Package ‘scaRabee’

October 14, 2022

Type Package

Title Optimization Toolkit for Pharmacokinetic-Pharmacodynamic Models

Version 1.1-4

Date 2022-02-04

Depends neldermead (>= 1.0-8), optimsimplex (>= 1.0-5), optimbase (>= 1.0-8)

Imports lattice, grid, deSolve, utils

Suggests knitr (>= 1.28),rmarkdown (>= 2.2)

Description A port of the Scarabee toolkit originally written as a Matlab-based application. scaRabee provides a framework for simulation and optimization of pharmacokinetic-pharmacodynamic models at the individual and population level. It is built on top of the neldermead package, which provides the direct search algorithm proposed by Nelder and Mead for model optimization.

License GPL-3

Encoding UTF-8

VignetteBuilder knitr

LazyLoad yes

NeedsCompilation no

Author Sebastien Bihorel [aut, cre]

Maintainer Sebastien Bihorel <sb.pmlab@gmail.com>

Repository CRAN

Date/Publication 2022-02-04 13:00:02 UTC

R topics documented:

`scaRabee-package` *scaRabee toolkit*

Description

Framework for Pharmacokinetic-Pharmacodynamic Model Simulation and Optimization

Details

Package:	scaRabee
Type:	Package
Version:	1.1-4
Date:	2022-02-04
License:	GPL-v3
LazyLoad:	yes

scaRabee is a toolkit for modeling and simulation primarily intended for the field of pharmacometrics. This package is a R port of Scarabee, a Matlab-based piece of software developed as a fairly simple application for the simulation and optimization of pharmacokinetic and/or pharmacodynamic models specified with explicit solutions, ordinary or delay differential equations.

The method of optimization used in scaRabee is based upon the Nelder-Mead simplex algorithm, as implemented by the `fminsearch` function from the **neldermead** package.

Please, refer to the vignette to learn how to run analyses with **scaRabee** and read more about the methods used in **scaRabee**.

scaRabee is available on the Comprehensive R Archive Network and also at: `https://github.com/sbihorel/scaRabee`

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

See Also

`neldermead`

`bound.parameters` *Forces parameter estimates between defined boundaries.*

Description

`bound.parameters` is a utility function called during estimation runs. It forces the parameter estimates to remain within the boundaries defined in the .csv file of initial estimates. `bound.parameters` is typically not called directly by users.

Usage

```
bound.parameters(x = NULL,
                 lb = NULL,
                 ub = NULL)
```

Arguments

`x`	A vector of *p* parameter estimates.
`lb`	A vector of *p* lower boundaries.
`ub`	A vector of *p* upper boundaries.

Value

Returns a vector of *p* values. The ith element of the returned vector is:

- x[i] if lb[i] < x[i] < ub[i]
- lb[i] if x[i] <= lb[i]
- ub[i] if ub[i] <= x[i]

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

Examples

```
bound.parameters(seq(1:5), lb=rep(3,5), ub=rep(4,5))

# The following call should return an error message
bound.parameters(1, lb=rep(3,5), ub=rep(4,5))
```

`compute.secondary` *Computes secondary parameter values*

Description

`compute.secondary` is a secondary function called during estimations runs. It evaluates the code provided in the $SECONDARY block of the model file; all parameters defined in this block are considered as secondary parameters at the initial and the final estimates of the model parameters. `compute.secondary` is typically not called directly by users.

Usage

```
compute.secondary(subproblem = NULL,
                  x = NULL)
```

Arguments

`subproblem` A list containing the following levels:

- **code** A list of R code extracted from the model file. Depending on content of the model file, the levels of this list could be: template, derived, lags, ode, dde, output, variance, and/or secondary.
- **method** A character string, indicating the scale of the analysis. Should be 'population' or 'subject'.
- **init** A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.
- **debugmode** Logical indicator of debugging mode.
- **modfun** Model function.
- **data** A list containing as many levels as there are treatment levels for the subject (or population) being evaluated, plus the `trts` level listing all treatments for this subject (or population), and the `id` level giving the identification number of the subject (or set to 1 if the analysis was run at the level of the population.
  Each treatment-specific level is a list containing the following levels:
  - **ana** mij x 3 data.frame containing the times of observations of the dependent variables (extracted from the TIME variable), the indicators of the type of dependent variables (extracted from the CMT variable), and the actual dependent variable observations (extracted from the DV variable) for this particular treatment.
  - **cov** mij x c data.frame containing the times of observations of the dependent variables (extracted from the TIME variable) and all the covariates identified for this particular treatment.
  - **bolus** bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.
  - **infusion** fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.
  - **trt** the particular treatment identifier.

`x` The vector of p final parameter estimates.

Value

Return a list of with the following elements:

- **init** The vector of s secondary parameter estimates derived from initial structural model parameter estimates.
- **estimates** The vector of s secondary parameter estimates derived from final structural model parameter estimates.
- **names** The vector of s secondary parameter names.

Author(s)

Sebastien Bihorel (`<sb.pmlab@gmail.com>`)

See Also

`scarabee.analysis`, `weighting`, `fitmle`, `get.secondary`

`convert.infusion` *Process Infusion Information*

Description

`convert.infusion` is a secondary function, which main purpose is to transform infusion information provided using NONMEM standards into an object that can be used by scaRabee model functions. `convert.infusion` is typically not called directly by users.

Usage

```
convert.infusion(infusion.data = NULL)
```

Arguments

`infusion.data`	A data.frame with the following variables: trt, time, cmt, amt, and rate.

Value

Return a data.frame with the following variables: trt, time, state, bolus, and infusion.

