

# Package ‘microsamplingDesign’

March 20, 2019

**Title** Finding Optimal Microsampling Designs for Non-Compartmental Pharmacokinetic Analysis

**Version** 1.0.6

**License** GPL-3

**Maintainer** Adriaan Blommaert <adriaan.blommaert@openanalytics.eu>

**Description** Find optimal microsampling designs for non-compartmental pharmacokinetic analysis using a general simulation methodology: Algorithm III of Barnett, Helen, Helena Geys, Tom Jacobs, and Thomas Jaki. (2017) "Optimal Designs for Non-Compartmental Analysis of Pharmacokinetic Studies. (currently unpublished)" This methodology consist of (1) specifying a pharmacokinetic model including variability among animals; (2) generating possible sampling times; (3) evaluating performance of each time point choice on simulated data; (4) generating possible schemes given a time point choice and additional constraints and finally (5) evaluating scheme performance on simulated data. The default settings differ from the article of Barnett and others, in the default pharmacokinetic model used and the parameterization of variability among animals. Details can be found in the package vignette. A 'shiny' web application is included, which guides users from model parametrization to optimal microsampling scheme.

**URL** <http://www.openanalytics.eu>

**Depends** R (>= 3.4.0), Rcpp

**Imports** abind, deSolve, devtools, ggplot2, gridExtra, gtools, knitr, MASS, matrixStats, matrixcalc, methods, parallel, plyr, readr, reshape2, shiny, stats, stringr, utils

**LinkingTo** Rcpp, RcppArmadillo

**ByteCompile** true

**LazyLoad** yes

**RoxygenNote** 6.1.0

**Suggests** bookdown, data.table, plotly, shinyjs, shinyBS, rmarkdown, rhandsontable, shinycssloaders, testthat

**Collate** 'RcppExports.R' 'aaaGenerics.R' 'appFunctions.R'  
 'constraintFunctions.R' 'fastRankSchemes.R' 'internalHelpers.R'  
 'objectPkModelParent.R' 'objectSetOfSchemes.R'  
 'objectPkModel.R' 'objectPkModelRange.R'  
 'objectSetOfTimePoints.R' 'pkFunctions.R' 'schemeStatistics.R'  
 'rankScheme.R' 'rankTimePoints.R' 'schemeGenerator.R'  
 'timePointGeneration.R'

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** Adriaan Blommaert [aut, cre],  
 Daan Seynaeve [ctb],  
 Helen Barnett [ctb],  
 Helena Geys [ctb],  
 Tom Jacobs [ctb],  
 Fetene Tekle [ctb],  
 Thomas Jaki [ctb]

**Repository** CRAN

**Date/Publication** 2019-03-20 15:50:03 UTC

## R topics documented:

addSchemes . . . . .	4
check_scheme_exactNumberObsPerTimePoint . . . . .	4
check_scheme_minObsPerTimePoint . . . . .	5
check_subject_maxConsecSamples . . . . .	5
construct2CompModel . . . . .	6
constructSetOfSchemes . . . . .	6
doAllSchemeChecks . . . . .	7
extractByRank . . . . .	8
flattenSetOfSchemes . . . . .	8
formatTimePoints . . . . .	9
get2ComptModelCurve . . . . .	9
getAllTimeOptions . . . . .	10
getCoeffVariationError . . . . .	11
getCombinationsWithMaxNRepetitions . . . . .	11
getConstraintsExample . . . . .	12
getCorrelationMatrix . . . . .	13
getData . . . . .	13
getDosingInfo . . . . .	14
getExampleData . . . . .	14
getExampleObjective . . . . .	14
getExampleParameters . . . . .	15
getExamplePkCurve . . . . .	15
getExamplePkModel . . . . .	16
getExamplePkModelRange . . . . .	16
getExampleSetOfSchemes . . . . .	16

getExampleSetOfTimePoints . . . . .	17
getExampleTimeData . . . . .	17
getExampleTimeZones . . . . .	17
getIndividualParameters . . . . .	18
getMMCurve . . . . .	19
getModelFunction . . . . .	20
getNames . . . . .	20
getNSchemes . . . . .	21
getNSubjects . . . . .	21
getParameters . . . . .	22
getPkData . . . . .	22
getPkModel . . . . .	23
getPkModelArticle . . . . .	24
getPkModels . . . . .	24
getRanking . . . . .	25
getSetOfSchemes . . . . .	25
getTimeChoicePerformance . . . . .	27
getTimePoints . . . . .	28
getTopNRanking . . . . .	29
oneCompartmentOralModel . . . . .	29
pkCurveStat . . . . .	30
PkData-class . . . . .	31
PkModel-class . . . . .	32
PkModelRange-class . . . . .	32
plotAverageRat . . . . .	33
plotMMCurve . . . . .	34
plotMMKinetics . . . . .	34
plotObject . . . . .	35
rankObject . . . . .	36
rankObjectWithRange . . . . .	39
setCoeffVariationError<- . . . . .	40
setCorrelationMatrix<- . . . . .	41
setDosingInfo<- . . . . .	41
setModelToAverageRat . . . . .	42
SetOfSchemes-class . . . . .	42
SetOfTimePoints-class . . . . .	43
setParameters<- . . . . .	43
setRanking<- . . . . .	44
setTimePoints<- . . . . .	44
subsetOnTimePoints . . . . .	45
summary,PkModelParent-method . . . . .	45
summary,SetOfSchemes-method . . . . .	46
%ARC% . . . . .	46

---

addSchemes	<i>add user defined scheme to an existing <a href="#">SetOfSchemes-class</a> or extend an existing set of schemes object with additional schemes</i>
------------	--

---

### Description

add user defined scheme to an existing [SetOfSchemes-class](#) or extend an existing set of schemes object with additional schemes

### Usage

```
addSchemes(setOfSchemes, extraSchemes)
```

### Arguments

setOfSchemes	<a href="#">SetOfSchemes-class</a> object or a matrix of individual schemes
extraSchemes	array of schemes to add, see code <a href="#">SetOfSchemes-class</a>

---

check_scheme_exactNumberObsPerTimePoint	<i>check the number of observations per time points is equal specified value</i>
---	--

---

### Description

check the number of observations per time points is equal specified value

### Usage

```
check_scheme_exactNumberObsPerTimePoint(scheme, value)
```

### Arguments

scheme	a microsampling scheme
value	numeric constant

---

check\_scheme\_minObsPerTimePoint

*check the minimum observations per time points is above a specified value*

---

### **Description**

check the minimum observations per time points is above a specified value

### **Usage**

check\_scheme\_minObsPerTimePoint(scheme, value)

### **Arguments**

scheme	a microsampling scheme
value	numeric constant

---

check\_subject\_maxConsecSamples

*check the maximum of consecutive samples per subject falls below the specified value*

---

### **Description**

check the maximum of consecutive samples per subject falls below the specified value

### **Usage**

check\_subject\_maxConsecSamples(subjectScheme, value)

### **Arguments**

subjectScheme	a one subject scheme, one line of a scheme
value	to compare scheme with

---

`construct2CompModel` *construct a 2 compartmental [PkModel-class](#) by providing parameters and dosing info*

---

### Description

construct a 2 compartmental [PkModel-class](#) by providing parameters and dosing info

### Usage

```
construct2CompModel(parameters, dosingInfo, correlationMatrix = NULL,
  coeffVariationError = 0)
```

### Arguments

`parameters`        see [PkModel-class](#)  
`dosingInfo`        see [PkModel-class](#)  
`correlationMatrix`  
                       see [PkModel-class](#), if NULL identity matrix is constructed  
`coeffVariationError`  
                       see [PkModel-class](#) , defaults to 0

### Note

model function is [get2ComptModelCurve](#)

### Examples

```
dosingInfo            <- data.frame( time = 0 , dose = 1 )
dataParametersFile   <- system.file( "extData",
"examplePkParameters.csv" , package = "microsamplingDesign" )
exampleParameters    <- read.csv( dataParametersFile ,
stringsAsFactors = FALSE , na.strings = NULL )
pkModel               <- construct2CompModel( exampleParameters , dosingInfo )
plotObject( pkModel , times = seq( 0, 5 , 0.1) , nSamplesIntegration = 12 )
```

