Package 'Certara.RsNLME'

November 20, 2024

Title Pharmacometric Modeling

Version 3.0.1

Description Facilitate Pharmacokinetic (PK) and Pharmacodynamic (PD) modeling and simulation with powerful tools for Nonlinear Mixed-Effects (NLME) modeling. The package provides access to the same advanced Maximum Likelihood algorithms used by the NLME-Engine in the Phoenix platform. These tools support a range of analyses, from parametric methods to individual and pooled data analysis <https://www.certara.com/app/uploads/2020/06/BR_PhoenixNLME-v4.pdf>. Execution is supported both locally or on remote machines.

Depends R (>= 4.0)

License LGPL-3

URL https://certara.github.io/R-RsNLME/

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Suggests rlang, knitr, rmarkdown, testthat, magrittr

Imports xml2, assertthat, Certara.NLME8, data.table, jsonlite, methods, utils, ssh

Collate 'BootstrapParams.R' 'NlmeParallelMethod.R' 'NlmeUserAuthentication.R' 'NlmeParallelHost.R' 'SimpleNlmeJob.R' 'BootNlmeJob.R' 'CovariateEffectModel.R' 'GenerateControlfile.R' 'GenerateParamsfile.R' 'NlmeColumnMapping.r' 'NlmeCovariateParameter.r' 'NlmeDataset.r' 'NlmeDoseMapping.R' 'NlmeEmaxParameters.R' 'NlmeIndirectParameters.R' 'NlmeModelAbsorption.R' 'NlmeModelParameterization.R' 'NlmeModelType.R' 'NlmeParamsMapping.R' 'NlmePkParameters.R' 'NlmePmlModelInfo.R' 'NlmeRandParamsMapping.R' 'NlmeRandomEffectBlock.r' 'NlmeRemoteExecutor.R' 'NlmeScenario.R' 'NlmeTableDef.R' 'SortColumns.R' 'ProfileParameters.R' 'ProfileNlmeJob.R' 'ProfileVar.R' 'error_model.r' 'pml_model.r' 'RandomEffectsMethods.R' 'ShotgunNlmeJob.R' 'SortByNlmeJob.R'

'StepwiseParams.R' 'StepwiseNlmeJob.R' 'acceptAllEffects.R'
'addInfusion.R' 'addLabel.R' 'addTablesToColumnMapping.R'
'add_input_dosingCycles.R' 'bootstrap.r' 'built_in_models.r'
'checkHostParams.R' 'colMapping.R' 'copyModel.R'
'covariateModel.R' 'covariateNames.R' 'createInitialMapping.R'
'create_model_from_metamodel.R' 'create_model_info.R' 'data.r'
'dosing.r' 'editModel.R' 'emaxmodel.R' 'engine_params.r'
'extract_mmdl.R' 'fitmodel.R' 'fitmodelHelperFunctions.R'
'fixedEffect.R' 'generateCovarSearchArgsFile.R' 'getThetas.R'
'get_omega_omegaSE.R' 'globals.R' 'hostParams.R' 'job.r'
'linearmodel.R' 'log_Execution.R' 'map_covariates.R'
'map_dosepoints.R' 'modelVariableNames.R' 'observation.r'
'obtain_NLMELicense.R' 'parameterNames.R' 'parseControlFile.R'
'parsePMLColMap.R' 'parse_mmdl.R' 'pkemaxmodel.R'
'pkindirectmodel.R' 'pklinearmodel.R' 'pkmodel.R'
'profile_estimation.r' 'randomEffect.R'
'readInitialEstimatesParams.R' 'run_metamodel.R' 'saveModel.R'
'saveUpdatedMetamodel.R' 'secondary_variable.r'
'shotgunSearch.R' 'simParams.R' 'sortfit.R' 'stepwiseSearch.R'
'structural_param.r' 'tableParams.R' 'update_PMLwithThetas.R'
'vpc.r' 'writeColumnMapping.R' 'writeDefaultFiles.R'

NeedsCompilation no

Author James Craig [aut, cre],
Michael Tomashevskiy [aut],
Vitalii Nazarov [aut],
Shuhua Hu [ctb],
Soltanshahi Fred [aut],
Certara USA, Inc. [cph, fnd]

Maintainer James Craig <james.craig@certara.com>

Repository CRAN

Date/Publication 2024-11-20 11:20:02 UTC

Contents

addADDL *Adds ADDL extra column definition to model object*

Description

Specify ADDL column definition in model object instead of specifying ADDL through addDoseCycle

Usage

```
addADDL(.Object, ADDL, II)
```

Arguments

.Object	Model object
ADDL	Column mapping argument specifying corresponding "ADDL" column in input data set
II	Column mapping argument specifying corresponding "II" column in input data set

Value

Modified NlmePmlModel object

Examples

```
model <- addADDL(model, ADDL = "addl", II = "ii")
```

addCovariate *Add covariate to model object*

Description

Add a continuous, categorical, or occasion covariate to model object and set covariate effect on structural parameters.

Usage

```
addCovariate(
  .Object,
  covariate,
  effect = NULL,
  type = c("Continuous", "Categorical", "Occasion"),
  direction = c("Forward", "Interpolate", "Backward"),
```

```
  option = c("Yes", "PlusOne", "No"),
  center = NULL,
  centerValue = NULL,
  levels = NULL,
  labels = NULL,
  isDiagonal = TRUE,
  values = NULL,
  isPositive = TRUE
)
```

Arguments

.Object	Model object
covariate	Name of covariate. If the involved model has columns mapped (i.e. model with columnMap = TRUE) use named character if the name of the covariate is different from the corresponding column in the input dataset, for example, covariate = c(BW = "BodyWeight"), where BW denotes the name of the covariate, and "BodyWeight" is the name of the corresponding column in the input dataset.
effect	Name of structural parameter(s) on which the covariate has an effect. Specify effect as character or character vector if the covariate has an effect on multiple structural parameters.
type	Type of covariate. Options are "Continuous", "Categorical", "Occasion".
direction	Direction of missing values propagation (if no covariate value is given). Options are "Forward", "Interpolate", "Backward", where "Interpolate" is only applicable to type = "Continuous".
option	Options are "Yes", "PlusOne", or "No", where option = "No" will remove the covariate effect from the specified structural parameter(s), but retain the covariate in the model. Note: option = "PlusOne" is only applicable to continuous and categorical covariates in the case where structural parameters have style = "LogNormal". Multiple options are not supported (i.e. all covariate effects in the call are supposed to have the same option. If different options are required for different covariate effects, sequential calls of current method could be done.
center	Centering method. Options are "Mean", "Median", "Value" or "None". Only applicable to covariate type = "Continuous". Must include argument centerValue if center = "Value".
centerValue	Value used to center covariate. Only applicable if argument center = "Value" and type = "Continuous".
levels	Unique values of categorical or occasion covariate. Only applicable to covariate type = "Categorical" or type = "Occasion".
labels	Label names (in the same order as levels) for unique levels of categorical or occasion covariate in data. Only applicable to covariate type = "Categorical" or type = "Occasion" where its corresponding column in the input dataset has character type.
isDiagonal	Set to FALSE if inter-occasion covariance matrix is not diagonal matrix. Only applicable to covariate type = "Occasion".

values	Initial values for the diagonal elements of the inter-occasion covariance matrix (if isDiagonal = TRUE) or initial values for the lower triangular elements (including diagonal elements) of inter-occasion covariance matrix (if isDiagonal = FALSE) in a row-wise order. Only applicable for covariate type = "Occasion".
isPositive	Set to FALSE if covariate contains negative values. Only applicable to covariate type = "Continuous".

Details

The following relationships are applicable for covariates:

- direction = "Forward" is equivalent to PML code 'fcovariate(CovName)';
- direction = "Backward" is equivalent to PML code 'covariate(CovName)';
- direction = "Interpolate" is equivalent to PML code 'interpolate(CovName)'.

If the structural parameter has style = "LogNormal", the options are reflected in PML code as follows:

- option = "Yes" is equivalent to stparm(V = tvV * wt^dVdwt * exp(dVdsex1*(sex==1)) * exp(nV));
- option = "PlusOne is equivalent to stparm(V = tvV * (1+wt*dVdwt) * (1+dVdsex1*(sex==1)) * exp(nV)).

Value

Modified NlmePmlModel object

Examples

```
model <- pkmodel(
  numCompartments = 2,
  data = pkData,
  ID = "Subject",
  Time = "Act_Time",
  A1 = "Amount",
  CObs = "Conc"
)

# Add Gender covariate of type categorical
model <- addCovariate(model,
  covariate = "Gender",
  type = "Categorical",
  effect = c("V2", "Cl2"),
  levels = c(0, 1),
  labels = c("Female", "Male")
)

# Add BodyWeight covariate of type continuous
model <- addCovariate(model,
  covariate = "BodyWeight",
  type = "Continuous",
  direction = "Backward",
```

```
  center = "Mean",
  effect = c("V", "Cl")
)
```

---

addDoseCycle *Adds a dosing cycle to model*

Description

Add Steady State or ADDL dosing cycle to model object.

Usage

```
addDoseCycle(
  .Object,
  type = "SteadyState",
  name,
  administration = "Bolus",
  amount = NULL,
  II = NULL,
  rate = NULL,
  duration = NULL,
  isSecondDose = FALSE,
  colName = NULL
)
```

Arguments

.Object	Model object
type	Specification of dose type. Options are "SteadyState"and "ADDL"
name	Dose point name. See doseNames
administration	Mechanism for administering dose. Options are "Bolus" or "Infusion"
amount	Optional. Column mapping argument specifying corresponding "ADDL" column in input data, or numeric value specifiying dose amount.
II	Optional. Column mapping argument specifying corresponding "II" column in input data, or numeric value specifying delta time.
rate	Optional. Column mapping argument specifying corresponding "Rate" column in input data, or numeric specifying dose rate.
duration	Optional. Column mapping argument specifying corresponding "Duration" column in data, or numeric specifying duration value.
isSecondDose	Use second dose point on compartment
colName	Column name in input data corresponding to column mapping for "SteadyState" or "ADDL" as supplied in type argument.

Value

Modified NlmePmlModel object

See Also

doseNames

Examples

```
model <- pkmodel(columnMap = FALSE) %>%
  addDoseCycle(type = "SteadyState", name = "A1", amount = "Amount", II = "II")
```

---

addExtraDef *Adds user defined extra column/table definitions to column definition file*

Description

Adds user defined extra column/table definitions to column definition file

Usage

```
addExtraDef(.Object, value)
```

Arguments

.Object	PK/PD model
value	Character vector of extra column/table definitions

Value

Modified NlmePmlModel object

Examples

```
model <- addExtraDef(model, c("addlcol(ADDL)", "table(file=\"res.csv\",time(0),Ka,V,Cl,Tlg)"))
```

addInfusion *Change existing dosing compartment to infusion*

Description

Allows user to switch any dosing compartment to infusion

Usage

```
addInfusion(
  .Object,
  doseCptName,
  isDuration = FALSE,
  isSecondDose = FALSE,
  colName = NULL
)
```

Arguments

.Object	Model object
doseCptName	Name of the compartment to which the dose is administered
isDuration	Set TRUE if duration is used to specify infusion information
isSecondDose	Set TRUE if doseCptName is specified in the model through dosepoint2 statement
colName	Name of the input data column that represents the corresponding infusion rate. If not provided, colName must be mapped through colMapping().

Value

Modified NlmePmlModel object

Examples

```
newModel <- addInfusion(model, "A1", FALSE, FALSE, "A1_1")
```

addLabel *Add levels and labels to categorical or occasion covariate*

Description

Allows users to specify the name and the associated value for each category/occasion of a categorical/occasion covariate in a textual model object. Only applicable to the case where the corresponding input data column of a categorical/occasion covariate is of class character.

Usage

```
addLabel(.Object, covariate, levels, labels)
```

Arguments

.Object	Model object
covariate	Existing covariate name
levels	Unique values of categorical or occasion covariate column specified as numeric vector
labels	Unique values specifying corresponding label names for levels of categorical or occasion covariate column in data specified as character vector.

Value

Modified NlmePmlModel object

Examples

```
model <- addLabel(model, covariate, c(1, 2, 3), c("a", "b", "c"))
```

---

addMDV *Adds MDV extra column definition to model object*

Description

Use to add MDV statement to model@userDefinedExtraDefs

Usage

```
addMDV(.Object, MDV)
```

Arguments

.Object	Model object
MDV	Column mapping argument specifying corresponding "MDV" column in input data set

Value

Modified NlmePmlModel object

Examples

```
model <- addMDV(model, MDV = "MDV")
```

addReset *Adds reset instructions to the model*

Description

Adds reset instructions to the model

Usage

```
addReset(.Object, low, hi, Reset = NULL)

## S4 method for signature 'NlmePmlModel'
addReset(.Object, low, hi, Reset = NULL)
```

Arguments

.Object	An 'NlmePmlModel' object to which you want to add reset instructions.
low	Lower value of reset range.
hi	Upper value of reset range.
Reset	Name of reset column in input data set for column mapping. The default is NULL.

Value

Depends on the specific methods

Returns the 'NlmePmlModel' object with updated reset information and definitions.

