeric(0),
  tau = 0
)
```

Arguments

outcome	Character string: model output receiving the dose. Library models typically use the catalogue name (typically "RespPK"). An outputs alias is accepted when it maps to a declared state; the dose is applied to that state. User-written ODEs may use the compartment name ("Cc") or that alias. Sampling times may use either the state or the alias.
timeDose	Numeric vector: administration times.
dose	Numeric vector: dose amounts (same length as <code>timeDose</code> , or longer when <code>timeDose</code> has length 1 - replicated for each dose).
Tinf	Numeric vector: infusion duration (0 or omitted for bolus).
tau	Numeric: dosing interval for repeated doses or steady-state models.

Value

An S7 object of class Administration.

Examples

```
Administration(outcome = "RespPK", timeDose = 0, dose = 100)
```

AdministrationConstraints
<i>AdministrationConstraints</i>

Description

Allowed dose values per outcome when optimizing or reporting designs.
Used by generateDosesCombination() to build the discrete dose grid for Fedorov-Wynn / multiplicative algorithms (and constraint reports).

Usage

```
AdministrationConstraints(outcome = character(0), doses = list())
```

Arguments

outcome	Character string: outcome name.
doses	List of admissible dose levels for that outcome.

Value

An S7 object of class AdministrationConstraints.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))  
# help(AdministrationConstraints)
```

Arm	<i>Arm</i>
-----	------------

Description

One experimental arm: subjects, dosing, sampling times, and slots filled during design evaluation (model predictions, gradients, variance, FIM).

Usage

```
Arm(  
  name = character(0),  
  size = numeric(0),  
  administrations = list(),  
  initialConditions = list(),  
  initialCondition = NULL,  
  samplingTimes = list(),
```

```

    administrationsConstraints = list(),
    samplingTimesConstraints = list(),
    evaluationModel = list(),
    evaluationGradients = list(),
    evaluationVariance = list(),
    evaluationFim = NULL
  )

```

Arguments

<code>name</code>	Character string: arm identifier.
<code>size</code>	Numeric: number of subjects in the arm.
<code>administrations</code>	List of Administration objects.
<code>initialConditions</code>	List of ODE initial conditions (name = state). Values may be numeric or a character expression in typical values and dose_<state> (e.g. "dose_Cc/V"). Expressions are evaluated in a sealed environment (no session symbols). List order does not set the deSolve state order (that follows Deriv_* declaration). In event-based bolus mode the value for a compartment dosed at $t = 0$ is replaced by 0: the dose mass enters through the event table.
<code>initialCondition</code>	Alias of <code>initialConditions</code> .
<code>samplingTimes</code>	List of SamplingTimes objects.
<code>administrationsConstraints</code>	List of AdministrationConstraints objects.
<code>samplingTimesConstraints</code>	List of SamplingTimeConstraints objects.
<code>evaluationModel</code>	Model predictions at sampling times (filled by <code>evaluateArm</code>).
<code>evaluationGradients</code>	Gradients of responses w.r.t. parameters (nested if covariates/IOV).
<code>evaluationVariance</code>	Residual variance structure for the arm.
<code>evaluationFim</code>	FIM object after <code>evaluateFim</code> .

Value

An S7 object of class `Arm`.

Examples

```

source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
arm = prop(prop(ev, "designs")[[1L]], "arms")[[1L]]
length(prop(prop(arm, "samplingTimes")[[1L]], "samplings"))

```

BayesianFim

*BayesianFim***Description**

Bayesian Fisher information matrix with shrinkage on random effects.

First-order linearization on the random-effect scale. Covariate beta is not a subject parameter (MAP conditions on β) and is omitted. Across strata / arms, subject FIMs keep their prior and are aggregated by averaging covariances: $\bar{C} = \sum w_s M_s^{-1}$, $M_{\text{eff}} = \bar{C}^{-1}$. A positive weight on a non-identifiable protocol yields Inf SE for the null-space parameters. This covariance mixture is not comparable to summing Fisher matrices (PFIM 6 / PopED on this case).

With IOV ($\gamma > 0$), each occasion uses the marginal FO residual $V_k = R_k + F_k \text{diag}(\gamma^2) F_k^\top$ on the η -scale (equivalent to the η block of the augmented (η, κ) system).

Usage

```
BayesianFim(
  fisherMatrix = numeric(0),
  fixedEffects = numeric(0),
  varianceEffects = numeric(0),
  SEAndRSE = list(),
  condNumberFixedEffects = 0,
  condNumberVarianceEffects = 0,
  shrinkage = numeric(0),
  singularFim = FALSE
)
```

Arguments

<code>fisherMatrix</code>	Labelled FIM matrix.
<code>fixedEffects</code>	Fixed-effects sub-block.
<code>varianceEffects</code>	Variance-effects sub-block.
<code>SEAndRSE</code>	List with SE, RSE, and combined tables.
<code>condNumberFixedEffects</code>	Condition number of the fixed-effects block.
<code>condNumberVarianceEffects</code>	Condition number of the variance-effects block.
<code>shrinkage</code>	Named shrinkage (percent) per parameter (Bayesian FIM).
<code>singularFim</code>	TRUE when SE/RSE used a pseudo-inverse (singular FIM). Set in <code>.fimStoreEvaluationResult</code> from <code>.fimBuildSeAndRse</code> ; D-criterion / log-det still use the raw matrix (may be 0 / -Inf when singular).

Value

A `BayesianFim` object (filled by `run()`).

Examples

```
## Not run:  
vignette("Example01")  
  
## End(Not run)
```

CategoricalCovariate *CategoricalCovariate*

Description

The class `CategoricalCovariate` represents a fixed categorical covariate (constant across occasions): binary or multi-level group membership (sex, genotype, centre, ...).

The constructor validates that `categoriesProportions` sums to 1 (within floating-point tolerance) before creating the object.

Usage

```
CategoricalCovariate(name, categories, categoriesProportions, effects = list())
```

Arguments

<code>name</code>	Covariate identifier string.
<code>categories</code>	Character vector of category labels. The first element is the reference level.
<code>categoriesProportions</code>	Numeric vector of proportions; must sum to 1.
<code>effects</code>	Named list of covariate effects per category.

Value

An S7 object of class `CategoricalCovariate`.

Examples

```
CategoricalCovariate(  
  name = "Sex", categories = c("M", "F"),  
  categoriesProportions = c(0.5, 0.5)  
)
```

CategoricalCovariateWithIOV

CategoricalCovariateWithIOV

Description

The class CategoricalCovariateWithIOV represents a categorical covariate that varies across occasions (inter-occasion variability design).

Typical use: crossover treatment sequences (AB/BA), period effects, etc.

Each element of sequences is a character vector of category labels, one per occasion. sequencesProportions gives the fraction of subjects following each sequence and must sum to 1.

Usage

```
CategoricalCovariateWithIOV(
  name,
  categories,
  sequences,
  sequencesProportions,
  effects = list()
)
```

Arguments

name	Character: covariate identifier.
categories	Character vector of category labels; first element is the reference level.
sequences	Named list of character vectors (one per sequence group); names are auto-generated as "sequence_1", "sequence_2", ...
sequencesProportions	Numeric vector summing to 1.
effects	Named list of covariate effects per category.

Value

An S7 object of class CategoricalCovariateWithIOV.

Examples

```
CategoricalCovariateWithIOV(
  name = "Trt",
  categories = c("A", "B"),
  sequences = list(c("A", "B"), c("B", "A")),
  sequencesProportions = c(0.5, 0.5)
)
```

Combined1

Combined1

Description

Combined residual error (additive + proportional SD): variance $(\sigma_{\text{Inter}} + \sigma_{\text{Slope}} * f^{\{c\text{Error}\}})^2$.

Usage

```
Combined1(
  output = character(0),
  sigmaInter = 0,
  sigmaSlope = 0,
  sigmaInterFixed = FALSE,
  sigmaSlopeFixed = FALSE,
  cError = 1,
  ...
)
```

Arguments

output	Outcome name.
sigmaInter	Additive residual SD.
sigmaSlope	Proportional residual SD.
sigmaInterFixed	If TRUE, sigmaInter is not estimated.
sigmaSlopeFixed	If TRUE, sigmaSlope is not estimated.
cError	Power on the proportional prediction term.
...	Legacy equation/derivatives warn and are ignored.

Value

An S7 object of class Combined1.

Examples

```
Combined1(output = "RespPK", sigmaInter = 0.5, sigmaSlope = 0.1)
```

Combined2

*Combined2***Description**

Combined residual error with independent additive and proportional SDs: variance $\sigma_{\text{Inter}}^2 + (\sigma_{\text{Slope}} * f^{\{c\text{Error}\}})^2$. Matches PopED $y*(1+e_{\text{prop}})+e_{\text{add}}$ when both sigmas are residual SDs.

Usage

```
Combined2(
  output = character(0),
  sigmaInter = 0,
  sigmaSlope = 0,
  sigmaInterFixed = FALSE,
  sigmaSlopeFixed = FALSE,
  cError = 1,
  ...
)
```

Arguments

output	Outcome name.
sigmaInter	Additive residual SD.
sigmaSlope	Proportional residual SD.
sigmaInterFixed	If TRUE, sigmaInter is not estimated.
sigmaSlopeFixed	If TRUE, sigmaSlope is not estimated.
cError	Power on the proportional prediction term.
...	Legacy equation/derivatives warn and are ignored.

Value

An S7 object of class Combined2.

Examples

```
Combined2(output = "RespPK", sigmaInter = 0.5, sigmaSlope = 0.1)
```

Constant	<i>Constant</i>
----------	-----------------

Description

Pure additive residual error: $V = \sigma_{\text{Inter}}^2$ (combined1 with σ_{Slope} locked to 0).

Usage

```
Constant(  
  output = character(0),  
  sigmaInter = 0,  
  sigmaSlope = 0,  
  sigmaInterFixed = FALSE,  
  sigmaSlopeFixed = FALSE,  
  cError = 1,  
  ...  
)
```

Arguments

- output Outcome name.
- sigmaInter Additive residual SD.
- sigmaSlope Proportional residual SD.
- sigmaInterFixed If TRUE, sigmaInter is not estimated.
- sigmaSlopeFixed If TRUE, sigmaSlope is not estimated.
- cError Power on the proportional prediction term.
- ... Legacy equation/derivatives warn and are ignored.

Value

An S7 object of class Constant.

Examples

```
Constant(output = "RespPK", sigmaInter = 0.1)
```

Covariate

*Covariate***Description**

The class `Covariate` is the base class for all covariate types.

The constructor doubles as a smart factory: when `categories` is supplied it dispatches to the correct concrete subclass:

- `sequences + sequencesProportions` -> `CategoricalCovariateWithIOV`
- `categoriesProportions` -> `CategoricalCovariate`
- otherwise -> plain `Covariate`

Usage

```
Covariate(
  name,
  effects = list(),
  categories = NULL,
  categoriesProportions = NULL,
  sequences = NULL,
  sequencesProportions = NULL
)
```

Arguments

<code>name</code>	Character: covariate identifier.
<code>effects</code>	Named list of covariate effects per category. A declared value of 0 still creates a β column in the population FIM (SE under the null); a warning is issued. Omit the parameter to drop it.
<code>categories</code>	(Optional) Character vector of category labels.
<code>categoriesProportions</code>	(Optional) Numeric vector summing to 1.
<code>sequences</code>	(Optional) List of category sequences (IOV).
<code>sequencesProportions</code>	(Optional) Numeric vector summing to 1.

Value

An S7 object of class `Covariate`.

Examples

```
Covariate(name = "Sex", categories = c("M", "F"),
  categoriesProportions = c(0.5, 0.5), effects = list(F = c(V = log(1.2))))
```

CovariateModelEquation	<i>CovariateModelEquation</i>
------------------------	-------------------------------

Description

The class CovariateModelEquation is the base class for the two covariate structural equations supported by PFIM: Additive and Exponential.

A covariate model equation adjusts the population mean for individual i with covariate value

- Additive: $\theta = \mu \cdot (1 + \beta \cdot cov)$
- Exponential: $\theta = \mu \cdot \exp(\beta \cdot cov)$

Concrete subclasses implement computeCovariateValue().

Usage

CovariateModelEquation(beta = 0, combinedEffect = 0, value = 0)

Arguments

beta	Fixed-effect coefficient (default 0).
combinedEffect	Combined covariate effect term (default 0).
value	Computed adjusted value (default 0).

Value

An S7 object of class CovariateModelEquation.

Examples

CovariateModelEquation(beta = 1, combinedEffect = 0.2)

CovariateTest-class	<i>CovariateTest Class</i>
---------------------	----------------------------

Description

CovariateTest stores covariate-test results.

Usage

```
CovariateTest(  
  covariate = data.frame(),  
  parameter = data.frame(),  
  nonRelevance = data.frame(),  
  relevance = data.frame(),  
  settings = list()  
)
```

Arguments

- `covariate` data.frame for β significance.
- `parameter` Deprecated; kept empty (typical μ are not reported).
- `nonRelevance` data.frame for TOST non-relevance on β .
- `relevance` data.frame for clinical relevance on β .
- `settings` List of run/display options.