---

`constructSetOfSchemes` *construct user defined [SetOfSchemes-class](#)*

---

### Description

construct user defined [SetOfSchemes-class](#)

### Usage

```
constructSetOfSchemes(schemes, timePoints)
```

**Arguments**

schemes            array representing .Data slot of [SetOfSchemes-class](#)  
timePoints        numeric vector, timePointst slot of [SetOfSchemes-class](#)

**Examples**

```
schemes                            <- getData( getExampleSetOfSchemes() )
timePoints                        <- exp(1:4)
constructSetOfSchemes( schemes , timePoints)
```

---

doAllSchemeChecks    *check whether either a 1 subject or multiple subject microsampling scheme meets imposed constraints*

---

**Description**

check whether either a 1 subject or multiple subject microsampling scheme meets imposed constraints

**Usage**

```
doAllSchemeChecks(object, level, checks)
```

**Arguments**

object            a logical vector or matrix, TRUE when a sample is taken for a subject (row) and time point (column) combination  
level             a character vector indicating either "subject" or "scheme" level  
checks            a dataframe with check definitions

**Value**

logical value TRUE when all checks are passed and FALSE if at least one check fails

**Examples**

```
exampleChecks        <- getConstraintsExample()
exampleSubject1     <- c( TRUE , TRUE , TRUE , FALSE , FALSE , TRUE )
exampleSubject2    <- c( FALSE , FALSE , TRUE , FALSE , FALSE , TRUE )
exampleScheme      <- rbind( exampleSubject1, exampleSubject2 )
doAllSchemeChecks( exampleSubject1 , "subject" , checks = exampleChecks )
doAllSchemeChecks( exampleSubject2 , "subject" , checks = exampleChecks )
doAllSchemeChecks( exampleScheme , "scheme" , checks = exampleChecks )
```

---

extractByRank	<i>extract a timepoint or Scheme choice by its rank</i>
---------------	---

---

**Description**

extract a timepoint or Scheme choice by its rank

**Usage**

```
extractByRank(object, rank)

## S4 method for signature 'SetOfSchemes,numeric'
extractByRank(object, rank)

## S4 method for signature 'SetOfTimePoints,numeric'
extractByRank(object, rank)
```

**Arguments**

object	an S4 object
rank	integer

**Examples**

```
object      <- getExampleSetOfSchemes()
pkData      <- getPkData( getExamplePkModel() ,
  getTimePoints( object ) , getNSubjects( object ) , nSamples = 10 )
objectRanked <- rankObject( object , pkData , data.frame(criterion = "auc" , weight = 1) )
extractByRank( object = objectRanked , rank = 1 )
extractByRank( objectRanked , rank = 5 )
object      <- getExampleSetOfTimePoints( 0 :10 )
pkData      <- getPkData(getExamplePkModel() ,
  getTimePoints( object ) , 1 , 5 )
objectRanked <- rankObject( object , pkData , nGrid = 20,
  nSamplesAvCurve = 25 )
extractByRank( objectRanked , 1)
extractByRank( object = objectRanked , rank = 5 )
```

---

flattenSetOfSchemes	<i>Transform 3 way array to 2 way array</i>
---------------------	---

---

**Description**

Transform 3 way array to 2 way array



**Usage**

```
flattenSetOfSchemes(object)
```

**Arguments**

object            [SetOfSchemes-class](#)

formatTimePoints        *Format time points as a set*

**Description**

Format time points as a set

**Usage**

```
formatTimePoints(timePoints)
```

**Arguments**

timePoints        numeric vector of timme points

get2ComptModelCurve    *provides solution of two compartmental pharmacodynamic model at specified time points*

**Description**

provides solution of two compartmental pharmacodynamic model at specified time points

**Usage**

```
get2ComptModelCurve(parameters, time, dosingInfo,
  internalODEs = pkOdeModel2Compartments, returnAll = FALSE)
```

**Arguments**

parameters        a list with correctly named input parameters

time                a numeric vector of times

dosingInfo        a data.frame with 2 columns

- time at which a dose is administered
- dose the amount administred to the gut

internalODEs      the model function used defaults to pkOdeModel2Compartments

returnAll         logical indicator if TRUE the solutions of all response variables is returned as a data.frame if FALSE only the plasma concentration is returned as a vector, defaults to FALSE

**Value**

data.frame or numeric vector of solutions, depending on the value of returnAll

**Examples**

```
pkModel          <- getExamplePkModel()
parameters       <- getParameters( pkModel )
testParameters   <- parameters[ , "value" ]
names(testParameters) <- parameters[ , "parameter" ]
time             <- seq( 0 , 3 , 0.1 )
dosingInfo       <- data.frame( time = c( 0 , 1 , 2 ) ,
                                dose = c( 5 , 2 , 1.5 ) )
get2ComptModelCurve( parameters = testParameters , time , dosingInfo )
get2ComptModelCurve( parameters = testParameters, time ,
                    dosingInfo , returnAll = TRUE )
```

---

getAllTimeOptions	<i>generate all possible time options from eligible time points and number of samples per time interval ( time zone )</i>
-------------------	---

---

**Description**

generate all possible time options from eligible time points and number of samples per time interval ( time zone )

**Usage**

```
getAllTimeOptions(timeZones, fullTimePoints)
```

**Arguments**

timeZones	a data.frame containing information on the number of points to be chosen in each time zone. Each row is a time zone. <ul style="list-style-type: none"> <li>• <code>startTime</code> the start time of each time zone assumed to be included in that zone</li> <li>• <code>endTime</code> the end time of the zone. It is not part of the current zone but the start time of the next zone</li> <li>• <code>nPointsPerZone</code> the number of time points to be chosen within each zone.</li> </ul>
fullTimePoints	a numeric vector containing all possible time points to be considered including time point zero and the last time point

**Details**

time point zero is never included in any time option and the last time point is always included. Note that the last time point is not a member of any zone. The number of time points in every time options is therefore the total number of time points specified in timeZone plus 1 for the last time point.

**Value**

[SetOfTimePoints-class](#)

**Examples**

```
timeZonesEx          <- getExampleTimeZones()
fullTimePointsEx    <- seq( 0 , 21 , 1 )
print(timeZonesEx)
setOfTimePoints      <- getAllTimeOptions( timeZones = timeZonesEx ,
      fullTimePoints = fullTimePointsEx )
setOfTimePoints      <- getAllTimeOptions(
      timeZones = data.frame(startTime = 0 , endTime = 21 , nPointsPerZone = 1) ,
      fullTimePoints = fullTimePointsEx
    )
```

---

getCoeffVariationError

*generic function to extract coeffVariationError slot*

---

**Description**

generic function to extract coeffVariationError slot

**Usage**

```
getCoeffVariationError(object, ...)
```

```
## S4 method for signature 'PkModelParent'
getCoeffVariationError(object)
```

**Arguments**

object	a S4 class object
...	additional parameters

---

getCombinationsWithMaxNRepetitions

*get all combinations with a maximum number of repetitions*

---

**Description**

get all combinations with a maximum number of repetitions

**Usage**

```
getCombinationsWithMaxNRepetitions(sourceVector, nDraws,  
  maxRepetitions = 1, nCombinationsOnly = FALSE)
```

**Arguments**

sourceVector is a vector with options to draw from

nDraws the combination size

maxRepetitions the number of times an element of the sampleVector can occur in a group

nCombinationsOnly if TRUE it returns the number of combinations instead of the combinations itself, defaults to FALSE

**Value**

a matrix with as a combination per row, unless nCombinationsOnly is TRUE

**Examples**

```
test1 <- getCombinationsWithMaxNRepetitions( c("a" , "b" , "c" ) ,  
  nDraws = 2, maxRepetitions = 2 )  
test2 <- getCombinationsWithMaxNRepetitions( 1:5 , nDraws = 3, maxRepetitions = 3 )  
test3 <- getCombinationsWithMaxNRepetitions( 1:5 , nDraws = 3, maxRepetitions = 3 ,  
  nCombinationsOnly = TRUE )
```