Author(s)

Sebastien Bihorel (`<sb.pmlab@gmail.com>`)

See Also

`scarabee.read.data`

`dde.model` *Delay Differential Equations*

Description

`dde.model` is the system evaluation function called when a $DDE block is detected in the model file, indicating that the model is defined by delay differential equations. `dde.model` is typically not called directly by users

Usage

```
dde.model(parms = NULL,
          derparms = NULL,
          code = NULL,
          bolus = NULL,
          infusion = NULL,
          xdata = NULL,
          covdata = NULL,
          issim = 0,
          check = FALSE,
          options = list(method='lsoda'))
```

Arguments

`parms`	A vector of primary parameters.
`derparms`	A list of derived parameters, specified in the $DERIVED block of code.
`code`	A list of R code extracted from the model file. Depending on content of the model file, the levels of this list could be: template, derived, lags, ode, dde, output, variance, and/or secondary.
`bolus`	A data.frame providing the instantaneous inputs entering the system of delay differential equations for the treatment and individual being evaluated.
`infusion`	A data.frame providing the zero-order inputs entering the system of delay differential equations for the treatment and individual being evaluated.
`xdata`	A vector of times at which the system is being evaluated.
`covdata`	A data.frame of covariate data for the treatment and individual being evaluated.
`issim`	An indicator for simulation or estimation runs.
`check`	An indicator whether checks should be performed to validate function inputs.
`options`	A list of delay differential equation solver options. Currently not modifiable by users.

Details

`dde.model` evaluates the model for each treatment of each individual contained in the dataset using, among other, the dedicated utility functions: `dde.syst`, and `dde.lags`. The actual evaluation of the system is performed by dede from the **deSolve** package.

`dde.model` also make use of utility functions which it shares with the other system evaluation functions `explicit.model`, and `ode.model`, such as `create.intervals`, `derived.parms`, `init.cond`, `input.scaling`, `make.dosing`, `init.update`, and `de.output`.

Value

Returns a matrix of system predictions.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

See Also

dede, dde.syst, dde.lags, explicit.model, ode.model, init.cond, input.scaling, make.dosing, init.update, de.output

---

dde.utils *Utility Functions for Delay Differential Equation Systems*

Description

This is a collection of utility functions called by `dde.model` when a model defined by delay differential equations is evaluated. None of these functions is typically called directly by users.

Usage

```
dde.syst(t = NULL,
         y = NULL,
         ic = NULL,
         parms = NULL,
         derparms = NULL,
         delags = NULL,
         codedde = NULL,
         dosing = NULL,
         has.dosing = NULL,
         dose.states = NULL,
         xdata = NULL,
         covdata = NULL,
         scale = NULL,
         check = FALSE)
dde.lags(parms = NULL,
         derparms = NULL,
         codelags = NULL,
         check = FALSE)
```

Arguments

`t`	A scalar or a vector of numerical time values.
`y`	A vector of system state values.
`ic`	A vector of initial conditions, typically returned by `init.cond`.
`parms`	A vector of primary parameters.
`derparms`	A list of derived parameters, specified in the $DERIVED block of code.
`delags`	A vector of delay parameters, typically returned by `dde.lags`.
`codedde`	The content of the R code specified within the $DDE block in the model file.
`dosing`	A data.frame of dosing information created by `make.dosing` from instantaneous and zero-order inputs into the system and containing the following columns:

TIME Dosing event times.

CMT State where the input should be assigned to.

AMT Amount that should be assigned to system `state` at the corresponding `TIME`.

RATE Rate of input that should be assigned to system `CMT` at the corresponding `TIME`. See `vignette('scaRabee',package='scaRabee')` for more details about the interpolation of the input rate at time not specified in `dosing`.

TYPE An indicator of the type of input. `TYPE` is set to 1 if the record in `dosing` correspond an original bolus input; it is set to 0 otherwise.

`has.dosing`	A logical variable, indicating whether any input has to be assigned to a system state.
`dose.states`	A vector of integers, indicating in which system state one or more doses have to be assigned to.
`xdata`	A vector of times at which the system is being evaluated.
`covdata`	A matrix of covariate data extracted from the dataset.
`scale`	A vector of system scale, typically returned by `input.scaling`
`check`	An indicator whether checks should be performed to validate function inputs
`codelags`	The content of the R code specified within the $LAGS block in the model file.

Details

`dde.syst` is the function which actually evaluates the system of delay differential equations specified in the $DDE block.

`dde.lags` is the function which evaluates the code specified in the $LAG block and defines the delays at which the system needs to be computed.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

See Also

`dde.model`, `make.dosing`

`estimation.plot` *Create Summary Estimation Plots*

Description

`estimation.plot` is a secondary function called at the end of the estimation runs. It generates plots from the iteration log file and the prediction & residual file. Those plots are: a figure summarizing the changes in the objective function and the estimated parameter values as a function of the iteration plus, for each subject and sub-problem (i.e. treatment), a figure overlaying model predictions and observed data, and another figure showing 4 goodness-of-fit plots (predictions vs observations, weighted residuals vs time, weighted residuals vs observations, weighted residuals vs predictions). See `vignette('scaRabee',package='scaRabee')` for more details. `estimation.plot` is typically not called directly by users.

Usage

```
estimation.plot(problem = NULL,
                Fit = NULL,
                files = NULL)
```

Arguments

`problem` A list containing the following levels:

data A list which content depends on the scope of the analysis. If the analysis was run at the level of the subject, `data` contains as many levels as the number of subjects in the dataset, plus the `ids` level containing the vector of identification numbers of all subjects included in the analysis population. If the analysis was run at the level of the population, `data` contains only one level of data and `ids` is set to 1.
Each subject-specific level contains as many levels as there are treatment levels for this subject, plus the `trts` level listing all treatments for this subject, and the `id` level giving the identification number of the subject.
Each treatment-specific levels is a list containing the following levels:

- **cov** mij x 3 data.frame containing the times of observations of the dependent variables (extracted from the TIME variable), the indicators of the type of dependent variables (extracted from the CMT variable), and the actual dependent variable observations (extracted from the DV variable) for this particular treatment and this particular subject.
- **cov** mij x c data.frame containing the times of observations of the dependent variables (extracted from the TIME variable) and all the covariates identified for this particular treatment and this particular subject.
- **bolus** bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.
- **infusion** fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.
- **trt** the particular treatment identifier.

method A character string, indicating the scale of the analysis. Should be 'population' or 'subject'.

init A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.

debugmode Logical indicator of debugging mode.

modfun Model function.

`Fit` A list containing the following elements:

estimations The vector of final parameter estimates.

fval The minimal value of the objective function.

cov The matrix of covariance for the parameter estimates.

orderedestimations A data.frame with the same structure as `problem$init` but only containing the sorted estimated estimates. The sorting is performed by `order.param.list`.

cor The upper triangle of the correlation matrix for the parameter estimates.

cv The coefficients of variations for the parameter estimates.

ci The confidence interval for the parameter estimates.