Details

- covariate** Significance on covariate effects β
- nonRelevance** TOST non-relevance on β
- relevance** Clinical relevance on β
- settings** Run options (tests, thetaL/U, targetPower, alpha, N0)

Value

An S7 object of class CovariateTest.

Examples

```
source(system.file("examples", "covariate-test-minimal.R", package = "PFIM"))  
class(ct)
```

Design	Design
--------	--------

Description

Experimental design: one or more arms, each with dosing and sampling schedules. Filled by `evaluateDesign()` with per-arm FIM results.

Usage

```
Design(  
  name = character(0),  
  size = 0,  
  arms = list(),  
  evaluationArms = list(),  
  numberOfArms = 0,  
  fim = NULL  
)
```

Arguments

name	Character string: design name.
size	Total number of subjects (sum of arm sizes).
arms	List of Arm objects.
evaluationArms	Evaluated arms after evaluateDesign().
numberOfArms	Optional total study size for discrete designs (historical CRAN field; Mult/FW prefer size / arm sizes / numberOfSubjects). Not the count of Arm objects.
fim	Aggregated Fim for the design.

Value

An S7 object of class Design.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))  
design = prop(ev, "designs")[[1L]]  
prop(prop(design, "arms")[[1L]], "size")
```

Distribution	<i>Distribution</i>
--------------	---------------------

Description

Base class for parameter distributions (Normal, LogNormal).

Usage

```
Distribution(name = character(0), mu = 0, omega = 0)
```

Arguments

name	Character string: distribution label.
mu	Typical value on the natural (untransformed) scale. For LogNormal, IIV is additive on $\log \theta$ with $\theta = \mu e^\eta$ at $\eta = 0$.
omega	Standard deviation of random effects (IIV).

Value

An S7 object of class `Distribution`.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
# help(Distribution)
```

Evaluation

Evaluation

Description

Design evaluation: build the model, compute the Fisher information matrix (FIM), and store results per design. Inherits `PFIMProject` fields directly. Use `prop` for project fields on both `Evaluation` and `Optimization`. After `run()`, `getEvaluationDesign` returns per-design results when several designs were evaluated.

Equation strings may use declared `ModelParameter` names, `t`, `dose_*` / `Tinf_*` for administered outcomes, and other outcome names. Free symbols resolve in `knitr::knit_global()` (while knitting) or `globalenv()`. Names that already exist on the search path (`beta`, `gamma`, `pi`) are not reported as missing; declare every model parameter explicitly.

Usage

```
Evaluation(
  evaluationDesign = list(),
  name = character(0),
  modelClass = character(0),
  modelParameters = list(),
  modelCovariates = list(),
  modelCovariatesEquation = character(0),
  modelEquations = list(),
  modelFromLibrary = list(),
  modelError = list(),
  designs = list(),
  outputs = list(),
  fimType = character(0),
  odeSolverParameters = list(),
  numberOfOccasions = NA_real_,
  fim = NULL
)
```

Arguments

<code>evaluationDesign</code>	List of evaluated designs (filled by <code>run()</code>).
<code>name</code>	Character string: project name.
<code>modelClass</code>	Model S7 class name; filled by <code>defineModelType()</code> when empty. Custom classes: pfim_register_model_class .
<code>modelParameters</code>	List of <code>ModelParameter</code> objects.
<code>modelCovariates</code>	List of covariate objects (from <code>Covariate()</code> factory).
<code>modelCovariatesEquation</code>	See PFIMProject (<code>modelCovariatesEquation</code>).
<code>modelEquations</code>	List of model equations (or empty if using the model library).
<code>modelFromLibrary</code>	List selecting a built-in PK/PD model.
<code>modelError</code>	List of residual error model objects.
<code>designs</code>	List of <code>Design</code> objects.
<code>outputs</code>	Named list mapping internal to user output names.
<code>fimType</code>	Character: "population", "individual", or "Bayesian".
<code>odeSolverParameters</code>	List with <code>atol</code> and <code>rtol</code> for ODE solvers. Finite-difference steps assume parameter mus are scaled to $O(1)$.
<code>numberOfOccasions</code>	Integer number of study occasions; NA to infer (see inferNumberOfOccasions). If set, must be consistent with IOV covariate sequences and gamma on parameters.
<code>fim</code>	Fim object filled after <code>run()</code> .

Value

An `Evaluation` object with `evaluationDesign` and `fim` filled.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
```

Exponential	<i>Exponential</i>
-------------	--------------------

Description

Exponential covariate link: $\text{theta}(\text{cov}) = \mu * \exp(\text{beta} * \text{cov})$.

Usage

`Exponential(beta = 0, combinedEffect = 0, value = 0)`

Arguments

- `beta` Fixed-effect coefficient (default 0).
- `combinedEffect` Combined covariate effect term (default 0).
- `value` Computed adjusted value (default 0).

Value

An S7 object of class `Exponential`.

Examples

`Exponential(beta = 1, combinedEffect = 0.2)`

FedorovWynnAlgorithm	<i>FedorovWynnAlgorithm</i>
----------------------	-----------------------------

Description

Fedorov-Wynn exchange algorithm for D-optimal sampling-time design (Rcpp kernel). Pass `elementaryProtocols`, `numberOfSubjects`, and `proportionsOfSubjects` via `optimizerParameters` on [Optimization](#). Optional `delta` (default 1e-4) is the relative optimality gap $\max_i \phi_i \leq p(1 + \delta)$, same role as in the multiplicative algorithm. Each elementary protocol may be a flat numeric vector (full grid row) or a list of per-outcome vectors that concatenate to that row (typical PK/PD). A vignette-style `list(pk, pd)` with a single proportion is accepted as one multi-outcome protocol.

Usage

`FedorovWynnAlgorithm(FedorovWynnAlgorithmOutputs = list())`

Arguments

- `FedorovWynnAlgorithmOutputs`
Output list from the Rcpp routine.

Details

FedorovWynnAlgorithm: S7 class

Value

A FedorovWynnAlgorithm specification object for run(Optimization).

Examples

```
## Not run:
vignette("Example01")

## End(Not run)
```

Fim

Fim

Description

Base class for Fisher information matrices (PopulationFim, IndividualFim, BayesianFim).

Usage

```
Fim(
  fisherMatrix = numeric(0),
  fixedEffects = numeric(0),
  varianceEffects = numeric(0),
  SEAndRSE = list(),
  condNumberFixedEffects = 0,
  condNumberVarianceEffects = 0,
  shrinkage = numeric(0),
  singularFim = FALSE
)
```

Arguments

fisherMatrix	Labelled FIM matrix.
fixedEffects	Fixed-effects sub-block.
varianceEffects	Variance-effects sub-block.
SEAndRSE	List with SE, RSE, and combined tables.
condNumberFixedEffects	Condition number of the fixed-effects block.
condNumberVarianceEffects	Condition number of the variance-effects block.
shrinkage	Named shrinkage (percent) per parameter (Bayesian FIM).
singularFim	TRUE when SE/RSE used a pseudo-inverse (singular FIM). Set in .fimStoreEvaluationResult from .fimBuildSeAndRse; D-criterion / log-det still use the raw matrix (may be 0 / -Inf when singular).

Value

An S7 object holding FIM matrices, SE/RSE tables, and condition numbers.

Examples

```
PopulationFim()
```

getCorrelationMatrix	<i>Retrieve correlation matrix from results</i>
----------------------	---

Description

Retrieve correlation matrix from results

Parameter correlations from the optimal design FIM.

Parameter correlations from the evaluated design FIM.

Usage

```
getCorrelationMatrix(pfimproject, ...)
```

Arguments

pfimproject	A PFIMProject object.
...	Optional method arguments.

Value

Correlation matrix of the parameter estimates.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))  
# help(getCorrelationMatrix)
```

getCovariateTestTables

Build HTML tables from a covariate test result

Description

Build HTML tables from a covariate test result

Usage

```
getCovariateTestTables(covariateTestResult, tests = NULL)
```

Arguments

covariateTestResult

A CovariateTest object.

tests

Optional character vector overriding settings\$tests.

Value

Named list of knitr_kable objects (or NULL if empty). Element significance is the covariate slot (report alias).

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
# help(getCovariateTestTables)
```

getDcriterion

Retrieve D-criterion from project results

Description

Retrieve D-criterion from project results

D-criterion of the optimal design.

D-criterion of the evaluated design.

Usage

```
getDcriterion(pfimproject, ...)
```

Arguments

pfimproject

A PFIMProject object.

...

Optional method arguments.

Value

Numeric D-optimality criterion value.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
getDcriterion(ev)
```

getDeterminant	<i>Retrieve determinant of the Fisher matrix</i>
----------------	--

Description

Retrieve determinant of the Fisher matrix

Determinant of the optimal design FIM.

Determinant of the evaluated design FIM.

Usage

```
getDeterminant(pfimproject, ...)
```

Arguments

pfimproject A PFIMProject object.

... Optional method arguments.

Value

Numeric determinant of the Fisher information matrix.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
getDeterminant(ev)
```

getEvaluationDesign	<i>Per-design evaluation result from a multi-design run().</i>
---------------------	--

Description

Per-design evaluation result from a multi-design run().

Usage

```
getEvaluationDesign(evaluation, design = 1L)
```

Arguments

evaluation	An Evaluation object after run().
design	Design index (integer) or name (character).

Value

One element of evaluationDesign (evaluated design with fim).

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
# help(getEvaluationDesign)
```

getFisherMatrix	<i>Retrieve the Fisher information matrix from results</i>
-----------------	--

Description

Runs setEvaluationFim() so dimnames and SE/RSE match the current model metadata, then returns the matrix blocks used by report helpers, plus singularFim.

Usage

```
getFisherMatrix(pfimproject, ...)
```

Arguments

pfimproject	A PFIMProject object.
...	Optional method arguments.

Value

A list with fisherMatrix, fixedEffects, varianceEffects, and singularFim.
List with fisherMatrix, fixedEffects, varianceEffects, and singularFim.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
getFisherMatrix(ev)
```

getMixtureDcriterion	<i>D-criterion of the continuous mixture over retained Multiplicative cells.</i>
----------------------	--

Description

Differs from getDcriterion() (implemented protocol) when joint cells or Hamilton allocation change the realised design. Requires MultiplicativeAlgorithm results.

Usage

```
getMixtureDcriterion(pfimproject, ...)
```

Arguments

pfimproject	An Optimization from MultiplicativeAlgorithm.
...	Optional method arguments.

Value

Numeric D-criterion of the continuous mixture.

getModelParametersData	<i>Parameter table for reports (mu, omega2, gamma2, fix flags, distribution)</i>
------------------------	--

Description

Parameter table for reports (mu, omega2, gamma2, fix flags, distribution).

Usage

```
getModelParametersData(modelParameter, ...)
```

Arguments

modelParameter	A ModelParameter object.
...	Passed to the S7 method.

Value

A data frame of model parameter values.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
# help(getModelParametersData)
```

getRealisedDcriterion *D-criterion of the single protocol implemented after Multiplicative allocation.*

Description

Matches getDcriterion() on the optimal evaluation. Requires MultiplicativeAlgorithm results.

Usage

```
getRealisedDcriterion(pfimproject, ...)
```

Arguments

pfimproject An Optimization from MultiplicativeAlgorithm.
 ... Optional method arguments.

Value

Numeric D-criterion of the implemented protocol.

getRSE *Retrieve relative standard errors from results*

Description

Retrieve relative standard errors from results
 Relative standard errors from the optimal design FIM (after run()).
 Relative standard errors (%) from the evaluated design FIM.

Usage

```
getRSE(pfimproject, ...)
```

Arguments

pfimproject A PFIMProject object.
 ... Optional method arguments.

Value

A list or matrix of relative standard errors (percent).

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
getRSE(ev)
```

getSE	<i>Retrieve standard errors from results</i>
-------	--

Description

Retrieve standard errors from results
 Standard errors from the optimal design FIM (after run()).
 Standard errors from the evaluated design FIM.

Usage

```
getSE(pfimproject, ...)
```

Arguments

pfimproject A PFIMProject object.
 ... Optional method arguments.

Value

A list or matrix of standard errors for the FIM parameters.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
getSE(ev)
```

getShrinkage	<i>Retrieve shrinkage metrics from results</i>
--------------	--

Description

Retrieve shrinkage metrics from results

Parameter shrinkage from the Bayesian optimal-design FIM.

Parameter shrinkage from a Bayesian evaluation FIM (empty for other types).

Usage

```
getShrinkage(pfimproject, ...)
```

Arguments

pfimproject	A PFIMProject object.
...	Optional method arguments.