---

getConstraintsExample *get a minimal example of a constraint data frame*

---

**Description**

get a minimal example of a constraint data frame

**Usage**

```
getConstraintsExample()
```

---

getCorrelationMatrix    *generic function to extract the correlationMatrix-slot*

---

**Description**

generic function to extract the correlationMatrix-slot

**Usage**

```
getCorrelationMatrix(object, ...)  
  
## S4 method for signature 'PkModelParent'  
getCorrelationMatrix(object)
```

**Arguments**

object	a S4 class object
...	additional parameters

---

getData                    *generic function to extract the .Data-slot*

---

**Description**

generic function to extract the .Data-slot

**Usage**

```
getData(object, ...)  
  
## S4 method for signature 'SetOfSchemes'  
getData(object)  
  
## S4 method for signature 'PkData'  
getData(object)  
  
## S4 method for signature 'SetOfTimePoints'  
getData(object)
```

**Arguments**

object	a S4 class object
...	additional parameters

---

getDosingInfo	<i>generic function to extract dosingInfo-slot</i>
---------------	--

---

**Description**

generic function to extract dosingInfo-slot

**Usage**

```
getDosingInfo(object, ...)

## S4 method for signature 'PkModelParent'
getDosingInfo(object)
```

**Arguments**

object	a S4 class object
...	additional parameters

---

getExampleData	<i>generate an minimal example of a Pk data without a model</i>
----------------	---

---

**Description**

generate an minimal example of a Pk data without a model

**Usage**

```
getExampleData()
```

**Examples**

```
getExampleData()
```

---

getExampleObjective	<i>example objective function for <a href="#">rankObject</a></i>
---------------------	--

---

**Description**

example objective function for [rankObject](#)

**Usage**

```
getExampleObjective()
```

---

getExampleParameters    *get example parameters to use in [pkOdeModel2Compartments](#) example*

---

**Description**

get example parameters to use in [pkOdeModel2Compartments](#) example

**Usage**

```
getExampleParameters()
```

---

getExamplePkCurve    *example of 1 pk curve to be used to test [pkCurveStat\\_\[function\]](#)*

---

**Description**

example of 1 pk curve to be used to test [pkCurveStat\\_\[function\]](#)

**Usage**

```
getExamplePkCurve(times)
```

**Arguments**

times            a numeric vector of timePoints

**Value**

a data.frame with time and concentration as columns

**Examples**

```
getExamplePkCurve( times = 0:10 )
```

getExamplePkModel      *get minimal example of PkModel-class*

---

**Description**

get minimal example of [PkModel-class](#)

**Usage**

getExamplePkModel()

**Examples**

getExamplePkModel()

---

getExamplePkModelRange  
*get minimal example of PkModelRange-class*

---

**Description**

get minimal example of [PkModelRange-class](#)

**Usage**

getExamplePkModelRange()

**Examples**

getExamplePkModelRange()

---

getExampleSetOfSchemes  
*get a minimal example of a set of schemes object*

---

**Description**

get a minimal example of a set of schemes object

**Usage**

getExampleSetOfSchemes()



---

`getExampleSetOfTimePoints`*get a minimal example set of time points to test functions with*

---

**Description**

get a minimal example set of time points to test functions with

**Usage**

```
getExampleSetOfTimePoints(fullTimePoints, nTimePointsSelect = 5,  
  nChoicesSubset = 7)
```

**Arguments**

`fullTimePoints` numeric vector of time points

`nTimePointsSelect`

number of time points to select from the full time points, defaults to 5

`nChoicesSubset` number of all selection to retain for the example to avoid a large object defaults to 7

**Examples**

```
getExampleSetOfTimePoints( fullTimePoints = 0:10, nTimePointsSelect = 5, nChoicesSubset = 7 )
```

---

`getExampleTimeData`*generate example PkData object to be used in example rankTimePoints*

---

**Description**

generate example PkData object to be used in example rankTimePoints

**Usage**

```
getExampleTimeData()
```

---

`getExampleTimeZones`*working example time zone dataframe to use in examples*

---

**Description**

working example time zone dataframe to use in examples

**Usage**

```
getExampleTimeZones()
```

---

```
getIndividualParameters
```

*sample subject specific parameters to input in pharmacodynamic model parameters are sample from a log-normal distribution*

---

### Description

sample subject specific parameters to input in pharmacodynamic model parameters are sample from a log-normal distribution

### Usage

```
getIndividualParameters(meanParam, coeffVariation, nSubjects,
  corrMatrix = NULL)
```

### Arguments

meanParam        numeric vector containing mean information of a set of parameters  
 coeffVariation   coefficient of variation to inform the variance of the subject  
 nSubjects        the number of subjects which should be sampled  
 corrMatrix       optional correlation matrix when not specified parameters are assumed independent

### Value

a matrix with rows subject and columns parameters

### Examples

```
parameters            <- c( 1 , 0.1 , 10 , 3 )
names( parameters )   <- c( "Ka", "Ke" , "volume" , "dose" )
coeffVariation        <- c( 0.05 , 0.05 , 0.05, 0 )
names(coeffVariation) <- names( parameters )
nSubjects             <- 9

# example correlation matrix
corrMatrix            <- matrix(0.2 , nrow = 4 , ncol = 4) +
  diag( rep( 0.8 , 4 ) ) # correlation on the the log scale

# assuming independence between parameters
getIndividualParameters( parameters , coeffVariation , nSubjects = 9 )

# assuming correlations between parameters
getIndividualParameters( parameters , coeffVariation , nSubjects = 9 , corrMatrix)
getIndividualParameters( meanParam = parameters , coeffVariation , nSubjects = 3 , corrMatrix)
```

---

getMMCurve	<i>calculate Michealis-Menten relation between x and velocity and rate</i>
------------	--

---

### Description

calculate Michealis-Menten relation between x and velocity and rate

### Usage

```
getMMCurve(x, Vmax, kappaMM, constantValue = NA)
```

### Arguments

x	numeric vector, independent variable in Michaelis-Menten function representing a concentration or dose
Vmax	is the maximum rate ( $x * Vmax / (kappaMM + x)$ ) with increasing x
kappaMM	scalar representing Michaelis-Menten constant wich is the x at the rate reaches half of Vmax
constantValue	numeric constant if not NULL , the rate equals $x * constantValue$ with Vmax and kappaMM are ignored, defaults to NA

### Value

data.frame given te relation between concentration and velocity and rate with columns

- x
- velocity wich is rate/concentration
- rate rate ( $x * Vmax / (kappaMM + x)$ )
- Vmax input value
- kappaMM input value

### Examples

```
getMMCurve( x = seq( 0 , 1 , 0.01 ) , Vmax = 5 , kappaMM = 0.3 )
getMMCurve( x = seq( 0 , 3 , 0.01 ) , Vmax = 5 , kappaMM = 0.3 )
getMMCurve( x = seq( 0 , 1 , 0.01 ) , Vmax = 5 , kappaMM = 0.3 , constantValue = 3 )
```

---

getModelFunction	<i>generic function to extract modelFunction slot from S4-class object</i>
------------------	--

---

**Description**

generic function to extract modelFunction slot from S4-class object

**Usage**

```
getModelFunction(object, ...)  
  
## S4 method for signature 'PkModelParent'  
getModelFunction(object)
```

**Arguments**

object	a S4 class object
...	additional parameters

---

getNames	<i>generic function extract the names of an S4-object</i>
----------	---

---

**Description**

generic function extract the names of an S4-object

**Usage**

```
getNames(object, ...)  
  
## S4 method for signature 'SetOfSchemes'  
getNames(object)  
  
## S4 method for signature 'SetOfTimePoints'  
getNames(object)
```

**Arguments**

object	a S4 class object
...	additional parameters

---

getNSchemes                    *generic function to extract nSchemes-slot*

---

**Description**

generic function to extract nSchemes-slot

**Usage**

```
getNSchemes(object, ...)
```

```
## S4 method for signature 'SetOfSchemes'  
getNSchemes(object)
```

**Arguments**

object	a S4 class object
...	additional parameters

---

getNSubjects                    *generic function to extract nSubjects-slot*

---

**Description**

generic function to extract nSubjects-slot

**Usage**

```
getNSubjects(object, ...)
```

```
## S4 method for signature 'SetOfSchemes'  
getNSubjects(object)
```

**Arguments**

object	a S4 class object
...	additional parameters

---

getParameters	<i>generic function to extract parameter-slot</i>
---------------	---