Functions

- addReset(NlmePmlModel): Method for the 'NlmePmlModel' class

  This method adds reset instructions to the NlmePmlModel object. It updates the reset information, checks column mappings if input data is not null, and adds a reset definition to user-defined extra definitions.

addSecondary *Adds a secondary parameter to model definition*

Description

Adds a secondary parameter to model definition

Usage

```
addSecondary(.Object, name, definition, unit = "")

## S4 method for signature 'NlmePmlModel'
addSecondary(.Object, name, definition, unit = "")
```

Arguments

.Object	An 'NlmePmlModel' object to which you want to add a secondary parameter.
name	Name of the secondary parameter.
definition	Definition of secondary parameter.
unit	Optional units of the secondary parameter. The default is "".

Value

Depends on the specific methods

Returns the 'NlmePmlModel' object with the added secondary parameter.

Functions

- addSecondary(NlmePmlModel): Method for the 'NlmePmlModel' class

  This method adds a secondary parameter to the NlmePmlModel object. It checks for duplicate parameter names, and if there is no duplicate, it adds the new secondary parameter to the object and updates the PML model.

Examples

```
model <- addSecondary(model, "Spc_Param", "log(2)/tvKe")
model <- addSecondary(
  model, "Tmax",
  "CalcTMax(tvA,tvAlpha,tvB,tvBeta,C,Gamma)"
)
```

---

addSteadyState *Adds Steady State extra column definition to model object*

Description

Use to add Steady State column definition statement to model@userDefinedExtraDefs

Usage

```
addSteadyState(.Object, SS, II, SSOffset = NULL)
```

Arguments

.Object	Model object
SS	Column mapping argument specifying corresponding "SS" column in input data set
II	Column mapping argument specifying corresponding "II" column in input data set
SSOffset	Optional. Column mapping argument specifying corresponding "SSOffset" column in input data set

Value

Modified NlmePmlModel object

Examples

```
model <- addSteadyState(model, SS = "ss", II = "ii")
```

---

bootstrap *Executes an NLME Bootstrap*

Description

Method to execute an NLME Bootstrap

Usage

```
bootstrap(
  model,
  hostPlatform = NULL,
  params,
  bootParams,
  runInBackground = FALSE,
  ...
)
```

Arguments

model	PK/PD model class object.
hostPlatform	Host definition for model execution. See hostParams. If missing, multicore local host with 4 threads is used.
params	Engine parameters. See engineParams. If missing, default parameters generated by engineParams(model) are used.
bootParams	Bootstrap parameters. See BootstrapParams. If missing, default parameters generated by BootstrapParams() are used.

runInBackground
: Set to TRUE to run in background and return prompt.

...
: Additional class initializer arguments for BootstrapParams or hostParams, or arguments available inside engineParams functions. If engineParams arguments are supplied through both params argument and additional argument (i.e., ellipsis), then the arguments in params will be ignored and only the additional arguments will be used with warning. If hostParams arguments are supplied through both hostPlatform argument and additional argument, then its values will be overridden by additional arguments. In addition, if BootstrapParams arguments are supplied through both bootParams argument and additional argument, then its slots will be overridden by additional arguments.

Value

if runInBackground = FALSE, a list is returned with bootstrap results, i.e. "BootOverall", "BootTheta", "BootOmega", "BootOmegaStderr", "BootVarCoVar" comma separated files. Otherwise the BootNlmeJob class object is returned.

See Also

hostParams, engineParams, BootstrapParams

Examples

```
input_data <- pkData

model <-
  pkmodel(
    numCompartments = 2,
    data = input_data,
    ID = "Subject",
    Time = "Act_Time",
    A1 = "Amount",
    CObs = "Conc"
  )

# multicore
multicoreHost <- hostParams(
  parallelMethod = "Multicore",
  hostName = "local_multicore",
  numCores = 4
)

bootstrapdf <- bootstrap(model,
  hostPlatform = multicoreHost,
  params = engineParams(model),
  numReplicates = 5,
  randomNumSeed = 1234,
  runInBackground = FALSE
)
```

cancelJob *Generic function for cancelling a job*

Description

Generic function for cancelling a job

Usage

```
cancelJob(.Object)

## S4 method for signature 'SimpleNlmeJob'
cancelJob(.Object)
```

Arguments

.Object	A 'SimpleNlmeJob' object that you want to cancel

Value

Depends on the specific methods

Prints the 'SimpleNlmeJob' object after attempting to cancel the job. No return value.

Functions

- `cancelJob(SimpleNlmeJob)`: Method for cancelling a job of the 'SimpleNlmeJob' class
  This method attempts to cancel a job of the 'SimpleNlmeJob' class. If the job is running on a local host or is not running in the background, it throws an error and does nothing. Otherwise, it uploads a 'STOP' command to the host's remote executor.

colMapping *Add column mappings*

Description

Piping compatible function for `modelColumnMapping` used to add column mappings from input data to model object

Usage

```
colMapping(.Object, mappings = NULL, ...)
```

Arguments

.Object	Model (NlmePmlModel) object
mappings	Named character vector specifying valid column names in the input data. Character vector names must be valid model variable names contained in modelVariableNames(model).
...	optional pairs ModelTerm = ColumnName or ModelTerm = "ColumnName". Has higher precedence than mappings if some ModelTerm is mapped twice in mappings and in .... For multiple mapping, i.e. id mapping, a vector should be provided with the names of columns. See example below.

Value

modified NlmePmlModel object

See Also

dataMapping modelVariableNames

Examples

```
pkData$id2 <- pkData$Subject
model <- pkmodel(columnMap = FALSE,
                 data = pkData)

modelvar <- unlist(modelVariableNames(model))

colnames <- c("Subject", "Act_Time", "Amount", "Conc")
names(colnames) <- modelvar
# will map subject directly
colnames <- colnames[-c(1)]

model <- colMapping(model, colnames, id = c(Subject, id2))
# also possible:
model <- colMapping(model, colnames, id = c("Subject", "id2"))
# not recommended since only not quoted names are identified
# if both types are provided:
model <- colMapping(model, colnames, id = c("Subject", id2))
```

---

copyModel *Copy model object to iterate over base model*

Description

Copies previously executed model into a new object and optionally accept all estimates returned from model execution. A new working directory is created and all files from base model are copied into it.

Usage

```
copyModel(model, acceptAllEffects = FALSE, modelName = "", workingDir = "")
```

Arguments

model	Model object to be copied
acceptAllEffects	Set to TRUE to accept all effects, update PML statements, and test.mdl file from original model run
modelName	New model name for subdirectory created for model output. Subdirectory is created in current working directory.
workingDir	Working directory to run the model. Current working directory will be used if workingDir not specified.

Value

Modified NlmePmlModel object

Examples

```
# Create initial model
model <- pkmodel(
  parameterization = "Clearance",
  absorption = "Intravenous",
  numCompartments = 2,
  data = pkData,
  ID = "Subject",
  A1 = "Amount",
  CObs = "Conc",
  Time = "Act_Time",
  modelName = "pk_model"
)

# Fit Model
job <- fitmodel(model)

# Copy model and accept all effects from the original model run
vpcModel <- copyModel(model, acceptAllEffects = TRUE, modelName = "vpc_model")
```

---

covariateNames *Return covariate names*

Description

Use to return character vector of covariate names available in model object.

Usage

```
covariateNames(model)
```

Arguments

model	Model object

Value

Character vector of covariate names defined in model

Examples

```
model <- pkmodel(columnMap = FALSE)
model <- addCovariate(model, covariate = "BW", effect = "V")
model <- addCovariate(model, covariate = "Age", effect = "Cl")

covariateNames(model)
```

---

createModelInfo *Parse the model and get the list of terms*

Description

Calls TDL5 to parse the model and get the list of terms

Usage

```
createModelInfo(model, ForceRun = FALSE)
```

Arguments

model	Model object
ForceRun	Set to TRUE to force run

Value

List of model information

Examples

```
createModelInfo(model)
```

dataMapping *Initialize input data for PK/PD model*

Description

Used to initialize input data for PK/PD model

Usage

```
dataMapping(.Object, data)
```

Arguments

.Object	Model object
data	Input data of class data.frame.

Value

Modified NlmePmlModel object

See Also

colMapping

Examples

```
model <- pkmodel(columnMap = FALSE)

model <- dataMapping(model, pkData)
```

doseNames *Return dose names*

Description

Use to return character vector of dose point names in model object.

Usage

```
doseNames(model)
```

Arguments

model	Model object

Value

Character vector of dose names defined in model

Examples

```
model <- pkmodel(columnMap = FALSE)

doses <- doseNames(model)
```

`editModel` *Directly edit PML text in model object*

Description

Allows user to edit PML text in model object using internal text editor and return a new textual model containing the edited PML statements.

Usage

```
editModel(.Object)
```

Arguments

`.Object`	Model object

Value

Modified `NlmePmlModel` object

Examples

```
model <- pkmodel(columnMap = FALSE)

newModel <- editModel(model)
```

`emaxmodel` *Create an Emax or Imax model*

Description

Use to create an Emax or Imax model

Usage

```
emaxmodel(
  isPopulation = TRUE,
  checkBaseline = FALSE,
  checkFractional = FALSE,
  checkInhibitory = FALSE,
  checkSigmoid = FALSE,
  data = NULL,
  columnMap = TRUE,
  modelName = "",
  workingDir = "",
  ...
)
```

Arguments

`isPopulation`	Is this a population model `TRUE` or individual model `FALSE`?
`checkBaseline`	Set to `TRUE` if the model contains a baseline response.
`checkFractional`	Set to `TRUE` to modify the default form for the model. Only applicable to models with `checkBaseline = TRUE`.
`checkInhibitory`	Set to `TRUE` to change the model from an Emax to an Imax model.
`checkSigmoid`	Set to `TRUE` to change the model to its corresponding signmoid form.
`data`	Input dataset
`columnMap`	If `TRUE` (default) column mapping arguments are required. Set to `FALSE` to manually map columns after defining model using `colMapping`.
`modelName`	Model name for subdirectory created for model output in current working directory.
`workingDir`	Working directory to run the model. Current working directory will be used if `workingDir` not specified.
`...`	Arguments passed on to `emaxmodel_MappingParameters` `ID` Column mapping argument for input dataset column(s) that identify individual data profiles. Only applicable to population models `isPopulation = TRUE`.

`C` Column mapping argument that represents the input dataset column for the independent variable that is treated as a covariate during the estimation/simulation process.

`EObs` Column mapping argument that represents the input dataset column for the observed drug effect (i.e., the dependent variable).

Value

`NlmePmlModel` object

Column mapping

Note that quoted and unquoted column names are supported. Please see `colMapping`.

Examples

```
model <- emaxmodel(data = pkpdData, ID = "ID", C = "CObs", EObs = "EObs")

model <- emaxmodel(
  checkBaseline = TRUE,
  checkFractional = TRUE,
  checkInhibitory = TRUE,
  data = pkpdData,
  ID = "ID",
  C = "CObs",
  EObs = "EObs"
)

# View PML Code
print(model)
```

`engineParams` *Specify engine parameters for model execution*

Description

Use to define extra engine parameters for model execution.

Usage

```
engineParams(
  model,
  sort = NULL,
  ODE = "MatrixExponent",
  rtolODE = 1e-06,
  atolODE = 1e-06,
  maxStepsODE = 50000,
  numIterations = 1000,
```

```
  method = NULL,
  stdErr = NULL,
  isCentralDiffStdErr = TRUE,
  stepSizeStdErr = NULL,
  numIntegratePtsAGQ = 1,
  numIterNonParametric = 0,
  allowSyntheticGradient = FALSE,
  numIterMAPNP = 0,
  numRepPCWRES = 0,
  stepSizeLinearize = 0.002,
  numDigitLaplacian = 7,
  numDigitBlup = 13,
  mapAssist = 0,
  iSample = 300,
  iAcceptRatio = 0.1,
  impDist = "Normal",
  tDOF = 4,
  numSampleSIR = 10,
  numBurnIn = 0,
  freezeOmega = FALSE,
  MCPEM = FALSE,
  runAllIterations = FALSE,
  scramble = "Owen",
  stepSizePartialDeriv = 1e-05,
  numTimeStepPartialDeriv = 20
)
```

Arguments

`model`	Model object
`sort`	Logical; Specifying whether or not to sort the input data by subject and time values. • If `model@hasResetInfo = TRUE`, then `sort` must be set to `FALSE` (default); • Otherwise, the default value for `sort` is `TRUE`.
`ODE`	Character; Specifying the solver used to numerically solve Ordinary Differential Equations (ODEs). Options are '"MatrixExponent"', '"Higham"', '"DVERK"', '"DOPRI5"', '"AutoDetect"', '"Stiff"'. See Details section.
`rtolODE`	Numeric; Specifying relative tolerance for the numerical ODE solver.
`atolODE`	Numeric; Specifying absolute tolerance for the numerical ODE solver.
`maxStepsODE`	Numeric; Specifying maximum number of allowable steps or function evaluations for the ODE solver.
`numIterations`	Numeric; Specifying maximum number of iterations for estimation.
`method`	Character; Specifying engine method for estimation. For population models, options are `"QRPEM"`, `"IT2S-EM"`, `"FOCE-LB"`, `"FO"`, `"FOCE-ELS"`, `"Laplacian"`, `"Naive-Pooled"`. While, for individual models, `"Naive-Pooled"` is the only option.