Value

Numeric vector or list of Bayesian shrinkage values.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
# help(getShrinkage)
```

IndividualFim	<i>IndividualFim</i>
---------------	----------------------

Description

Individual (subject-level) Fisher information matrix.

Across covariate strata (and arms at design level), subject FIMs are combined by averaging covariances: $\bar{C} = \sum_s w_s M_s^{-1}$, $M_{\text{eff}} = \bar{C}^{-1}$. A positive weight on a non-identifiable protocol yields Inf SE. This mixture is not comparable to summing Fisher matrices (PFIM 6 / PopED). Covariate beta effects are not subject parameters and are omitted. With IOV, each occasion uses $V_k = R_k + F_k \text{diag}(\gamma^2) F_k^\top$ for the μ and residual σ blocks.

Arm-level matrices are per-subject.

Usage

```
IndividualFim(
  fisherMatrix = numeric(0),
  fixedEffects = numeric(0),
  varianceEffects = numeric(0),
  SEAndRSE = list(),
  condNumberFixedEffects = 0,
  condNumberVarianceEffects = 0,
  shrinkage = numeric(0),
  singularFim = FALSE
)
```

Arguments

<code>fisherMatrix</code>	Labelled FIM matrix.
<code>fixedEffects</code>	Fixed-effects sub-block.
<code>varianceEffects</code>	Variance-effects sub-block.
<code>SEAndRSE</code>	List with SE, RSE, and combined tables.
<code>condNumberFixedEffects</code>	Condition number of the fixed-effects block.
<code>condNumberVarianceEffects</code>	Condition number of the variance-effects block.
<code>shrinkage</code>	Named shrinkage (percent) per parameter (Bayesian FIM).
<code>singularFim</code>	TRUE when SE/RSE used a pseudo-inverse (singular FIM). Set in <code>.fimStoreEvaluationResult</code> from <code>.fimBuildSeAndRse</code> ; D-criterion / log-det still use the raw matrix (may be 0 / -Inf when singular).

Details

Block-diagonal structure for a single stratum:

$$M_I = \text{bdiag}(M_\mu, M_\sigma)$$

where $M_\mu = G^\top V^{-1}G$ and M_σ is the residual variance-effects block (with the same V under IOV).

Value

An `IndividualFim` object (filled by `run()`).

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
```

LibraryOfModels	<i>LibraryOfModels</i>
-----------------	------------------------

Description

Registry of built-in PK and PD model definitions.

Subclasses `LibraryOfPKModels` and `LibraryOfPDModels` store named equation lists; `defineModelEquationsFromLibrary` looks them up via the package instances `pkModelLibrary` / `pdModelLibrary`.

Usage

```
LibraryOfModels(models = list())
```

Arguments

`models` List of PK and PD model metadata objects.

Value

An S7 object of class `LibraryOfModels`.

Examples

```
length(prop(pkModelLibrary, "models"))
length(prop(pdModelLibrary, "models"))
```

LibraryOfPDModels	<i>LibraryOfPDModels</i>
-------------------	--------------------------

Description

Built-in pharmacodynamic (PD) model library (Emax, turnover, etc.).

Entries are named lists of equation strings (`RespPD` for immediate response, `Deriv_E` for turnover ODEs). Combined with a PK model via `definePKPDModel()`.

Usage

```
LibraryOfPDModels(models = list())
```

Arguments

`models` Named list of built-in PD model equation definitions.

Value

An S7 object of class LibraryOfPDMODELS.

Examples

```
names(prop(pdModelLibrary, "models"))[1:5]
```

LibraryOfPKModels	<i>LibraryOfPKModels</i>
-------------------	--------------------------

Description

Built-in pharmacokinetic (PK) model library (bolus, infusion, absorption, one- and two-compartment and Michaelis-Menten structures).

Exported Linear* helpers return equation *strings* (not S7 objects): two-compartment forms expand hybrid rates (alpha/beta) and partial-fraction coefficients (A/B) via string substitution. Prefer looking up names in pkModelLibrary for new code.

Usage

```
LibraryOfPKModels(models = list())
```

Arguments

models	Named list of built-in PK model equation definitions.
--------	---

Value

An S7 object of class LibraryOfPKModels.

Examples

```
names(prop(pkModelLibrary, "models"))[1:5]  
nzchar(Linear2BolusSingleDose_C1QV1V2())
```

`Linear2BolusSingleDose_ClQV1V2`*Analytic PK equation: Linear2BolusSingleDose_ClQV1V2.*

Description

Two-compartment IV bolus (Cl, Q, V1, V2 parameterization).

Usage`Linear2BolusSingleDose_ClQV1V2()`**Value**

Character string: closed-form PK equation.

Examples`nzchar(Linear2BolusSingleDose_ClQV1V2())`

`Linear2BolusSingleDose_kk12k21V`*Analytic PK equation: Linear2BolusSingleDose_kk12k21V (micro-constant form).*

Description

Analytic PK equation: Linear2BolusSingleDose_kk12k21V (micro-constant form).

Usage`Linear2BolusSingleDose_kk12k21V()`**Value**

Character string: closed-form PK equation.

Examples`nzchar(Linear2BolusSingleDose_kk12k21V())`

Linear2BolusSteadyState_C1QV1V2tau

Analytic PK equation: Linear2BolusSteadyState_C1QV1V2tau.

Description

Analytic PK equation: Linear2BolusSteadyState_C1QV1V2tau.

Usage

Linear2BolusSteadyState_C1QV1V2tau()

Value

Character string: closed-form PK equation.

Examples

`nzchar(Linear2BolusSteadyState_C1QV1V2tau())`

Linear2BolusSteadyState_kk12k21Vtau

Analytic PK equation: Linear2BolusSteadyState_kk12k21Vtau.

Description

Analytic PK equation: Linear2BolusSteadyState_kk12k21Vtau.

Usage

Linear2BolusSteadyState_kk12k21Vtau()

Value

Character string: closed-form PK equation.

Examples

`nzchar(Linear2BolusSteadyState_kk12k21Vtau())`

`Linear2FirstOrderSingleDose_kaClQV1V2`*Analytic PK equation: Linear2FirstOrderSingleDose_kaClQV1V2.*

Description

Analytic PK equation: Linear2FirstOrderSingleDose_kaClQV1V2.

Usage

`Linear2FirstOrderSingleDose_kaClQV1V2()`

Value

Character string: closed-form PK equation.

Examples

`nzchar(Linear2FirstOrderSingleDose_kaClQV1V2())`

`Linear2FirstOrderSingleDose_kakk12k21V`*Analytic PK equation: Linear2FirstOrderSingleDose_kakk12k21V.*

Description

Analytic PK equation: Linear2FirstOrderSingleDose_kakk12k21V.

Usage

`Linear2FirstOrderSingleDose_kakk12k21V()`

Value

Character string: closed-form PK equation.

Examples

`nzchar(Linear2FirstOrderSingleDose_kakk12k21V())`

Linear2FirstOrderSteadyState_kaClQV1V2tau

Analytic PK equation: Linear2FirstOrderSteadyState_kaClQV1V2tau.

Description

Analytic PK equation: Linear2FirstOrderSteadyState_kaClQV1V2tau.

Usage

Linear2FirstOrderSteadyState_kaClQV1V2tau()

Value

Character string: closed-form PK equation.

Examples

nzchar(Linear2FirstOrderSteadyState_kaClQV1V2tau())

Linear2FirstOrderSteadyState_kakk12k21Vtau

Analytic PK equation: Linear2FirstOrderSteadyState_kakk12k21Vtau.

Description

Analytic PK equation: Linear2FirstOrderSteadyState_kakk12k21Vtau.

Usage

Linear2FirstOrderSteadyState_kakk12k21Vtau()

Value

Character string: closed-form PK equation.

Examples

nzchar(Linear2FirstOrderSteadyState_kakk12k21Vtau())

Linear2InfusionSingleDose_ClQV1V2

Analytic PK equation: Linear2InfusionSingleDose_ClQV1V2.

Description

Analytic PK equation: Linear2InfusionSingleDose_ClQV1V2.

Usage

Linear2InfusionSingleDose_ClQV1V2()

Value

Named list with duringInfusion / afterInfusion equation strings.

Examples

```
nzchar(Linear2InfusionSingleDose_ClQV1V2())$duringInfusion$RespPK)
```

Linear2InfusionSingleDose_kk12k21V

Analytic PK equation: Linear2InfusionSingleDose_kk12k21V.

Description

Analytic PK equation: Linear2InfusionSingleDose_kk12k21V.

Usage

Linear2InfusionSingleDose_kk12k21V()

Value

Named list with duringInfusion / afterInfusion equation strings.

Examples

```
nzchar(Linear2InfusionSingleDose_kk12k21V())$duringInfusion$RespPK)
```

Linear2InfusionSteadyState_ClQV1V2tau

Analytic PK equation: Linear2InfusionSteadyState_ClQV1V2tau.

Description

Analytic PK equation: Linear2InfusionSteadyState_ClQV1V2tau.

Usage

Linear2InfusionSteadyState_ClQV1V2tau()

Value

Named list with duringInfusion / afterInfusion equation strings.

Examples

`nzchar(Linear2InfusionSteadyState_ClQV1V2tau())$duringInfusion$RespPK)`

Linear2InfusionSteadyState_kk12k21Vtau

Analytic PK equation: Linear2InfusionSteadyState_kk12k21Vtau.

Description

Analytic PK equation: Linear2InfusionSteadyState_kk12k21Vtau.

Usage

Linear2InfusionSteadyState_kk12k21Vtau()

Value

Named list with duringInfusion / afterInfusion equation strings.

Examples

`nzchar(Linear2InfusionSteadyState_kk12k21Vtau())$duringInfusion$RespPK)`

LogNormal

LogNormal

Description

Log-normal distribution for positive PK/PD parameters (default in PFIM).

Usage

```
LogNormal(name = character(0), mu = 0, omega = 0)
```

Arguments

name	Character string: distribution label.
mu	Typical value on the natural (untransformed) scale. For LogNormal, IIV is additive on $\log \theta$ with $\theta = \mu e^\eta$ at $\eta = 0$.
omega	Standard deviation of random effects (IIV).

Value

An S7 object of class LogNormal.

Examples

```
LogNormal(mu = 1, omega = 0.3)
```

Model

Model

Description

Internal model object built from a PFIMProject: parameters, covariates, equations, and evaluation hooks (gradients, variance).

Usage

```
Model(
  name = character(0),
  modelParameters = list(),
  modelCovariatesEquation = NULL,
  modelCovariates = list(),
  covariatesEffect = list(),
  covariatesCombination = list(),
  modelParametersWithCovariates = list(),
  numberOfOccasions = 1,
```

```

    omegaWithIOV = numeric(0),
    samplings = numeric(0),
    modelEquations = list(),
    wrapper = function() NULL,
    outputFormula = list(),
    outputNames = character(0),
    variableNames = character(0),
    outcomesWithAdministration = character(0),
    outcomesWithNoAdministration = character(0),
    modelError = list(),
    odeSolverParameters = list(),
    parametersForComputingGradient = list(),
    initialConditions = numeric(0),
    functionArguments = character(0),
    functionArgumentsSymbol = list()
  )

```

Arguments

<code>name</code>	Model name.
<code>modelParameters</code>	List of <code>ModelParameter</code> objects.
<code>modelCovariatesEquation</code>	Additive or Exponential covariate link.
<code>modelCovariates</code>	List of covariate objects.
<code>covariatesEffect</code>	Nested covariate effect structure (internal).
<code>covariatesCombination</code>	Covariate combination table (internal).
<code>modelParametersWithCovariates</code>	Per-combination/occasion parameters (internal).
<code>numberOfOccasions</code>	Number of study occasions (set on the <code>Model</code> in <code>defineModelType()</code> from <code>Evaluation/Optimization</code> ; validated against inferNumberOfOccasions).
<code>omegaWithIOV</code>	IIV/IOV variance vector for population FIM (internal).
<code>samplings</code>	Sampling times for the current arm.
<code>modelEquations</code>	Model equations (analytic or ODE).
<code>wrapper</code>	Compiled or R function wrapper for predictions.
<code>outputFormula</code>	Output expressions per outcome (character strings).
<code>outputNames</code>	Character vector of outcome names.
<code>variableNames</code>	State variable names.
<code>outcomesWithAdministration</code>	Outcomes linked to dosing.
<code>outcomesWithNoAdministration</code>	Outcomes without dosing.

modelError List of residual error models.
 odeSolverParameters atol and rtol for deSolve. Finite-difference gradients assume parameter mus are $O(1)$; values with $|\mu| < 10^{-4}$ use an absolute step floor.
 parametersForComputingGradient Internal FD stencil ($\text{eps}^{\{1/3\}}$ relative steps).
 initialConditions ODE initial conditions.
 functionArguments Names passed to the model function.
 functionArgumentsSymbol Symbolic argument list for parsing.

Value

An S7 object of class Model.

Examples

```
Model()
```

ModelAnalytic

ModelAnalytic

Description

Closed-form (non-ODE) PK/PD model with bolus or explicit dosing.