---

**Description**

generic function to extract parameter-slot

**Usage**

```
getParameters(object, ...)

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

**Arguments**

object	a S4 class object
...	additional parameters

---

getPkData	<i>simulate <a href="#">PkData-class</a> from <a href="#">PkModel-class</a></i>
-----------	---

---

**Description**

simulate [PkData-class](#) from [PkModel-class](#)

**Usage**

```
getPkData(pkModel, timePoints, nSubjectsPerScheme, nSamples,
  errorCorrelationMatrixIntime = diag(1, length(timePoints)),
  nCores = 1, dirIntermediateOutput = NULL)
```

**Arguments**

pkModel	an object of <a href="#">PkModel-class</a>
timePoints	numeric vector of time points
nSubjectsPerScheme	numeric constant, number of subjects per dataset on which a sampling scheme can be applied
nSamples	number of datasets to sample
errorCorrelationMatrixIntime	the correlation between additive error terms within a subject, by default no correlation

nCores            number of cores used for parallel computing, defaults to 1 (remark no random numbers are generated in parallel)

dirIntermediateOutput    directory to write intermediate output to for debugging, defaults to NULL, when no intermediate output is written down

## Value

[PkData-class](#) object

## Examples

```
getPkData( getExamplePkModel() , 0:5 , nSubjectsPerScheme = 3 , nSamples = 4 )
getPkData( getExamplePkModel() , 0:5 , nSubjectsPerScheme = 7 , nSamples = 1 )
```

---

getPkModel            *generic function extract a [PkModel-class](#)*

---

## Description

generic function extract a [PkModel-class](#)

## Usage

```
getPkModel(object, ...)
```

## S4 method for signature 'PkData'

```
getPkModel(object)
```

## Arguments

object            an S4 object

...                additional parameters

## Examples

```
getPkModel( getExampleData() )
```

---

getPkModelArticle      *reproduce the example of the article of Helen Barnet et al.*

---

### Description

reproduce the example of the article of Helen Barnet et al.

### Usage

```
getPkModelArticle()
```

### Note

this models serves only to reproduce results of the article, and allows only 1 dose administered at time 0.

### Examples

```
model      <- getPkModelArticle()
summary( model )
testData   <- getPkData( model , 1:12 , nSubjectsPerScheme = 3 , nSamples = 7 )
plotObject( model , times = 0:12 )
plotAverageRat( model , doseZero = 100 , timePoints = seq(0,12,0.5) )
```

---

getPkModels              *Generate all possible [PkModel-class](#) from [PkModelRange-class](#) combination of ranges*

---

### Description

Generate all possible [PkModel-class](#) from [PkModelRange-class](#) combination of ranges

### Usage

```
getPkModels(object, outputDirectory = NULL)
```

### Arguments

object                  [PkModelRange-class](#)  
outputDirectory  
                          directory to save models as .Rds objects, defaults to NULL when a temporary directory is made to save models

### Value

[PkModelRange-class](#) objects saved as a subdirectory of the outputdirectory



**Note**

the outputDirectory is should be empty

---

getRanking	<i>generic function to extract the ranking-slot</i>
------------	---

---

**Description**

generic function to extract the ranking-slot

**Usage**

```
getRanking(object, ...)

## S4 method for signature 'SetOfSchemes'
getRanking(object)

## S4 method for signature 'SetOfTimePoints'
getRanking(object)
```

**Arguments**

object	a S4 class object
...	additional parameters

---

getSetOfSchemes	<i>Generate a <a href="#">SetOfSchemes-class</a> object of specified dimensions ( subjects, observations per t) for a given set of time points which meets user specified constraints</i>
-----------------	---

---

**Description**

Generate a [SetOfSchemes-class](#) object of specified dimensions ( subjects, observations per t) for a given set of time points which meets user specified constraints

**Usage**

```
getSetOfSchemes(minNSubjects, maxNSubjects, minObsPerSubject,
  maxObsPerSubject, timePoints, constraints = NULL,
  maxRepetitionIndSchemes = 1, maxNumberOfSchemesBeforeChecks = 10^5,
  returnNSchemesBeforeConstraints = FALSE)
```

**Arguments**

<code>minNSubjects</code>	numeric, the minimum number of subjects per scheme
<code>maxNSubjects</code>	numeric, the maximum number of subjects per scheme
<code>minObsPerSubject</code>	numeric, the minimum number of sampling occasions per subject
<code>maxObsPerSubject</code>	numeric, the maximum number of sampling occasions per subject
<code>timePoints</code>	numeric vector of time points larger than zero, at which subject can be sampled
<code>constraints</code>	data.frame specifying constraints the scheme should meet. with columns: <ul style="list-style-type: none"> <li>• <code>check</code>: identifier of the function to perform the check</li> <li>• <code>level</code>: the level at which the check is applied: either at the subject level or scheme level</li> <li>• <code>value</code>: input value used by the check function</li> </ul> <p>(a user can add constraint functions following naming convention <code>check_[level]_[check]</code> see examples: ( <a href="#">check_scheme_minObsPerTimePoint</a> and <a href="#">check_subject_maxConsecSamples</a> )  remark: number of subjects per scheme or number of observations per scheme should not be specified in constraints</p>
<code>maxRepetitionIndSchemes</code>	the maximum number of times an individual subject scheme can be repeated, defaults to 1
<code>maxNumberOfSchemesBeforeChecks</code>	the maximum number of schemes to consider before applying scheme constraints, to avoid long processing and using up memory. defaults to 10 <sup>5</sup>
<code>returnNSchemesBeforeConstraints</code>	if TRUE return only number of schemes before checking constraints instead of the schemes themselves, defaults to FALSE

**Note**

keep number of subjects , range of number of subjects and observations per subject and number of time points restricted to avoid a large number of potential schemes slowing down computation and increasing memory usage

only schemes with minimal one observation per subject are contained even if not specified in constraints

**Examples**

```
timePoints      <- c( 1.2 , 1.3 , 2, 5 )
constraints     <- getConstraintsExample()
ex1 <- getSetOfSchemes( minNSubjects = 4 , maxNSubjects = 4 ,
  minObsPerSubject = 3 , maxObsPerSubject = 3 , timePoints , constraints )
ex2 <- getSetOfSchemes( minNSubjects = 4 , maxNSubjects = 4 ,
  minObsPerSubject = 3 , maxObsPerSubject = 3 , timePoints ,
  constraints , maxRepetitionIndSchemes = 1 )
ex3 <- getSetOfSchemes( minNSubjects = 4 , maxNSubjects = 4 ,
  minObsPerSubject = 2 , maxObsPerSubject = 3 , timePoints ,
```

```

constraints , maxRepetitionIndSchemes = 1 )
ex4 <- getSetOfSchemes( minNSubjects = 2 , maxNSubjects = 5 ,
  minObsPerSubject = 2 , maxObsPerSubject = 3 , timePoints ,
  constraints , maxRepetitionIndSchemes = 1 )
ex5 <- getSetOfSchemes( minNSubjects = 2 , maxNSubjects = 5 ,
  minObsPerSubject = 2 , maxObsPerSubject = 3 , timePoints ,
  maxRepetitionIndSchemes = 2 )
## Not run:
# this should throw an error (to many combinations required )
ex6 <- getSetOfSchemes( minNSubjects = 2 , maxNSubjects = 5 ,
  minObsPerSubject = 2 , maxObsPerSubject = 3 , timePoints ,
  maxRepetitionIndSchemes = 2 , maxNumberOfSchemesBeforeChecks = 1000 )

## End(Not run)

```

---

getTimeChoicePerformance

*estimate the distance between population average and average over sample datasets with given time points (zero point included)*

---

### Description

estimate the distance between population average and average over sample datasets with given time points (zero point included)

### Usage

