

# Package ‘Certara.RsNLME’

March 18, 2026

**Title** Pharmacometric Modeling

**Version** 3.1.1

**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, and support integrated use within the Pirana pharmacometric workbench <[doi:10.1002/psp4.70067](https://doi.org/10.1002/psp4.70067)>. 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 (>= 3.0.2), 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** 2026-03-18 06:13:33 UTC

## Contents

addADDL . . . . .	4
addCovariate . . . . .	5
addDoseCycle . . . . .	7
addExtraDef . . . . .	9
addInfusion . . . . .	9
addLabel . . . . .	10
addMDV . . . . .	11
addReset . . . . .	12
addSecondary . . . . .	13
addSteadyState . . . . .	14
bootstrap . . . . .	15

cancelJob . . . . .	16
colMapping . . . . .	17
copyModel . . . . .	18
covariateNames . . . . .	19
createModelInfo . . . . .	20
dataMapping . . . . .	21
doseNames . . . . .	21
editModel . . . . .	22
emaxmodel . . . . .	23
engineParams . . . . .	24
extraDoseLines . . . . .	30
extraDoseNames . . . . .	31
fitmodel . . . . .	32
fixedEffect . . . . .	35
getRandomEffectNames . . . . .	37
getThetas . . . . .	37
hostParams . . . . .	38
initFixedEffects . . . . .	40
linearmodel . . . . .	41
listCovariateEffectNames . . . . .	42
modelVariableNames . . . . .	43
obtain_NLMELicense . . . . .	44
OneCpt_IVInfusionData . . . . .	45
parsePMLColMap . . . . .	46
pkcovbqlData . . . . .	46
pkData . . . . .	47
pkemaxmodel . . . . .	48
pkindirectmodel . . . . .	52
pklinearmodel . . . . .	57
pkmodel . . . . .	62
pkpdData . . . . .	65
print.NlmePmlModel . . . . .	66
randomEffect . . . . .	66
removeCovariate . . . . .	67
remove_NLMELicense . . . . .	68
residualEffectNames . . . . .	69
residualError . . . . .	70
secondaryParameterNames . . . . .	71
shotgunSearch . . . . .	72
simmodel . . . . .	74
sortfit . . . . .	75
stepwiseSearch . . . . .	79
structuralParameter . . . . .	81
structuralParameterNames . . . . .	83
tableParams . . . . .	84
textualmodel . . . . .	85
vpcmodel . . . . .	86

---

addADDL	<i>Adds ADDL extra column definition to model object</i>
---------	--

---

### 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

```
pkData1 <- pkData
pkData1$ii <- 0
pkData1$addl <- 0
model <- pkmodel(numComp = 2,
                 absorption = "FirstOrder",
                 ID = "Subject",
                 Time = "Act_Time",
                 CObs = "Conc",
                 Aa = "Amount",
                 data = pkData1,
                 modelName = "PkModel",
                 workingDir = tempdir())
model <- addADDL(model, ADDL = "addl", II = "ii")
```

---

addCovariate	<i>Add covariate to model object</i>
--------------	--------------------------------------

---

### 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 <code>columnMap = TRUE</code> ) use named character if the name of the covariate is different from the corresponding column in the input dataset, for example, <code>covariate = c(BW = "BodyWeight")</code> , where <code>BW</code> denotes the name of the covariate, and <code>"BodyWeight"</code> 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. <b>Important for Occasion Covariates:</b> When modifying an <i>existing</i> occasion covariate (e.g., changing <code>option</code> or <code>values</code> ), you must list <i>all</i> structural parameters currently affected by this occasion covariate in the <code>effect</code> argument, even those whose effect relationship is not being changed in this specific call.
type	Type of covariate. Options are <code>"Continuous"</code> , <code>"Categorical"</code> , <code>"Occasion"</code> .
direction	Direction of missing values propagation (if no covariate value is given). Options are <code>"Forward"</code> , <code>"Interpolate"</code> , <code>"Backward"</code> , where <code>"Interpolate"</code> is only applicable to <code>type = "Continuous"</code> .

option	Options are "Yes", "PlusOne", or "No". <ul style="list-style-type: none"> <li>• "Yes": Apply the covariate effect using the standard method (multiplicative for LogNormal style, additive for Normal style).</li> <li>• "PlusOne": Apply the covariate effect using the "1 + effect" formulation. This is only applicable to continuous and categorical covariates where the affected structural parameter has style = "LogNormal".</li> <li>• "No": Remove the specified covariate effect from the specified structural parameter(s). The covariate itself (and its definition, e.g., fcovariate(Occ1)) remains part of the model, but the link between this covariate and the specified parameter(s) in the stparm() statement is removed. See the note for the effect argument regarding occasion covariates. Multiple options are not supported within a single call (i.e., all covariate effects listed in the effect argument for a single call must use the same option). If different options are required, use sequential calls to addCovariate.</li> </ul>
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",  
  workingDir = tempdir()  
)  
  