AIC The Akaike Information Criterion.

sec A list of data related to the secondary parameters, containing the following elements:

- **estimates** The vector of secondary parameter estimates calculated using the initial estimates of the primary model parameters.
- **names** The vector of names of the secondary parameter estimates.
- **pder** The matrix of partial derivatives for the secondary parameter estimates.
- **cov** The matrix of covariance for the secondary parameter estimates.
- **cv** The coefficients of variations for the secondary parameter estimates.
- **ci** The confidence interval for the secondary parameter estimates.

orderedfixed A data.frame with the same structure as `problem$init` but only containing the sorted fixed estimates. The sorting is performed by `order.param.list`.

orderedinitial A data.frame with the same content as `problem$init` but sorted by `order.param.list`.

`files` A list of input used for the analysis. The following elements are expected and none of them could be null:

data A .csv file located in the working directory, which contains the dosing information, the observations of the dependent variable(s) to be modeled, and possibly covariate information. The expected format of this file is described in details in `vignette('scaRabee', package='scaRabee')`.

param A .csv file located in the working directory, which contains the initial guess(es) for the model parameter(s) to be optimized or used for model simulation. The expected format of this file is described in details in `vignette('scaRabee',package='`

model A text file located in the working directory, which defines the model. Models specified with explicit, ordinary or delay differential equations are expected to respect a certain syntax and organization detailed in `vignette('scaRabee',package='s`

iter A .csv file reporting the values of the objective function and estimates of model parameters at each iteration.

report A text file reporting for each individual in the dataset the final parameter estimates for structural model parameters, residual variability and secondary parameters as well as the related statistics (coefficients of variation, confidence intervals, covariance and correlation matrix).

pred A .csv file reporting the predictions and calculated residuals for each individual in the dataset.

est A .csv file reporting the final parameter estimates for each individual in the dataset.

sim A .csv file reporting the simulated model predictions for each individual in the dataset. (Not used for estimation runs).

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

`examples.data` *Datasets for scaRabee demo.*

Description

Datasets for scaRabee demo.

Usage

```
data(example1.data)
data(example1.initials)

data(example2.data)
data(example2.initials)

data(example3.data)
data(example3.initials)

data(example4.data)
data(example4.initials)

data(example6.data)
data(example6.initials)

data(example8.data)
data(example8.initials)
```

Format

All example#.data are data.frame of observations and dosing events organized as typical scaRabee datasets.

All example#.initials are data.frame of parameter names and values organized as typical scaRabee parameter files.

Details

These datasets are intended to be used in the scaRabee package demos.

Examples

```
data(example1.data)
data(example1.initials)
```

`explicit.model` *Explicit Equations*

Description

`explicit.model` is the system evaluation function called when neither $ODE or $DDE blocks are detected in the model file, indicating that the model is defined by explicit equations. `explicit.model` is typically not called directly by users.

Usage

```
explicit.model(parms = NULL,
               derparms = NULL,
               code = NULL,
               bolus = NULL,
               infusion = NULL,
               xdata = NULL,
               covdata = NULL,
               issim = 0,
               check = FALSE)
```

Arguments

`parms`	A vector of primary parameters.
`derparms`	A list of derived parameters, specified in the $DERIVED block of code. (NULL for `explicit.model`).
`code`	A list of R code extracted from the model file. Depending on content of the model file, the levels of this list could be: template, derived, lags, ode, dde, output, variance, and/or secondary.
`bolus`	A data.frame providing the instantaneous inputs entering the system of delay differential equations for the treatment and individual being evaluated.
`infusion`	A data.frame providing the zero-order inputs entering the system of delay differential equations for the treatment and individual being evaluated.
`xdata`	A vector of times at which the system is being evaluated.
`covdata`	A data.frame of covariate data for the treatment and individual being evaluated.
`issim`	An indicator for simulation or estimation runs.
`check`	An indicator whether checks should be performed to validate function inputs.

Details

`explicit.model` evaluates the model for each treatment of each individual contained in the dataset, based upon the code specified in the $OUTPUT block in the model file.

Value

Returns a matrix of system predictions.

Author(s)

Sebastien Bihorel (<sb.pmlab@gmail.com>)

`finalize.grid.report` *Finalize Direct Grid Search Report*

Description

`finalize.grid.report` is a secondary function called at the end of the direct grid search step of the direct grid search runs (this step is actually by a simulation step). It outputs to the report file the grid search summary table produced by `scarabee.gridsearch`. `finalize.grid.report` is typically not called directly by users.

Usage

```
finalize.grid.report(problem = NULL,
                     fgrid = NULL,
                     files = NULL)
```

Arguments

`problem` A list containing the following levels:

data A list which content depends on the scope of the analysis. If the analysis was run at the level of the subject, `data` contains as many levels as the number of subjects in the dataset, plus the `ids` level containing the vector of identification numbers of all subjects included in the analysis population. If the analysis was run at the level of the population, `data` contains only one level of data and `ids` is set to 1.
Each subject-specific level contains as many levels as there are treatment levels for this subject, plus the `trts` level listing all treatments for this subject, and the `id` level giving the identification number of the subject.
Each treatment-specific levels is a list containing the following levels:

- **cov** mij x 3 data.frame containing the times of observations of the dependent variables (extracted from the TIME variable), the indicators of the type of dependent variables (extracted from the CMT variable), and the actual dependent variable observations (extracted from the DV variable) for this particular treatment and this particular subject.
- **cov** mij x c data.frame containing the times of observations of the dependent variables (extracted from the TIME variable) and all the covariates identified for this particular treatment and this particular subject.
- **bolus** bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.
- **infusion** fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.
- **trt** the particular treatment identifier.

- **method** A character string, indicating the scale of the analysis. Should be 'population' or 'subject'.
- **init** A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.
- **debugmode** Logical indicator of debugging mode.
- **modfun** Model function.

`fgrid` A data.frame with pe+2 columns. The last 2 columns report the value and the feasibility of the objective function at each specific vector of parameter estimates which is documented in the first pe columns. This data.frame must be ordered by feasibility and increasing value of the objective function.