```
getTimeChoicePerformance(timePointInd, pkData, popAvCurve, timeGrid,
  printMCError = FALSE)
```

### Arguments

timePointInd	a vector indicating time points indicator selection of time points from fullTimePoints
pkData	<a href="#">PkData-class</a>
popAvCurve	an interpolated population average curve
timeGrid	the grid point at which to interpolate the curve
printMCError	logical indicator when true the MC error is printed to the terminal, defaults to FALSE

### Value

numeric value of the timePoint choice performance

**Examples**

```

# get example inputs
fullPkData          <-  getExampleTimeData() # PkData object
fullTimePoints      <-  getTimePoints(fullPkData)
examplePopAvCurve   <-  fullTimePoints^2
timePointIndicators <-  c( 1 , 5, 21 ) # zero point included
nGridPoints         <-  25
timeGrid            <-  seq( min( fullTimePoints ),
  max( fullTimePoints ) , length.out = nGridPoints )
popCurveInterpolated <-  microsamplingDesign::interpolateVec( fullTimePoints ,
  examplePopAvCurve, timeGrid )

getTimeChoicePerformance( timePointInd = timePointIndicators, pkData = fullPkData ,
  popAvCurve = popCurveInterpolated, timeGrid )

getTimeChoicePerformance( timePointInd = timePointIndicators, pkData = fullPkData ,
  popAvCurve = popCurveInterpolated, timeGrid, printMCErrors = TRUE )

```

---

getTimePoints	<i>generic function to extract timePoints-slot</i>
---------------	--

---

**Description**

generic function to extract timePoints-slot

**Usage**

```

getTimePoints(object, ...)

## S4 method for signature 'SetOfSchemes'
getTimePoints(object)

## S4 method for signature 'PkData'
getTimePoints(object)

## S4 method for signature 'SetOfTimePoints'
getTimePoints(object)

```

**Arguments**

object	a S4 class object
...	additional parameters

---

getTopNRanking	<i>extract the top n rankings as numeric vector</i>
----------------	---

---

**Description**

extract the top n rankings as numeric vector

**Usage**

```
getTopNRanking(ranking, nSelect, top = TRUE)
```

**Arguments**

ranking	ranking slot of a <a href="#">SetOfTimePoints-class</a> or <a href="#">SetOfSchemes-class</a>
nSelect	the number of items to select
top	logical value if TRUE the top of the ranking is selected, if FALSE the bottom of the ranking is selected, defaults to TRUE

**Value**

numeric vector of items (number of timePointOption or scheme ) from highest to lowest rank

---

oneCompartmentOralModel	<i>solution of one compartmental oral administration model only use one set of parameters, times can input can be an numeric array</i>
-------------------------	--

---

**Description**

solution of one compartmental oral administration model only use one set of parameters, times can input can be an numeric array

**Usage**

```
oneCompartmentOralModel(parameters, time, dosingInfo)
```

**Arguments**

parameters	a numeric vector of parameters as input to the model with names <ul style="list-style-type: none"> <li>• Ka: constant absorption rate</li> <li>• Ke: constant elimination rate</li> <li>• dose: initial dose</li> <li>• volume: volume to which the dose is administered</li> </ul>
time	a numeric vector containing timePoints at which the concentration should be predicted timepoint zero is defined as the moment the dose is administered
dosingInfo	see <a href="#">link{PkModel-class}</a> but opnly one dose at time zero allowed

**Value**

vector of concentrations corresponding to the input timePoints

---

pkCurveStat	<i>calculate summary statistics from a pkCurve</i>
-------------	--

---

**Description**

implemented statistics:

- auc area under the curve , between first and last time points
- cMax maximum concentration
- tMax time at maximum concentration

**Usage**

```
pkCurveStat_auc(concentration, timePoints)
```

```
pkCurveStat_cMax(concentration, timePoints)
```

```
pkCurveStat_tMax(concentration, timePoints)
```

**Arguments**

concentration    numeric vector of concentrations corresponding to timePoints  
timePoints       time and concentration

**Value**

a numeric value

**Examples**

```
## toy example
timeToy          <- 1:2
concToy          <- 1:2

pkCurveStat_auc( concToy , timeToy )
pkCurveStat_cMax( concToy , timeToy )
pkCurveStat_tMax( concToy , timeToy )

## real example
times           <- c(0 , 1.5 , 2:10)
concentration    <- getExamplePkCurve( times )
pkCurveStat_auc( concentration , times )
pkCurveStat_cMax( concentration , times )
pkCurveStat_tMax( concentration , times )
```

---

PkData-class	<i>An S4 object containing samples from a Pk model</i>
--------------	--

---

**Description**

An S4 object containing samples from a Pk model

**Slots**

`modelFunction` a function of parameters and hyperparameters

`parameters` a data.frame of parameters of mean parameters as input to the `modelFunction` with columns:

- `parameter`: parameter name for
- `explanation`: optional explanation
- `value`: fixed parameter value for [PkModel-class](#), for [PkModel-class](#) split up between `minValue` and `maxValue`
- `coeffVariation`: the coefficient of variation ( standard deviation / mean ) specifying between-subject variation, for [PkModelRange-class](#) split up into `minValue` and `maxValue`

`correlationMatrix` correlation matrix of parameters at the log-scale

`coeffVariationError` the coefficient of variation for residual normally distributed error, for [PkModelRange-class](#) split up into `minCoeffVariationError` and `maxCoeffVariationError` @slot `dosingInfo` data frame containing:

- `time` numeric, times when a dose is administered
- `dose` numeric, with an amount of dose

`timePoints` vector of time points

`.Data` a numerical array of 3 dimensions ( `nSubjects` x `nTimePoints` x `nSamples` )

**Note**

other slots are inherited from [{PkModel-class}](#)

**Author(s)**

Adriaan Blommaert

---

PkModel-class	<i>S4 class PkModel representing a pharmacokinetic model and its parameters</i>
---------------	---

---

### Description

S4 class PkModel representing a pharmacokinetic model and its parameters

### Slots

modelFunction a function of parameters and hyperparameters

parameters a data.frame of parameters of mean parameters as input to the modelFunction with columns:

- parameter: parameter name for
- explanation: optional explanation
- value: fixed parameter value for [PkModel-class](#), for [PkModel-class](#) split up between minValue and maxValue
- coeffVariation: the coefficient of variation ( standard deviation / mean ) specifying between-subject variation, for [PkModelRange-class](#) split up into minValue and maxValue

correlationMatrix correlation matrix of parameters at the log-scale

coeffVariationError the coefficient of variation for residual normally distributed error, for [PkModelRange-class](#) split up into minCoeffVariationError and maxCoeffVariationError @slot dosingInfo data frame containing:

- time numeric, times when a dose is administered
- dose numeric, with an amount of dose

### Author(s)

Adriaan Blommaert

---

PkModelRange-class	<i>S4 class PkModel representing a pharmacokinetic model and its parameters and uncertainty of parameter choices by ranges</i>
--------------------	--

---

### Description

S4 class PkModel representing a pharmacokinetic model and its parameters and uncertainty of parameter choices by ranges



**Slots**

modelFunction a function of parameters and hyperparameters

parameters a data.frame of parameters of mean parameters as input to the modelFunction with columns:

- parameter: parameter name for
- explanation: optional explanation
- value: fixed parameter value for [PkModel-class](#), for [PkModel-class](#) split up between minValue and maxValue
- coeffVariation: the coefficient of variation ( standard deviation / mean ) specifying between-subject variation, for [PkModelRange-class](#) split up into minValue and maxValue

correlationMatrix correlation matrix of parameters at the log-scale

coeffVariationError the coefficient of variation for residual normally distributed error, for [PkModelRange-class](#) split up into minCoeffVariationError and maxCoeffVariationError @slot dosingInfo

data frame containing:

- time numeric, times when a dose is administered
- dose numeric, with an amount of dose

---

plotAverageRat	<i>plot plasma concentration for average individual (i.e average parameter values) in function of dose at time zero</i>
----------------	---

---

**Description**

plot plasma concentration for average individual (i.e average parameter values) in function of dose at time zero

**Usage**