Usage

```

ModelAnalytic(
  name = character(0),
  modelParameters = list(),
  modelCovariatesEquation = NULL,
  modelCovariates = list(),
  covariatesEffect = list(),
  covariatesCombination = list(),
  modelParametersWithCovariates = list(),
  numberOfOccasions = 1,
  omegaWithIOV = numeric(0),
  samplings = numeric(0),
  modelEquations = list(),
  wrapper = function() NULL,
  outputFormula = list(),
  outputNames = character(0),
  variableNames = character(0),
  outcomesWithAdministration = character(0),

```

```

    outcomesWithNoAdministration = character(0),
    modelError = list(),
    odeSolverParameters = list(),
    parametersForComputingGradient = list(),
    initialConditions = numeric(0),
    functionArguments = character(0),
    functionArgumentsSymbol = list(),
    wrapperModelAnalytic = list(),
    functionArgumentsModelAnalytic = list(),
    functionArgumentsSymbolModelAnalytic = list(),
    solverInputs = list()
)

```

Arguments

<code>name</code>	Model name.
<code>modelParameters</code>	List of <code>ModelParameter</code> objects.
<code>modelCovariatesEquation</code>	Additive or Exponential covariate link.
<code>modelCovariates</code>	List of covariate objects.
<code>covariatesEffect</code>	Nested covariate effect structure (internal).
<code>covariatesCombination</code>	Covariate combination table (internal).
<code>modelParametersWithCovariates</code>	Per-combination/occasion parameters (internal).
<code>numberOfOccasions</code>	Number of study occasions (set on the Model in <code>defineModelType()</code> from Evaluation/Optimization; validated against <code>inferNumberOfOccasions</code>).
<code>omegaWithIOV</code>	IIV/IOV variance vector for population FIM (internal).
<code>samplings</code>	Sampling times for the current arm.
<code>modelEquations</code>	Model equations (analytic or ODE).
<code>wrapper</code>	Compiled or R function wrapper for predictions.
<code>outputFormula</code>	Output expressions per outcome (character strings).
<code>outputNames</code>	Character vector of outcome names.
<code>variableNames</code>	State variable names.
<code>outcomesWithAdministration</code>	Outcomes linked to dosing.
<code>outcomesWithNoAdministration</code>	Outcomes without dosing.
<code>modelError</code>	List of residual error models.
<code>odeSolverParameters</code>	<code>atol</code> and <code>rtol</code> for <code>deSolve</code> . Finite-difference gradients assume parameter must be $O(1)$; values with $ \mu < 10^{-4}$ use an absolute step floor.

`parametersForComputingGradient`
 Internal FD stencil ($\epsilon^{1/3}$ relative steps).
`initialConditions`
 ODE initial conditions.
`functionArguments`
 Names passed to the model function.
`functionArgumentsSymbol`
 Symbolic argument list for parsing.
`wrapperModelAnalytic`
 Wrapper for the analytic solver.
`functionArgumentsModelAnalytic`
 A list with the function arguments of the wrapper.
`functionArgumentsSymbolModelAnalytic`
 A list with the function argument symbols.
`solverInputs` A list with the solver inputs.

Value

An S7 object of class `ModelAnalytic`.

Examples

```
ModelAnalytic()
```

ModelAnalyticInfusion *ModelAnalyticInfusion*

Description

Closed-form model with infusion (dose in equations). Equations come as `duringInfusion` / `afterInfusion` pairs; evaluation superposes past doses (after formula) with the active infusion (during formula) using half-open windows $[t_{\text{dose}}, t_{\text{dose}} + T_{\text{inf}})$.

Usage

```
ModelAnalyticInfusion(
  name = character(0),
  modelParameters = list(),
  modelCovariatesEquation = NULL,
  modelCovariates = list(),
  covariatesEffect = list(),
  covariatesCombination = list(),
  modelParametersWithCovariates = list(),
  numberOfOccasions = 1,
  omegaWithIOV = numeric(0),
  samplings = numeric(0),
```

```

modelEquations = list(),
wrapper = function() NULL,
outputFormula = list(),
outputNames = character(0),
variableNames = character(0),
outcomesWithAdministration = character(0),
outcomesWithNoAdministration = character(0),
modelError = list(),
odeSolverParameters = list(),
parametersForComputingGradient = list(),
initialConditions = numeric(0),
functionArguments = character(0),
functionArgumentsSymbol = list(),
wrapperModelAnalyticInfusion = list(),
functionArgumentsModelAnalyticInfusion = list(),
functionArgumentsSymbolModelAnalyticInfusion = list(),
solverInputs = list()
)

```

Arguments

name	Model name.
modelParameters	List of ModelParameter objects.
modelCovariatesEquation	Additive or Exponential covariate link.
modelCovariates	List of covariate objects.
covariatesEffect	Nested covariate effect structure (internal).
covariatesCombination	Covariate combination table (internal).
modelParametersWithCovariates	Per-combination/occasion parameters (internal).
numberOfOccasions	Number of study occasions (set on the Model in defineModelType() from Evaluation/Optimization; validated against inferNumberOfOccasions).
omegaWithIOV	IIV/IOV variance vector for population FIM (internal).
samplings	Sampling times for the current arm.
modelEquations	Model equations (analytic or ODE).
wrapper	Compiled or R function wrapper for predictions.
outputFormula	Output expressions per outcome (character strings).
outputNames	Character vector of outcome names.
variableNames	State variable names.
outcomesWithAdministration	Outcomes linked to dosing.

outcomesWithNoAdministration	Outcomes without dosing.
modelError	List of residual error models.
odeSolverParameters	atol and rtol for deSolve. Finite-difference gradients assume parameter mus are $O(1)$; values with $ \mu < 10^{-4}$ use an absolute step floor.
parametersForComputingGradient	Internal FD stencil ($\text{eps}^{\{1/3\}}$ relative steps).
initialConditions	ODE initial conditions.
functionArguments	Names passed to the model function.
functionArgumentsSymbol	Symbolic argument list for parsing.
wrapperModelAnalyticInfusion	Wrapper for the analytic solver.
functionArgumentsModelAnalyticInfusion	A list with the function arguments.
functionArgumentsSymbolModelAnalyticInfusion	A list with the function argument symbols.
solverInputs	A list with the solver inputs.

Value

The requested PFIM result.

Examples

```
ModelAnalyticInfusion()
```

```
ModelAnalyticInfusionSteadyState
```

```
ModelAnalyticInfusionSteadyState
```

Description

Analytic infusion model at steady state.

Usage

```
ModelAnalyticInfusionSteadyState(
  name = character(0),
  modelParameters = list(),
  modelCovariatesEquation = NULL,
  modelCovariates = list(),
  covariatesEffect = list(),
```

```

covariatesCombination = list(),
modelParametersWithCovariates = list(),
numberOfOccasions = 1,
omegaWithIOV = numeric(0),
samplings = numeric(0),
modelEquations = list(),
wrapper = function() NULL,
outputFormula = list(),
outputNames = character(0),
variableNames = character(0),
outcomesWithAdministration = character(0),
outcomesWithNoAdministration = character(0),
modelError = list(),
odeSolverParameters = list(),
parametersForComputingGradient = list(),
initialConditions = numeric(0),
functionArguments = character(0),
functionArgumentsSymbol = list(),
wrapperModelAnalyticInfusion = list(),
functionArgumentsModelAnalyticInfusion = list(),
functionArgumentsSymbolModelAnalyticInfusion = list(),
solverInputs = list()
)

```

Arguments

name	Model name.
modelParameters	List of ModelParameter objects.
modelCovariatesEquation	Additive or Exponential covariate link.
modelCovariates	List of covariate objects.
covariatesEffect	Nested covariate effect structure (internal).
covariatesCombination	Covariate combination table (internal).
modelParametersWithCovariates	Per-combination/occasion parameters (internal).
numberOfOccasions	Number of study occasions (set on the Model in defineModelType() from Evaluation/Optimization; validated against inferNumberOfOccasions).
omegaWithIOV	IIV/IOV variance vector for population FIM (internal).
samplings	Sampling times for the current arm.
modelEquations	Model equations (analytic or ODE).
wrapper	Compiled or R function wrapper for predictions.

outputFormula	Output expressions per outcome (character strings).
outputNames	Character vector of outcome names.
variableNames	State variable names.
outcomesWithAdministration	Outcomes linked to dosing.
outcomesWithNoAdministration	Outcomes without dosing.
modelError	List of residual error models.
odeSolverParameters	atol and rtol for deSolve. Finite-difference gradients assume parameter mus are O(1); values with $ \mu < 10^{-4}$ use an absolute step floor.
parametersForComputingGradient	Internal FD stencil ($\text{eps}^{\{1/3\}}$ relative steps).
initialConditions	ODE initial conditions.
functionArguments	Names passed to the model function.
functionArgumentsSymbol	Symbolic argument list for parsing.
wrapperModelAnalyticInfusion	Wrapper for the analytic solver.
functionArgumentsModelAnalyticInfusion	A list with the function arguments.
functionArgumentsSymbolModelAnalyticInfusion	A list with the function argument symbols.
solverInputs	A list with the solver inputs.

Value

An S7 object of class ModelAnalyticInfusionSteadyState.

Examples

```
ModelAnalyticInfusionSteadyState()
```

ModelAnalyticSteadyState

ModelAnalyticSteadyState

Description

Analytic model at steady state (tau in equations).

Usage

```

ModelAnalyticSteadyState(
  name = character(0),
  modelParameters = list(),
  modelCovariatesEquation = NULL,
  modelCovariates = list(),
  covariatesEffect = list(),
  covariatesCombination = list(),
  modelParametersWithCovariates = list(),
  numberOfOccasions = 1,
  omegaWithIOV = numeric(0),
  samplings = numeric(0),
  modelEquations = list(),
  wrapper = function() NULL,
  outputFormula = list(),
  outputNames = character(0),
  variableNames = character(0),
  outcomesWithAdministration = character(0),
  outcomesWithNoAdministration = character(0),
  modelError = list(),
  odeSolverParameters = list(),
  parametersForComputingGradient = list(),
  initialConditions = numeric(0),
  functionArguments = character(0),
  functionArgumentsSymbol = list(),
  wrapperModelAnalytic = list(),
  functionArgumentsModelAnalytic = list(),
  functionArgumentsSymbolModelAnalytic = list(),
  solverInputs = list()
)

```

Arguments

<code>name</code>	Model name.
<code>modelParameters</code>	List of <code>ModelParameter</code> objects.
<code>modelCovariatesEquation</code>	Additive or Exponential covariate link.
<code>modelCovariates</code>	List of covariate objects.
<code>covariatesEffect</code>	Nested covariate effect structure (internal).
<code>covariatesCombination</code>	Covariate combination table (internal).
<code>modelParametersWithCovariates</code>	Per-combination/occasion parameters (internal).

numberOfOccasions	Number of study occasions (set on the Model in <code>defineModelType()</code> from Evaluation/Optimization; validated against <code>inferNumberOfOccasions</code>).
omegaWithIOV	IIV/IOV variance vector for population FIM (internal).
samplings	Sampling times for the current arm.
modelEquations	Model equations (analytic or ODE).
wrapper	Compiled or R function wrapper for predictions.
outputFormula	Output expressions per outcome (character strings).
outputNames	Character vector of outcome names.
variableNames	State variable names.
outcomesWithAdministration	Outcomes linked to dosing.
outcomesWithNoAdministration	Outcomes without dosing.
modelError	List of residual error models.
odeSolverParameters	atol and rtol for <code>deSolve</code> . Finite-difference gradients assume parameter must be $O(1)$; values with $ \mu < 10^{-4}$ use an absolute step floor.
parametersForComputingGradient	Internal FD stencil ($\epsilon^{1/3}$ relative steps).
initialConditions	ODE initial conditions.
functionArguments	Names passed to the model function.
functionArgumentsSymbol	Symbolic argument list for parsing.
wrapperModelAnalytic	Wrapper for the analytic solver.
functionArgumentsModelAnalytic	A list with the function arguments.
functionArgumentsSymbolModelAnalytic	A list with the function argument symbols.
solverInputs	A list with the solver inputs.

Details

ModelAnalyticSteadyState

Value

An S7 object of class `ModelAnalyticSteadyState`.

Examples

```
ModelAnalyticSteadyState()
```

 ModelError

ModelError

Description

Base class for residual error models (Constant, Combined1, Combined2, Proportional).

Variance is determined by sigmaInter (additive SD), sigmaSlope (proportional SD), cError, and varianceForm:

- "combined1": $V = (a + bf^c)^2$
- "combined2": $V = a^2 + (bf^c)^2$ (PopED-style)

A sigma is estimable in the FIM iff it is non-zero and not marked fixed (sigmaInterFixed / sigmaSlopeFixed).

Usage

```
ModelError(
  output = "output",
  sigmaInter = 0,
  sigmaSlope = 0,
  sigmaInterFixed = FALSE,
  sigmaSlopeFixed = FALSE,
  cError = 1,
  varianceForm = "combined1",
  ...
)
```

Arguments

output	Outcome name.
sigmaInter	Additive residual SD.
sigmaSlope	Proportional residual SD.
sigmaInterFixed	If TRUE, sigmaInter is not estimated.
sigmaSlopeFixed	If TRUE, sigmaSlope is not estimated.
cError	Power on the proportional prediction term.
varianceForm	"combined1" or "combined2" (subclasses set this).
...	Legacy equation/derivatives warn and are ignored.

Value

An S7 object of class ModelError.

Examples

```
ModelError()
```

ModelInfusion

ModelInfusion

Description

Base class for infusion administration models.