Note: For population models, if model involves any discontinuous observed variable (e.g., count data) or BQL data, the default method is "Laplacian"; otherwise, the default method is "FOCE-ELS".

stdErr Character; Specifying method for standard error computations.

- For individual models, options are "Hessian" (default) and "None";
- For population models with method = "QRPEM", options are "Fisher-Score" (default) and "None";
- For population models with method = "IT2s-EM", the only option is "None";
- For population models with method set to either "FOCE-LB", "FO", "FOCE-ELS", "Laplacian", or "Naive-Pooled", options are "Sandwich" (default), "Hessian", "Fisher-Score", "Auto-Detect", and "None".

Here "None" means that standard error calculations are not performed.

isCentralDiffStdErr
Logical; Default TRUE uses central difference for stdErr calculations. Set to FALSE for forward difference method.

stepSizeStdErr Numeric; Specifying the step size used for stdErr calculations. If not specified, 0.01 is used for population models and 0.001 for individual models.

numIntegratePtsAGQ
Numeric; Specifying the number of integration points for adaptive Gaussian quadrature (AGQ) algorithm. Only applicable to population models with method set to either "FOCE-ELS" or "Laplacian".

numIterNonParametric
Numeric; Specifying the number of iterations to perform non-parametric estimation. Only applicable to population models when method is not set to Naive-Pooled.

allowSyntheticGradient
Logical, Set to TRUE to use synthetic gradient during the estimation process. Only applicable to population models when method is not set to Naive-Pooled.

numIterMAPNP Numeric; Specifying the number of iterations to perform Maximum A Posterior (MAP) initial Naive Pooling (NP) run before estimation. Only applicable to population models when method is not set to Naive-Pooled.

numRepPCWRES Numeric; Specifying the number of replicates to generate the PCWRES after the simple estimation. Only applicable to population models when method is not set to Naive-Pooled.

stepSizeLinearize
Numeric; Specifying the step size used for numerical differentiation when linearizing the model function during the estimation process.

numDigitLaplacian
Numeric; Specifying the number of significant decimal digits for the Laplacian algorithm to use to reach convergence. Only applicable to population models.

numDigitBlup Numeric; Specifying the number of significant decimal digits for the individual estimation to use to reach convergence. Only applicable to population models.

mapAssist Numeric; Specifying the period used to perform MAP assistance (mapAssist = 0 means that MAP assistance is not performed). Only applicable to population models with method = "QRPEM".

`iSample` Numeric; Specifying the number of samples. Only applicable to population models with `method = "QRPEM"`.

`iAcceptRatio` Numeric; Specifying the acceptance ratio. Only applicable to population models with `method = "QRPEM"`.

`impDist` Character; Specifying the distribution used for important sampling, and options are `"Normal"` (default), `"DoubleExponential"`, `"Direct"`, `"T"`, `"Mixture-2"`, `Mixture-3`. Only applicable to population models with `method = "QRPEM"`.

`tDOF` Numeric; Specifing the degree of freedom (allowed value is between 3 and 30) for T distribution. Only applicable to population models with `method = "QRPEM"` and `impDist = "T"`.

`numSampleSIR` Numeric; Specifying the number of samples per subject used in the Sampling Importance Re-Sampling (SIR) algorithm to determine the number of SIR samples taken from the empirical discrete distribution that approximates the target conditional distribution. Only applicable to population models with `method = "QRPEM"`.

`numBurnIn` Numeric; Specifying the number of burn-in iterations to perform at startup to adjust certain internal parameters. Only applicable to population models with `method = "QRPEM"`.

`freezeOmega` Logical; Set to `TRUE` to freeze Omega but not Theta for the number of iterations specified in the `numBurnIn`. Only applicable to population models with `method = "QRPEM"`.

`MCPEM` Logical; Set to `TRUE` to use Monte-Carlo sampling instead of Quasi-Random. Only applicable to population models with `method = "QRPEM"`.

`runAllIterations`
Logical; Set to `TRUE` to execute all requested iterations specified in `numIterations`. Only applicable to population models with `method = "QRPEM"`.

`scramble` Character; Specifying the quasi-random scrambling method to use, and options are `"Owen"`, `"Tezuka-Faur"`, or `"None"`. Only applicable to population models with `method = "QRPEM"`.

`stepSizePartialDeriv`
Numeric; Specifying the step size used to numerically calculate the partial derivatives of observed variables with respect to parameters. Only applicable to individual models.

`numTimeStepPartialDeriv`
Numeric; Specifying the number of time steps used to output the partial derivatives of observed variables with respect to parameters. Only applicable to individual models.

Details

Both ‘"DVERK"‘ and ‘"DOPRI5"‘ are non-stiff solvers. ‘"Higham"‘ is a matrix exponent based ODE solver which could be useful when overscaling issue should be avoided, i.e. the ratio between observed values and doses is too high or too low. ‘"AutoDetect"‘ represents LSODA solver implemenation, which solves the initial value problem for stiff or nonstiff systems of first order ordinary differential equations. ‘"Stiff"‘ is a LSODE (Livermore solver). It is best suited for stiff problems.

Value

List of engine parameters to be used during fitting or simulation

---

`extraDoseLines` *Return extra dose lines*

Description

Use to return extra dose lines for model object

Usage

```
extraDoseLines(model)
```

Arguments

`model`	Model object

Value

List of extra dose information

Examples

```
data <- pkData
data$II <- 24
data$ADDL <- 1

model <-
pkmodel(
  parameterization = "Clearance",
  numCompartments = 2,
  data = data,
  ID = "Subject",
  Time = "Act_Time",
  A1 = "Amount",
  CObs = "Conc") |>
addDoseCycle(
  name = "A1",
  amount = 30000,
  II = 24,
  type = "ADDL",
  colName = "ADDL")

extraDoseLines(model)
```

`extraDoseNames` *Return extra dose names*

Description

Use to return extra dose names for model object

Usage

```
extraDoseNames(model)
```

Arguments

`model`	Model object

Value

Character vector of extra dose names

Examples

```
data <- pkData
data$II <- 24
data$ADDL <- 1

model <-
pkmodel(
  parameterization = "Clearance",
  numCompartments = 2,
  data = data,
  ID = "Subject",
  Time = "Act_Time",
  A1 = "Amount",
  CObs = "Conc") |>
addDoseCycle(
  name = "A1",
  amount = 30000,
  II = 24,
  type = "ADDL",
  colName = "ADDL")

extraDoseNames(model)
```

`fitmodel` *Executes an NLME simple estimation*

Description

Executes an NLME simple estimation

Usage

```
fitmodel(
  model,
  hostPlatform = NULL,
  params,
  simpleTables,
  runInBackground = FALSE,
  filesToReturn = "*",
  ...
)
```

Arguments

`model`	PK/PD model class object.
`hostPlatform`	Host definition for model execution. See `hostParams`. If `missing`, PhoenixMPIDir64 is given and MPI is installed, MPI local host with 4 threads is used. If MPI is not found, local host without parallelization is used.
`params`	Engine parameters. See `engineParams`. If `missing`, default parameters generated by engineParams(model) are used.
`simpleTables`	Optional list of simple tables. See `tableParams`. By default a table named 'posthoc.csv' is returned with structural parameters values for all source data rows.
`runInBackground`	Set to `TRUE` to run in background and return prompt.
`filesToReturn`	Used to specify which files to be outputted to the model directory and loaded as returned value. By default, all the applicable files listed in the `Value` section will be outputted to the model directory and loaded as returned value. Only those files listed in the `Value` section can be specified. Simple regex patterns are supported for the specification.
`...`	Additional arguments for `hostParams` or arguments available inside `engineParams` functions. If `engineParams` arguments are supplied through both `params` argument and additional argument (i.e., ellipsis), then the arguments in `params` will be ignored and only the additional arguments will be used with warning. If `hostParams` arguments are supplied through both the `hostPlatform` argument and the ellipses, values supplied to `hostPlatform` will be overridden by additional arguments supplied via the ellipses e.g., `....`

Value

if `runInBackground` is `FALSE`, a list with main resulted dataframes is returned:

- Overall
- ConvergenceData
- residuals
- Secondary
- StrCovariate - if continuous covariates presented
- StrCovariateCat - if categorical covariates presented
- theta
- posthoc table
- posthocStacked table
- Requested tables

`nlme7engine.log` textual output is returned and loaded with the main information related to fitting. `dmp.txt` structure with the results of fitting (including LL by subject information) is returned and loaded. These 2 files are returned and loaded irrespective of `filesToReturn` argument value.

For individual models, additional dataframe with partial derivatives is returned:

- ParDer

For population models and the method specified is NOT `Naive-Pooled`, additional dataframes are returned:

- omega
- Eta
- EtaStacked
- EtaEta
- EtaCov
- EtaCovariate - if continuous covariates presented
- EtaCovariateCat - if categorical covariates presented
- bluptable.dat

If standard error computation was requested and it was successful, additional dataframes are returned:

- thetaCorrelation
- thetaCovariance
- Covariance
- omega_stderr

If nonparametric method was requested (`numIterNonParametric` > 0) and the `method` specified in `engineParams` is NOT `Naive-Pooled`, additional dataframes are returned:

- nonParSupportResult

- nonParStackedResult
- nonParEtaResult
- nonParOverallResult

if `runInBackground` is `TRUE`, only current status of job is returned.

`filesToReturn` **with** `Certara.Xpose.NLME`

If `filesToReturn` is used and "ConvergenceData.csv" and "residuals.csv" are not in the patterns, these files won't be returned and loaded. These files are essential for `Certara.Xpose.NLME::xposeNlmeModel` and `Certara.Xpose.NLME::xposeNlme` functions. This makes impossible to use the resulted object in `Certara.Xpose.NLME` functions.

Non-loaded but returned files

The non-loaded but returned files in the model working directory are:

- err1.txt - concatenated for all runs detailed logs for all steps of optimization,
- out.txt - general pivoted information about results,
- doses.csv - information about doses given for all subjects,
- iniest.csv - information about initial estimates

See Also

`tableParams`, `hostParams`,`engineParams`

Examples

```
# Define the host
host <- hostParams(parallelMethod = "None",
                   hostName = "local",
                   numCores = 1)
# Define the model
model <- pkmodel(numComp = 2,
                 absorption = "FirstOrder",
                 ID = "Subject",
                 Time = "Act_Time",
                 CObs = "Conc",
                 Aa = "Amount",
                 data = pkData,
                 modelName = "PkModel")

Table01 <- tableParams(name = "SimTableObs.csv",
                       timesList = "0,1,2,4,4.9,55.1,56,57,59,60",
                       variablesList = "C, CObs",
                       timeAfterDose = FALSE,
                       forSimulation = FALSE)

# Update fixed effects
```

```
model <- fixedEffect(model,
                     effect = c("tvV", "tvCl", "tvV2", "tvCl2"),
                     value = c(16, 41, 7, 14))

# Define the engine parameters
params <- engineParams(model)

# Fit model
res <- fitmodel(model = model,
                hostPlatform = host,
                params = params,
                simpleTables = Table01)
```

`fixedEffect` *Specifies the initial values, lower bounds, upper bounds, and units for fixed effects in a model*

Description

Specifies the initial values, lower bounds, upper bounds, and units for fixed effects in a model

Usage

```
fixedEffect(
  .Object,
  effect,
  value = NULL,
  lowerBound = NULL,
  upperBound = NULL,
  isFrozen = NULL,
  unit = NULL
)
```

Arguments

`.Object`	Model object in which to define fixed effects values
`effect`	Character or character vector specifying names of fixed effects
`value`	Numeric or numeric vector specifying the initial values of fixed effects. If supplying vector, must be in the same order/length as corresponding `effect`.
`lowerBound`	Numeric or numeric vector specifying the lower limit values of fixed effects. If supplying vector, must be in the same order as `effect`.
`upperBound`	Numeric or numeric vector specifying the upper limit values of fixed effects. If supplying vector, must be in the same order as `effect`.
`isFrozen`	Logical or logical vector. Set to `TRUE` to freeze the fixed effect to the specified initial value. If supplying vector, must be in the same order as `effect`.
`unit`	Character or character vector specifying units of measurement for the fixed effects. If supplying a vector, must be in the same order as `effect`.