```
plotAverageRat(pkModel, doseZero, timePoints)
```

**Arguments**

pkModel	<a href="#">PkModel-class</a>
doseZero	numeric value, dose given at time zero
timePoints	a numeric vector of time points to plot the plasma concentration at

**Value**

ggplot object

**Note**

dose inside de pkModel is not used

**Examples**

```
plotAverageRat( getExamplePkModel() , 2 , seq( 0 , 20 , 0.1 ) )
```

---

plotMMCurve	<i>plot Michealis-Menten curve for either capacity dependent absorption or clearance</i>
-------------	--

---

**Description**

plot Michealis-Menten curve for either capacity dependent absorption or clearance

**Usage**

```
plotMMCurve(dataInput, parameter)
```

**Arguments**

dataInput	output of function <a href="#">getMMCurve</a>
parameter	character value indicating either absorption or clearance

**Value**

ggplot2-object

**Examples**

```
plotMMCurve( dataInput = getMMCurve( seq(0, 5, 0.01) ,
  Vmax = 5, kappaMM = 0.3 ) , parameter = "absorption" )
plotMMCurve( dataInput = getMMCurve( seq(0, 5, 0.01) ,
  Vmax = 5, kappaMM = 0.3, constantValue = 4 ) , parameter = "absorption" )
plotMMCurve( dataInput = getMMCurve( seq(0, 1, 0.01) ,
  Vmax = 2, kappaMM = 0.3 ) , parameter = "clearance" )
plotMMCurve( dataInput = getMMCurve( seq(0, 1, 0.01) ,
  Vmax = 2, kappaMM = 0.3, constantValue = 1.5 ) , parameter = "clearance" )
```

---

plotMMKinetics	<i>plot MM kinetics of both absorption and clearance</i>
----------------	--

---

**Description**

plot MM kinetics of both absorption and clearance

**Usage**

```
plotMMKinetics(pkModel, doseRange, concentrationRange,
  absorptionYRange = NULL, clearanceYRange = NULL)
```

**Arguments**

pkModel            an object of `PkModel-class`

doseRange          numeric vector representing the range of doses for absorption plot

concentrationRange  
                  numeric vector representing the range of concentrations for the clearance plot

absorptionYRange  
                  numeric vector of size 2 specifying y-limits for the absorption plot, defaults to  
                  NULL

clearanceYRange  
                  numeric vector of size 2 specifying y-limits for the clearance plot, defaults to  
                  NULL

**Value**

ggplot2 object

**Examples**

```
plotMMKinetics( pkModel = getExamplePkModel() ,
  doseRange = seq( 0 , 5 , 0.1 ) ,
  concentrationRange = seq( 0 , 2.5 , 0.1 ) )
plotMMKinetics( pkModel = getExamplePkModel() ,
  doseRange = seq( 0 , 5 , 0.1 ) ,
  concentrationRange = seq( 0 , 2.5 , 0.1 ) ,
  clearanceYRange = c( 0 , 50 ) , absorptionYRange = c( 0 , 10 ) )
```

---

plotObject            *generic function to plot an object*

---

**Description**

generic function to plot an object

**Usage**

```
plotObject(object, ...)

## S4 method for signature 'PkModel'
plotObject(object, times, nCurves = 12,
  nSamplesIntegration = 1000, seed = 134, sampleCurvesOnly = FALSE,
  indSamplingPoints = FALSE)

## S4 method for signature 'PkData'
plotObject(object, nCurves = NULL,
  nSamplesIntegration = 1000, sampleCurvesOnly = TRUE, seed = NULL,
  indSamplingPoints = TRUE, addZeroIsZero = FALSE)
```

**Arguments**

object	a S4 class object
...	additional parameters
times	numeric vector at of times at which the model should be simulated for <a href="#">PkModel-class</a>
nCurves	the number of sample curves defaults to 12 for <a href="#">PkModel-class</a> , if <a href="#">PkData-class</a> defaults to NULL meaning all data are plotted
nSamplesIntegration	number of simulated curves to calculate averaged curve, defaults to 1000
seed	specify the random seed to draw samples to get the same plot each time
sampleCurvesOnly	logical value if TRUE only sample curves are displayed and the averaged curve omitted , defaults to FALSE for <a href="#">PkModel-class</a> and TRUE for <a href="#">PkData-class</a>
indSamplingPoints	logical indicator if TRUE sample times are indicated on the plot, defaults to FALSE for <a href="#">PkModel-class</a> and TRUE for <a href="#">PkData-class</a>
addZeroIsZero	logical value, when TRUE the zero point is added to the plot with value zero ( only for <a href="#">PkData-class</a> , defaults to FALSE )

**Examples**

```
## Not run:
# examples with limited number of samples, increase samples in practice
plotObject( object = getExamplePkModel() ,
  times = seq( 0 , 10 , 1 ) , nSamplesIntegration = 25 )
plotObject( object = getExamplePkModel() ,
  times = seq( 0 , 10 , 1 ) , nCurves = 3 , nSamplesIntegration = 5 )
plotObject( object = getExamplePkModel() ,
  times = seq( 0 , 10 , 1 ) , nCurves = 3 , sampleCurvesOnly = TRUE )

## End(Not run)
## Not run:
pkData <- getPkData( getExamplePkModel() , 1:10 , 5 , 10 )
plotObject( object = pkData )
plotObject( object = pkData , nCurves = 2 )
plotObject( object = pkData , nCurves = 2 , addZeroIsZero = TRUE )
plotObject( object = pkData , nCurves = 3 ,
  sampleCurvesOnly = FALSE , nSamplesIntegration = 25 )

## End(Not run)
```

rankObject

*generic function to calculate a ranking-slot***Description**

generic function to calculate a ranking-slot

**Usage**

```
rankObject(object, ...)

fastRankSchemes(object, pkData, objective, nCores = 1)

## S4 method for signature 'SetOfSchemes'
rankObject(object, pkData, objective,
  varianceMeasure = "var", scaleWith = "max", skipTests = FALSE,
  nCores = 1)

## S4 method for signature 'SetOfTimePoints'
rankObject(object, pkData, nGrid = 100,
  nSamplesAvCurve = 1000, useAverageRat = FALSE, avCurve = NULL,
  nCores = 1)
```

**Arguments**

object	a S4 class object
...	additional parameters
pkData	<a href="#">PkData-class</a>
objective	a data.frame with columns: <ul style="list-style-type: none"> <li>• criterion summary function of an estimated pkCurve (data frame with columns time and concentration): area under the curve (auc) ; maximum concentration (cMax) and time when the maximum concentration is reached (tMax); user defined functions are allowed but prefix pkCurveStat_ should be added in function definition, see examples <a href="#">pkCurveStat</a></li> <li>• weight relative importance of the different criteria</li> </ul>
nCores	number of cores used in parallel processing, defaults to 1
varianceMeasure	variance criteria applied to the objective, defaults to summarise objective over sample data, defaults to var
scaleWith	function to scale different criteria in objective before combining results by taking a weighted sum
skipTests	if TRUE object validity and compatibility is not tested, defaults to FALSE , doing these tests is slow
nGrid	number of equally spaced point to calculate the distance between sample and population averaged kinetic curve, defaults to 100
nSamplesAvCurve	the number of samples to calculate the averaged curve ( only to rank <a href="#">SetOfTimePoints-class</a> ), defaults to 1000
useAverageRat	logical value if TRUE, the average rat (with random effects equal to zero and no additional error) is used instead of the integrated out population averaged curve, defaults to FALSE; this is faster but biased
avCurve	a user specified averaged curve, when specified, the average curve is no longer calculated from the pkModel, defaults to NULL

## Details

fastRankSchemes is a faster version to rank [SetOfSchemes-class](#) objects , with fixed settings ( objective AUC and cMax , summary measure is variance and scale measure is maximum ). It is meant to be used inside the shiny application

## Value

[SetOfSchemes-class](#) object

## Note

when ranking [SetOfSchemes-class](#) using if multiple criteria, the combined criterion is rescaled such that the best result is 1

if [SetOfTimePoints-class](#) timePoints are ranked according to minimal distance between population average curve and the estimate of the population average curve based on a selection of time points.

## Examples