# Add Gender covariate of type categorical  
model <- addCovariate(model,  
  covariate = "Gender",  
  type = "Categorical",  
  effect = c("V2", "C12"),  
  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", "C1")  
)
```

---

addDoseCycle	<i>Adds a dosing cycle to model</i>
--------------	-------------------------------------

---

**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

<code>.Object</code>	Model object
<code>type</code>	Specification of dose type. Options are "SteadyState" and "ADDL"
<code>name</code>	Dose point name. See <a href="#">doseNames</a>
<code>administration</code>	Mechanism for administering dose. Options are "Bolus" or "Infusion"
<code>amount</code>	Optional. Column mapping argument specifying corresponding "ADDL" column in input data, or numeric value specifying dose amount.
<code>II</code>	Optional. Column mapping argument specifying corresponding "II" column in input data, or numeric value specifying delta time.
<code>rate</code>	Optional. Column mapping argument specifying corresponding "Rate" column in input data, or numeric specifying dose rate.
<code>duration</code>	Optional. Column mapping argument specifying corresponding "Duration" column in data, or numeric specifying duration value.
<code>isSecondDose</code>	Use second dose point on compartment
<code>colName</code>	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 <- addDoseCycle(pkmodel(columnMap = FALSE,
                             isPopulation = FALSE,
                             workingDir = tempdir()),
                     type = "SteadyState",
                     name = "A1",
                     amount = "Amount",
                     II = "II")

```



---

addExtraDef	<i>Adds user defined extra column/table definitions to column definition file</i>
-------------	---

---

**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 <- pkmodel(columnMap = FALSE,  
                 workingDir = tempdir())  
model <- addExtraDef(model, c("addlcol(ADDL)",  
                             "table(file=\"res.csv\",time(0),Ka,V,C1,Tlg)"))
```

---

addInfusion	<i>Change existing dosing compartment to infusion</i>
-------------	---

---

**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**

```
pkData1 <- pkData
pkData1$A1_1 <- 0
model <- pkmodel(numComp = 2,
  absorption = "Intravenous",
  ID = "Subject",
  Time = "Act_Time",
  CObs = "Conc",
  A1 = "Amount",
  data = pkData1,
  modelName = "PkModel",
  workingDir = tempdir())
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 <- pkmodel(columnMap = FALSE,
                 isPopulation = FALSE,
                 workingDir = tempdir())

model <- suppressWarnings(addCovariate(model,
                                       covariate = "Gender",
                                       type = "Categorical",
                                       effect = c("V"),
                                       levels = c(0, 1)))

model@isTextual <- TRUE
model <- addLabel(model, "Gender", c(1, 2), c("male", "female"))

```

---

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**

```
pkData1 <- pkData
pkData1$MDV <- 0
model <- pkmodel(data = pkData1,
                 ID = "Subject",
                 Time = "Act_Time",
                 A1 = "Amount",
                 CObs = "Conc",
                 workingDir = tempdir()
                 )
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	<i>Adds a secondary parameter to model definition</i>
--------------	---

---

**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 <- pkmodel(columnMap = FALSE,
                 absorption = "FirstOrder",
                 workingDir = tempdir())
model <- addSecondary(model, "Ke", "tvCl/tvV")
model <- addSecondary(
  model, "Tmax",
  "CalcTMax(tvA, tvCl/tvV)"
)
```

---

addSteadyState	<i>Adds Steady State extra column definition to model object</i>
----------------	--

---

**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 NlmePnlModel object

**Examples**

```
pkData1 <- pkData
pkData1$SS <- 0
pkData1$II <- 0
model <- pkmodel(data = pkData1,
  ID = "Subject",
  Time = "Act_Time",
  A1 = "Amount",
  CObs = "Conc",
  workingDir = tempdir()
)
model <- addSteadyState(model, SS = "SS", II = "II")
```

---

bootstrap	<i>Executes an NLME Bootstrap</i>
-----------	-----------------------------------

---

## 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 <a href="#">hostParams</a> . If missing, multicore local host with 4 threads is used.
params	Engine parameters. See <a href="#">engineParams</a> . If missing, default parameters generated by <code>engineParams(model)</code> are used.
bootParams	Bootstrap parameters. See <a href="#">BootstrapParams</a> . If missing, default parameters generated by <code>BootstrapParams()</code> are used.
runInBackground	Set to TRUE to run in background and return prompt.
...	Additional class initializer arguments for <a href="#">BootstrapParams</a> or <a href="#">hostParams</a> , or arguments available inside <a href="#">engineParams</a> functions. If <a href="#">engineParams</a> arguments are supplied through both <code>params</code> argument and additional argument (i.e., ellipsis), then the arguments in <code>params</code> will be ignored and only the additional arguments will be used with warning. If <a href="#">hostParams</a> arguments are supplied through both <code>hostPlatform</code> argument and additional argument, then its values will be overridden by additional arguments. In addition, if <a href="#">BootstrapParams</a> arguments are supplied through both <code>bootParams</code> 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", "Boot-Theta", "BootOmega", "BootOmegaStderr", "BootVarCoVar" comma separated files. Otherwise the `BootNLmeJob` class object is returned.

**See Also**

[hostParams](#), [engineParams](#), [BootstrapParams](#)

**Examples**

```
## Not run:
input_data <- pkData

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

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

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

## End(Not run)
```

---

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,
                 workingDir = tempdir())

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

```
## Not run:
model <- pkmodel(
  parameterization = "Clearance",
  numCompartments = 2,
  data = pkData,
  ID = "Subject",
  Time = "Act_Time",
  A1 = "Amount",
  CObs = "Conc",
  workingDir = tempdir()
)
host <- hostParams(sharedDirectory = tempdir(),
  parallelMethod = "None",
  hostName = "local",
  numCores = 1)
job <- fitmodel(model,
  numIterations = 3,
  hostPlatform = host)

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

## End(Not run)
```

---

covariateNames	<i>Return covariate names</i>
----------------	-------------------------------

---

## 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,
                 workingDir = tempdir())
model <- addCovariate(model, covariate = "BW", effect = "V")
model <- addCovariate(model, covariate = "Age", effect = "Cl")

covariateNames(model)
```