`files` A list of input used for the analysis. The following elements are expected and none of them could be null:

- **data** A .csv file located in the working directory, which contains the dosing information, the observations of the dependent variable(s) to be modeled, and possibly covariate information. The expected format of this file is described in details in `vignette('scaRabee', package='scaRabee')`.
- **param** A .csv file located in the working directory, which contains the initial guess(es) for the model parameter(s) to be optimized or used for model simulation. The expected format of this file is described in details in `vignette('scaRabee',package='`
- **model** A text file located in the working directory, which defines the model. Models specified with explicit, ordinary or delay differential equations are expected to respect a certain syntax and organization detailed in `vignette('scaRabee',package='s`
- **iter** A .csv file reporting the values of the objective function and estimates of model parameters at each iteration. (Not used for direct grid search runs).
- **report** A text file reporting the summary tables of ordered objective function values for the various tested vectors of model parameters.
- **pred** A .csv file reporting the predictions and calculated residuals for each individual in the dataset. (Not used for direct grid search runs).
- **est** A .csv file reporting the final parameter estimates for each individual in the dataset. (Not used for direct grid search runs).
- **sim** A .csv file reporting the simulated model predictions for each individual in the dataset. (Not used for direct grid search runs).

Value

Does not return any value.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

See Also

`scarabee.gridsearch`,

`finalize.report` *Finalize Estimation Report*

Description

`finalize.report` is a secondary function called at the end of the estimation runs. It outputs to the report file the final parameter estimates for structural model parameters, residual variability and secondary parameters as well as the related statistics (coefficients of variation, confidence intervals, covariance and correlation matrix). `finalize.report` is typically not called directly by users.

Usage

```
finalize.report(problem = NULL,
                Fit = NULL,
                files = NULL)
```

Arguments

`problem` A list containing the following levels:

- **data** A list containing as many levels as there are treatment levels for the subject (or population) being evaluated, plus the `trts` level listing all treatments for this subject (or population), and the `id` level giving the identification number of the subject (or set to 1 if the analysis was run at the level of the population.
  Each treatment-specific level is a list containing the following levels:
  - **cov** mij x 3 data.frame containing the times of observations of the dependent variables (extracted from the TIME variable), the indicators of the type of dependent variables (extracted from the CMT variable), and the actual dependent variable observations (extracted from the DV variable) for this particular treatment.
  - **cov** mij x c data.frame containing the times of observations of the dependent variables (extracted from the TIME variable) and all the covariates identified for this particular treatment.
  - **bolus** bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.
  - **infusion** fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.
  - **trt** the particular treatment identifier.
- **method** A character string, indicating the scale of the analysis. Should be 'population' or 'subject'.
- **init** A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.
- **debugmode** Logical indicator of debugging mode.
- **modfun** Model function.

`Fit` A list containing the following elements:

estimations The vector of final parameter estimates.

fval The minimal value of the objective function.

cov The matrix of covariance for the parameter estimates.

orderedestimations A data.frame with the same structure as `problem$init` but only containing the sorted estimated estimates. The sorting is performed by `order.param.list`.

cor The upper triangle of the correlation matrix for the parameter estimates.

cv The coefficients of variations for the parameter estimates.

ci The confidence interval for the parameter estimates.

AIC The Akaike Information Criterion.

sec A list of data related to the secondary parameters, containing the following elements:

- **estimates** The vector of secondary parameter estimates calculated using the initial estimates of the primary model parameters.
- **estimates** The vector of secondary parameter estimates calculated using the final estimates of the primary model parameters.
- **names** The vector of names of the secondary parameter estimates.
- **pder** The matrix of partial derivatives for the secondary parameter estimates.
- **cov** The matrix of covariance for the secondary parameter estimates.
- **cv** The coefficients of variations for the secondary parameter estimates.
- **ci** The confidence interval for the secondary parameter estimates.

`files` A list of input used for the analysis. The following elements are expected and none of them could be null:

data A .csv file located in the working directory, which contains the dosing information, the observations of the dependent variable(s) to be modeled, and possibly covariate information. The expected format of this file is described in details in `vignette('scaRabee', package='scaRabee')`.

param A .csv file located in the working directory, which contains the initial guess(es) for the model parameter(s) to be optimized or used for model simulation. The expected format of this file is described in details in `vignette('scaRabee',package='`

model A text file located in the working directory, which defines the model. Models specified with explicit, ordinary or delay differential equations are expected to respect a certain syntax and organization detailed in `vignette('scaRabee',package='s`

iter A .csv file reporting the values of the objective function and estimates of model parameters at each iteration.

report A text file reporting for each individual in the dataset the final parameter estimates for structural model parameters, residual variability and secondary parameters as well as the related statistics (coefficients of variation, confidence intervals, covariance and correlation matrix).

pred A .csv file reporting the predictions and calculated residuals for each individual in the dataset.

est A .csv file reporting the final parameter estimates for each individual in the dataset.

sim A .csv file reporting the simulated model predictions for each individual in the dataset. (Not used for estimation runs).

Value

Return a modified version of `Fit`, containing the following elements:

estimations The vector of final parameter estimates.

fval The minimal value of the objective function.

cov The matrix of covariance for the parameter estimates.

orderedestimations A data.frame with the same structure as `problem$init` but only containing the sorted estimated estimates. The sorting is performed by `order.param.list`.

cor The upper triangle of the correlation matrix for the parameter estimates.

cv The coefficients of variations for the parameter estimates.

ci The confidence interval for the parameter estimates.

AIC The Akaike Information Criterion.

sec A list of data related to the secondary parameters, containing the following elements:

- **estimates** A vector of secondary parameter estimates.
- **cov** The matrix of covariance for the secondary parameter estimates.
- **cv** The coefficients of variations for the secondary parameter estimates.
- **ci** The confidence interval for the secondary parameter estimates.

orderedfixed A data.frame with the same structure as `problem$init` but only containing the sorted fixed estimates. The sorting is performed by `order.param.list`.

orderedinitial A data.frame with the same content as `problem$init` but sorted by `order.param.list`.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

See Also

`order.parms.list`,

`fitmle` *Maximum Likelihood Estimator*

Description

`fitmle` is a secondary function called during estimation runs. It performs the optimization of the model parameters by the method of the maximum likelihood, i.e. the minimization of an objective function defined as the exact negative log likelihood of the observed data, given the structural model, the model of residual variability, and the parameter estimates. This minimization is performed by the Nelder-Mead simplex algorithm implemented in `fminsearch` from the **neldermead** package. `fitmle` is typically not called directly by users.