```
## Not run:
setOfSchemes      <- getExampleSetOfSchemes()
dataForSchemes    <- getExampleData()
ex1               <- rankObject( object = setOfSchemes, dataForSchemes ,
  objective = data.frame( criterion = "auc" , weight = 1 ) )
getRanking(ex1) # to get the dataframe and not the whole object
ex2               <- rankObject( object = setOfSchemes, dataForSchemes ,
  objective = data.frame( criterion = "auc" , weight = 1 ) ,
  varianceMeasure = "sd" , scaleWith = "min" )
getRanking(ex2)
ex3               <- rankObject( object = setOfSchemes, dataForSchemes ,
  objective = data.frame( criterion = c( "auc" , "cMax" , "tMax" ) ,
  weight = c( 9 , 1 , 1 ) ) )
getRanking(ex3)

# example with own defined varianceMeasure
rangeWidth        <- function( x ){
  range <- range(x) ;
  rangeWith <- range[2] - range[1]; rangeWith
}
ex4               <- rankObject( object = setOfSchemes, dataForSchemes ,
  objective = data.frame( criterion = c( "auc" , "cMax" , "tMax" ) ,
  weight = c( 9 , 1 , 1 ) ) ,
  varianceMeasure = "rangeWidth" ,
  scaleWith = "mean" )

## End(Not run)
## Not run:
fullTimePoints    <- 0:10
setOfTimePoints   <- getExampleSetOfTimePoints( fullTimePoints)
pkDataExample     <- getPkData( getExamplePkModel() , getTimePoints( setOfTimePoints ) ,
  nSubjectsPerScheme = 5 , nSamples = 17 )
```

```

ex1          <- rankObject( object = setOfTimePoints , pkData = pkDataExample ,
  nGrid = 75 , nSamplesAvCurve = 13)
ex2          <- rankObject( object = setOfTimePoints ,  pkData = pkDataExample ,
  nGrid = 75 , nSamplesAvCurve = 13 , useAverageRat = TRUE )
ex3          <- rankObject( object = setOfTimePoints ,  pkData = pkDataExample ,
  nGrid = 75 , avCurve = rep(0 , length(fullTimePoints) ) )

## End(Not run)

```

---

rankObjectWithRange    *Rank a [SetOfSchemes-class](#) or a [SetOfTimePoints](#) object using data generated per scenario defined by [PkModelRange-class](#)*

---

### Description

Rank a [SetOfSchemes-class](#) or a [SetOfTimePoints](#) object using data generated per scenario defined by [PkModelRange-class](#)

### Usage

```
rankObjectWithRange(object, pkModelRange, nSim,
  summaryFunctionOverScenarios = "max", directory = NULL, nCores = 1,
  seed = 123, ...)
```

### Arguments

object	to be ranked
pkModelRange	see <a href="#">PkModelRange-class</a>
nSim	number of samples dataset to generate per scenario (= combination of uncertain parameters)
summaryFunctionOverScenarios	function to summarize performance over different scenarios, defaults to max which corresponds to the min-max criterion
directory	directory to save models as .Rds objects, defaults to NULL when a temporary directory is made to save models and additional info on simulation settings, ranks, ...
nCores	number of cores used internally for ranking
seed	random seed reset when ranking on each directory ( for reproducibility ) , defaults to 123
...	additional parameters to pass to <a href="#">rankObject</a>

### Note

parallel computing at level of individual ranking and data generation

see [rankObject](#) for additional arguments, when ranking a `link{SetOfTimePoints-class}` , nSubjectsPerScheme should be included

The same random seed is used when using parallel computations

**Examples**

```

## Not run: # takes too much time for CRAN
## rank SetOfSchemes
setOfSchemesExample <- getExampleSetOfSchemes()
pkModelRange <- getExamplePkModelRange()
nSim <- 13
testDirectory1 <- file.path( tempdir() , "test1" )
dir.create( testDirectory1 )
rankObjectWithRange( object = setOfSchemesExample , pkModelRange , nSim = 13 ,
  summaryFunctionOverScenarios = "max" ,
  directory = testDirectory1 , varianceMeasure = "sd" , objective = getExampleObjective()
  , nCores = 1 )

## rank set of timePoints
timePoints <- getExampleSetOfTimePoints( 0:10 )
testDirectory2 <- file.path( tempdir() , "test2" )
dir.create( testDirectory2 )
rankObjectWithRange( object = timePoints , pkModelRange , nSim = 13 ,
  summaryFunctionOverScenarios = "max" , directory = testDirectory2 , nGrid = 20 ,
  nSamplesAvCurve = 25 , nSubjectsPerScheme = 3
  , nCores = 1 )

# remark : use larger number of simulation in realistic context

## clean up directories
unlink( testDirectory1 , recursive = TRUE )
unlink( testDirectory2 , recursive = TRUE )

## End(Not run)

```

---

```

setCoeffVariationError<-

```

```

  replace coeffVariationError-slot

```

---

**Description**

replace coeffVariationError-slot

**Usage**

```

setCoeffVariationError( object ) <- value

```

```

## S4 replacement method for signature 'PkModelParent'

```

```

setCoeffVariationError(object) <- value

```



**Arguments**

object	a S4 class object
value	a value containing the coefficient of variation of the error term

---

```
setCorrelationMatrix<-  
    replace correlationMatrix-slot
```

---

**Description**

replace correlationMatrix-slot

**Usage**

```
setCorrelationMatrix( object ) <- value  
  
## S4 replacement method for signature 'PkModelParent'  
setCorrelationMatrix(object) <- value
```

**Arguments**

object	a S4 class object
value	a matrix containing correlations between parameters

---

```
setDosingInfo<-    replace dosingInfo-slot
```

---

**Description**

replace dosingInfo-slot

**Usage**

```
setDosingInfo(object) <- value  
  
## S4 replacement method for signature 'PkModelParent'  
setDosingInfo(object) <- value
```

**Arguments**

object	a S4 class object
value	a data.frame containing dosing information

---

setModelToAverageRat    *get a model with all variances to zero*

---

### Description

get a model with all variances to zero

### Usage

```
setModelToAverageRat(pkModel)
```

### Arguments

pkModel            [PkModel-class](#)

---

SetOfSchemes-class    *S4 class SetOfSchemes representing a set of designs with given time points*

---

### Description

S4 class SetOfSchemes representing a set of designs with given time points

### Slots

.Data a logical array of 3 dimensions ( nSubjects x nTimePoints x nSchemes )

timePoints numeric vector of time Points

nSchemes integer value number of schemes

nSubjects numeric maximum number of subjects per scheme

designConstraints a data.frame of constraints on possible sampling schemes as background information

ranking is a data.frame which is the rank of the schemes according to a specific criterion

### Author(s)

Adriaan Blommaert

---

SetOfTimePoints-class *S4 class SetOfTimePoints representing a set of designs with given time points*

---

### Description

S4 class SetOfTimePoints representing a set of designs with given time points

### Slots

.Data a numerics array of 2 dimensions ( nTimePointChoices x nTimePointsSelect) contains per time point choice the selected time points in hours

fullTimePoints numeric vector of all time points one is willing to consider

nFullTimePoints number of all time points one is willing to consider

nTimePointsSelect number of time points selected from the fullTimePoints

nTimePointOptions number of possible timePoint choices

ranking is a data.frame which is the rank of the timePointChoices according to a specific criterion.

### Author(s)

Adriaan Blommaert

---

setParameters<- *replace parameters-slot*

---

### Description

replace parameters-slot

### Usage

```
setParameters( object ) <- value
```

```
## S4 replacement method for signature 'PkModelParent'
setParameters(object) <- value
```

### Arguments

object a S4 class object

value a data.frame containing parameters

---

setRanking<-            *replace ranking-slot*

---

**Description**

replace ranking-slot

**Usage**

```
setRanking(object) <- value

## S4 replacement method for signature 'SetOfSchemes'
setRanking(object) <- value

## S4 replacement method for signature 'SetOfTimePoints'
setRanking(object) <- value
```

**Arguments**

object	a S4 class object
value	a data.frame containing a ranking

---

setTimePoints<-            *generic function to replace timePoints-slot*

---

**Description**

generic function to replace timePoints-slot

**Usage**

```
setTimePoints(object) <- value

## S4 replacement method for signature 'SetOfSchemes'
setTimePoints(object) <- value
```

**Arguments**

object	a S4 class object
value	a vector of time points

---

subsetOnTimePoints     *generic function to subset the timePoints-slot and generate an object of the same class*

---

**Description**

generic function to subset the timePoints-slot and generate an object of the same class

**Usage**

```
subsetOnTimePoints(object, ...)  
  