---

createModelInfo	<i>Parse the model and get the list of terms</i>
-----------------	--

---

## 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

```
## Not run:
model <- pkmodel(columnMap = FALSE,
                 workingDir = tempdir())
createModelInfo(model)

## End(Not run)
```

---

dataMapping	<i>Initialize input data for PK/PD model</i>
-------------	--

---

**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,  
                 workingDir = tempdir())  
  
model <- dataMapping(model, pkData)
```

---

doseNames	<i>Return dose names</i>
-----------	--------------------------

---

**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,
                 workingDir = tempdir())

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,
                 workingDir = tempdir())

if (FALSE) {
  # will open an additional window with the model text:
  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

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

- 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

*Main function to specify engine parameters*

---

**Description**

Use to define 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,
logTransform = NULL,
numIntegratePtsAGQ = 1,
numIterNonParametric = 0,
allowSyntheticGradient = FALSE,
fastOptimization = FALSE,
numIterMAPNP = 0,
numRepPCWRES = 0,
stepSizeLinearize = 0.002,
numDigitLaplacian = 7,
numDigitBlup = 13,
gradTolOuter = 2e-04,
stepTolOuter = 1e-04,
gradTolInner = 1.71e-05,
stepTolInner = 7.07e-08,
refDeltaLag1 = 0.001,
mapAssist = 0,
iSample = 300,
iAcceptRatio = 0.1,
impDist = "Normal",
tDOF = 4,
numSampleSIR = 10,
numBurnIn = 0,
freezeOmega = FALSE,
MCPEM = FALSE,
runAllIterations = FALSE,
scramble = "Owen",
emTolType = 0,
emConvLen = 10,
emConvCritVal = 5,
stepSizePartialDeriv = 1e-05,
numTimeStepPartialDeriv = 20
)

```

### Arguments

model	Model object. The type of model (population or individual) is determined by the <code>model@isPopulation</code> slot. If <code>model@isPopulation</code> is <code>TRUE</code> , the model is treated as a population model; otherwise, it's treated as an individual model.
sort	Logical; Specifies whether to sort the input data by subject and time. If <code>TRUE</code> , data are sorted. If <code>FALSE</code> , data are not sorted. Defaults to <code>FALSE</code> if the model contains reset information ( <code>model@hasResetInfo = TRUE</code> ); otherwise, defaults to <code>TRUE</code> .
ODE	Character; Specifies the ODE solver to be used. Options are: "MatrixExponent", "DVERK", "DOPRI5", "AutoDetect", "Stiff", "LSODE". See Details section for

	a description of each solver.
rtolODE	Numeric; Specifying relative tolerance for the numerical ODE solver.
atolODE	Numeric; Specifying absolute tolerance for the numerical ODE solver.
maxStepsODE	Numeric; Specifies the maximum number of steps allowed for the ODE solver.
numIterations	Integer; Specifies the maximum number of iterations for the estimation algorithm. Must be a non-negative integer, with a maximum value of 10000.
method	Character; Specifies the estimation method. For population models, options are: "QRPEM", "IT2S-EM", "FOCE-LB", "FO", "FOCE-ELS", "Laplacian", and "Naive-Pooled". For individual models, only "Naive-Pooled" is available. The default for population models depends on model characteristics: <ul style="list-style-type: none"> <li>• If the model includes discontinuous observed variables (e.g., count data), Below Quantifiable Limit (BQL) data, or has no unfrozen sigmas, the default is "Laplacian".</li> <li>• Otherwise, the default is "FOCE-ELS".</li> </ul>
stdErr	Character; Specifies the method for standard error computations. Options vary depending on the model type and estimation method: <ul style="list-style-type: none"> <li>• Individual models: "Hessian" (default) or "None".</li> <li>• Population models with method = "QRPEM": "Fisher-Score" (default) or "None".</li> <li>• Population models with method = "IT2S-EM": "None" only.</li> <li>• Population models with method in c("FOCE-LB", "FO", "FOCE-ELS", "Laplacian", "Naive-Pooled"): "Sandwich" (default), "Hessian", "Fisher-Score", "Auto-Detect", or "None".</li> </ul> <p>"None" means that standard error calculations are not performed.</p>
isCentralDiffStdErr	Logical; If TRUE (default), uses central difference for standard error calculations when applicable. If FALSE, uses forward difference.
stepSizeStdErr	Numeric; Specifies the relative step size used for the numerical computation of the Hessian matrix during standard error calculations. If not specified, a default value is used (0.001 for "Naive-Pooled" method, and 0.01 otherwise).
logTransform	Logical or NULL; Controls log-transformation behavior, particularly for models with a LogAdditive residual error (e.g., $C \cdot \exp(\epsilon)$ ). The internal engine parameter 'logtran' is set based on this argument and specific model characteristics as detailed below. <ul style="list-style-type: none"> <li>• NULL (default) or TRUE: When the model has exactly one residual error model and it is LogAdditive, this setting enables Log-Transform Both Sides (LTBS). In LTBS, predictions and observations are log-transformed, and the model is fit in the log-domain. This results in the internal 'logtran' engine parameter being set to 1.</li> <li>• FALSE: When the model has exactly one residual error model and it is LogAdditive, this setting results in the LogAdditive error being treated as a proportional/multiplicative error during fitting (by neglecting third and higher-order terms in the Taylor expansion of <math>\exp(\epsilon)</math>). This sets the internal 'logtran' engine parameter to 0. For simulation, the error is treated as <math>\exp(\epsilon)</math>.</li> </ul>