Usage

```
fitmle(problem = NULL,
       estim.options = NULL,
       files = NULL)
```

Arguments

`problem` A list containing the following levels:

- **data** A list containing as many levels as there are treatment levels for the subject (or population) being evaluated, plus the `trts` level listing all treatments for this subject (or population), and the `id` level giving the identification number of the subject (or set to 1 if the analysis was run at the level of the population.
  Each treatment-specific level is a list containing the following levels:
  - **cov** mij x 3 data.frame containing the times of observations of the dependent variables (extracted from the TIME variable), the indicators of the type of dependent variables (extracted from the CMT variable), and the actual dependent variable observations (extracted from the DV variable) for this particular treatment.
  - **cov** mij x c data.frame containing the times of observations of the dependent variables (extracted from the TIME variable) and all the covariates identified for this particular treatment.
  - **bolus** bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.
  - **infusion** fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.
  - **trt** the particular treatment identifier.
- **method** A character string, indicating the scale of the analysis. Should be 'population' or 'subject'.
- **init** A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.
- **debugmode** Logical indicator of debugging mode.
- **modfun** Model function.

`estim.options` A list of estimation options containing two elements `maxiter` (the maximum number of iterations) and `maxeval` (the maximum number of function evaluations).

`files` A list of input used for the analysis. The following elements are expected and none of them could be null:

- **data** A .csv file located in the working directory, which contains the dosing information, the observations of the dependent variable(s) to be modeled, and possibly covariate information. The expected format of this file is described in details in `vignette('scaRabee', package='scaRabee')`.
- **param** A .csv file located in the working directory, which contains the initial guess(es) for the model parameter(s) to be optimized or used for model simulation. The expected format of this file is described in details in `vignette('scaRabee',package='`

model A text file located in the working directory, which defines the model. Models specified with explicit, ordinary or delay differential equations are expected to respect a certain syntax and organization detailed in `vignette('scaRabee',package='s`

iter A .csv file reporting the values of the objective function and estimates of model parameters at each iteration.

report A text file reporting for each individual in the dataset the final parameter estimates for structural model parameters, residual variability and secondary parameters as well as the related statistics (coefficients of variation, confidence intervals, covariance and correlation matrix).

pred A .csv file reporting the predictions and calculated residuals for each individual in the dataset.

est A .csv file reporting the final parameter estimates for each individual in the dataset.

sim A .csv file reporting the simulated model predictions for each individual in the dataset. (Not used for estimation runs).

Value

Return a list with two elements: `estimations` which contains the vector of final parameter estimates and `fval` the minimal value of the objective function.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

Pawel Wiczling

See Also

`fminsearch`

`fitmle.cov` *Computation of the Covariance Matrix*

Description

`fitmle.cov` is a secondary function called during estimation runs. It performs multiple tasks after completion of the model optimization by `fitmle`:

1- It computes the matrix of covariance (as described by D'Argenio and Schumitzky) by calling `get.cov.matrix` and derives some related statistics: correlation matrix, coefficient of variation of parameter estimates, confidence intervals and Akaike Information criterion,

2- It estimates secondary parameters and computes the coefficient of variation of those estimates, as well as the confidence intervals.

`fitmle.cov` is typically not called directly by users.

Usage

```
fitmle.cov(problem = NULL,
           Fit = NULL)
```

Arguments

`problem` A list containing the following levels:

- **data** A list containing as many levels as there are treatment levels for the subject (or population) being evaluated, plus the `trts` level listing all treatments for this subject (or population), and the `id` level giving the identification number of the subject (or set to 1 if the analysis was run at the level of the population.

  Each treatment-specific level is a list containing the following levels:

  - **cov** mij x 3 data.frame containing the times of observations of the dependent variables (extracted from the TIME variable), the indicators of the type of dependent variables (extracted from the CMT variable), and the actual dependent variable observations (extracted from the DV variable) for this particular treatment.
  - **cov** mij x c data.frame containing the times of observations of the dependent variables (extracted from the TIME variable) and all the covariates identified for this particular treatment.
  - **bolus** bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.
  - **infusion** fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.
  - **trt** the particular treatment identifier.
- **method** A character string, indicating the scale of the analysis. Should be 'population' or 'subject'.
- **init** A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.
- **debugmode** Logical indicator of debugging mode.
- **modfun** Model function.

`Fit` A list of containing the following levels:

- **estimations** The vector of final parameter estimates.
- **fval** The minimal value of the objective function.

Value

Return a list containing the following elements:

estimations The vector of final parameter estimates.

fval The minimal value of the objective function.

cov The matrix of covariance for the parameter estimates.

orderedestimations A data.frame with the same structure as `problem$init` but only containing the sorted estimated estimates. The sorting is performed by `order.param.list`.

- **cor** The upper triangle of the correlation matrix for the parameter estimates.
- **cv** The coefficients of variations for the parameter estimates.
- **ci** The confidence interval for the parameter estimates.
- **AIC** The Akaike Information Criterion.
- **sec** A list of data related to the secondary parameters, containing the following elements:
    - **estimates** The vector of secondary parameter estimates calculated using the initial estimates of the primary model parameters.
    - **estimates** The vector of secondary parameter estimates calculated using the final estimates of the primary model parameters.
    - **names** The vector of names of the secondary parameter estimates.
    - **pder** The matrix of partial derivatives for the secondary parameter estimates.
    - **cov** The matrix of covariance for the secondary parameter estimates.
    - **cv** The coefficients of variations for the secondary parameter estimates.
    - **ci** The confidence interval for the secondary parameter estimates.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

Pawel Wiczling

References

D.Z. D'Argenio and A. Schumitzky. ADAPT II User's Guide: Pharmacokinetic/ Pharmacodynamic Systems Analysis Software. Biomedical Simulations Resource, Los Angeles, 1997.