Usage

```
ModelInfusion(
  name = character(0),
  modelParameters = list(),
  modelCovariatesEquation = NULL,
  modelCovariates = list(),
  covariatesEffect = list(),
  covariatesCombination = list(),
  modelParametersWithCovariates = list(),
  numberOfOccasions = 1,
  omegaWithIOV = numeric(0),
  samplings = numeric(0),
  modelEquations = list(),
  wrapper = function() NULL,
  outputFormula = list(),
  outputNames = character(0),
  variableNames = character(0),
  outcomesWithAdministration = character(0),
  outcomesWithNoAdministration = character(0),
  modelError = list(),
  odeSolverParameters = list(),
  parametersForComputingGradient = list(),
  initialConditions = numeric(0),
  functionArguments = character(0),
  functionArgumentsSymbol = list()
)
```

Arguments

name	Model name.
modelParameters	List of ModelParameter objects.
modelCovariatesEquation	Additive or Exponential covariate link.
modelCovariates	List of covariate objects.

covariatesEffect	Nested covariate effect structure (internal).
covariatesCombination	Covariate combination table (internal).
modelParametersWithCovariates	Per-combination/occasion parameters (internal).
numberOfOccasions	Number of study occasions (set on the Model in <code>defineModelType()</code> from Evaluation/Optimization; validated against <code>inferNumberOfOccasions</code>).
omegaWithIOV	IIV/IOV variance vector for population FIM (internal).
samplings	Sampling times for the current arm.
modelEquations	Model equations (analytic or ODE).
wrapper	Compiled or R function wrapper for predictions.
outputFormula	Output expressions per outcome (character strings).
outputNames	Character vector of outcome names.
variableNames	State variable names.
outcomesWithAdministration	Outcomes linked to dosing.
outcomesWithNoAdministration	Outcomes without dosing.
modelError	List of residual error models.
odeSolverParameters	<code>atol</code> and <code>rtol</code> for <code>deSolve</code> . Finite-difference gradients assume parameter mus are $O(1)$; values with $ \mu < 10^{-4}$ use an absolute step floor.
parametersForComputingGradient	Internal FD stencil ($\text{eps}^{\{1/3\}}$ relative steps).
initialConditions	ODE initial conditions.
functionArguments	Names passed to the model function.
functionArgumentsSymbol	Symbolic argument list for parsing.

Value

An S7 object of class `ModelInfusion`.

Examples

```
ModelInfusion()
```

ModelODE

ModelODE

Description

Base class for models integrated with deSolve.

Usage

```
ModelODE(
  name = character(0),
  modelParameters = list(),
  modelCovariatesEquation = NULL,
  modelCovariates = list(),
  covariatesEffect = list(),
  covariatesCombination = list(),
  modelParametersWithCovariates = list(),
  numberOfOccasions = 1,
  omegaWithIOV = numeric(0),
  samplings = numeric(0),
  modelEquations = list(),
  wrapper = function() NULL,
  outputFormula = list(),
  outputNames = character(0),
  variableNames = character(0),
  outcomesWithAdministration = character(0),
  outcomesWithNoAdministration = character(0),
  modelError = list(),
  odeSolverParameters = list(),
  parametersForComputingGradient = list(),
  initialConditions = numeric(0),
  functionArguments = character(0),
  functionArgumentsSymbol = list()
)
```

Arguments

name	Model name.
modelParameters	List of ModelParameter objects.
modelCovariatesEquation	Additive or Exponential covariate link.
modelCovariates	List of covariate objects.
covariatesEffect	Nested covariate effect structure (internal).

covariatesCombination	Covariate combination table (internal).
modelParametersWithCovariates	Per-combination/occasion parameters (internal).
numberOfOccasions	Number of study occasions (set on the Model in <code>defineModelType()</code> from Evaluation/Optimization; validated against <code>inferNumberOfOccasions</code>).
omegaWithIOV	IIV/IOV variance vector for population FIM (internal).
samplings	Sampling times for the current arm.
modelEquations	Model equations (analytic or ODE).
wrapper	Compiled or R function wrapper for predictions.
outputFormula	Output expressions per outcome (character strings).
outputNames	Character vector of outcome names.
variableNames	State variable names.
outcomesWithAdministration	Outcomes linked to dosing.
outcomesWithNoAdministration	Outcomes without dosing.
modelError	List of residual error models.
odeSolverParameters	<code>atol</code> and <code>rtol</code> for <code>deSolve</code> . Finite-difference gradients assume parameter μ are $O(1)$; values with $ \mu < 10^{-4}$ use an absolute step floor.
parametersForComputingGradient	Internal FD stencil ($\epsilon^{1/3}$ relative steps).
initialConditions	ODE initial conditions.
functionArguments	Names passed to the model function.
functionArgumentsSymbol	Symbolic argument list for parsing.

Value

An S7 object of class `ModelODE`.

Examples

```
ModelODE()
```

ModelODEBolus

ModelODEBolus

Description

ODE bolus model with doses in initial conditions.

Usage

```
ModelODEBolus(
  name = character(0),
  modelParameters = list(),
  modelCovariatesEquation = NULL,
  modelCovariates = list(),
  covariatesEffect = list(),
  covariatesCombination = list(),
  modelParametersWithCovariates = list(),
  numberOfOccasions = 1,
  omegaWithIOV = numeric(0),
  samplings = numeric(0),
  modelEquations = list(),
  wrapper = function() NULL,
  outputFormula = list(),
  outputNames = character(0),
  variableNames = character(0),
  outcomesWithAdministration = character(0),
  outcomesWithNoAdministration = character(0),
  modelError = list(),
  odeSolverParameters = list(),
  parametersForComputingGradient = list(),
  initialConditions = numeric(0),
  functionArguments = character(0),
  functionArgumentsSymbol = list(),
  modelODE = function() NULL,
  doseEvent = list(),
  solverInputs = list()
)
```

Arguments

name	Model name.
modelParameters	List of ModelParameter objects.
modelCovariatesEquation	Additive or Exponential covariate link.
modelCovariates	List of covariate objects.

covariatesEffect	Nested covariate effect structure (internal).
covariatesCombination	Covariate combination table (internal).
modelParametersWithCovariates	Per-combination/occasion parameters (internal).
numberOfOccasions	Number of study occasions (set on the Model in <code>defineModelType()</code> from Evaluation/Optimization; validated against <code>inferNumberOfOccasions</code>).
omegaWithIOV	IIV/IOV variance vector for population FIM (internal).
samplings	Sampling times for the current arm.
modelEquations	Model equations (analytic or ODE).
wrapper	Compiled or R function wrapper for predictions.
outputFormula	Output expressions per outcome (character strings).
outputNames	Character vector of outcome names.
variableNames	State variable names.
outcomesWithAdministration	Outcomes linked to dosing.
outcomesWithNoAdministration	Outcomes without dosing.
modelError	List of residual error models.
odeSolverParameters	<code>atol</code> and <code>rtol</code> for <code>deSolve</code> . Finite-difference gradients assume parameter mus are $O(1)$; values with $ \mu < 10^{-4}$ use an absolute step floor.
parametersForComputingGradient	Internal FD stencil ($\epsilon^{\{1/3\}}$ relative steps).
initialConditions	ODE initial conditions.
functionArguments	Names passed to the model function.
functionArgumentsSymbol	Symbolic argument list for parsing.
modelODE	<code>deSolve</code> right-hand side built at administration time.
doseEvent	Bolus dose event table passed to <code>deSolve</code> .
solverInputs	Reserved; not used for bolus initial-condition dosing.

Value

An S7 object of class `ModelODEBolus`.

Examples

```
ModelODEBolus()
```

ModelODEDoseInEquations

ModelODEDoseInEquations

Description

ODE model with bolus dose terms inside the differential equations.

Usage

```
ModelODEDoseInEquations(
  name = character(0),
  modelParameters = list(),
  modelCovariatesEquation = NULL,
  modelCovariates = list(),
  covariatesEffect = list(),
  covariatesCombination = list(),
  modelParametersWithCovariates = list(),
  numberOfOccasions = 1,
  omegaWithIOV = numeric(0),
  samplings = numeric(0),
  modelEquations = list(),
  wrapper = function() NULL,
  outputFormula = list(),
  outputNames = character(0),
  variableNames = character(0),
  outcomesWithAdministration = character(0),
  outcomesWithNoAdministration = character(0),
  modelError = list(),
  odeSolverParameters = list(),
  parametersForComputingGradient = list(),
  initialConditions = numeric(0),
  functionArguments = character(0),
  functionArgumentsSymbol = list(),
  modelODEDoseInEquations = function() NULL,
  solverInputs = list()
)
```

Arguments

name	Model name.
modelParameters	List of ModelParameter objects.
modelCovariatesEquation	Additive or Exponential covariate link.
modelCovariates	List of covariate objects.

covariatesEffect	Nested covariate effect structure (internal).
covariatesCombination	Covariate combination table (internal).
modelParametersWithCovariates	Per-combination/occasion parameters (internal).
numberOfOccasions	Number of study occasions (set on the Model in <code>defineModelType()</code> from Evaluation/Optimization; validated against <code>inferNumberOfOccasions</code>).
omegaWithIOV	IIV/IOV variance vector for population FIM (internal).
samplings	Sampling times for the current arm.
modelEquations	Model equations (analytic or ODE).
wrapper	Compiled or R function wrapper for predictions.
outputFormula	Output expressions per outcome (character strings).
outputNames	Character vector of outcome names.
variableNames	State variable names.
outcomesWithAdministration	Outcomes linked to dosing.
outcomesWithNoAdministration	Outcomes without dosing.
modelError	List of residual error models.
odeSolverParameters	<code>atol</code> and <code>rtol</code> for <code>deSolve</code> . Finite-difference gradients assume parameter μ are $O(1)$; values with $ \mu < 10^{-4}$ use an absolute step floor.
parametersForComputingGradient	Internal FD stencil ($\epsilon^{1/3}$ relative steps).
initialConditions	ODE initial conditions.
functionArguments	Names passed to the model function.
functionArgumentsSymbol	Symbolic argument list for parsing.
modelODEDoseInEquations	<code>deSolve</code> right-hand side with <code>dose_</code> terms.
solverInputs	Per-outcome administration windows and dose levels.

Value

An S7 object of class `ModelODEDoseInEquations`.

Examples

```
ModelODEDoseInEquations()
```

ModelODEDoseNotInEquations

ModelODEDoseNotInEquations

Description

ODE model with bolus doses added as compartment events.

Usage

```
ModelODEDoseNotInEquations(
  name = character(0),
  modelParameters = list(),
  modelCovariatesEquation = NULL,
  modelCovariates = list(),
  covariatesEffect = list(),
  covariatesCombination = list(),
  modelParametersWithCovariates = list(),
  numberOfOccasions = 1,
  omegaWithIOV = numeric(0),
  samplings = numeric(0),
  modelEquations = list(),
  wrapper = function() NULL,
  outputFormula = list(),
  outputNames = character(0),
  variableNames = character(0),
  outcomesWithAdministration = character(0),
  outcomesWithNoAdministration = character(0),
  modelError = list(),
  odeSolverParameters = list(),
  parametersForComputingGradient = list(),
  initialConditions = numeric(0),
  functionArguments = character(0),
  functionArgumentsSymbol = list(),
  modelODE = function() NULL,
  doseEvent = list(),
  solverInputs = list()
)
```

Arguments

name	Model name.
modelParameters	List of ModelParameter objects.
modelCovariatesEquation	Additive or Exponential covariate link.

modelCovariates	List of covariate objects.
covariatesEffect	Nested covariate effect structure (internal).
covariatesCombination	Covariate combination table (internal).
modelParametersWithCovariates	Per-combination/occasion parameters (internal).
numberOfOccasions	Number of study occasions (set on the Model in <code>defineModelType()</code> from Evaluation/Optimization; validated against <code>inferNumberOfOccasions</code>).
omegaWithIOV	IIV/IOV variance vector for population FIM (internal).
samplings	Sampling times for the current arm.
modelEquations	Model equations (analytic or ODE).
wrapper	Compiled or R function wrapper for predictions.
outputFormula	Output expressions per outcome (character strings).
outputNames	Character vector of outcome names.
variableNames	State variable names.
outcomesWithAdministration	Outcomes linked to dosing.
outcomesWithNoAdministration	Outcomes without dosing.
modelError	List of residual error models.
odeSolverParameters	<code>atol</code> and <code>rtol</code> for <code>deSolve</code> . Finite-difference gradients assume parameter <code>mus</code> are $O(1)$; values with $ \mu < 10^{-4}$ use an absolute step floor.
parametersForComputingGradient	Internal FD stencil ($\text{eps}^{1/3}$ relative steps).
initialConditions	ODE initial conditions.
functionArguments	Names passed to the model function.
functionArgumentsSymbol	Symbolic argument list for parsing.
modelODE	<code>deSolve</code> right-hand side built at administration time.
doseEvent	Bolus dose event table passed to <code>deSolve</code> .
solverInputs	Reserved; not used for compartment-event bolus dosing.

Value

An S7 object of class `ModelODEDoseNotInEquations`.

Examples

```
ModelODEDoseNotInEquations()
```

ModelODEInfusion	<i>ModelODEInfusion</i>
------------------	-------------------------

Description

ODE model with infusion inputs.