Value

Modified NlmePmlModel object

Examples

```
model <- pkmodel(
  numCompartments = 2,
  data = pkData,
  ID = "Subject",
  Time = "Act_Time",
  A1 = "Amount",
  CObs = "Conc",
  modelName = "TwCpt_IVBolus_FOCE_ELS"
  )

 # View initial/current fixed effect values
 initFixedEffects(model)

model <- model |>
fixedEffect(
  effect = c("tvV", "tvCl", "tvV2", "tvCl2"),
  value = c(15, 5, 40, 15)
  )
```

---

getRandomEffectNames *Return random effect names in model*

Description

Use to return character vector of random effect names (if available) in model object

Usage

```
getRandomEffectNames(model)
```

Arguments

model	Model object

Value

Characters vector of random effect names

Examples

```
model <- pkmodel(columnMap = FALSE)
getRandomEffectNames(model)
```

getThetas *Return theta names and values*

Description

Returns named character vector of theta values by parsing PML fixed effect statements

Usage

```
getThetas(model)
```

Arguments

model	PK/PD model

Value

Character vector of theta names defined in model

Examples

```
getThetas(pkpdmodel)
```

hostParams *Initialize for NlmeParallelHost*

Description

Initialize for NlmeParallelHost

Usage

```
hostParams(
  sharedDirectory,
  installationDirectory = Sys.getenv("INSTALLDIR"),
  hostName = Sys.info()[["nodename"]],
  machineName = "127.0.0.1",
  hostType = Sys.info()[["sysname"]],
  numCores = 4,
  parallelMethod = "LOCAL_MPI",
  userName = "",
  privateKeyFile = NULL,
  userPassword = "",
  scriptPath = "",
  rLocation = "",
  isLocal = TRUE
)
```

Arguments

sharedDirectory	Directory where temporary NLME run folder is created during execution. If missing, the current working directory will be used.
installationDirectory	Directory containing NLME libraries/scripts
hostName	Visual name of the host (default A name by which the machine is known on the network)
machineName	IP address or name of the host(default 127.0.0.1)
hostType	windows or linux. Current OS by default. For remote runs it is possible to point the distro suppported, i.e. RHEL8 or UBUNTU2204. In such case the corresponding PML_BIN_DIR variable will be created and NLME Engine libraries will be looked in installationDirectory/{$PML_BIN_DIR}.
numCores	Integer; Number of compute cores. 4 by default
parallelMethod	String; Options are: None|Multicore|LOCAL_MPI|SGE|SGE_MPI| TORQUE|TORQUE_MPI|LSF|LSF_MPI|SLURM_
userName	String; How the user is identified to the remote system
privateKeyFile	Path to private key file, see ssh::ssh_connect() for details
userPassword	Either a string or a callback function for password prompt, see ssh::ssh_connect() for details
scriptPath	a path to the script to be executed before starting Rscript within Certara.NLME8 package on the remote host. Ignored when running locally.
rLocation	Path to Rscript executable on remote host; ignored on local host
isLocal	Is this a local TRUE or remote FALSE host?

Value

NlmeParallelHost class instance

Examples

```
host <- hostParams(sharedDirectory = tempdir(),
                   parallelMethod = "LOCAL_MPI",
                   hostName = "Local",
                   numCores = 4)
```

---

initFixedEffects *Display/Set initial estimates for fixed effects*

Description

Display/Set initial estimates for fixed effects

Usage

```
initFixedEffects(.Object)

## S4 method for signature 'NlmePmlModel'
initFixedEffects(.Object)

initFixedEffects(.Object) <- value

## S4 replacement method for signature 'NlmePmlModel'
initFixedEffects(.Object) <- value
```

Arguments

.Object	PK/PD model
value	Named numeric vector

Value

Named numeric vector of fixed effects estimates

See Also

fixedEffect

Examples

```
model <- pkmodel(
  numCompartments = 2,
  data = pkData,
  ID = "Subject",
  Time = "Act_Time",
  A1 = "Amount",
  CObs = "Conc",
  modelName = "TwCpt_IVBolus_FOCE_ELS"
  )

# View initial/current fixed effect values
 initFixedEffects(model)

# May also use as a 'replacement function' to set the values
initFixedEffects(model) <- c(tvV = 15, tvCl = 5, tvV2 = 40, tvCl2 = 15)
```

`linearmodel` *Create linear model*

Description

Use to create a constant, linear, or quadratic PD model

Usage

```
linearmodel(
  isPopulation = TRUE,
  type = "Constant",
  data = NULL,
  columnMap = TRUE,
  modelName = "",
  workingDir = "",
  ...
)
```

Arguments

`isPopulation`	Is this a population model `TRUE` or individual model `FALSE`?
`type`	Model type. Options are `"Constant"`, `"Linear"`, `"Quadratic"`.
`data`	Input dataset
`columnMap`	If `TRUE` (default) column mapping arguments are required. Set to `FALSE` to manually map columns after defining model using `colMapping`.
`modelName`	Model name for subdirectory created for model output in current working directory.
`workingDir`	Working directory to run the model. Current working directory will be used if `workingDir` not specified.
`...`	Arguments passed on to `linearmodel_MappingParameters` `ID` Column mapping argument for input dataset column(s) that identify individual data profiles. Only applicable to population models `isPopulation = TRUE`. `C` Column mapping argument that represents the input dataset column for the independent variable that is treated as a covariate during the estimation/simulation process. `EObs` Column mapping argument that represents the input dataset column for the observed drug effect (i.e., the dependent variable).

Value

`NlmePmlModel` object

Column mapping

Note that quoted and unquoted column names are supported. Please see colMapping.

Examples

```
model <- linearmodel(type = "Linear", data = pkpdData, ID = "ID", C = "CObs", EObs = "EObs")

# View PML Code
print(model)
```

---

listCovariateEffectNames

Lists covariate effect names in the model

Description

This function lists the names of covariate effects in a provided pharmacokinetic/pharmacodynamic (PK/PD) model.

Usage

```
listCovariateEffectNames(.Object)

## S4 method for signature 'NlmePmlModel'
listCovariateEffectNames(.Object)
```

Arguments

.Object	PK/PD model

Value

A vector of character strings containing the names of the covariate effects in the model.

Examples

```
listCovariateEffectNames(model)
```

modelVariableNames *Return model variable names*

Description

Return a vector of model variable names from model object

Usage

```
modelVariableNames(model)
```

Arguments

model	Model object

Value

Character vector of required model variable names

Examples

```
modelVariableNames(model)
```

obtain_NLMELicense *Obtain NLME License*

Description

This function attempts to authenticate and obtain an NLME license using the specified installation directory and licensing tool.

Usage

```
obtain_NLMELicense(
  InstallDir = Sys.getenv("INSTALLDIR"),
  ForceAuth = FALSE,
  ForceLicenseGet = FALSE,
  verbose = getOption("verbose")
)
```

Arguments

InstallDir	A character string specifying the directory where the NLME Engine is installed e.g., INSTALLDIR environment variable. The cadlicensingtool executable is expected to be located within this directory, or within a subdirectory specified by the PML_BIN_DIR environment variable.
ForceAuth	A logical value indicating whether to force re-authentication even if already authenticated. Default is FALSE.
ForceLicenseGet	A logical value indicating whether to force obtaining the license even if already licensed. Default is FALSE.
verbose	A logical value indicating whether to print verbose output. Default is getOption("verbose").

Details

This function checks for the presence of the necessary appsettings.json file as indicated by the CAD_CONFIG_FILE environment variable, runs the licensing tool to authenticate the user, and attempts to obtain an NLME license. It prints detailed messages if the verbose parameter is set to TRUE.

Value

A logical value indicating whether the license was successfully obtained.

Examples

```
result <- obtain_NLMELicense("C:/Program Files/Certara/NLME_Engine", verbose = TRUE)
if (result) {
  message("License obtained successfully!")
} else {
  message("Failed to obtain license.")
}
```

OneCpt_IVInfusionData *Pharmacokinetic dataset containing 100 subjects with single dose given by infusion*

Description

Pharmacokinetic dataset containing 16 subjects with single dose given by infusion.

Usage

```
OneCpt_IVInfusionData
```

Format

A data frame with 800 rows and 6 variables:

Subject Subject ID

Time Time point

Dose Amount of dose

CObs Observations of drug concentration in blood

Rate Rate of infusion

Duration Duration of infusion

Source

The data is simulated using a PK model described by a one-compartment model with IV infusion

---

parsePMLColMap *Embed column definition info into the model*

Description

Add/update column definition information for the model object

Usage

```
parsePMLColMap(.Object, ForceRun = TRUE)
```

Arguments

`.Object`	Model (`NlmePmlModel`) object
`ForceRun`	Set to `TRUE` to force run

Details

Intended to be used by other packages

Value

modified `NLMEPmlModel` object with column mapping definitions

pkcovbqlData *Pharmacokinetic pediatric dataset containing 80 subjects with single bolus dose.*

Description

Pharmacokinetic pediatric dataset containing 80 subjects with single bolus dose. Dataset includes covariates and observations Below Quantification Limit (BQL).

Usage

```
pkcovbqlData
```

Format

A data frame with 880 rows and 8 variables:

ID Subject ID

Time Nominal Time

Dose Amount of dose

CObs Observations of drug concentration in blood

LLOQ Lower Limit of Quantification

CObsBQL Variable that indicates whether the observed drug concentration is below the limit of quantification

BW Body weight

PMA Postmenstrual age

Source

The data is simulated using a one-compartment model with IV bolus, where the central volume is allometric weight scaled, and the clearance is scaled by a combination of allometric weight scaling and a sigmoidal maturation function driven by PMA. Germovsek E., et al, Pharmacokinetic–Pharmacodynamic Modeling in Pediatric Drug Development, and the Importance of Standardized Scaling of Clearance, Clin Pharmacokinet (2019) 58:39–52.

pkData — *Pharmacokinetic dataset containing 16 subjects with single bolus dose*

Description

Pharmacokinetic dataset containing 16 subjects with single bolus dose.

Usage

```
pkData
```

Format

A data frame with 112 rows and 8 variables:

Subject Subject ID

Nom_Time Nominal Time

Act_Time Actual Time

Amount Amount of dose

Conc Observations of drug concentration in blood

Age Age

BodyWeight Body weight

Gender Gender ("male", "female")

Source

Certara University

pkemaxmodel — *Create a PK/Emax or PK/Imax model*

Description

Use to create a PK/Emax or PK/Imax model

Usage

```
pkemaxmodel(
  isPopulation = TRUE,
  parameterization = "Clearance",
  absorption = "Intravenous",
  numCompartments = 1,
  isClosedForm = TRUE,
  isTlag = FALSE,
  hasEliminationComp = FALSE,
  isFractionExcreted = FALSE,
  isSaturating = FALSE,
  infusionAllowed = FALSE,
  isDuration = FALSE,
  isSequential = FALSE,
  isPkFrozen = FALSE,
  hasEffectsCompartment = FALSE,
  checkBaseline = FALSE,
  checkFractional = FALSE,
  checkInhibitory = FALSE,
  checkSigmoid = FALSE,
  isEmaxFrozen = FALSE,
  data = NULL,
  columnMap = TRUE,
  modelName = "",
  workingDir = "",
  ...
)
```

Arguments

isPopulation	Is this a population model TRUE or individual model FALSE?
parameterization	Type of parameterization. Options are "Clearance", "Micro", "Macro", or "Macro1".
absorption	Type of absorption. Options are "Intravenous", "FirstOrder", "Gamma", "InverseGaussian", "Weibull" .
numCompartments	Value of either 1, 2, or 3.
isClosedForm	Set to TRUE to convert model from a differential equation to close form.
isTlag	Set to TRUE to add a lag time parameter to the model.
hasEliminationComp	Set to TRUE to add an elimination compartment to the model.
isFractionExcreted	Set to TRUE if elimination compartment (hasEliminationComp = TRUE) contains a fraction excreted parameter.
isSaturating	Set to TRUE to use Michaelis-Menten kinetics for elimination. Only applicable to models with paramteterization = "Clearance"

infusionAllowed
: Set to TRUE if infusions allowed.

isDuration
: Set to TRUE if infusions use duration instead of rate (must also set infusionAllowed = TRUE).

isSequential
: Set to TRUE to freeze PK fixed effects and convert the corresponding random effects into covariates as well as remove the PK observed variable from the model.

isPkFrozen
: Set to TRUE to freeze PK fixed effects and remove the corresponding random effects as well as the PK observed variable from the model.

hasEffectsCompartment
: Set to TRUE to include an effect compartment into the model.

checkBaseline
: Does Emax/Imax model have a baseline response?

checkFractional
: Set to TRUE to modify the default form for the Emax/Imax model. Only applicable to models with checkBaseline = TRUE.

checkInhibitory
: Set to TRUE to change the default Emax to Imax model.

checkSigmoid
: Set to TRUE to change the Emax/Imax to its corresponding sigmoid form.

isEmaxFrozen
: Set to TRUE to freeze PD fixed effects and remove the corresponding random effects as well as the PD observed variable from the model.

data
: Input dataset

columnMap
: If TRUE (default) column mapping arguments are required. Set to FALSE to manually map columns after defining model using colMapping.

modelName
: Model name for subdirectory created for model output in current working directory.

workingDir
: Working directory to run the model. Current working directory will be used if workingDir not specified.

...
: Arguments passed on to pkindirectmodel_MappingParameters

  ID Column mapping argument for input dataset column(s) that identify individual data profiles. Only applicable to population models isPopulation = TRUE.

  Time Column mapping argument that represents the input dataset column for the relative time used in a study and only applicable to time-based models.