For other model configurations, the 'logtran' parameter is determined as follows:

- If there are multiple residual error models or no residual error models, 'logtran' is set to 0, irrespective of the 'logTransform' value. (In the case of multiple errors, any LogAdditive errors present are treated as proportional).
- If there is a single residual error model that is not LogAdditive:
  - For built-in models: 'logtran' is set to 0.
  - For textual models: 'logtran' reflects the 'logTransform' setting (it becomes 1 if 'logTransform' is 'NULL' or 'TRUE', and 0 if 'logTransform' is 'FALSE'). A warning is issued if 'logTransform' is 'NULL' or 'TRUE' in this scenario, highlighting that LTBS is typically for LogAdditive errors and that error type identification can be challenging in textual models.

numIntegratePtsAGQ

Integer; Specifies the number of quadrature points per dimension to use for Adaptive Gaussian Quadrature (AGQ). Only applicable to population models when method is "FOCE-ELS" or "Laplacian".

- 1: Standard FOCE-ELS/LAPLACIAN computation (no AGQ).
- >1: AGQ is performed. The total number of quadrature points used is  $(\text{number of ETAs})^{\text{numIntegratePtsAGQ}}$ .

numIterNonParametric

Integer; Controls non-parametric (NP) optimization.

- 0: Disables NP optimization.
- 1: Enables NONMEM-style NP optimization using posthoc estimates as support points.
- >1: Enables an evolutionary NP algorithm, using numIterNonParametric as the number of generations.

Only applicable to population models when method is not "Naive-Pooled".

allowSyntheticGradient

Deprecated.

fastOptimization

Logical; Controls the differentiation method used during the optimization of random effects (etas). If TRUE, automatic differentiation is used where possible. If FALSE, a finite difference approach is used. Only applicable to population models when method is "FOCE-ELS" or "Laplacian".

numIterMAPNP

Integer; Specifies the number of iterations for a preliminary Naive-Pooled (NP) optimization run before the main estimation. Applicable when the method is not "NAIVE-POOLED".

numRepPCWRES

Integer; Specifies the number of replicates to generate for Population Conditional Weighted Residuals (PCWRES) calculations. Setting this value to 0 disables PCWRES computation. Only applicable to population models when method is not set to "Naive-Pooled".

stepSizeLinearize

Numeric; Specifies the relative step size for numerical differentiation during model linearization.

numDigitLaplacian	Numeric; Specifies the optimization accuracy (NDIGIT) for the outer loop (thetas and sigmas) when using "FOCE-ELS" or "Laplacian" methods. Only applicable to population models.
numDigitBlup	Numeric; Specifies the optimization accuracy (NDIGIT) for the inner loop (optimization of etas). Also applies to the single optimization loop in the "NAIVE-POOLED" method.
gradTolOuter	Numeric; maximum gradient tolerance for the outer loop (Theta/Omega/Sigma optimization) of "FOCE-ELS" or "Laplacian" method. This tolerance controls how close the gradient must be to zero before the outer optimization is considered converged.
stepTolOuter	Numeric; maximum step tolerance for the outer loop (Theta/Omega/Sigma optimization) of "FOCE-ELS" or "Laplacian" method. This measures the relative change in the solution vector between iterations.
gradTolInner	Numeric; maximum gradient tolerance for the inner loop (Eta optimization) of "FOCE-ELS" or "Laplacian" method. A smaller value forces the algorithm to iterate until a very small gradient is achieved.
stepTolInner	Numeric; maximum step tolerance for the inner loop (Eta optimization) of "FOCE-ELS" or "Laplacian" method. This determines when the algorithm will terminate based on minimal changes in the solution vector.
refDeltaLagl	Numeric; tolerance for the change in the log-likelihood (LL) value during outer loop optimization of "FOCE-ELS" or "Laplacian" method. This parameter is used to check convergence by comparing the absolute change in LL between major iterations. If the change in LL is less than refDeltaLagl and the optimization driver returns a specific termination code, the algorithm considers the solution sufficiently converged. This tolerance helps to avoid unnecessary iterations when improvements in LL become marginal.
mapAssist	Numeric; Controls the use of MAP assistance in the QRPEM algorithm. <ul style="list-style-type: none"> <li>• 0: No MAP assistance.</li> <li>• &gt;0: The inner ETAs optimization loop is used in the QRPEM outer optimization loop with a periodicity equal to the value of mapAssist.</li> </ul> Only applicable to population models with method = "QRPEM".
iSample	Numeric; Specifies the number of sample points used in the QRPEM algorithm. Only applicable to population models with method = "QRPEM".
iAcceptRatio	Numeric; Specifies the acceptance ratio used in the QRPEM algorithm for scaling the covariance matrix. Only applicable to population models with method = "QRPEM".
impDist	Character; Specifies the importance sampling distribution used in the QRPEM algorithm. Options are: "Normal", "DoubleExponential", "Direct", "T", "Mixture-2", "Mixture-3". Only applicable to population models with method = "QRPEM". See Details for further information.
tDOF	Numeric; Specifies the degrees of freedom for the multivariate T distribution used in importance sampling. Only applicable when method = "QRPEM" and impDist = "T". Must be between 3 and 30.