Inherits from ModelODE (numerical nature) rather than ModelInfusion, so `S7_inherits(x, ModelODE)` is TRUE for infusion ODEs. Use `.pfimIsInfusionModel()` for the infusion capability.

Usage

```
ModelODEInfusion(
  name = character(0),
  modelParameters = list(),
  modelCovariatesEquation = NULL,
  modelCovariates = list(),
  covariatesEffect = list(),
  covariatesCombination = list(),
  modelParametersWithCovariates = list(),
  numberOfOccasions = 1,
  omegaWithIOV = numeric(0),
  samplings = numeric(0),
  modelEquations = list(),
  wrapper = function() NULL,
  outputFormula = list(),
  outputNames = character(0),
  variableNames = character(0),
  outcomesWithAdministration = character(0),
  outcomesWithNoAdministration = character(0),
  modelError = list(),
  odeSolverParameters = list(),
  parametersForComputingGradient = list(),
  initialConditions = numeric(0),
  functionArguments = character(0),
  functionArgumentsSymbol = list()
)
```

Arguments

<code>name</code>	Model name.
<code>modelParameters</code>	List of ModelParameter objects.
<code>modelCovariatesEquation</code>	Additive or Exponential covariate link.
<code>modelCovariates</code>	List of covariate objects.

covariatesEffect	Nested covariate effect structure (internal).
covariatesCombination	Covariate combination table (internal).
modelParametersWithCovariates	Per-combination/occasion parameters (internal).
numberOfOccasions	Number of study occasions (set on the Model in <code>defineModelType()</code> from Evaluation/Optimization; validated against <code>inferNumberOfOccasions</code>).
omegaWithIOV	IIV/IOV variance vector for population FIM (internal).
samplings	Sampling times for the current arm.
modelEquations	Model equations (analytic or ODE).
wrapper	Compiled or R function wrapper for predictions.
outputFormula	Output expressions per outcome (character strings).
outputNames	Character vector of outcome names.
variableNames	State variable names.
outcomesWithAdministration	Outcomes linked to dosing.
outcomesWithNoAdministration	Outcomes without dosing.
modelError	List of residual error models.
odeSolverParameters	<code>atol</code> and <code>rtol</code> for <code>deSolve</code> . Finite-difference gradients assume parameter mus are $O(1)$; values with $ \mu < 10^{-4}$ use an absolute step floor.
parametersForComputingGradient	Internal FD stencil ($\text{eps}^{\{1/3\}}$ relative steps).
initialConditions	ODE initial conditions.
functionArguments	Names passed to the model function.
functionArgumentsSymbol	Symbolic argument list for parsing.

Value

An S7 object of class `ModelODEInfusion`.

Examples

```
ModelODEInfusion()
```

```
ModelODEInfusionDoseInEquation
      ModelODEInfusionDoseInEquation
```

Description

ODE infusion model with dose terms in the equations.

Usage

```
ModelODEInfusionDoseInEquation(
  name = character(0),
  modelParameters = list(),
  modelCovariatesEquation = NULL,
  modelCovariates = list(),
  covariatesEffect = list(),
  covariatesCombination = list(),
  modelParametersWithCovariates = list(),
  numberOfOccasions = 1,
  omegaWithIOV = numeric(0),
  samplings = numeric(0),
  modelEquations = list(),
  wrapper = function() NULL,
  outputFormula = list(),
  outputNames = character(0),
  variableNames = character(0),
  outcomesWithAdministration = character(0),
  outcomesWithNoAdministration = character(0),
  modelError = list(),
  odeSolverParameters = list(),
  parametersForComputingGradient = list(),
  initialConditions = numeric(0),
  functionArguments = character(0),
  functionArgumentsSymbol = list(),
  modelODE = function() NULL,
  wrapperModelInfusion = list(),
  solverInputs = list()
)
```

Arguments

name	Model name.
modelParameters	List of ModelParameter objects.
modelCovariatesEquation	Additive or Exponential covariate link.

modelCovariates	List of covariate objects.
covariatesEffect	Nested covariate effect structure (internal).
covariatesCombination	Covariate combination table (internal).
modelParametersWithCovariates	Per-combination/occasion parameters (internal).
numberOfOccasions	Number of study occasions (set on the Model in <code>defineModelType()</code> from Evaluation/Optimization; validated against <code>inferNumberOfOccasions</code>).
omegaWithIOV	IIV/IOV variance vector for population FIM (internal).
samplings	Sampling times for the current arm.
modelEquations	Model equations (analytic or ODE).
wrapper	Compiled or R function wrapper for predictions.
outputFormula	Output expressions per outcome (character strings).
outputNames	Character vector of outcome names.
variableNames	State variable names.
outcomesWithAdministration	Outcomes linked to dosing.
outcomesWithNoAdministration	Outcomes without dosing.
modelError	List of residual error models.
odeSolverParameters	<code>atol</code> and <code>rtol</code> for <code>deSolve</code> . Finite-difference gradients assume parameter mus are $O(1)$; values with $ \mu < 10^{-4}$ use an absolute step floor.
parametersForComputingGradient	Internal FD stencil ($\text{eps}^{\{1/3\}}$ relative steps).
initialConditions	ODE initial conditions.
functionArguments	Names passed to the model function.
functionArgumentsSymbol	Symbolic argument list for parsing.
modelODE	An object <code>modelODE</code> .
wrapperModelInfusion	Wrapper for solver.
solverInputs	A list with the solver inputs.

Value

An S7 object of class `ModelODEInfusionDoseInEquation`.

Examples

```
ModelODEInfusionDoseInEquation()
```

ModelParameter	<i>ModelParameter</i>
----------------	-----------------------

Description

One population model parameter: distribution (mu, omega), optional IOV (gamma), and fix flags.

Usage

```
ModelParameter(  
  name = character(0),  
  gamma = 0,  
  distribution = NULL,  
  fixedMu = FALSE,  
  fixedOmega = FALSE,  
  value = numeric(0)  
)
```

Arguments

name	Character string: name of the parameter.
gamma	Numeric: the SD for inter-occasion variability (IOV). γ^2 is the IOV variance component; 0 means no IOV.
distribution	An object of class <code>Distribution</code> for this parameter.
fixedMu	Logical: TRUE if mu is fixed (not estimated).
fixedOmega	Logical: TRUE if omega is fixed (not estimated).
value	Reserved (unused); use <code>prop(distribution, "mu")</code> for reports and FIM.

Value

An S7 object of class `ModelParameter`.

Examples

```
ModelParameter(name = "C1", distribution = LogNormal(mu = 1, omega = 0.3))
```

MultiplicativeAlgorithm
<i>MultiplicativeAlgorithm</i>

Description

Multiplicative weight algorithm for optimal sampling-time allocation (Rcpp kernel). Pass lambda, delta, numberOfIterations, and weightThreshold via optimizerParameters on [Optimization](#).
Outputs store two D-values: mixtureDcriterion is the D of the continuous mixture over retained candidate cells; realisedDcriterion is the D of the single protocol actually implemented. For joint multi-outcome cells (several arms sharing one protocol in the constraint grid), only the best-weighted cell is kept - mixture weights describe the optimisation simplex, not a multi-protocol subject allocation across joint arms.

Usage

```
MultiplicativeAlgorithm(multiplicativeAlgorithmOutputs = list())
```

Arguments

```
multiplicativeAlgorithmOutputs  
  Compact outputs after optimize (retained listArms, weights, D-criteria, algorithmOutput).  
  Filled by optimizeDesign().
```

Value

A MultiplicativeAlgorithm specification object.

Examples

```
## Not run:  
vignette("Example01")  
  
## End(Not run)
```

Normal	<i>Normal</i>
--------	---------------

Description

Normal distribution for model parameters (identity link on the mean).

Usage

```
Normal(name = character(0), mu = 0, omega = 0)
```

Arguments

name	Character string: distribution label.
mu	Typical value on the natural (untransformed) scale. For LogNormal, IIV is additive on $\log \theta$ with $\theta = \mu e^\eta$ at $\eta = 0$.
omega	Standard deviation of random effects (IIV).

Value

An S7 object of class Normal.

Examples

```
Normal(mu = 0, omega = 1)
```

Optimization

Optimization

Description

Design optimization: search over sampling times (or related design variables) using a metaheuristic or exchange algorithm.

Project fields use [prop](#). Several designs: all optimizers (discrete and continuous) optimize each in turn (optimisationAlgorithmOutputs\$perDesign).

Usage

```
Optimization(
  name = character(0),
  modelClass = character(0),
  modelEquations = list(),
  modelCovariatesEquation = character(0),
  modelFromLibrary = list(),
  modelParameters = list(),
  modelCovariates = list(),
  modelError = list(),
  optimizer = character(0),
  optimizerParameters = list(),
  outputs = list(),
  designs = list(),
  fimType = character(0),
  fim = NULL,
  odeSolverParameters = list(),
  numberOfOccasions = NA_real_,
  project = NULL,
  optimisationDesign = list(),
  optimisationAlgorithmOutputs = list()
)
```

Arguments

<code>name</code>	Character string: project name.
<code>modelClass</code>	Model S7 class name; filled by <code>defineModelType()</code> when empty. Custom classes: pfim_register_model_class .
<code>modelEquations</code>	List of model equations (or empty if using the model library).
<code>modelCovariatesEquation</code>	Character: "additive" or "exponential".
<code>modelFromLibrary</code>	List selecting a built-in PK/PD model.
<code>modelParameters</code>	List of <code>ModelParameter</code> objects.
<code>modelCovariates</code>	List of covariate objects (from <code>Covariate()</code> factory).
<code>modelError</code>	List of residual error model objects.
<code>optimizer</code>	Character: optimization algorithm name (for <code>Optimization</code>).
<code>optimizerParameters</code>	See PFIMProject ; validated when optimizer names a built-in algorithm.
<code>outputs</code>	Named list mapping internal to user output names.
<code>designs</code>	List of <code>Design</code> objects.
<code>fimType</code>	Character: "population", "individual", or "Bayesian".
<code>fim</code>	<code>Fim</code> object filled after <code>run()</code> .
<code>odeSolverParameters</code>	List with <code>atol</code> and <code>rtol</code> for ODE solvers. Finite-difference steps assume parameter <code>mus</code> are scaled to $O(1)$.
<code>numberOfOccasions</code>	Integer number of study occasions; NA to infer (see inferNumberOfOccasions). If set, must be consistent with IOV covariate sequences and gamma on parameters.
<code>project</code>	Nested <code>PFIMProject</code> (filled by the constructor).
<code>optimisationDesign</code>	List with initial and optimal design evaluations.
<code>optimisationAlgorithmOutputs</code>	Raw outputs from the optimization algorithm.

Value

An `Optimization` object; after `run()`, see `optimisationDesign` and `optimisationAlgorithmOutputs`.

Examples

```
## Not run:
vignette("Example01")

## End(Not run)
```

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
class(ev)
```

pdModelLibrary	<i>Default built-in PD model library instance.</i>
----------------	--

Description

Default built-in PD model library instance.

Usage

```
pdModelLibrary
```

Format

An S7 object of class LibraryOfPDModels.

Value

A LibraryOfPDModels object.

See Also

[LibraryOfPDModels](#) for the class constructor.

Examples

```
length(prop(pdModelLibrary, "models"))
```

pfim-constraints-helpers	<i>Constraint tables for optimization reports.</i>
--------------------------	--

Description

Builds kableExtra tables of arm-level dose/sampling constraints for discrete (Fedorov-Wynn / Multiplicative) and continuous (PSO / PGBO / Simplex) optimizers. Uses getArmConstraints() from pfim-arm-constraints.R.

PFIMProject

PFIMProject

Description

Base S7 class for PFIM design evaluation and optimization projects.

Set numberOfOccasions on Evaluation or Optimization (optional: NA infers from covariates and IOV). When set explicitly, it must match [inferNumberOfOccasions](#). The resolved value is stored on the Model in defineModelType().

All fimType values support covariates and IOV; population adds omega and gamma blocks to the FIM.

Usage

```
PFIMProject(
  name = character(0),
  modelClass = character(0),
  modelEquations = list(),
  modelCovariatesEquation = character(0),
  modelFromLibrary = list(),
  modelParameters = list(),
  modelCovariates = list(),
  modelError = list(),
  optimizer = character(0),
  optimizerParameters = list(),
  outputs = list(),
  designs = list(),
  fimType = character(0),
  fim = NULL,
  odeSolverParameters = list(),
  numberOfOccasions = NA_real_,
  cacheScope = ""
)
```

Arguments

name	Character string: project name.
modelClass	Model S7 class name; filled by defineModelType() when empty. Custom classes: pfim_register_model_class .
modelEquations	List of model equations (or empty if using the model library).
modelCovariatesEquation	Character: "additive" or "exponential".
modelFromLibrary	List selecting a built-in PK/PD model.

modelParameters	List of ModelParameter objects.
modelCovariates	List of covariate objects (from Covariate() factory).
modelError	List of residual error model objects.
optimizer	Character: optimization algorithm name (for Optimization).
optimizerParameters	List of algorithm-specific settings passed to run(Optimization). Names and types are checked at construction for built-in optimizers; see MultiplicativeAlgorithm , FedorovWynnAlgorithm , etc. Do not set these on the *Algorithm object - that object only stores outputs after run().
outputs	Named list mapping internal to user output names.
designs	List of Design objects.
fimType	Character: "population", "individual", or "Bayesian".
fim	Fim object filled after run().
odeSolverParameters	List with atol and rtol for ODE solvers. Finite-difference steps assume parameter mus are scaled to O(1).
numberOfOccasions	Integer number of study occasions; NA to infer (see inferNumberOfOccasions). If set, must be consistent with IOV covariate sequences and gamma on parameters.
cacheScope	Reserved (internal FIM cache id).

Value

An S7 object of class PFIMProject.

Examples