See Also

`fitmle`, `order.parms.list`

`get.cov.matrix` *Computation of the Covariance Matrix*

Description

`get.cov.matrix` is a secondary function called during estimation runs by `fitmle.cov`. It computes the covariance matrix for the parameter estimates. `get.cov.matrix` is typically not called directly by users.

Usage

```
get.cov.matrix(problem = NULL,
               Fit = NULL)
```

Arguments

`problem` A list containing the following levels:

- **data** A list containing as many levels as there are treatment levels for the subject (or population) being evaluated, plus the `trts` level listing all treatments for this subject (or population), and the `id` level giving the identification number of the subject (or set to 1 if the analysis was run at the level of the population.

  Each treatment-specific level is a list containing the following levels:

  - **cov** mij x 3 data.frame containing the times of observations of the dependent variables (extracted from the TIME variable), the indicators of the type of dependent variables (extracted from the CMT variable), and the actual dependent variable observations (extracted from the DV variable) for this particular treatment.
  - **cov** mij x c data.frame containing the times of observations of the dependent variables (extracted from the TIME variable) and all the covariates identified for this particular treatment.
  - **bolus** bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.
  - **infusion** fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.
  - **trt** the particular treatment identifier.
- **method** A character string, indicating the scale of the analysis. Should be 'population' or 'subject'.
- **init** A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.
- **debugmode** Logical indicator of debugging mode.
- **modfun** Model function.

`Fit` A list of containing the following levels:

- **estimations** The vector of final parameter estimates.
- **fval** The minimal value of the objective function.

Value

Return a list with the following elements:

covmatrix The matrix of covariance for the parameter estimates.

orderedestimations A data.frame with the same structure as `problem$init` but only containing the sorted estimated estimates. The sorting is performed by `order.param.list`.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

See Also

`fitmle.cov`

get.events *Create events from bolus dosing records.*

Description

`get.events` is a secondary function called by `dde.model`. It creates a data.frame of events from the bolus dosing records found in the dataset. `get.events` is typically not called directly by users.

Usage

```
get.events(bolus = NULL,
           scale = NULL)
```

Arguments

`bolus`	b x 4 data.frame providing the instantaneous inputs
`scale`	s x 1 vector of scaling factors

Value

Return a data.frame of events with the following elements:

var A name of the state affected by the event

time The time of the event

value The value associated with the event

method How the event affects the state ('add' by default)

See `events` for more details

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

See Also

codeevents

get.layout *Layout for Lattice Functions*

Description

get.layout is a utility function called by estimation.plot and simulation.plot. It provides a layout for **lattice** functions based upon a user-defined number of plots per page. get.layout is typically not called directly by users.

Usage

```
get.layout(nplot = NULL)
```

Arguments

nplot	A integer scalar defining the number of plots per page.

Value

Return a vector of two integers (nx,ny), where nx is the number of rows and ny the number of columns for the **lattice** layout.

Author(s)

Sebastien Bihorel (<sb.pmlab@gmail.com>)

See Also

estimation.plot, simulation.plot

Examples

```
get.layout(1)
get.layout(7)
## Not run: get.layout(1:5)
## Not run: get.layout(NA)
```

`get.parms.data` *Extract data from scaRabee parameter table*

Description

`get.parms.data` is a utility function in **scaRabee**. It allows to extract from a parameter table `x` all the data of a given type `which` for a set of parameter of type `type`. `get.parms.data` is typically not called directly by users.

Usage

```
  get.parms.data(x = NULL,
                 which = NULL,
                 type = NULL)
```

Arguments

`x`	A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.
`which`	A single string of characters, either 'names', 'type', 'value' or 'isfix'.
`type`	A single string of characters, either 'P', 'L', 'V' or 'IC'.

Value

Return as a vector the content of the `which` column of `x` corresponding to the `type` parameters.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

`get.secondary` *Computation of Secondary Parameter Estimates*

Description

`get.secondary` is a secondary function called during estimation runs by `fitmle.cov`. It computes estimates of secondary parameters and related statistics (covariance, coefficient of variations, and confidence intervals). `get.secondary` is typically not called directly by users.

Usage

```
get.secondary(subproblem = NULL,
              x = NULL)
```

Arguments

`subproblem` A list containing the following levels:

data A list containing as many levels as there are treatment levels for the subject (or population) being evaluated, plus the `trts` level listing all treatments for this subject (or population), and the `id` level giving the identification number of the subject (or set to 1 if the analysis was run at the level of the population.

Each treatment-specific level is a list containing the following levels:

- **cov** mij x 3 data.frame containing the times of observations of the dependent variables (extracted from the TIME variable), the indicators of the type of dependent variables (extracted from the CMT variable), and the actual dependent variable observations (extracted from the DV variable) for this particular treatment.
- **cov** mij x c data.frame containing the times of observations of the dependent variables (extracted from the TIME variable) and all the covariates identified for this particular treatment.
- **bolus** bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.
- **infusion** fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.
- **trt** the particular treatment identifier.

method A character string, indicating the scale of the analysis. Should be ’population’ or ’subject’.

init A data.frame of parameter data with the following columns: ’names’, ’type’, ’value’, ’isfix’, ’lb’, and ’ub’.

debugmode Logical indicator of debugging mode.

modfun Model function.

`x` The vector of *p* parameter estimates.

Value

Return a list of with the following elements:

init The vector of *s* secondary parameter estimates derived from initial structural model parameter estimates.

estimates The vector of *s* secondary parameter estimates derived from final structural model parameter estimates.

names The vector of *s* secondary parameter names.

pder The *p* x *s* matrix of partial derivatives for the secondary parameters.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

See Also

`fitmle.cov`

`initialize.report` *Initialize Report Files*

Description

`initialize.report` is a secondary function called during direct grid search or estimation runs.

For direct grid search runs, `initialize.report` creates the report file in the results & backup directory. This report file reports information about the run, including a summary table of the grid search which gives the value of the objective function for the different vector of estimates tested.

For estimation runs, `initialize.report` creates the log file and the report file in the results & backup directory. The log file stores, for each subject in the analysis, the values of the objective function and the parameter estimates at each iteration of the estimation process. The report file reports information about the estimation run, including the final estimates and some related statistics.

`initialize.report` is typically not called directly by users.