numSampleSIR	Numeric; Specifies the number of samples per eta per subject used in the Sampling Importance Resampling (SIR) algorithm within QRPEM. Only applicable to population models with method = "QRPEM".
numBurnIn	Numeric; Specifies the number of burn-in iterations in the QRPEM algorithm. During burn-in, omegas can be frozen (see freezeOmega parameter). 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; Controls the sampling method used in the QRPEM algorithm. <ul style="list-style-type: none"> <li>• FALSE: Quasi-Random sampling.</li> <li>• TRUE: Monte-Carlo sampling.</li> </ul> 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; Specifies the scrambling method for quasi-random number generation in the QRPEM algorithm. Options are: "None", "Owen", "Faure-Tezuka". Only applicable to population models with method = "QRPEM".
emTolType	Numeric; QRPEM convergence check type. Options: <ul style="list-style-type: none"> <li>• 0: Default (no rollout, LL &amp; Theta and Sigma).</li> <li>• 1: LL &amp; All Population Params (Theta, Omega, and Sigma) with rollout.</li> <li>• 2: LL with rollout.</li> <li>• 3: All Population Params with rollout.</li> </ul> Only applicable to population models with method = "QRPEM".
emConvLen	Numeric; number of iterations over which convergence is checked in the QRPEM method. Only applicable to population models with method = "QRPEM" and emTolType being nonzero.
emConvCritVal	Numeric; critical value used in the QRPEM convergence check. It specifies the threshold improvement required to continue iterating. Only applicable to population models with method = "QRPEM" and emTolType being nonzero.
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. "AutoDetect" represents LSODA solver implementation, 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. "MatrixExponent" is a matrix exponential solver.

For the QRPEM method, the impDist parameter controls the importance sampling distribution. The ximpsampdof slot in the internal NlmeEngineExtraParams object is set based on impDist as follows:

- "Normal": ximpsampdof = 0
- "DoubleExponential": ximpsampdof = 1
- "Direct": ximpsampdof = 2
- "T": ximpsampdof is set to the value of tDOF.
- "Mixture-2": ximpsampdof = -2
- "Mixture-3": ximpsampdof = -3

### Value

List of engine parameters to be used during fitting or simulation

---

<code>extraDoseLines</code>	<i>Return extra dose lines</i>
-----------------------------	--------------------------------

---

### Description

Use to return extra dose lines for model object

### Usage

```
extraDoseLines(model)
```

### Arguments

<code>model</code>	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",
workingDir = tempdir())
addDoseCycle(
  model,
  name = "A1",
  amount = 30000,
  II = 24,
  type = "ADDL",
  colName = "ADDL")

extraDoseLines(model)
```

---

extraDoseNames	<i>Return extra dose names</i>
----------------	--------------------------------

---

### 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",
```

```

    workingDir = tempdir()
addDoseCycle(
  model,
  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 <a href="#">hostParams</a> . If missing, PhoenixM-PIDir64 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 <a href="#">engineParams</a> . If missing, default parameters generated by <code>engineParams(model)</code> are used.
simpleTables	Optional list of simple tables. See <a href="#">tableParams</a> . 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 <a href="#">hostParams</a> or arguments available inside <a href="#">engineParams</a> functions. If <a href="#">engineParams</a> 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 <a href="#">hostParams</a> 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

n1me7engine.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

```
## Not run:

# Define the host
host <- hostParams(sharedDirectory = tempdir(),
                  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",
  workingDir = tempdir())

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)

## End(Not run)

```

---

fixedEffect	<i>Specifies the initial values, lower bounds, upper bounds, and units for fixed effects in a model</i>
-------------	---

---

### 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

<code>.Object</code>	Model object in which to define fixed effects values
<code>effect</code>	Character or character vector specifying names of fixed effects
<code>value</code>	Numeric or numeric vector specifying the initial values of fixed effects. If supplying vector, must be in the same order/length as corresponding effect.
<code>lowerBound</code>	Numeric or numeric vector specifying the lower limit values of fixed effects. If supplying vector, must be in the same order as effect.
<code>upperBound</code>	Numeric or numeric vector specifying the upper limit values of fixed effects. If supplying vector, must be in the same order as effect.
<code>isFrozen</code>	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.
<code>unit</code>	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",
  workingDir = tempdir()
)

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

model <-
fixedEffect(
  model,
  effect = c("tvV", "tvC1", "tvV2", "tvC12"),
  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,  
                 workingDir = tempdir())  
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

```
## Not run:
model <- emaxmodel(
  checkBaseline = TRUE,
  checkFractional = TRUE,
  checkInhibitory = TRUE,
  data = pkpdData,
  ID = "ID",
  C = "CObs",
  EObs = "EObs"
)
getThetas(model)

## End(Not run)
```

---

hostParams

*Create an NLME Parallel Host Configuration*

---

## Description

This helper function simplifies the creation and configuration of an `NlmeParallelHost` object, which defines the environment for running NLME jobs.

## 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`

character. The directory where temporary run folders are created. Defaults to the current working directory for local runs and to `"~"` for remote runs.

installationDirectory	character. The directory containing NLME libraries/scripts. Defaults to the INSTALLDIR environment variable for local runs and to file.path(sharedDirectory, "InstallDirNLME") for remote runs.
hostName	character. A display name for the host. Defaults to the system's network name (from Sys.info()[["nodename"]]) for local runs and to machineName for remote runs.
machineName	character. The IP address or network name of the host. Defaults to "127.0.0.1".
hostType	character. The host operating system. Defaults to the current OS (Sys.info()[["sysname"]]) for local runs and to "linux" for remote runs. While "windows" or "linux" are valid for local runs, for remote Linux hosts the following are officially supported: "RHEL" (for RHEL 8 and 9) and "UBUNTU" (for Ubuntu 22.04 and 24.04). Specifying one of these values correctly sets the PML_BIN_DIR variable.
numCores	numeric. The number of CPU cores to utilize. Defaults to 4.
parallelMethod	character. The parallel execution method. Options include: "None", "Multicore", "LOCAL_MPI", "SGE", "SGE_MPI", "TORQUE", "TORQUE_MPI", "LSF", "LSF_MPI", "SLURM", "SLURM_MPI". Defaults to "LOCAL_MPI".
userName	character. The username for remote host authentication.
privateKeyFile	character. The path to an SSH private key file for remote authentication. See ssh::ssh_connect() for more details.
userPassword	character or function. The password or a callback function for remote authentication. See ssh::ssh_connect() for details.
scriptPath	character. The path to a script to run on a remote host before the main job starts. Ignored for local runs.
rLocation	character. The path to the Rscript executable on a remote host. Ignored for local runs.
isLocal	logical. Set to TRUE for a local host or FALSE for a remote host. Defaults to TRUE.

### Value

An NlmeParallelHost object configured with the specified parameters.

### 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",
  workingDir = tempdir()
)

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