```
PFIMProject(name = "demo")
```

pfim_get_option	<i>Read a PFIM session option</i>
-----------------	-----------------------------------

Description

Read a PFIM session option

Usage

```
pfim_get_option(name, default = NULL)
```

Arguments

name	Option name, with or without PFIM. prefix.
default	Value when the option is unset.

Value

Option value.

See Also

[pfim_set_option](#), [pfim_reset_session](#)

Examples

```
pfim_get_option("fim.cache.maxEntries")
```

pfim_list_extensions *List registered extensions*

Description

List registered extensions

Usage

```
pfim_list_extensions()
```

Value

A list of registered extension names.

Examples

```
pfim_list_extensions()
```

pfim_registered_model_classes
Registered model class names

Description

Registered model class names

Usage

```
pfim_registered_model_classes()
pfim_registered_fim_types()
pfim_registered_optimizers()
```

Value

A character vector of registered class names.

Examples

```
pfim_registered_model_classes()
pfim_registered_fim_types()
pfim_registered_optimizers()
```

pfim_register_fim_type
Register a custom FIM type

Description

Extends PFIM with a new fimType string for [Evaluation](#) / [Optimization](#). The factory is called whenever a project needs a fresh FIM of that type. Overwrites silently if name already exists.

Usage

```
pfim_register_fim_type(name, factory, duplicator = .fimDuplicateReset)
```

Arguments

name	Character label (e.g. "population").
factory	Zero-argument function returning a Fim object.
duplicator	Optional function(fim) for design-level FIM copies; defaults to .fimDuplicateReset.

Value

Invisibly, NULL.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))  
# help(pfim_register_fim_type)
```

pfim_register_model_class

Register a model class

Description

Register a model class

Usage

```
pfim_register_model_class(className, factory, detect = NULL)
```

Arguments

className	Character S7 class name (e.g. "ModelAnalytic").
factory	Zero-argument function returning a Model object.
detect	Optional function(pfimproject) returning logical. Legacy function(equations, initialConditions) is still accepted.

Value

Invisibly, NULL.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))  
# help(pfim_register_model_class)
```

pfim_register_optimizer

Register a custom optimizer

Description

Makes optimizer = name valid on [Optimization](#). The factory must return an S7 optimizer object with an optimizeDesign method. Overwrites silently if name already exists.

Usage

```
pfim_register_optimizer(name, factory)
```

Arguments

name	Character optimizer name (e.g. "PSOAlgorithm").
factory	Zero-argument function returning an optimizer object.

Value

Invisibly, NULL.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
# help(pfim_register_optimizer)
```

pfim_reset_session

Reset PFIM session options and clear all computed-data caches.

Description

Clears .pfimSession options and empties the design FIM cache (.pfimFimDesignCache), covariate/occasion cache, eval-model rebuild cache, and gradient/ODE performance caches (FD scheme, admin, ODE time grids). Call between unrelated runs in a long-lived R session (Shiny, services).

Usage

```
pfim_reset_session(closeDevices = FALSE)
```

Arguments

closeDevices	Logical. If TRUE, close all open graphics devices except the null device. Default FALSE.
--------------	--

Value

Invisibly NULL.

Examples

```
pfim_reset_session()
```

```
pfim_resolve_model_class
```

Resolve model class name for a project

Description

Resolve model class name for a project

Usage

```
pfim_resolve_model_class(pfimproject)
```

Arguments

pfimproject A PFIMProject, Evaluation, or Optimization.

Value

Character class name with attribute source ("registry" or "legacy").

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
# help(pfim_resolve_model_class)
```

```
pfim_set_option
```

Set PFIM session options

Description

Options apply for the current R session until [pfim_reset_session](#).

Usage

```
pfim_set_option(...)
```

Arguments

... Named option values. Common options:

`fim.cache` Cache FIM results during optimization (default TRUE).

`fim.cache.maxEntries` LRU cap on design and covariate/occasion cache entries (default 2048L). Set NULL for no limit.

`eval.batch` Batch model rebuilds on constraint grids (default FALSE; set automatically during [generateFimsFromConstraints](#)).

`constraints.maxTasks` NULL (default) evaluates every cell of the dose $\times$ sampling grid in discrete optimization. Set to a positive integer to cap the number of FIM evaluations: cells are subsampled deterministically (evenly spaced within each dose stratum), so Fedorov-Wynn and Multiplicative runs remain reproducible without touching the global RNG.

`perf.adminCache`, `perf.fdCache`, `perf.odeTimesCache` ODE / FD caches (default TRUE).

`perf.odeTimesCache.maxEntries` LRU cap on the C++ ODE sim-time cache (default 2048L). Set NULL for no limit.

`verbose` Extra messages (default FALSE).

Value

Invisibly NULL.

See Also

[pfim_get_option](#), [pfim_reset_session](#)

Examples

```
pfim_set_option(fim.cache.maxEntries = 2048L)
```

PGBOAlgorithm

PGBOAlgorithm

Description

Population-based global optimization for sampling times (Rcpp kernel). Pass `N`, `muteEffect`, `maxIteration`, `purgeIteration`, and `seed` via `optimizerParameters` on [Optimization](#). `muteEffect` is an additive mutation scale in the same time unit as the sampling windows (e.g. hours), not a ratio in $[0, 1]$. Search starts from each outcome's `initialSamplings`; `seed` controls mutation draws only. Optional `fitBase` (default 0.03) and `cauchyProb` (default 0.8, probability of a Cauchy jump vs Gaussian group mutation). Optional `tolerance` (default 0, stall stop disabled $\rightarrow$ converged = NA) and `stallIterations` (default 5) enable early stopping when `tolerance > 0`.

Usage

```
PGBOAlgorithm(optimizerOutputs = list())
```

Arguments

optimizerOutputs

List filled by optimizeDesign() (optimal arms, etc.).

Value

A PGBOAlgorithm specification object.

Examples

```
## Not run:  
vignette("Example02")  
  
## End(Not run)
```

pkModelLibrary

Default built-in PK model library instance.

Description

Default built-in PK model library instance.

Usage

pkModelLibrary

Format

An S7 object of class LibraryOfPKModels.

Value

A LibraryOfPKModels object.

See Also

[LibraryOfPKModels](#) for the class constructor.

Examples

```
length(prop(pkModelLibrary, "models"))
```

plotEvaluation	<i>Plot evaluation outputs for a project</i>
----------------	--

Description

Plot evaluation outputs for a project
Predicted responses for the optimal design evaluation.
Predicted responses with sampling markers for all designs/arms.

Usage

```
plotEvaluation(pfimproject, ...)
```

Arguments

pfimproject	A PFIMProject object.
...	Method-specific arguments. For Evaluation and Optimization, plotOptions list with unitTime, unitOutcomes. Response curves are densified for display with sampling markers at design times.

Value

A nested list of ggplot2 plot objects (by design / arm / outcome).

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))  
names(plotEvaluation(ev, list()))
```

plotFrequencies	<i>Plot optimized sampling frequencies</i>
-----------------	--

Description

Plot optimized sampling frequencies
Fedorov-Wynn optimal frequencies

Usage

```
plotFrequencies(optimization, ...)
```

Arguments

optimization	An Optimization object.
...	Optional method arguments.

Value

A ggplot2 plot object.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
# help(plotFrequencies)
```

plotRSE

Plot relative standard errors for project results

Description

Plot relative standard errors for project results

RSE bar chart for the optimal design FIM.

RSE bar chart for the primary evaluation FIM.

Usage

```
plotRSE(pfimproject, ...)
```

Arguments

pfimproject A PFIMProject object.

... Optional method arguments.

Value

A ggplot2 plot object.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
inherits(plotRSE(ev), "ggplot")
```

plotSE	<i>Plot standard errors for project results</i>
--------	---

Description

Plot standard errors for project results
 SE bar chart for the optimal design FIM.
 SE bar chart for the primary evaluation FIM.

Usage

```
plotSE(pfimproject, ...)
```

Arguments

pfimproject	A PFIMProject object.
...	Optional method arguments.

Value

A ggplot2 plot object.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
inherits(plotSE(ev), "ggplot")
```

plotSensitivityIndices	<i>Plot sensitivity indices for a project</i>
------------------------	---

Description

Plot sensitivity indices for a project
 Sensitivity-index plots for the optimal design evaluation.
 Sensitivity indices over time for all designs/arms.

Usage

```
plotSensitivityIndices(pfimproject, ...)
```

Arguments

pfimproject A PFIMProject object.
 ... Method-specific arguments. For Evaluation, plotOptions list with unitTime, unitOutcomes. Gradients are plotted at design sampling times.

Value

A ggplot2 plot object.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
names(plotSensitivityIndices(ev, list()))
```

plotShrinkage	<i>Plot Bayesian shrinkage from a FIM</i>
---------------	---

Description

Plot Bayesian shrinkage from a FIM
 Default method for BayesianFim.

Usage

```
plotShrinkage(fim, evaluation, ...)
```

Arguments

fim First argument of generic.
 evaluation PFIMProject providing model parameter labels.
 ... Graphics parameters passed to ggplot2.

Value

ggplot object showing Bayesian shrinkage by parameter.

Examples

```
## Not run:
source(system.file("examples", "covariate-test-minimal.R", package = "PFIM"))
# plotShrinkage(prop(ev, "fim"), ev) # BayesianFim

## End(Not run)
```

plotWeights	<i>Plot optimized arm weights</i>
-------------	-----------------------------------

Description

Plot optimized arm weights
 Multiplicative algorithm weights by iteration

Usage

```
plotWeights(optimization, ...)
```

Arguments

optimization	An Optimization object.
...	Optional method arguments.

Value

A ggplot2 plot object.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
# help(plotWeights)
```

PopulationFim	<i>PopulationFim</i>
---------------	----------------------

Description

Population Fisher information matrix for nonlinear mixed-effects models.

Usage

```
PopulationFim(
  fisherMatrix = numeric(0),
  fixedEffects = numeric(0),
  varianceEffects = numeric(0),
  SEAndRSE = list(),
  condNumberFixedEffects = 0,
  condNumberVarianceEffects = 0,
  shrinkage = numeric(0),
  singularFim = FALSE
)
```

Arguments

fisherMatrix	Labelled FIM matrix.
fixedEffects	Fixed-effects sub-block.
varianceEffects	Variance-effects sub-block.
SEAndRSE	List with SE, RSE, and combined tables.
condNumberFixedEffects	Condition number of the fixed-effects block.
condNumberVarianceEffects	Condition number of the variance-effects block.
shrinkage	Named shrinkage (percent) per parameter (Bayesian FIM).
singularFim	TRUE when SE/RSE used a pseudo-inverse (singular FIM). Set in .fimStoreEvaluationResult from .fimBuildSeAndRse; D-criterion / log-det still use the raw matrix (may be 0 / -Inf when singular).

Details

The population FIM has fixed-effects (μ, β) and variance-effects $(\omega^2, \gamma^2, \sigma)$ blocks:

$$M_P = \text{bdiag}(N \cdot M_\mu, N \cdot M_\lambda)$$

where N is the arm size and $M_\mu = G_\mu^\top V^{-1} G_\mu$.

Value

A PopulationFim object (filled by run()).

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
S7::S7_inherits(prop(ev, "fim"), PopulationFim)
```

projectOf

Nested PFIMProject for an Optimization

Description

Nested PFIMProject for an Optimization

Usage

```
projectOf(x)
```

Arguments

x An Evaluation, Optimization, or PFIMProject object.

Value

The underlying PFIMProject.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
projectOf(ev)
```

projectProp	<i>Read a project field</i>
-------------	-----------------------------

Description

Read a project field

Write a project field

Usage

```
projectProp(x, name)
projectProp(x, name) <- value

projectProp(x, name) <- value
```

Arguments

x	An Evaluation, Optimization, or PFIMProject object.
name	Character name of the project property.
value	Value to assign.

Value

The requested project property value, or the modified object for replacement.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
projectProp(ev, "name")
projectProp(ev, "name") = "renamed"
```

prop	<i>Property accessor (S7); project fields on Optimization read slot project.</i>
------	--

Description

Property accessor (S7); project fields on Optimization read slot project.

Usage

```
prop(object, name)
prop(object, name) <- value

prop(object, name) <- value
```

Arguments

object	An S7 object.
name	Property name.
value	Value to assign.