  A1 Column mapping argument that represents the input dataset column for the amount of drug administered. Only applicable to the following types of models:

  - Models with absorption = "Intravenous" and parameterization set to either "Clearance","Micro", or "Macro"
  - Models with absorption set to either "Gamma", "InverseGaussian", or "Weibull"

  Aa Column mapping argument that represents the input dataset column for the amount of drug administered and only applicable to models with absorption = "FirstOrder".

`A` Column mapping argument that represents the input dataset column for the amount of drug administered and only applicable to models with `absorption = "Intravenous"` and `parameterization = "Macro1"`.

`A1_Rate` Column mapping argument that represents the input dataset column for the rate of drug administered. Only applicable to the following types of models:

- Models with `absorption = "Intravenous"`, `infusionAllowed = TRUE` and parameterization set to either `"Clearance"`,`"Micro"` or `"Macro"`
- Models with `absorption` set to either `"Gamma"`, `"InverseGaussian"`, or `"Weibull"` and `infusionAllowed = TRUE`

`A1_Duration` Column mapping argument that represents the input dataset column for the duration of drug administered. Only applicable to the following types of models:

- Models with `absorption = "Intravenous"`, `infusionAllowed = TRUE` with `isDuration = TRUE` and parameterization set to either `"Clearance"`,`"Micro"` or `"Macro"`
- Models with `absorption` set to either `"Gamma"`, `"InverseGaussian"`, or `"Weibull"` and `infusionAllowed = TRUE` with `isDuration = TRUE`

`Aa_Rate` Column mapping argument that represents the input dataset column for the rate of drug administered and only applicable to models with `absorption = "FirstOrder"`, `infusionAllowed = TRUE`.

`Aa_Duration` Column mapping argument that represents the input dataset column for the duration of drug administered and only applicable to models with `absorption = "FirstOrder"`, `infusionAllowed = TRUE`, and `isDuration = TRUE`.

`A_Rate` Column mapping argument that represents the input dataset column for the rate of drug administered and only applicable to models with `absorption = "Intravenous"`, `infusionAllowed = TRUE`, and `parameterization = "Macro1"`.

`A_Duration` Column mapping argument that represents the input dataset column for the duration of drug administered and only applicable to models with `absorption = "Intravenous"`, `infusionAllowed = TRUE`, `isDuration = TRUE`, and `parameterization = "Macro1"`.

`A1Strip` Column mapping argument that represents the input dataset column for the stripping dose and only applicable to models with `parameterization = "Macro"`.

`C0bs` Column mapping argument that represents the input dataset column for the observations of drug concentration in the central compartment and only applicable to models with `parameterization` being either set to either `"Clearance"` or `"Micro"`.

`C1Obs` Column mapping argument that represents the input dataset column for the observations of drug concentration in the central compartment and only applicable to models with `parameterization` being either set to either `"Macro"` or `"Macro1"`.

`A0Obs` Column mapping argument that represents the input dataset column for the observed amount of drug in the elimination compartment. (`hasEliminationComp = TRUE`).

EObs Column mapping argument that represents the input dataset column for the observed drug effect.

nV If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nV.

nV2 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nV2.

nV3 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nV3.

nCl If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nCl.

nCl2 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nCl2.

nCl3 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nCl3.

nKa If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nKa.

nA If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nA.

nAlpha If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nAlpha.

nB If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nB.

nBeta If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nBeta.

nC If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nC.

nGamma If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nGamma.

nKe If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nKe.

nK12 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nK12.

nK21 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nK21.

nK13 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nK13.

nK31 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nK31.

nTlag If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nTlag.

nKm If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nKm.

nVmax If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nVmax.

nFe If isSequential = TRUE and isFractionExcreted = TRUE, mapped to the input dataset column that lists the values for random effect nFe.

nMeanDelayTime If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nMeanDelayTime.

nShapeParam If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nShapeParam.

nShapeParamMinusOne If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nShapeParamMinusOne.

Value

NlmePmlModel object

Column mapping

Note that quoted and unquoted column names are supported. Please see colMapping.

Examples

```
model <- pkemaxmodel(
  parameterization = "Macro",
  data = pkpdData,
  Time = "Time",
  ID = "ID",
  A1 = "Dose",
  C1Obs = "CObs",
  EObs = "EObs"
)

# View the model as well as its associated column mappings
print(model)
```

---

pkindirectmodel *Create a PK/Indirect response model*

Description

Use to create a PK/Indirect response model.

Usage

```
pkindirectmodel(
  isPopulation = TRUE,
  parameterization = "Clearance",
  absorption = "Intravenous",
  numCompartments = 1,
  isClosedForm = TRUE,
  isTlag = FALSE,
  hasEliminationComp = FALSE,
```

```
  isFractionExcreted = FALSE,
  isSaturating = FALSE,
  infusionAllowed = FALSE,
  isDuration = FALSE,
  isSequential = FALSE,
  isPkFrozen = FALSE,
  hasEffectsCompartment = FALSE,
  indirectType = "LimitedStimulation",
  isBuildup = TRUE,
  isExponent = FALSE,
  indirectFrozen = FALSE,
  data = NULL,
  columnMap = TRUE,
  modelName = "",
  workingDir = "",
  ...
)
```

Arguments

isPopulation	Is this a population model TRUE or individual model FALSE?
parameterization	Type of parameterization. Options are "Clearance", "Micro", "Macro", or "Macro1".
absorption	Type of absorption. Options are "Intravenous", "FirstOrder", "Gamma", "InverseGaussian", "Weibull" .
numCompartments	Value of either 1, 2, or 3.
isClosedForm	Set to TRUE to convert model from a differential equation to close form.
isTlag	Set to TRUE to add a lag time parameter to the model.
hasEliminationComp	Set to TRUE to add an elimination compartment to the model.
isFractionExcreted	Set to TRUE if elimination compartment (hasEliminationComp = TRUE) contains a fraction excreted parameter.
isSaturating	Set to TRUE to use Michaelis-Menten kinetics for elimination. Only applicable to models with paramteterization = "Clearance"
infusionAllowed	Set to TRUE if infusions allowed.
isDuration	Set to TRUE if infusions use duration instead of rate (must also set infusionAllowed = TRUE).
isSequential	Set to TRUE to freeze PK fixed effects and convert the corresponding random effects into covariates as well as remove the PK observed variable from the model.
isPkFrozen	Set to TRUE to freeze PK fixed effects and remove the corresponding random effects as well as the PK observed variable from the model.

`hasEffectsCompartment`
: Set to `TRUE` to include an effect compartment into the model.

`indirectType`
: Type of drug actions for the indirect response model. Options are `"LimitedStimulation"`, `"InfiniteStimulation"`, `"LimitedInhibition"`, `"InverseInhibition"`, `"LinearStimulation"`, or `"LogLinearStimulation"`.

`isBuildup`
: Set to `FALSE` to have the drug actions affect the loss/degradation instead of the production.

`isExponent`
: Set to `TRUE` to add an exponent parameter to the drug action term.

`indirectFrozen`
: Set to `TRUE` to freeze PD fixed effects and remove the corresponding random effects as well as the PD observed variable from the model.

`data`
: Input dataset

`columnMap`
: If `TRUE` (default) column mapping arguments are required. Set to `FALSE` to manually map columns after defining model using `colMapping`.

`modelName`
: Model name for subdirectory created for model output in current working directory.

`workingDir`
: Working directory to run the model. Current working directory will be used if `workingDir` not specified.

`...`
: Arguments passed on to `pkindirectmodel_MappingParameters`

  `ID` Column mapping argument for input dataset column(s) that identify individual data profiles. Only applicable to population models `isPopulation = TRUE`.

  `Time` Column mapping argument that represents the input dataset column for the relative time used in a study and only applicable to time-based models.

  `A1` Column mapping argument that represents the input dataset column for the amount of drug administered. Only applicable to the following types of models:

  - Models with `absorption = "Intravenous"` and parameterization set to either `"Clearance"`,`"Micro"`, or `"Macro"`
  - Models with `absorption` set to either `"Gamma"`, `"InverseGaussian"`, or `"Weibull"`

  `Aa` Column mapping argument that represents the input dataset column for the amount of drug administered and only applicable to models with `absorption = "FirstOrder"`.

  `A` Column mapping argument that represents the input dataset column for the amount of drug administered and only applicable to models with `absorption = "Intravenous"` and `parameterization = "Macro1"`.

  `A1_Rate` Column mapping argument that represents the input dataset column for the rate of drug administered. Only applicable to the following types of models:

  - Models with `absorption = "Intravenous"`, `infusionAllowed = TRUE` and parameterization set to either `"Clearance"`,`"Micro"` or `"Macro"`
  - Models with `absorption` set to either `"Gamma"`, `"InverseGaussian"`, or `"Weibull"` and `infusionAllowed = TRUE`

A1_Duration Column mapping argument that represents the input dataset column for the duration of drug administered. Only applicable to the following types of models:

- Models with absorption = "Intravenous", infusionAllowed = TRUE with isDuration = TRUE and parameterization set to either "Clearance","Micro" or "Macro"
- Models with absorption set to either "Gamma", "InverseGaussian", or "Weibull" and infusionAllowed = TRUE with isDuration = TRUE

Aa_Rate Column mapping argument that represents the input dataset column for the rate of drug administered and only applicable to models with absorption = "FirstOrder", infusionAllowed = TRUE.

Aa_Duration Column mapping argument that represents the input dataset column for the duration of drug administered and only applicable to models with absorption = "FirstOrder", infusionAllowed = TRUE, and isDuration = TRUE.

A_Rate Column mapping argument that represents the input dataset column for the rate of drug administered and only applicable to models with absorption = "Intravenous", infusionAllowed = TRUE, and parameterization = "Macro1".

A_Duration Column mapping argument that represents the input dataset column for the duration of drug administered and only applicable to models with absorption = "Intravenous", infusionAllowed = TRUE, isDuration = TRUE, and parameterization = "Macro1".

A1Strip Column mapping argument that represents the input dataset column for the stripping dose and only applicable to models with parameterization = "Macro".

CObs Column mapping argument that represents the input dataset column for the observations of drug concentration in the central compartment and only applicable to models with parameterization being either set to either "Clearance" or "Micro".

C1Obs Column mapping argument that represents the input dataset column for the observations of drug concentration in the central compartment and only applicable to models with parameterization being either set to either "Macro" or "Macro1".

A0Obs Column mapping argument that represents the input dataset column for the observed amount of drug in the elimination compartment. (hasEliminationComp = TRUE).

EObs Column mapping argument that represents the input dataset column for the observed drug effect.

nV If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nV.

nV2 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nV2.

nV3 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nV3.

nCl If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nCl.

nCl2 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nCl2.

nCl3 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nCl3.

nKa If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nKa.

nA If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nA.

nAlpha If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nAlpha.

nB If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nB.

nBeta If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nBeta.

nC If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nC.

nGamma If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nGamma.

nKe If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nKe.

nK12 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nK12.

nK21 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nK21.

nK13 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nK13.

nK31 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nK31.

nTlag If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nTlag.

nKm If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nKm.

nVmax If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nVmax.

nFe If isSequential = TRUE and isFractionExcreted = TRUE, mapped to the input dataset column that lists the values for random effect nFe.

nMeanDelayTime If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nMeanDelayTime.

nShapeParam If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nShapeParam.

nShapeParamMinusOne If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nShapeParamMinusOne.

Value

NlmePmlModel object

Column mapping

Note that quoted and unquoted column names are supported. Please see colMapping.

Examples

```
model <- pkindirectmodel(
  parameterization = "Micro",
  data = pkpdData,
  ID = "ID",
  Time = "Time",
  A1 = "Dose",
  CObs = "CObs",
  EObs = "EObs"
)

# View PML Code
print(model)
```

---

pklinearmodel *Create PK linear model*

Description

Use to create a PK/PD model with PD described by either constant, linear, or quadratic model

Usage

```
pklinearmodel(
  isPopulation = TRUE,
  parameterization = "Clearance",
  absorption = "Intravenous",
  numCompartments = 1,
  isClosedForm = TRUE,
  isTlag = FALSE,
  hasEliminationComp = FALSE,
  isFractionExcreted = FALSE,
  isSaturating = FALSE,
  infusionAllowed = FALSE,
  isDuration = FALSE,
  isSequential = FALSE,
  isPkFrozen = FALSE,
  hasEffectsCompartment = FALSE,
  linearType = "Constant",
  isLinearFrozen = FALSE,
  data = NULL,
  columnMap = TRUE,
  modelName = "",
```

```
  workingDir = "",
  ...
)
```

Arguments

isPopulation	Is this a population model TRUE or individual model FALSE?
parameterization	Type of parameterization. Options are "Clearance", "Micro", "Macro", or "Macro1".
absorption	Type of absorption. Options are "Intravenous", "FirstOrder", "Gamma", "InverseGaussian", "Weibull" .
numCompartments	Value of either 1, 2, or 3.
isClosedForm	Set to TRUE to convert model from a differential equation to close form.
isTlag	Set to TRUE to add a lag time parameter to the model.
hasEliminationComp	Set to TRUE to add an elimination compartment to the model.
isFractionExcreted	Set to TRUE if elimination compartment (hasEliminationComp = TRUE) contains a fraction excreted parameter.
isSaturating	Set to TRUE to use Michaelis-Menten kinetics for elimination. Only applicable to models with paramteterization = "Clearance"
infusionAllowed	Set to TRUE if infusions allowed.
isDuration	Set to TRUE if infusions use duration instead of rate (must also set infusionAllowed = TRUE).
isSequential	Set to TRUE to freeze PK fixed effects and convert the corresponding random effects into covariates as well as remove the PK observed variable from the model.
isPkFrozen	Set to TRUE to freeze PK fixed effects and remove the corresponding random effects as well as the PK observed variable from the model.
hasEffectsCompartment	Set to TRUE to include an effect compartment into the model.
linearType	Type of PD model; Options are "Constant", "Linear", "Quadratic".
isLinearFrozen	Set to TRUE to freeze PD fixed effects and remove the corresponding random effects as well as the PD observed variable from the model.
data	Input dataset
columnMap	If TRUE (default) column mapping arguments are required. Set to FALSE to manually map columns after defining model using colMapping.
modelName	Model name for subdirectory created for model output in current working directory.
workingDir	Working directory to run the model. Current working directory will be used if workingDir not specified.