```
# May also use as a 'replacement function' to set the values
initFixedEffects(model) <- c(tvV = 15, tvC1 = 5, tvV2 = 40, tvC12 = 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 <a href="#">colMapping</a> .
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 <a href="#">linearmodel_MappingParameters</a>
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**

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

```
    workingDir = tempdir()
  )
model <- addCovariate(model,
  covariate = "Gender",
  type = "Categorical",
  effect = c("V2", "C12"),
  levels = c(0, 1),
  labels = c("Female", "Male")
)
listCovariateEffectNames(model)
```

---

modelVariableNames	<i>Return model variable names</i>
--------------------	------------------------------------

---

### 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

```
model <- pkmodel(columnMap = FALSE,
  workingDir = tempdir())
modelVariableNames(model)
```

---

obtain_NLMELicense	<i>Obtain NLME License</i>
--------------------	----------------------------

---

### 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 <code>getOption("verbose")</code> .

### 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

```
## Not run:
INSTALLDIR <- Sys.getenv("INSTALLDIR")
if (INSTALLDIR == "") INSTALLDIR <- "C:/Program Files/Certara/NLME_Engine"
result <- obtain_NLMELicense(INSTALLDIR, verbose = TRUE)
```

```
if (result) {  
  message("License obtained successfully!")  
} else {  
  message("Failed to obtain license.")  
}  
  
## End(Not run)
```

---

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	<i>Embed column definition info into the model</i>
----------------	--

---

**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	<i>Pharmacokinetic pediatric dataset containing 80 subjects with single bolus dose.</i>
--------------	---

---

**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 <a href="#">colMapping</a> .
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 <a href="#">pkindirectmodel_MappingParameters</a>
	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".

C10bs 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".

A00bs Column mapping argument that represents the input dataset column for the observed amount of drug in the elimination compartment. (hasEliminationComp = TRUE).

E0bs 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.

nC1 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nC1.

nC12 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nC12.

nC13 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nC13.

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 parameterization = "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 <a href="#">colMapping</a> .
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 <a href="#">pkindirectmodel_MappingParameters</a>
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: <ul style="list-style-type: none"> <li>• Models with absorption = "Intravenous" and parameterization set to either "Clearance", "Micro", or "Macro"</li> <li>• Models with absorption set to either "Gamma", "InverseGaussian", or "Weibull"</li> </ul>
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".
- C10bs 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".
- A00bs Column mapping argument that represents the input dataset column for the observed amount of drug in the elimination compartment. (hasEliminationComp = TRUE).
- E0bs 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.

nC1 If `isSequential = TRUE`, mapped to the input dataset column that lists the values for random effect nC1.

nC12 If `isSequential = TRUE`, mapped to the input dataset column that lists the values for random effect nC12.

nC13 If `isSequential = TRUE`, mapped to the input dataset column that lists the values for random effect nC13.

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 parameterization = "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 <a href="#">colMapping</a> .
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 <a href="#">pkindirectmodel_MappingParameters</a>
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: <ul style="list-style-type: none"> <li>• Models with absorption = "Intravenous" and parameterization set to either "Clearance", "Micro", or "Macro"</li> <li>• Models with absorption set to either "Gamma", "InverseGaussian", or "Weibull"</li> </ul>
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: <ul style="list-style-type: none"> <li>• Models with absorption = "Intravenous", infusionAllowed = TRUE and parameterization set to either "Clearance", "Micro" or "Macro"</li> <li>• Models with absorption set to either "Gamma", "InverseGaussian", or "Weibull" and infusionAllowed = TRUE</li> </ul>
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: <ul style="list-style-type: none"> <li>• Models with absorption = "Intravenous", infusionAllowed = TRUE with isDuration = TRUE and parameterization set to either "Clearance", "Micro" or "Macro"</li> <li>• Models with absorption set to either "Gamma", "InverseGaussian", or "Weibull" and infusionAllowed = TRUE with isDuration = TRUE</li> </ul>
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".

C10bs 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".

A00bs Column mapping argument that represents the input dataset column for the observed amount of drug in the elimination compartment. (hasEliminationComp = TRUE).

E0bs 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.

nC1 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nC1.

nC12 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nC12.

nC13 If isSequential = TRUE, mapped to the input dataset column that lists the values for random effect nC13.