Value

The requested property value, or the modified object for replacement.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
prop(ev, "name")
prop(ev, "name") = "renamed"
```

Proportional	<i>Proportional</i>
--------------	---------------------

Description

Pure proportional residual error: $V = (\sigma_{Slope} * f^{cError})^2$ (combined1 with sigmaInter locked to 0).

Usage

```
Proportional(  
  output = character(0),  
  sigmaInter = 0,  
  sigmaSlope = 0,  
  sigmaInterFixed = FALSE,  
  sigmaSlopeFixed = FALSE,  
  cError = 1,  
  ...  
)
```

Arguments

output	Outcome name.
sigmaInter	Additive residual SD.
sigmaSlope	Proportional residual SD.
sigmaInterFixed	If TRUE, sigmaInter is not estimated.
sigmaSlopeFixed	If TRUE, sigmaSlope is not estimated.
cError	Power on the proportional prediction term.
...	Legacy equation/derivatives warn and are ignored.

Value

An S7 object of class Proportional.

Examples

```
Proportional(output = "RespPK", sigmaSlope = 0.15)
```

PSOAlgorithm	<i>PSOAlgorithm</i>
--------------	---------------------

Description

Particle Swarm Optimization for sampling-time search (Rcpp kernel in src/PSOKernel.cpp). Pass maxIteration, populationSize, seed, and learning coefficients via optimizerParameters on [Optimization](#). Search starts from each outcome's initialSamplings (reported initial design); seed controls swarm exploration after that start. Optional tolerance (default 0, stall stop disabled -> converged = NA) and stallIterations (default 5) enable early stopping when tolerance > 0.

Usage

```
PSOAlgorithm(optimizerOutputs = list())
```

Arguments

optimizerOutputs

List filled by optimizeDesign() (optimal arms, etc.).

Value

A PSOAlgorithm specification object.

Examples

```
## Not run:
vignette("Example02")

## End(Not run)
```

Report

Generate an HTML evaluation or optimization report

Description

Requires **rmarkdown** and pandoc (<https://pandoc.org>).

Rebuilds kable tables and plots for both the initial and optimal evaluations, then hands everything to generateReportOptimization() with the FIM-type R Markdown template.

Assembles model / design / FIM kables and response / SI / SE / RSE plots, then dispatches to the FIM-type R Markdown template via generateReportEvaluation(). Administration / initial-design tables use design 1 only; response and SI plots cover every design in designs.

Usage

```
Report(pfimproject, ...)
```

Arguments

pfimproject A PFIMProject, Evaluation, or Optimization object after run().
 ... outputPath, outputFile, plotOptions for HTML reports.

Value

Invisibly, the path or result from the report generator.

Examples

```
## Not run:
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
tmp = tempfile(fileext = ".html")
Report(ev, outputPath = tempdir(), outputFile = basename(tmp), plotOptions = list())

## End(Not run)
```

run*Run a PFIM evaluation or optimization project*

Description

Pipeline: set FIM cache scope -> rebuild model (with finite differences) -> instantiate FIM type -> evaluateDesign() per design -> post-process the primary FIM via setEvaluationFim(). Additional designs remain in evaluationDesign; use getEvaluationDesign() to retrieve them.

Pipeline: invalidate model cache -> instantiate algorithm -> define FIM -> resolve library equations if needed -> optimizeDesign(). Results land in optimisationDesign (initial + optimal evaluations) and optimisationAlgorithmOutputs.

Usage

```
run(pfimproject, ...)
```

Arguments

pfimproject	An Optimization object.
...	Optional method arguments.

Value

The same Evaluation with evaluationDesign and fim updated.

The same Optimization with optimal design evaluations stored.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
getDeterminant(ev)

source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))

## Not run:
vignette("Example01")

## End(Not run)
```

SamplingTimeConstraints

SamplingTimeConstraints

Description

Constraints on sampling times for design optimization (windows, fixed times, number of points to optimize, minimum spacing).

Metaheuristics (Simplex / PSO / PGB0) draw candidates with `generateSamplingsFromSamplingConstraints()` and reject infeasible ones via `checkSamplingTimeConstraintsForMetaheuristic()`. Discrete algorithms instead enumerate combinations with `generateSamplingTimesCombination()`.

Usage

```
SamplingTimeConstraints(
  outcome = character(0),
  initialSamplings = numeric(0),
  fixedTimes = numeric(0),
  numberOfsamplingsOptimisable = 0,
  numberOfSamplingsOptimisable = NULL,
  samplingsWindows = list(),
  numberOfTimesByWindows = numeric(0),
  minSampling = numeric(0)
)
```

Arguments

<code>outcome</code>	Character string: outcome name.
<code>initialSamplings</code>	Initial sampling-time vector. Continuous optimizers start from this vector (not from <code>Arm</code> <code>samplingTimes</code>); a warning is issued when the two differ. When <code>samplingsWindows</code> is set, <code>initialSamplings</code> must already satisfy the window counts and <code>minSampling</code> .
<code>fixedTimes</code>	Times that must remain fixed during optimization.
<code>numberOfsamplingsOptimisable</code>	Total protocol size including <code>fixedTimes</code> , not the number of free times ($k = \text{numberOfsamplingsOptimisable} - \text{length}(\text{fixedTimes})$). Historical property name (s lowercase); <code>numberOfSamplingsOptimisable</code> is accepted as an alias.
<code>numberOfSamplingsOptimisable</code>	Alias of <code>numberOfsamplingsOptimisable</code> .
<code>samplingsWindows</code>	List of time windows, in increasing time order.
<code>numberOfTimesByWindows</code>	Number of samples per window.
<code>minSampling</code>	Minimum spacing between consecutive samples.

Value

An S7 object of class `SamplingTimeConstraints`.

Examples

```
SamplingTimeConstraints(
  outcome = "RespPK",
  initialSamplings = c(0.5, 2, 8),
  numberOfSamplingsOptimisable = 2
)
```

SamplingTimes

SamplingTimes

Description

Observation times for one outcome within an arm.

Times are in the model's time unit (often hours). Multiple `SamplingTimes` objects on an arm cover multi-response designs.

Usage

```
SamplingTimes(outcome = character(0), samplings = numeric(0))
```

Arguments

`outcome` Character string: outcome name.

`samplings` Numeric vector: sampling times (hours or model time unit).

Value

An S7 object of class `SamplingTimes`.

Examples

```
SamplingTimes(outcome = "RespPK", samplings = c(1, 4, 8))
```

saveCovariateTest	<i>Save covariate test results to a text file</i>
-------------------	---

Description

Save covariate test results to a text file

Usage

```
saveCovariateTest(relevanceResults, fileName, folder = NULL, tests = NULL)
```

Arguments

relevanceResults	A CovariateTest object.
fileName	Output file name (or full path when folder is NULL).
folder	Optional directory; created if missing.
tests	Optional character vector overriding settings\$tests.

Value

Invisibly, the output file path.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
# help(saveCovariateTest)
```

showFIM	<i>Print FIM summaries to the console</i>
---------	---

Description

Print FIM summaries to the console
 Print FIM summaries to the console
 Print FIM summaries to the console
 Print FIM summaries to the console

Usage

```
showFIM(fim, ...)
```

Arguments

fim A Fim object.
 ... Not used by current methods.

Value

Invisibly, the input Fim object.

Examples

```
source(system.file("examples", "evaluation-minimal.R", package = "PFIM"))
showFIM(prop(ev, "fim"))
```

SimplexAlgorithm	<i>SimplexAlgorithm</i>
------------------	-------------------------

Description

Nelder-Mead (amoeba) optimization for sampling times (Rcpp kernel). Pass `pctInitialSimplexBuilding`, `tolerance`, and `maxIteration` via `optimizerParameters` on [Optimization](#). Search starts from each outcome's `initialSamplings` (same flat layout as PSO/PGB0). Extra vertices move one coordinate toward the far end of its sampling window (not toward 0), so `unequalNumberOfTimesByWindows` stays feasible. Infeasible `initialSamplings` are replaced by a feasible window start with a warning. `tolerance <= 0` disables the relative stop (`converged = NA`, no warning), matching PSO/PGB0 when stall stop is off.

Usage

```
SimplexAlgorithm(optimizerOutputs = list())
```

Arguments

optimizerOutputs
 List filled by `optimizeDesign()` (optimal arms, etc.).

Value

A SimplexAlgorithm specification object.

Examples

```
## Not run:
vignette("Example02")

## End(Not run)
```

Index

Additive, [6](#)
Administration, [7](#)
AdministrationConstraints, [8](#)
Arm, [5](#), [8](#)

BayesianFim, [10](#)

CategoricalCovariate, [11](#)
CategoricalCovariateWithIOV, [12](#)
Combined1, [13](#)
Combined2, [14](#)
Constant, [15](#)
Covariate, [16](#)
CovariateModelEquation, [17](#)
CovariateTest (CovariateTest-class), [17](#)
covariateTest, [5](#)
CovariateTest-class, [17](#)

defineFim, [5](#)
definePKModel, [5](#)
definePKPDMModel, [5](#)
Design, [5](#), [18](#)
Distribution, [19](#)

Evaluation, [5](#), [20](#), [75](#)
Exponential, [22](#)

FedorovWynnAlgorithm, [22](#), [73](#)
Fim, [23](#)

generateFimsFromConstraints, [79](#)
getCorrelationMatrix, [24](#)
getCovariateTestTables, [25](#)
getDcriterion, [25](#)
getDeterminant, [26](#)
getEvaluationDesign, [20](#), [27](#)
getFisherMatrix, [27](#)
getMixtureDcriterion, [28](#)
getModelParametersData, [28](#)
getRealisedDcriterion, [29](#)
getRSE, [29](#)

getSE, [30](#)
getShrinkage, [31](#)

IndividualFim, [31](#)
inferNumberOfOccasions, [21](#), [42](#), [44](#), [46](#), [48](#),
[51](#), [54](#), [56](#), [58](#), [60](#), [62](#), [64](#), [66](#), [70](#), [72](#),
[73](#)

LibraryOfModels, [33](#)
LibraryOfPDModels, [33](#), [71](#)
LibraryOfPKModels, [34](#), [80](#)
Linear2BolusSingleDose_C1QV1V2, [35](#)
Linear2BolusSingleDose_kk12k21V, [35](#)
Linear2BolusSteadyState_C1QV1V2tau, [36](#)
Linear2BolusSteadyState_kk12k21Vtau,
[36](#)
Linear2FirstOrderSingleDose_kaC1QV1V2,
[37](#)
Linear2FirstOrderSingleDose_kakk12k21V,
[37](#)
Linear2FirstOrderSteadyState_kaC1QV1V2tau,
[38](#)
Linear2FirstOrderSteadyState_kakk12k21Vtau,
[38](#)
Linear2InfusionSingleDose_C1QV1V2, [39](#)
Linear2InfusionSingleDose_kk12k21V, [39](#)
Linear2InfusionSteadyState_C1QV1V2tau,
[40](#)
Linear2InfusionSteadyState_kk12k21Vtau,
[40](#)
LogNormal, [41](#)

Model, [41](#)
ModelAnalytic, [43](#)
ModelAnalyticInfusion, [45](#)
ModelAnalyticInfusionSteadyState, [47](#)
ModelAnalyticSteadyState, [49](#)
ModelError, [52](#)
ModelInfusion, [53](#)
ModelODE, [55](#)

ModelODEBolus, 57
 ModelODEDoseInEquations, 59
 ModelODEDoseNotInEquations, 61
 ModelODEInfusion, 63
 ModelODEInfusionDoseInEquation, 65
 ModelParameter, 5, 20, 67
 MultiplicativeAlgorithm, 68, 73

 Normal, 68

 Optimization, 5, 22, 68, 69, 75, 77, 79, 89, 95

 package-PFIM (PFIM-package), 5
 pdModelLibrary, 71
 PFIM (PFIM-package), 5
 pfim-constraints-helpers, 71
 PFIM-package, 5
 pfim_get_option, 5, 73, 79
 pfim_list_extensions, 74
 pfim_register_fim_type, 75
 pfim_register_model_class, 21, 70, 72, 76
 pfim_register_optimizer, 77
 pfim_registered_fim_types
 (pfim_registered_model_classes),
 75
 pfim_registered_model_classes, 75
 pfim_registered_optimizers
 (pfim_registered_model_classes),
 75
 pfim_reset_session, 5, 74, 77, 78, 79
 pfim_resolve_model_class, 5, 78
 pfim_set_option, 5, 74, 78
 PFIMProject, 21, 70, 72
 PGB0Algorithm, 79
 pkModelLibrary, 80
 plotEvaluation, 81
 plotFrequencies, 81
 plotRSE, 82
 plotSE, 83
 plotSensitivityIndices, 83
 plotShrinkage, 84
 plotWeights, 85
 PopulationFim, 85
 projectOf, 86
 projectProp, 87
 projectProp<- (projectProp), 87
 prop, 20, 69, 88
 prop<- (prop), 88
 Proportional, 88

 PSOAlgorithm, 89

 Report, 5, 90
 run, 5, 91

 SamplingTimeConstraints, 92
 SamplingTimes, 93
 saveCovariateTest, 94
 showFIM, 5, 94
 SimplexAlgorithm, 95