Usage

```
initialize.report(problem = NULL,
                  param = NULL,
                  files = NULL,
                  isgrid = 0)
```

Arguments

`problem` A list containing the following levels:

data A list which content depends on the scope of the analysis. If the analysis was run at the level of the subject, `data` contains as many levels as the number of subjects in the dataset, plus the `ids` level containing the vector of identification numbers of all subjects included in the analysis population. If the analysis was run at the level of the population, `data` contains only one level of data and `ids` is set to 1.
Each subject-specific level contains as many levels as there are treatment levels for this subject, plus the `trts` level listing all treatments for this subject, and the `id` level giving the identification number of the subject.
Each treatment-specific levels is a list containing the following levels:

- **cov** mij x 3 data.frame containing the times of observations of the dependent variables (extracted from the TIME variable), the indicators of the type of dependent variables (extracted from the CMT variable), and the actual dependent variable observations (extracted from the DV variable) for this particular treatment and this particular subject.
- **cov** mij x c data.frame containing the times of observations of the dependent variables (extracted from the TIME variable) and all the covariates identified for this particular treatment and this particular subject.
- **bolus** bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.

- **infusion** fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.
- **trt** the particular treatment identifier.

method A character string, indicating the scale of the analysis. Should be 'population' or 'subject'.

init A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.

debugmode Logical indicator of debugging mode.

modfun Model function.

`param` A data.frame containing the values of fixed and variable parameter estimates. Expected to contain the 'names', 'type', 'value', 'isfix', 'lb', and 'ub' columns.

`files` A list of input used for the analysis. The following elements are expected and none of them could be null:

data A .csv file located in the working directory, which contains the dosing information, the observations of the dependent variable(s) to be modeled, and possibly covariate information. The expected format of this file is described in details in `vignette('scaRabee', package='scaRabee')`.

param A .csv file located in the working directory, which contains the initial guess(es) for the model parameter(s) to be optimized or used for model simulation. The expected format of this file is described in details in `vignette('scaRabee',package='`

model A text file located in the working directory, which defines the model. Models specified with explicit, ordinary or delay differential equations are expected to respect a certain syntax and organization detailed in `vignette('scaRabee',package='s`

iter A .csv file reporting the values of the objective function and estimates of model parameters at each iteration.

report A text file reporting for each individual in the dataset the final parameter estimates for structural model parameters, residual variability and secondary parameters as well as the related statistics (coefficients of variation, confidence intervals, covariance and correlation matrix).

pred A .csv file reporting the predictions and calculated residuals for each individual in the dataset.

est A .csv file reporting the final parameter estimates for each individual in the dataset.

sim A .csv file reporting the simulated model predictions for each individual in the dataset. (Not used for estimation runs).

`isgrid` Indicator of direct grid search runs.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

ode.model *Ordinary Differential Equations*

Description

`ode.model` is the system evaluation function called when an $ODE block is detected in the model file, indicating that the model is defined by ordinary differential equations. `ode.model` is typically not called directly by users

Usage

```
ode.model(parms = NULL,
          derparms = NULL,
          code = NULL,
          bolus = NULL,
          infusion = NULL,
          xdata = NULL,
          covdata = NULL,
          issim = 0,
          check = FALSE,
          options = list(method='lsoda'))
```

Arguments

`parms`	A vector of primary parameters.
`derparms`	A list of derived parameters, specified in the $DERIVED block of code.
`code`	A list of R code extracted from the model file. Depending on content of the model file, the levels of this list could be: template, derived, lags, ode, dde, output, variance, and/or secondary.
`bolus`	A data.frame providing the instantaneous inputs entering the system of delay differential equations for the treatment and individual being evaluated.
`infusion`	A data.frame providing the zero-order inputs entering the system of delay differential equations for the treatment and individual being evaluated.
`xdata`	A vector of times at which the system is being evaluated.
`covdata`	A data.frame of covariate data for the treatment and individual being evaluated.
`issim`	An indicator for simulation or estimation runs.
`check`	An indicator whether checks should be performed to validate function inputs.
`options`	A list of differential equation solver options. Currently not modifiable by users.

Details

`ode.model` evaluates the model for each treatment of each individual contained in the dataset using several utility functions: `derived.parms`, `init.cond`, `input.scaling`, `make.dosing`, `init.update`, and `de.output`.

The actual evaluation of the system is performed by `ode` from the **deSolve** package.

Value

Returns a matrix of system predictions.

Author(s)

Sebastien Bihorel (<sb.pmlab@gmail.com>)

See Also

ode, init.cond, input.scaling, make.dosing, init.update, de.output

---

ode.utils *Utility Functions for Ordinary Differential Equation Systems*

Description

This is a collection of utility functions called by ode.model (and for some by dde.model when a model defined by ordinary (or delay) differential equations is evaluated. None of these functions is typically called directly by users.

Usage

```
ode.syst(t = NULL,
         y = NULL,
         parms = NULL,
         derparms = NULL,
         codeode = NULL,
         dosing = NULL,
         has.dosing = NULL,
         dose.states = NULL,
         covdata = NULL,
         scale = NULL,
         check = FALSE)
create.intervals(xdata = NULL,
                 bolus = NULL,
                 infusion = NULL)
de.output(f = NULL,
          parms = NULL,
          derparms = NULL,
          codeoutput = NULL,
          dosing = NULL,
          xdata = NULL,
          check = FALSE)
derived.parms(parms = NULL,
              covdata,
              codederiv = NULL,
              check = FALSE)
```

```
init.cond(parms = NULL,
          derparms = NULL,
          codeic = NULL,
          dosing = NULL,
          check = FALSE)
input.scaling(parms = NULL,
              derparms = NULL,
              codescale = NULL,
              ic = NULL,
              check = FALSE)
make.dosing(allparms = NULL,
            bolus = NULL,
            infusion = NULL,
            check = FALSE)
init.update(a = NULL,
            t = NULL,
            dosing = NULL,
            scale = NULL)
```