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 parameterization = "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 <a href="#">colMapping</a> .
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 <a href="#">pkmodel_MappingParameters</a>
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: <ul style="list-style-type: none"> <li>• Models with absorption = "Intravenous" and parameterization set to either "Clearance", "Micro", or "Macro"</li> <li>• Models with absorption set to either "Gamma", "InverseGaussian", or "Weibull"</li> </ul>
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".
- C10bs 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".
- A00bs 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",  
  workingDir = tempdir()  
)  
  
# View the model as well as its associated column mappings  
print(model)
```

---

pkpdData	<i>Pharmacokinetic/Pharmacodynamic dataset containing 200 subjects with single bolus dose</i>
----------	---

---

**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,  
                 workingDir = tempdir())  
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",
  workingDir = tempdir()
)

model <-
  randomEffect(model,
    effect = c("nV", "nC1", "nC12"), 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 NlmePnlModel object

**Examples**

```

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

# Add Gender covariate of type categorical
model <- addCovariate(model,
  covariate = "Gender",
  type = "Categorical",
  effect = c("V2", "C12"),
  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", "C1")
)

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

```

**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**

```
## Not run:
INSTALLDIR <- Sys.getenv("INSTALLDIR")
if (INSTALLDIR == "") INSTALLDIR <- "C:/Program Files/Certara/NLME_Engine"
if (FALSE) { # to prevent unintended logout
  result <- remove_NLMELicense(INSTALLDIR)
}

## End(Not run)
```

---

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	<i>Assign residual error model to model object</i>
---------------	--

---

**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,
  E0bsBQL = NULL,
  C0bsBQL = NULL,
  C10bsBQL = NULL,
  A00bsBQL = NULL,
  exponent = NULL
)
```

**Arguments**

.Object	Model object
predName	Name of the predicted variable as returned in <a href="#">residualEffectNames</a> .
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
E0bsBQL	Column mapping argument that represents the input dataset column that contains the BQL flag for observation values corresponding to E0bs. Only applicable to isBQL = TRUE.
C0bsBQL	Column mapping argument that represents the input dataset column that contains the BQL flag for observation values corresponding to C0bs. Only applicable to isBQL = TRUE.
C10bsBQL	Column mapping argument that represents the input dataset column that contains the BQL flag for observation values corresponding to C10bs. Only applicable to isBQL = TRUE.
A00bsBQL	Column mapping argument that represents the input dataset column that contains the BQL flag for observation values corresponding to A0bs. 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", C0bs = "C0bs", E0bs = "E0bs")

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)
secondaryparams <- secondaryParameterNames(model)
```

---

<code>shotgunSearch</code>	<i>Executes an NLME shotgun covariate search</i>
----------------------------	--

---

**Description**

Executes an NLME shotgun covariate search

**Usage**

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

**Arguments**

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



**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**

```
## Not run:
# Define the model
model <- pkmodel(numCompartments = 2,
                 data = pkData,
                 ID = "Subject",
                 Time = "Act_Time",
                 A1 = "Amount",
                 CObs = "Conc",
                 workingDir = tempdir())

# 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 = "MULTICORE",
                  hostName = "local",
                  numCores = 8,
                  sharedDirectory = tempdir())

# Define the engine parameters
params <- engineParams(model, fastOptimization = TRUE, numIterations = 7)

# Define covariate model
cp <- covariateModel(model)

# Perform shotgun search
OverallDF <- shotgunSearch(model = model,
                          hostPlatform = host,
```

```

        params = params,
        covariateModel = cp,
        runInBackground = FALSE)

## End(Not run)

```

---

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

```
## Not run:
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",
  workingDir = tempdir()
)

host <- hostParams(
  sharedDirectory = tempdir(),
  parallelMethod = "NONE",
  hostName = "local",
  numCores = 1
)

results <- simmodel(model, simParams, hostPlatform = host)
# with seed given additionally:
results <- simmodel(model, simParams, hostPlatform = host, seed = 3527)

## End(Not run)
```

**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 <a href="#">hostParams</a> . If missing, PhoenixM-PIDir64 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 <a href="#">engineParams</a> . If missing, default parameters generated by <code>engineParams(model)</code> are used.
sortColumns	List of sort columns. See <a href="#">SortColumns</a> . 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 <a href="#">NlmeScenario</a> If missing, all covariates effects are considered as enabled.
simpleTables	Optional list of simple tables. See <a href="#">tableParams</a> . 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 <a href="#">hostParams</a> or arguments available inside <a href="#">engineParams</a> functions. If <a href="#">engineParams</a> arguments are supplied through both <code>params</code> argument and additional argument (i.e., ellipsis), then the arguments in <code>params</code> will be ignored and only the additional arguments will be used with warning. If <a href="#">hostParams</a> arguments are supplied through both the <code>hostPlatform</code> argument and the ellipses, values supplied to <code>hostPlatform</code> 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

```
## Not run:
input_data <- pkData

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

model <-
  addCovariate(model,
              covariate = "BodyWeight",
              direction = "Backward",
              center = "Mean",
              effect = c("V", "C1"))

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

```

sharedDirectory = tempdir()

# 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, numIterations = 5, fastOptimization = TRUE),
          sortColumns = SortColumns("Gender"),
          scenarios = scenarios)

## End(Not run)

```

---

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

```

## Not run:
# Define the model
model <- pkmodel(numCompartments = 1,
                 data = pkData,
                 ID = "Subject",
                 Time = "Act_Time",
                 A1 = "Amount",
                 CObs = "Conc",
                 workingDir = tempdir())

# Add Gender covariate of type categorical
model <- addCovariate(model,
                     covariate = "Gender",
                     type = "Categorical",

```



```

        effect = c("V", "C1"),
        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", "C1"))

# Define the host
defaultHost <- hostParams(parallelMethod = "MULTICORE",
  hostName = "local",
  numCores = 8,
  sharedDirectory = tempdir())

# Define the engine parameters
params <- engineParams(model, numIterations = 6)

# 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)

## End(Not run)