... Arguments passed on to pkindirectmodel_MappingParameters

ID Column mapping argument for input dataset column(s) that identify individual data profiles. Only applicable to population models isPopulation = TRUE.

Time Column mapping argument that represents the input dataset column for the relative time used in a study and only applicable to time-based models.

A1 Column mapping argument that represents the input dataset column for the amount of drug administered. Only applicable to the following types of models:

- Models with absorption = "Intravenous" and parameterization set to either "Clearance","Micro", or "Macro"
- Models with absorption set to either "Gamma", "InverseGaussian", or "Weibull"

Aa Column mapping argument that represents the input dataset column for the amount of drug administered and only applicable to models with absorption = "FirstOrder".

A Column mapping argument that represents the input dataset column for the amount of drug administered and only applicable to models with absorption = "Intravenous" and parameterization = "Macro1".

A1_Rate Column mapping argument that represents the input dataset column for the rate of drug administered. Only applicable to the following types of models:

- Models with absorption = "Intravenous", infusionAllowed = TRUE and parameterization set to either "Clearance","Micro" or "Macro"
- Models with absorption set to either "Gamma", "InverseGaussian", or "Weibull" and infusionAllowed = TRUE

A1_Duration Column mapping argument that represents the input dataset column for the duration of drug administered. Only applicable to the following types of models:

- Models with absorption = "Intravenous", infusionAllowed = TRUE with isDuration = TRUE and parameterization set to either "Clearance","Micro" or "Macro"
- Models with absorption set to either "Gamma", "InverseGaussian", or "Weibull" and infusionAllowed = TRUE with isDuration = TRUE

Aa_Rate Column mapping argument that represents the input dataset column for the rate of drug administered and only applicable to models with absorption = "FirstOrder", infusionAllowed = TRUE.

Aa_Duration Column mapping argument that represents the input dataset column for the duration of drug administered and only applicable to models with absorption = "FirstOrder", infusionAllowed = TRUE, and isDuration = TRUE.

A_Rate Column mapping argument that represents the input dataset column for the rate of drug administered and only applicable to models with absorption = "Intravenous", infusionAllowed = TRUE, and parameterization = "Macro1".

A_Duration Column mapping argument that represents the input dataset column for the duration of drug administered and only applicable to models

with absorption = "Intravenous", infusionAllowed = TRUE, isDuration = TRUE, and parameterization = "Macro1".

A1Strip Column mapping argument that represents the input dataset column for the stripping dose and only applicable to models with parameterization = "Macro".

CObs Column mapping argument that represents the input dataset column for the observations of drug concentration in the central compartment and only applicable to models with parameterization being either set to either "Clearance" or "Micro".

C1Obs Column mapping argument that represents the input dataset column for the observations of drug concentration in the central compartment and only applicable to models with parameterization being either set to either "Macro" or "Macro1".

A0Obs Column mapping argument that represents the input dataset column for the observed amount of drug in the elimination compartment. (hasEliminationComp = TRUE).

EObs Column mapping argument that represents the input dataset column for the observed drug effect.

nV If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nV.

nV2 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nV2.

nV3 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nV3.

nCl If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nCl.

nCl2 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nCl2.

nCl3 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nCl3.

nKa If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nKa.

nA If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nA.

nAlpha If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nAlpha.

nB If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nB.

nBeta If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nBeta.

nC If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nC.

nGamma If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nGamma.

nKe If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nKe.

nK12 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nK12.

nK21 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nK21.

nK13 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nK13.

nK31 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nK31.

nTlag If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nTlag.

nKm If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nKm.

nVmax If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nVmax.

nFe If isSequential = TRUE and isFractionExcreted = TRUE, mapped to the input dataset column that lists the values for random effect nFe.

nMeanDelayTime If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nMeanDelayTime.

nShapeParam If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nShapeParam.

nShapeParamMinusOne If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nShapeParamMinusOne.

Value

NlmePmlModel object

Column mapping

Note that quoted and unquoted column names are supported. Please see colMapping.

Examples

```
model <- pklinearmodel(
  parameterization = "Clearance",
  linearType = "Constant",
  data = pkpdData,
  ID = "ID",
  Time = "Time",
  A1 = "Dose",
  CObs = "CObs",
  EObs = "EObs"
)

# View the model as well as its associated column mappings
print(model)
```

pkmodel *Creates a PK model*

Description

Use to create a PK model

Usage

```
pkmodel(
  isPopulation = TRUE,
  parameterization = "Clearance",
  absorption = "Intravenous",
  numCompartments = 1,
  isClosedForm = TRUE,
  isTlag = FALSE,
  hasEliminationComp = FALSE,
  isFractionExcreted = FALSE,
  isSaturating = FALSE,
  infusionAllowed = FALSE,
  isDuration = FALSE,
  isStdevFrozen = FALSE,
  data = NULL,
  columnMap = TRUE,
  modelName = "",
  workingDir = "",
  ...
)
```

Arguments

isPopulation	Is this a population model TRUE or individual model FALSE?
parameterization	Type of parameterization. Options are "Clearance", "Micro", "Macro", or "Macro1".
absorption	Type of absorption. Options are "Intravenous", "FirstOrder", "Gamma", "InverseGaussian", "Weibull" .
numCompartments	Value of either 1, 2, or 3.
isClosedForm	Set to TRUE to convert model from a differential equation to close form.
isTlag	Set to TRUE to add a lag time parameter to the model.
hasEliminationComp	Set to TRUE to add an elimination compartment to the model.
isFractionExcreted	Set to TRUE if elimination compartment (hasEliminationComp = TRUE) contains a fraction excreted parameter.

isSaturating Set to TRUE to use Michaelis-Menten kinetics for elimination. Only applicable to models with paramteterization = "Clearance"

infusionAllowed
Set to TRUE if infusions allowed.

isDuration Set to TRUE if infusions use duration instead of rate (must also set infusionAllowed = TRUE).

isStdevFrozen Set to TRUE to freeze value of standard deviation of residual error variable.

data Input dataset

columnMap If TRUE (default) column mapping arguments are required. Set to FALSE to manually map columns after defining model using colMapping.

modelName Model name for subdirectory created for model output in current working directory.

workingDir Working directory to run the model. Current working directory will be used if workingDir not specified.

... Arguments passed on to pkmodel_MappingParameters

ID Column mapping argument for input dataset column(s) that identify individual data profiles. Only applicable to population models isPopulation = TRUE.

Time Column mapping argument that represents the input dataset column for the relative time used in a study and only applicable to time-based models.

A1 Column mapping argument that represents the input dataset column for the amount of drug administered. Only applicable to the following types of models:

- Models with absorption = "Intravenous" and parameterization set to either "Clearance","Micro", or "Macro"
- Models with absorption set to either "Gamma", "InverseGaussian", or "Weibull"

Aa Column mapping argument that represents the input dataset column for the amount of drug administered and only applicable to models with absorption = "FirstOrder".

A Column mapping argument that represents the input dataset column for the amount of drug administered and only applicable to models with absorption = "Intravenous" and parameterization = "Macro1".

A1_Rate Column mapping argument that represents the input dataset column for the rate of drug administered. Only applicable to the following types of models:

- Models with absorption = "Intravenous", infusionAllowed = TRUE and parameterization set to either "Clearance","Micro" or "Macro"
- Models with absorption set to either "Gamma", "InverseGaussian", or "Weibull" and infusionAllowed = TRUE

A1_Duration Column mapping argument that represents the input dataset column for the duration of drug administered. Only applicable to the following types of models:

- Models with absorption = "Intravenous",infusionAllowed = TRUE with isDuration = TRUE and parameterization set to either "Clearance", "Micro" or "Macro"
- Models with absorption set to either "Gamma", "InverseGaussian", or "Weibull" and infusionAllowed = TRUE with isDuration = TRUE

Aa_Rate Column mapping argument that represents the input dataset column for the rate of drug administered and only applicable to models with absorption = "FirstOrder", infusionAllowed = TRUE.

Aa_Duration Column mapping argument that represents the input dataset column for the duration of drug administered and only applicable to models with absorption = "FirstOrder", infusionAllowed = TRUE, and isDuration = TRUE.

A_Rate Column mapping argument that represents the input dataset column for the rate of drug administered and only applicable to models with absorption = "Intravenous", infusionAllowed = TRUE, and parameterization = "Macro1".

A_Duration Column mapping argument that represents the input dataset column for the duration of drug administered and only applicable to models with absorption = "Intravenous", infusionAllowed = TRUE, isDuration = TRUE, and parameterization = "Macro1".

A1Strip Column mapping argument that represents the input dataset column for the stripping dose and only applicable to models with parameterization = "Macro".

CObs Column mapping argument that represents the input dataset column for the observations of drug concentration in the central compartment and only applicable to models with parameterization being either set to either "Clearance" or "Micro".

C1Obs Column mapping argument that represents the input dataset column for the observations of drug concentration in the central compartment and only applicable to models with parameterization being either set to either "Macro" or "Macro1".

A0Obs Column mapping argument that represents the input dataset column for the observed amount of drug in the elimination compartment. (hasEliminationComp = TRUE).

Value

NlmePmlModel object

Column mapping

Note that quoted and unquoted column names are supported. Please see colMapping.

Examples

```
model <- pkmodel(
  parameterization = "Clearance",
  numCompartments = 2,
  data = pkData,
```

```
  ID = "Subject",
  Time = "Act_Time",
  A1 = "Amount",
  CObs = "Conc"
)

# View the model as well as its associated column mappings
print(model)
```

`pkpdData` *Pharmacokinetic/Pharmacodynamic dataset containing 200 subjects with single bolus dose*

Description

Pharmacokinetic/Pharmacodynamic dataset containing 200 subjects with single bolus dose.

Usage

```
pkpdData
```

Format

A data frame with 2600 rows and 5 variables:

ID Subject ID

Time Nominal Time

Dose Amount of dose

CObs Observations of drug concentration in blood

EObs Observations of drug effect

Source

The data is simulated using a PKPD model with PK described by a one-compartment model with IV bolus and PD described by an indirect response model with the loss inhibited.

print.NlmePmlModel *Print generic for class NlmePmlModel*

Description

Prints model information, including PML and column mappings.

Usage

```
## S3 method for class 'NlmePmlModel'
print(x, ...)
```

Arguments

x	NlmePmlModel class instance
...	Arguments passed to methods.

Value

NULL

Examples

```
model <- pkmodel(columnMap = FALSE, data = pkData)
print(model)
```

randomEffect *Sets or updates the covariance matrix of random effects*

Description

Use to set or update the covariance matrix of random effects in a model object.

Usage

```
randomEffect(
  .Object,
  effect,
  value = NULL,
  isDiagonal = TRUE,
  isFrozen = FALSE,
  ...
)
```

Arguments

.Object	Model object
effect	One or more names of available random effects.
value	Initial values for the diagonal elements of the covariance matrix of random effects (if isDiagonal = TRUE, or initial values for the lower triangular elements (including diagonal elements) of the covariance matrix (if isDiagonal = FALSE) in a row-wise order.
isDiagonal	Set to TRUE to if the covariance matrix of the specified random effects is a diagonal matrix. or FALSE if not.
isFrozen	Set to TRUE to freeze the covariance matrix of random effects.
...	Additional arguments

Value

Modified NlmePmlModel object

Examples

```
model <- pkmodel(
  numCompartments = 2,
  data = pkData,
  ID = "Subject",
  Time = "Act_Time",
  A1 = "Amount",
  CObs = "Conc",
  modelName = "TwCpt_IVBolus_FOCE_ELS"
  )

model <- model |>
  randomEffect(effect = c("nV", "nCl", "nCl2"), value = rep(0.1, 3))
```

---

removeCovariate *Remove covariate from structural parameters in a model object.*

Description

Remove one or more covariates from structural parameters in a model object.

Usage

```
removeCovariate(.Object, covariate = NULL, paramName = NULL)
```

Arguments

.Object	Model object
covariate	Covariates to remove from model. If NULL all covariates will be removed from model.
paramName	Structural parameters for which to remove covariate effect(s) from. If NULL covariate effect will be removed from all structural parameters.