## S4 method for signature 'PkModel'  
subsetOnTimePoints(object, timePointsSelect)
```

**Arguments**

object             a S4 class object  
...                additional parameters  
timePointsSelect   a subset of time points to select data for

**Examples**

```
subsetOnTimePoints( pkData, c( 1 , 2 ) )  
subsetOnTimePoints( object = pkData, timePointsSelect = c( 1 , 2 ) )
```

---

summary,PkModelParent-method  
*function to summarize an object*

---

**Description**

function to summarize an object

**Usage**

```
## S4 method for signature 'PkModelParent'  
summary(object)
```

**Arguments**

object             [PkModel-class](#)

---

summary, SetOfSchemes-method  
*summarize object*

---

### Description

summarize object

### Usage

```
## S4 method for signature 'SetOfSchemes'
summary(object, printToConsole = TRUE)
```

### Arguments

object                    [SetOfSchemes-class](#)  
 printToConsole logical value if TRUE prints to console , if FALSE outputs text element , defaults to TRUE

---

%ARC%                    *All Row Combinations (ARC) function take all combination of rows of 2 matrices and bind them together*

---

### Description

All Row Combinations (ARC) function take all combination of rows of 2 matrices and bind them together

### Usage

```
matrix1 %ARC% matrix2
```

### Arguments

matrix1                    numeric matrix  
 matrix2                    numeric matrix

### Value

numeric matrix

# Index

## \*Topic **export**

- rankObject, [36](#)
- %ARC%, [46](#)
- addSchemes, [4](#)
- auc (pkCurveStat), [30](#)
- check\_scheme\_exactNumberObsPerTimePoint, [4](#)
- check\_scheme\_minObsPerTimePoint, [5](#), [26](#)
- check\_subject\_maxConsecSamples, [5](#), [26](#)
- cMax (pkCurveStat), [30](#)
- construct2CompModel, [6](#)
- constructSetOfSchemes, [6](#)
- doAllSchemeChecks, [7](#)
- extractByRank, [8](#)
- extractByRank, SetOfSchemes, numeric-method (extractByRank), [8](#)
- extractByRank, SetOfTimePoints, numeric-method (extractByRank), [8](#)
- fastRankSchemes (rankObject), [36](#)
- flattenSetOfSchemes, [8](#)
- formatTimePoints, [9](#)
- get2ComptModelCurve, [6](#), [9](#)
- getAllTimeOptions, [10](#)
- getCoeffVariationError, [11](#)
- getCoeffVariationError, PkModelParent-method (getCoeffVariationError), [11](#)
- getCombinationsWithMaxNR repetitions, [11](#)
- getConstraintsExample, [12](#)
- getCorrelationMatrix, [13](#)
- getCorrelationMatrix, PkModelParent-method (getCorrelationMatrix), [13](#)
- getData, [13](#)
- getData, PkData-method (getData), [13](#)
- getData, SetOfSchemes-method (getData), [13](#)
- getData, SetOfTimePoints-method (getData), [13](#)
- getDosingInfo, [14](#)
- getDosingInfo, PkModelParent-method (getDosingInfo), [14](#)
- getExampleData, [14](#)
- getExampleObjective, [14](#)
- getExampleParameters, [15](#)
- getExamplePkCurve, [15](#)
- getExamplePkModel, [16](#)
- getExamplePkModelRange, [16](#)
- getExampleSetOfSchemes, [16](#)
- getExampleSetOfTimePoints, [17](#)
- getExampleTimeData, [17](#)
- getExampleTimeZones, [17](#)
- getIndividualParameters, [18](#)
- getMMCurve, [19](#), [34](#)
- getModelFunction, [20](#)
- getModelFunction, PkModelParent-method (getModelFunction), [20](#)
- getNames, [20](#)
- getNames, SetOfSchemes-method (getNames), [20](#)
- getNames, SetOfTimePoints-method (getNames), [20](#)
- getNSchemes, [21](#)
- getNSchemes, SetOfSchemes-method (getNSchemes), [21](#)
- getNSubjects, [21](#)
- getNSubjects, SetOfSchemes-method (getNSubjects), [21](#)
- getParameters, [22](#)
- getParameters, PkModelParent-method (getParameters), [22](#)
- getPkData, [22](#)
- getPkModel, [23](#)
- getPkModel, PkData-method (getPkModel), [23](#)
- getPkModelArticle, [24](#)

- getPkModels, 24
- getRanking, 25
- getRanking, SetOfSchemes-method  
(getRanking), 25
- getRanking, SetOfTimePoints-method  
(getRanking), 25
- getSetOfSchemes, 25
- getTimeChoicePerformance, 27
- getTimePoints, 28
- getTimePoints, PkData-method  
(getTimePoints), 28
- getTimePoints, SetOfSchemes-method  
(getTimePoints), 28
- getTimePoints, SetOfTimePoints-method  
(getTimePoints), 28
- getTopNRanking, 29
  
- oneCompartmentOralModel, 29
  
- pkCurveStat, 30, 37
- pkCurveStat\_auc (pkCurveStat), 30
- pkCurveStat\_cMax (pkCurveStat), 30
- pkCurveStat\_tMax (pkCurveStat), 30
- PkData (PkData-class), 31
- pkData (PkData-class), 31
- pkdata (PkData-class), 31
- PkData-class, 22, 31
- PkModel (PkModel-class), 32
- pkModel (PkModel-class), 32
- pkmodel (PkModel-class), 32
- PkModel-class, 6, 16, 22–24, 32
- PkModelRange (PkModelRange-class), 32
- pkModelRange (PkModelRange-class), 32
- pkmodelrange (PkModelRange-class), 32
- PkModelRange-class, 16, 24, 32, 39
- pkOdeModel2Compartments, 15
- plotAverageRat, 33
- plotMMCurve, 34
- plotMMKinetics, 34
- plotObject, 35
- plotObject, PkData-method (plotObject),  
35
- plotObject, PkModel-method (plotObject),  
35
  
- rankObject, 14, 36, 39
- rankObject, SetOfSchemes-method  
(rankObject), 36
- rankObject, SetOfTimePoints-method  
(rankObject), 36
- rankObjectWithRange, 39
  
- setCoeffVariationError<- , 40
- setCoeffVariationError<- , PkModelParent-method  
(setCoeffVariationError<-), 40
- setCorrelationMatrix<- , 41
- setCorrelationMatrix<- , PkModelParent-method  
(setCorrelationMatrix<-), 41
- setDosingInfo<- , 41
- setDosingInfo<- , PkModelParent-method  
(setDosingInfo<-), 41
- setModelToAverageRat, 42
- SetOfSchemes (SetOfSchemes-class), 42
- setOfSchemes (SetOfSchemes-class), 42
- SetOfSchemes-class, 4, 6, 25, 39, 42
- SetOfTimePoints, 39
- SetOfTimePoints  
(SetOfTimePoints-class), 43
- setOfTimePoints  
(SetOfTimePoints-class), 43
- SetOfTimePoints-class, 43
- setParameters<- , 43
- setParameters<- , PkModelParent-method  
(setParameters<-), 43
- setRanking<- , 44
- setRanking<- , SetOfSchemes-method  
(setRanking<-), 44
- setRanking<- , SetOfTimePoints-method  
(setRanking<-), 44
- setTimePoints<- , 44
- setTimePoints<- , SetOfSchemes-method  
(setTimePoints<-), 44
- subsetOnTimePoints, 45
- subsetOnTimePoints, PkModel-method  
(subsetOnTimePoints), 45
- summary, PkModelParent-method, 45
- summary, SetOfSchemes-method, 46
  
- tMax (pkCurveStat), 30