```

---

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. <ul style="list-style-type: none"> <li>• "LogNormal" (Default): The structural parameter is defined as <math>\text{Product} * \exp(\text{Eta})</math></li> <li>• "LogNormal1": The structural parameter is defined as <math>\text{Sum} * \exp(\text{Eta})</math></li> <li>• "LogNormal2": The structural parameter is defined as <math>\exp(\text{Sum} + \text{Eta})</math></li> <li>• "LogitNormal": The structural parameter is defined as <math>\text{ilogit}(\text{Sum} + \text{Eta})</math></li> <li>• "Normal": The structural parameter is defined as <math>\text{Sum} + \text{Eta}</math></li> </ul> <p>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).</p>
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 <code>NlmePmlModel</code>
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 <a href="#">covariateNames</a>
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 <a href="#">NlmeVpcParams</a> . If missing, default values generated by <code>NlmeVpcParams()</code> are used.
params	Engine argument setup. See <a href="#">engineParams</a> . The following arguments are the subject of interest: <code>sort</code> , <code>ODE</code> , <code>rtolODE</code> , <code>atolODE</code> , <code>maxStepsODE</code> . If missing, default values generated by <code>engineParams(model)</code> are used.
hostPlatform	Host definition for model execution. See <a href="#">hostParams</a> . 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

```
## Not run:
model <- pkmodel(
  numComp = 1,
  absorption = "Extravascular",
  ID = "Subject",
  Time = "Act_Time",
  CObs = "Conc",
  Aa = "Amount",
  data = pkData,
  modelName = "PkModel",
  workingDir = tempdir()
)

host <- hostParams(
  sharedDirectory = tempdir(),
  parallelMethod = "NONE",
  hostName = "local",
  numCores = 1
)

job <- fitmodel(model = model,
               hostPlatform = host)

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

```
# 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, hostPlatform = host)
# 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)

## End(Not run)
```



# Index

- \* **NLME**
  - hostParams, 38
- \* **NlmeParallelHost**
  - hostParams, 38
- \* **datasets**
  - OneCpt\_IVInfusionData, 45
  - pkcovbqlData, 46
  - pkData, 47
  - pkpdData, 65
  
- addADDL, 4
- addCovariate, 5
- addDoseCycle, 4, 7
- addExtraDef, 9
- addInfusion, 9
- addLabel, 10
- addMDV, 11
- addReset, 12
- addReset, NlmePmlModel-method  
(addReset), 12
- addSecondary, 13
- addSecondary, NlmePmlModel-method  
(addSecondary), 13
- addSteadyState, 14
  
- bootstrap, 15
- BootstrapParams, 15, 16
  
- cancelJob, 16
- cancelJob, SimpleNlmeJob-method  
(cancelJob), 16
- colMapping, 17, 21, 23, 24, 41, 42, 49, 52, 54,  
57, 59, 61, 63, 64
- copyModel, 18
- covariateNames, 19, 84
- createModelInfo, 20
  
- dataMapping, 17, 21
- doseNames, 8, 21
  
- editModel, 22
  
- emaxmodel, 23
- emaxmodel\_MappingParameters, 23
- engineParams, 15, 16, 24, 32–34, 72–74, 76,  
78, 80, 86, 87
- extraDoseLines, 30
- extraDoseNames, 31
  
- fitmodel, 32
- fixedEffect, 35, 40
  
- getRandomEffectNames, 37
- getThetas, 37
  
- hostParams, 15, 16, 32–34, 38, 72–74, 76, 78,  
80, 86, 87
  
- initFixedEffects, 40
- initFixedEffects, NlmePmlModel-method  
(initFixedEffects), 40
- initFixedEffects<- (initFixedEffects),  
40
- initFixedEffects<- , NlmePmlModel-method  
(initFixedEffects), 40
  
- linearmodel, 41
- linearmodel\_MappingParameters, 41
- listCovariateEffectNames, 42
- listCovariateEffectNames, NlmePmlModel-method  
(listCovariateEffectNames), 42
  
- modelVariableNames, 17, 43
  
- NlmeScenario, 76, 78
- NlmeSimulationParams, 74
- NlmeVpcParams, 86, 87
  
- obtain\_NLMELicense, 44
- OneCpt\_IVInfusionData, 45
  
- parsePMLColMap, 46
- pkcovbqlData, 46

pkData, [47](#)  
pkemaxmodel, [48](#)  
pkindirectmodel, [52](#)  
pkindirectmodel\_MappingParameters, [49](#),  
[54](#), [59](#)  
pklinearmodel, [57](#)  
pkmodel, [62](#)  
pkmodel\_MappingParameters, [63](#)  
pkpdData, [65](#)  
print.NlmePmlModel, [66](#)  
  
randomEffect, [66](#)  
remove\_NLMELicense, [68](#)  
removeCovariate, [67](#)  
residualEffectNames, [69](#), [70](#)  
residualError, [70](#)  
  
secondaryParameterNames, [71](#)  
shotgunSearch, [72](#)  
simmodel, [74](#)  
SortColumns, [76](#), [78](#)  
sortfit, [75](#)  
StepwiseParams, [80](#)  
stepwiseSearch, [79](#)  
structuralParameter, [81](#)  
structuralParameterNames, [83](#)  
  
tableParams, [32](#), [34](#), [76](#), [78](#), [84](#)  
textualmodel, [85](#)  
  
vpcmodel, [86](#)