Value

Modified NlmePmlModel object

Examples

```
model <- pkmodel(
  numCompartments = 2,
  data = pkData,
  ID = "Subject",
  Time = "Act_Time",
  A1 = "Amount",
  CObs = "Conc"
)

# Add Gender covariate of type categorical
model <- addCovariate(model,
  covariate = "Gender",
  type = "Categorical",
  effect = c("V2", "Cl2"),
  levels = c(0, 1),
  labels = c("Female", "Male")
)

# Add BodyWeight covariate of type continuous
model <- addCovariate(model,
  covariate = "BodyWeight",
  type = "Continuous",
  direction = "Backward",
  center = "Mean",
  effect = c("V", "Cl")
)

# Remove all covariates from model
model <- removeCovariate(model)
```

---

remove_NLMELicense *Remove NLME License*

Description

This function attempts to remove an NLME license using the specified installation directory and licensing tool.

Usage

```
remove_NLMELicense(InstallDir = Sys.getenv("INSTALLDIR"))
```

Arguments

InstallDir	A character string specifying the directory where the NLME Engine is installed e.g., INSTALLDIR environment variable. The cadlicensingtool executable is expected to be located within this directory, or within a subdirectory specified by the PML_BIN_DIR environment variable.

Details

The function checks for the presence of the necessary 'appsettings.json' file in the specified directory or the CAD config file specified by the 'CAD_CONFIG_FILE' environment variable, runs the licensing tool to log out the user, and attempts to remove the NLME license.

Value

A logical value indicating whether the license information was successfully removed.

Examples

```
result <- remove_NLMELicense("/path/to/install/dir")
if (result) {
  message("License removed successfully!")
} else {
  message("Failed to remove license.")
}
```

---

residualEffectNames *Return residual effect terms available in model*

Description

Use to return character vector of residual effect names in model object

Usage

```
residualEffectNames(model)
```

Arguments

model	Object of class NlmePmlModel

Value

Character vector of residual effect names

Examples

```
model <- pkemaxmodel(columnMap = FALSE)
residualEffectNames(model)
```

`residualError` *Assign residual error model to model object*

Description

Use to change or update residual error model for model object

Usage

```
residualError(
  .Object,
  predName = "C",
  errorType = NULL,
  SD = NULL,
  isFrozen = FALSE,
  isBQL = FALSE,
  staticLLOQ = NULL,
  EObsBQL = NULL,
  CObsBQL = NULL,
  C1ObsBQL = NULL,
  A0ObsBQL = NULL,
  exponent = NULL
)
```

Arguments

`.Object`	Model object
`predName`	Name of the predicted variable as returned in `residualEffectNames`.
`errorType`	Options are `"Additive"`, `"LogAdditive"`, `"Multiplicative"`, `"AdditiveMultiplicative"`, `"MixRatio"`, `"Power"`.
`SD`	Value for the standard deviation of the residual error variable.
`isFrozen`	Set to `TRUE` to freeze the standard deviation to the value specified for `SD`.
`isBQL`	Set to `TRUE` if BQL values present in the observation data.
`staticLLOQ`	Optional LLOQ value if `isBQL = TRUE`
`EObsBQL`	Column mapping argument that represents the input dataset column that contains the BQL flag for observation values corresponding to `EObs`. Only applicable to `isBQL = TRUE`.

CObsBQL	Column mapping argument that represents the input dataset column that contains the BQL flag for observation values corresponding to CObs. Only applicable to isBQL = TRUE.
C1ObsBQL	Column mapping argument that represents the input dataset column that contains the BQL flag for observation values corresponding to C1Obs. Only applicable to isBQL = TRUE.
A0ObsBQL	Column mapping argument that represents the input dataset column that contains the BQL flag for observation values corresponding to AObs. Only applicable to isBQL = TRUE.
exponent	Value of exponent. Only applicable to errorType = "Power".

Value

Modified NlmePmlModel object

Examples

```
model <- pkindirectmodel(indirectType = "LimitedInhibition", isBuildup = FALSE,
 data = pkpdData, ID = "ID", Time = "Time", A1 = "Dose", CObs = "CObs", EObs = "EObs")

residualEffectNames(model)

# Change error type to "Multiplicative" and value of SD to 0.1 for "E"
model <- residualError(model, predName = "E", errorType = "Multiplicative", SD = 0.1)

# Change error type to "Power", value of SD to 0.15, and set exponent = 2 for "C"
model <- residualError(model, predName = "C", errorType = "Power", SD = 0.15, exponent = 2)
```

---

secondaryParameterNames

Get secondary parameter names

Description

Returns character vector of secondary parameter names for model object.

Usage

```
secondaryParameterNames(model)
```

Arguments

model	Object of class NlmePmlModel

Value

Character vector of secondary parameter names defined in model

Examples

```
model <- pkemaxmodel(columnMap = FALSE)
secondaryparms <- secondaryParameterNames(model)
```

---

shotgunSearch *Executes an NLME shotgun covariate search*

Description

Executes an NLME shotgun covariate search

Usage

```
shotgunSearch(
  model,
  hostPlatform = NULL,
  params,
  covariateModel,
  runInBackground = FALSE,
  ...
)
```

Arguments

model	PK/PD model class object.
hostPlatform	Host definition for model execution. See hostParams. If missing, multicore local host with 4 threads is used.
params	Engine parameters. See engineParams. If missing, default parameters generated by engineParams(model) are used.
covariateModel	Covariate Effects Model providing the relationship between covariates and structural parameters to test (covariateModel(model)).
runInBackground	Set to TRUE to run in background and return prompt.
...	Additional arguments for hostParams or arguments available inside engineParams functions. If engineParams arguments are supplied through both params argument and additional argument (i.e., ellipsis), then the arguments in params will be ignored and only the additional arguments will be used with warning. If hostParams arguments are supplied through both the hostPlatform argument and the ellipses, values supplied to hostPlatform will be overridden by additional arguments supplied via the ellipses e.g., ...

Value

if runInBackground = FALSE, a data frame is returned with shotgun (all combinations given the covariate model) search results, i.e. "Overall" comma separated file. Otherwise the ShotgunNlmeJob class object is returned.

See Also

hostParams, engineParams

Examples

```
# Define the model
model <- pkmodel(numCompartments = 2,
                 data = pkData,
                 ID = "Subject",
                 Time = "Act_Time",
                 A1 = "Amount",
                 CObs = "Conc")

# Add Gender covariate of type categorical
model <- addCovariate(model,
                      covariate = "Gender",
                      type = "Categorical",
                      effect = c("V2", "Cl2"),
                      levels = c(0, 1),
                      labels = c("Female", "Male"))

# Add Bodyweight covariate of type continuous
model <- addCovariate(model,
             covariate = "BodyWeight",
             type = "Continuous",
             direction = "Backward",
             center = "Mean",
             effect = c("V", "Cl"))

# Define the host
host <- hostParams(parallelMethod = "None",
                   hostName = "local",
                   numCores = 1)

# Define the engine parameters
params <- engineParams(model)

# Define covariate model
cp <- covariateModel(model)

# Perform shotgun search
OverallDF <-  shotgunSearch(model = model,
                            hostPlatform = host,
                            params = params,
                            covariateModel = cp,
                            runInBackground = FALSE)
```

simmodel *Executes an NLME simulation*

Description

Executes an NLME simulation

Usage

```
simmodel(
  model,
  simParams,
  params,
  hostPlatform = NULL,
  runInBackground = FALSE,
  ...
)
```

Arguments

model	PK/PD model class object.
simParams	Simulation parameters. See NlmeSimulationParams. If missing, default parameters generated by NlmeSimulationParams() are used.
params	Engine parameters. See engineParams. The common parameters include: sort, ODE, rtolODE, atolODE, maxStepsODE. If missing, default parameters generated by engineParams(model) are used.
hostPlatform	Host definition for model execution. See hostParams. If missing, simple local host is used.
runInBackground	Set to TRUE to run in background and return prompt.
...	Additional class initializer arguments for NlmeSimulationParams, or arguments available inside hostParams or engineParams functions. If engineParams arguments are supplied through both params argument and additional argument (i.e., ellipsis), then the arguments in params will be ignored and only the additional arguments will be used with warning. If hostParams arguments are supplied through both hostPlatform argument and additional argument, then its slots will be overridden by additional arguments. In addition, if NlmeSimulationParams arguments are supplied through both simParams argument and additional argument, then its slots will be overridden by additional arguments.

Value

returns job properties if runInBackground is TRUE; if runInBackground is FALSE and the function is called in interactive mode, the resulted simulated tables wil be loaded and presented as a list; if runInBackground is FALSE and the function is called in non-interactive mode, the list returned will have just the full paths of the tables generated.

Examples

```
SimTableObs <- tableParams(
  name = "SimTableObs.csv",
  timesList = "0,1,2,4,4.9,55.1,56,57,59,60",
  variablesList = "C, CObs",
  timeAfterDose = FALSE,
  forSimulation = TRUE
)

simParams <- NlmeSimulationParams(
  numReplicates = 2,
  simulationTables = SimTableObs
)
# Define the model
model <- pkmodel(
  numComp = 2,
  absorption = "Extravascular",
  ID = "Subject",
  Time = "Act_Time",
  CObs = "Conc",
  Aa = "Amount",
  data = pkData,
  modelName = "PkModel"
)
results <- simmodel(model, simParams)
# with seed given additionally:
results <- simmodel(model, simParams, seed = 3527)
```

---

sortfit *Executes an NLME simple estimation with sort keys and given scenarios*

Description

Executes an NLME simple estimation with sort keys and given scenarios

Usage

```
sortfit(
  model,
  hostPlatform = NULL,
  params,
  sortColumns,
  scenarios = list(),
  simpleTables,
  runInBackground = FALSE,
  filesToReturn = "*",
  ...
)
```

Arguments

model	PK/PD model class object.
hostPlatform	Host definition for model execution. See hostParams. If missing, PhoenixMPIDir64 is given and MPI is installed, MPI local host with 4 threads is used. If MPI is not found, local host without parallelization is used.
params	Engine parameters. See engineParams. If missing, default parameters generated by engineParams(model) are used.
sortColumns	List of sort columns. See SortColumns. If missing, empty sort columns argument is used and NLME dataset is used as is.
scenarios	List of scenarios with different sets of covariates. See NlmeScenario If missing, all covariates effects are considered as enabled.
simpleTables	Optional list of simple tables. See tableParams. By default a table named 'posthoc.csv' is returned with structural parameters values for all source data rows.
runInBackground	Set to TRUE to run in background and return prompt.
filesToReturn	Used to specify which files to be outputted to the model directory and loaded as returned value. By default, all the applicable files listed in the Value section will be outputted to the model directory and loaded as returned value. Only those files listed in the Value section can be specified. Simple regex patterns are supported for the specification.
...	Additional arguments for hostParams or arguments available inside engineParams functions. If engineParams arguments are supplied through both params argument and additional argument (i.e., ellipsis), then the arguments in params will be ignored and only the additional arguments will be used with warning. If hostParams arguments are supplied through both the hostPlatform argument and the ellipses, values supplied to hostPlatform will be overridden by additional arguments supplied via the ellipses e.g., ...

Details

All the results in tabular format have scenario column and sorts columns appended. The resulted logs (nlme7engine.log, err1.txt, dmp.txt, out.txt) are appended with a row delimiter where the name of the Scenario and sort values are specified.

Value

if runInBackground is FALSE, a list with main resulted dataframes is returned:

- Overall
- ConvergenceData
- residuals
- Secondary
- StrCovariate - if continuous covariates presented
- StrCovariateCat - if categorical covariates presented

- theta
- posthoc table
- posthocStacked table
- Requested tables

`nlme7engine.log` textual output is returned and loaded with the main information related to fitting. `dmp.txt` structure with the results of fitting (including LL by subject information) is returned and loaded. These 2 files are returned and loaded irrespective of `filesToReturn` argument value.

For individual models, additional dataframe with partial derivatives is returned:

- ParDer

For population models and the method specified is NOT `Naive-Pooled`, additional dataframes are returned:

- omega
- Eta
- EtaStacked
- EtaEta
- EtaCov
- EtaCovariate - if continuous covariates presented
- EtaCovariateCat - if categorical covariates presented
- bluptable.dat

If standard error computation was requested and it was successful, additional dataframes are returned:

- thetaCorrelation
- thetaCovariance
- Covariance
- omega_stderr

If nonparametric method was requested (`numIterNonParametric` > 0) and the `method` specified in `engineParams` is NOT `Naive-Pooled`, additional dataframes are returned:

- nonParSupportResult
- nonParStackedResult
- nonParEtaResult
- nonParOverallResult

if `runInBackground` is `TRUE`, only current status of job is returned.

Non-loaded but returned files

The non-loaded but returned files in the model working directory are:

- err1.txt - concatenated for all runs detailed logs for all steps of optimization,
- out.txt - general pivoted information about results,
- doses.csv - information about doses given for all subjects,
- iniest.csv - information about initial estimates

See Also

hostParams, engineParams, SortColumns,NlmeScenario, tableParams

Examples

```
input_data <- pkData

model <-
  pkmodel(numCompartments = 2,
          data = input_data,
          ID = "Subject",
          Time = "Act_Time",
          A1 = "Amount",
          CObs = "Conc")

model <-
  addCovariate(model,
               covariate = "BodyWeight",
               direction = "Backward",
               center = "Mean",
               effect = c("V", "Cl"))

# multicore
multicoreHost <-
   hostParams(parallelMethod = "Multicore",
              hostName = "multicore",
              numCores = 4)

# specify scenarios
CovariateEffectNames <- listCovariateEffectNames(model)
combinations <-
  combn(c("", CovariateEffectNames),
        length(CovariateEffectNames),
        simplify = FALSE)

scenarioNames <-
  lapply(combinations,
         function(x) {paste(x, collapse = " ")})

scenarios <-
  lapply(scenarioNames,
         function(x, CovariateEffectNames) {
           CovariateCombinations <- unlist(strsplit(x, " ", fixed = TRUE))
           scenarioIndex <-
             paste(which(CovariateEffectNames %in% CovariateCombinations,
                         arr.ind = TRUE),
                         collapse = ", ")
           NlmeScenario(trimws(x), scenarioIndex)
         },
         CovariateEffectNames)

res <-
```

```
sortfit(model,
        hostPlatform = multicoreHost,
        params = engineParams(model),
        sortColumns = SortColumns("Gender"),
        scenarios = scenarios)
```

`stepwiseSearch` *Executes an NLME stepwise covariate search*

Description

Executes an NLME stepwise covariate search

Usage

```
stepwiseSearch(
  model,
  hostPlatform = NULL,
  params,
  covariateModel,
  stepwiseParams,
  runInBackground = FALSE,
  ...
)
```

Arguments

`model`	PK/PD model class object.
`hostPlatform`	Host definition for model execution. See `hostParams`. If `missing`, multicore local host with 4 threads is used.
`params`	Engine parameters. See `engineParams`. If `missing`, default parameters generated by engineParams(model) are used.
`covariateModel`	Covariate Effects Model providing the relationship between covariates and structural parameters to test (`covariateModel(model)`).
`stepwiseParams`	Stepwise parameters defining decision tree. See `StepwiseParams`
`runInBackground`	Set to `TRUE` to run in background and return prompt.
`...`	Additional arguments for `hostParams` or arguments available inside `engineParams` functions. If `engineParams` arguments are supplied through both `params` argument and additional argument (i.e., ellipsis), then the arguments in `params` will be ignored and only the additional arguments will be used with warning. If `hostParams` arguments are supplied through both the `hostPlatform` argument and the ellipses, values supplied to `hostPlatform` will be overridden by additional arguments supplied via the ellipses e.g., `....`

Value

if runInBackground = FALSE, a data frame is returned with stepwise search results, i.e. "Overall" comma separated file. Otherwise the StepwiseNlmeJob class object is returned.

See Also

hostParams, engineParams

Examples

```
# Define the model
model <- pkmodel(numCompartments = 2,
                 data = pkData,
                 ID = "Subject",
                 Time = "Act_Time",
                 A1 = "Amount",
                 CObs = "Conc")

# Add Gender covariate of type categorical
model <- addCovariate(model,
                      covariate = "Gender",
                      type = "Categorical",
                      effect = c("V2", "Cl2"),
                      levels = c(0, 1),
                      labels = c("Female", "Male"))

# Add Bodyweight covariate of type continuous
model <- addCovariate(model,
             covariate = "BodyWeight",
             type = "Continuous",
             direction = "Backward",
             center = "Mean",
             effect = c("V", "Cl"))

# Define the host
defaultHost <- hostParams(parallelMethod = "None",
                   hostName = "local",
                   numCores = 1)

# Define the engine parameters
params <- engineParams(model)

# Define covariate model
cp <- covariateModel(model)

# Define the stepwise parameters
sp <- StepwiseParams(0.01, 0.001, "-2LL")

# Perform stepwise search
OverallDF <-  stepwiseSearch(model = model,
                      hostPlatform = defaultHost,
                      params = params,
```

```
                    covariateModel = cp,
                    stepwiseParams = sp,
                    runInBackground = FALSE)
```

`structuralParameter` *Set structural parameter in model object*

Description

Use to specify the relationship of the structural parameter with corresponding fixed effect, random effect, and covariate.

Usage

```
structuralParameter(
  .Object,
  paramName,
  fixedEffName = NULL,
  randomEffName = NULL,
  style = "LogNormal",
  hasRandomEffect = NULL
)
```

Arguments

`.Object`	Model object
`paramName`	Name of the structural parameter
`fixedEffName`	Name of the corresponding fixed effect
`randomEffName`	Name of the corresponding random effect; only applicable to population models.
`style`	Use to specify the relationship of the structural parameter with its corresponding fixed effect, random effect, and covariate, if exists. • `"LogNormal"` (Default): The structural parameter is defined as `Product * exp(Eta)` • `"LogNormal1"`: The structural parameter is defined as `Sum * exp(Eta)` • `"LogNormal2"`: The structural parameter is defined as `exp(Sum + Eta)` • `"LogitNormal"`: The structural parameter is defined as `ilogit(Sum + Eta)` • `"Normal"`: The structural parameter is defined as `Sum + Eta)` `Product` denotes the product of the corresponding fixed effect and covariate effect terms (if exists), `Eta` represents the corresponding random effect, and `Sum` denotes the sum of its corresponding fixed effect and covariate effect terms (if exists).

hasRandomEffect

Set to FALSE to remove the corresponding random effect from the model. Only applicable to population models. If NULL the system will automatically set hasRandomEffect = TRUE for population models, and hasRandomEffect = FALSE for individual models.

Value

Modified NlmePmlModel object

Examples

```
model <- pkindirectmodel(
  indirectType = "LimitedInhibition",
  isBuildup = FALSE,
  data = pkpdData,
  ID = "ID",
  Time = "Time",
  A1 = "Dose",
  CObs = "CObs",
  EObs = "EObs"
)

# Change style of structural parameter "Imax" to "LogitNormal"
# and rename fixed effect to "tvlogitImax"
model <- structuralParameter(model,
  paramName = "Imax",
  style = "LogitNormal", fixedEffName = "tvlogitImax"
)

# Remove random effect for structural parameter "IC50"
model <- structuralParameter(model,
  paramName = "IC50",
  hasRandomEffect = FALSE
)
```

---

structuralParameterNames

Get structural parameter names

Description

Returns character vector of structural parameter names for model object.

Usage

```
structuralParameterNames(model, omitEmpties = TRUE)
```

Arguments

model	Object of class NlmePmlModel
omitEmpties	Set to TRUE to omit empty names

Value

Character vector of structural parameter names defined in model

Examples

```
model <- pkemaxmodel(columnMap = FALSE)
stparms <- structuralParameterNames(model)
```

---

tableParams *Wrapper around NlmeTableDef/NlmeSimTableDef-classes initializers.*

Description

Wrapper around NlmeTableDef/NlmeSimTableDef-classes initializers.

Usage

```
tableParams(
  name = "",
  timesList = numeric(0),
  covrSet = "",
  whenDose = "",
  whenObs = "",
  variablesList = "",
  keepSource = FALSE,
  timeAfterDose = FALSE,
  IRES = FALSE,
  Weight = FALSE,
  IWRES = FALSE,
  mode = "all",
  forSimulation = FALSE
)
```

Arguments

name	Name of the generated simulation file.
timesList	Numeric; Time values for simulation. Applicable for time-based models only. Ignored when "keepSource=TRUE"
covrSet	Character; Vector of covariate names. Simulation point is added when the covariate value is set. See covariateNames

whenDose	Character; Vector of dosing compartment names. Simulation point is added when the dose value is set.
whenObs	Character; String of observed variables names. Simulation point is added when the observation value is set.
variablesList	Character; List of variables from the model for simulation.
keepSource	Logical; Set to TRUE to keep the number of rows appearing in the table the same as the number of rows in the input dataset.
timeAfterDose	Set to TRUE to output time after dose.
IRES	Logical; Set to TRUE to output individual residuals. Valid only if whenObs is specified.
Weight	Logical; Set to TRUE to output the weight of current observation. Valid only if whenObs is specified.
IWRES	Logical; Set to TRUE to output individual weighted residuals. Valid only if whenObs is specified.
mode	Character; The mode of output. Options are "all" (default), "unique", "first". Only applicable to non time-based models for the case where only covrSet is defined or the case where only covrSet and variablesList are defined. Option "all" (default): it outputs all the rows invoked by specified covariates. Option "unique": if the values in a row are the same as the ones in the previous row for the current subject, then the row is omitted; otherwise, it is printed out. Option "first": it outputs only the first row for each subject.
forSimulation	logical. Defining whether the table is for simulation purposes or for postprocessing after fit. Default is FALSE.

Value

NlmeTableDef object if forSimulation is FALSE, NlmeSimTableDef object otherwise.

Examples

```
Table1 <- tableParams(
  name = "Table1.csv",
  timesList = seq(0, 24, 2),
  whenObs = c("CObs"),
  variablesList = "C",
  IRES = TRUE,
  IWRES = TRUE,
  Weight = TRUE)

SimTable1 <- tableParams(
  name = "SimTable1.csv",
  variablesList = "CL, V",
  keepSource = TRUE,
  forSimulation = TRUE)
```

textualmodel *Create a textual model object*

Description

Use to create an empty model object and optionally supply location of .mdl file to initialize model with PML statements.

Usage

```
textualmodel(modelName = "", workingDir = "", data, mdl = NULL)
```

Arguments

modelName	Model name to create subdirectory for model output in current working directory.
workingDir	Working directory to run the model. Current working directory will be used if workingDir not specified.
data	Input dataset
mdl	File path specifying location of test.mdl file

Value

NlmePmlModel object

Examples

```
model <- textualmodel(data = pkData)
```

vpcmodel *Perform visual predictive check for NLME models*

Description

Perform visual predictive check for NLME models

Usage

```
vpcmodel(
  model,
  vpcParams,
  params,
  hostPlatform = NULL,
  runInBackground = FALSE,
  ...
)
```

Arguments

model	PK/PD model class object.
vpcParams	VPC argument setup. See NlmeVpcParams. If missing, default values generated by NlmeVpcParams() are used.
params	Engine argument setup. See engineParams. The following arguments are the subject of interest: sort, ODE, rtolODE, atolODE, maxStepsODE. If missing, default values generated by engineParams(model) are used.
hostPlatform	Host definition for model execution. See hostParams. If missing, simple local host is used.
runInBackground	Set to TRUE to run in background and return prompt.
...	Additional class initializer arguments for NlmeVpcParams or hostParams, or arguments available inside engineParams functions. If engineParams arguments are supplied through both params argument and additional argument (i.e., ellipsis), then the arguments in params will be ignored and only the additional arguments will be used with warning. If hostParams arguments are supplied through both hostPlatform argument and additional argument, then its values will be overridden by additional arguments. In addition, if NlmeVpcParams arguments are supplied through both vpcParams argument and additional argument, then its slots will be overridden by additional arguments.

Value

if runInBackground is TRUE, it returns job properties. Otherwise,

- If the function is called in an interactive mode, the resulting simulated tables and summary statistics tables will be loaded and presented as a list;
- If the function is called in a non-interactive mode, it returns the full paths of the tables generated

Examples

```
job <- fitmodel(model)

# View estimation results
print(job)

finalModelVPC <- copyModel(model, acceptAllEffects = TRUE, modelName = "model_VPC")

# View the model
print(finalModelVPC)

# Set up VPC arguments to have PRED outputted to simulation output dataset "predout.csv"
vpcSetup <- NlmeVpcParams(outputPRED = TRUE)

# Run VPC using the default host, default values for the relevant NLME engine arguments
finalVPCJob <- vpcmodel(model = finalModelVPC, vpcParams = vpcSetup)
# the same as:
```

```
finalVPCJob <- vpcmodel(model = finalModelVPC, outputPRED = TRUE)

# Observed dataset predcheck0.csv
dt_ObsData <- finalVPCJob$predcheck0

# Simulation output dataset predout.csv
dt_SimData <- finalVPCJob$predout

# Add PRED from REPLICATE = 0 of simulation output dataset to observed input dataset
dt_ObsData$PRED <- dt_SimData[REPLICATE == 0]$PRED

# tidyvpc package VPC example:
# library(tidyvpc)
library(magrittr)
# Create a regular VPC plot with binning method set to be "jenks"
binned_VPC <- observed(dt_ObsData, x = IVAR, yobs = DV) %>%
  simulated(dt_SimData, ysim = DV) %>%
  binning(bin = "jenks") %>%
  vpcstats()

plot_binned_VPC <- plot(binned_VPC)

# Create a pcVPC plot with binning method set to be "jenks"
binned_pcVPC <- observed(dt_ObsData, x = IVAR, yobs = DV) %>%
  simulated(dt_SimData, ysim = DV) %>%
  binning(bin = "jenks") %>%
  predcorrect(pred = PRED) %>%
  vpcstats()

plot_binned_pcVPC <- plot(binned_pcVPC)
```

Index