Arguments

`t`	A scalar or a vector of numerical time values.
`y`	A vector of system state values.
`parms`	A vector of primary parameters.
`derparms`	A list of derived parameters, specified in the $DERIVED block of code.
`codeode`	The content of the R code specified within the $ODE block in the model file.
`dosing`	A data.frame of dosing information created by `make.dosing` from instantaneous and zero-order inputs into the system and containing the following columns: **TIME** Dosing event times. **CMT** State where the input should be assigned to. **AMT** Amount that should be assigned to system `state` at the corresponding `TIME`. **RATE** Rate of input that should be assigned to system `CMT` at the corresponding `TIME`. See `vignette('scaRabee',package='scaRabee')` for more details about the interpolation of the input rate at time not specified in `dosing`. **TYPE** An indicator of the type of input. `TYPE` is set to 1 if the record in `dosing` correspond an original bolus input; it is set to 0 otherwise.
`has.dosing`	A logical variable, indicating whether any input has to be assigned to a system state.
`dose.states`	A vector of integers, indicating in which system state one or more doses have to be assigned to.
`covdata`	A matrix of covariate data extracted from the dataset.
`scale`	A vector of system scale, typically returned by `input.scaling`
`check`	An indicator whether checks should be performed to validate function inputs
`xdata`	A vector of times at which the system is being evaluated.

`bolus`	bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.
`infusion`	fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.
`f`	A matrix of time (first row) and system predictions. In the `de.output` function, the first row is deleted so that `f` has the same number of rows as in `dadt` defined in the $ODE or $DDE block.
`codeoutput`	The content of the R code specified within the $OUTPUT block in the model file.
`codederiv`	The content of the R code specified within the $DERIVED block in the model file.
`codeic`	The content of the R code specified within the $IC block in the model file.
`codescale`	The content of the R code specified within the $SCALE block in the model file.
`ic`	A vector of initial conditions, typically returned by `init.cond`
`allparms`	A vector of parameters (primary+derived).
`a`	A vector of system state values, similar to `y`.

Details

`ode.syst` is the function which actually evaluates the system of ordinary differential equations specified in the $ODE block.

`create.intervals` is a function that allows the overall integration interval to be split into sub-intervals based upon dosing history. This allows for the exact implementation of bolus inputs into the system. It determines the number of unique bolus dosing events there is by system state in `dosing`. It then creates the sub-intervals using these unique event times. If the first dosing events occurs after the first observation time, an initial sub-interval is added.

`de.output` is the function which evaluates the code specified in the $OUTPUT block and, thus, defines the output of the model.

`derived.parms` is the function which evaluates the code provided in the $DERIVED block and calculate derived parameters. It prevents primary parameters and covariates from being edited.

`init.cond` is the function which evaluates the code provided in the $IC block, and, thus, defines the initial conditions of the system.

`input.scaling` is the function which evaluates the code provided in the $SCALE block, and, thus, defines how system inputs are scaled.

`make.dosing` is the function which processes the instantaneous and zero-order inputs provided in the dataset and creates the `dosing` object. This function also implements absorption lags if the user included ALAGx parameters in `parms` or `derpamrs`.

`init.update` is a function that updates the system states at the beginning of each integration interval created by `create.intervals` to provide an exact implementation of bolus inputs into the system.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

See Also

`dde.model`

`order.parms.list` *Sort a scaRabee parameter table*

Description

`order.parms.list` is a secondary function called during estimation runs. It reorder a data.frame of initial parameter estimates by type: structural ('P'), delays ('L'), initial conditions ('IC'), and finally variance ('V'). `order.parms.list` is typically not called directly by users.

Usage

```
order.parms.list(x = NULL)
```

Arguments

`x`	A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.

Value

A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

`pder` *Compute Matrix of Partial Derivatives*

Description

`pder` is a secondary function called by `get.cov.matrix`. It computes the matrix of partial derivatives for the model predictions and the residual variability. `pder` is typically not called directly by users.

Usage

```
pder(subproblem = NULL,
     x = NULL)
```

Arguments

subproblem — A list containing the following levels:

code A list of R code extracted from the model file. Depending on content of the model file, the levels of this list could be: template, derived, lags, ode, dde, output, variance, and/or secondary.

method A character string, indicating the scale of the analysis. Should be 'population' or 'subject'.

init A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.

debugmode Logical indicator of debugging mode.

modfun Model function.

data A list containing the following levels:

- **xdata** 1 x m matrix of time of observations of the dependent variables.
- **data** m x 3 data.frame containing the times of observations of the dependent variables (extracted from the TIME variable), the indicators of the type of dependent variables (extracted from the CMT variable), and the actual dependent variable observations (extracted from the DV variable).

bolus bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.

infusion fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.

cov mij x c data.frame containing the times of observations of the dependent variables (extracted from the TIME variable) and all the covariates identified for this particular treatment.

x — The vector of *p* final parameter estimates.

Value

Return a list containing the *p* x *p* matrices of partial derivatives for model predictions (`mpder`) and residual variability (`wpder`).

Author(s)

Sebastien Bihorel (<sb.pmlab@gmail.com>)

See Also

get.cov.matrix

`problem.eval` *Evaluation of structural and residual variability models*

Description

`problem.eval` is a secondary function called during estimation runs. It evaluates the structural model and the residual variability model at given point estimates and at given values of the time variable. `problem.eval` is typically not called directly by users.

Usage

```
problem.eval(subproblem = NULL,
             x = NULL,
             grid = FALSE,
             check = FALSE)
```

Arguments

`subproblem` A list containing the following levels:

- **code** A list of R code extracted from the model file. Depending on content of the model file, the levels of this list could be: template, derived, lags, ode, dde, output, variance, and/or secondary.
- **method** A character string, indicating the scale of the analysis. Should be 'population' or 'subject'.
- **init** A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.
- **debugmode** Logical indicator of debugging mode.
- **modfun** Model function.
- **data** A list containing the following levels:
  - **xdata** 1 x m matrix of time of observations of the dependent variables.
  - **data** m x 3 data.frame containing the times of observations of the dependent variables (extracted from the TIME variable), the indicators of the type of dependent variables (extracted from the CMT variable), and the actual dependent variable observations (extracted from the DV variable).
- **bolus** bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.
- **infusion** fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.
- **cov** mij x c data.frame containing the times of observations of the dependent variables (extracted from the TIME variable) and all the covariates identified for this particular treatment.

`x` A vector of numerical estimates of numerical parameters.

`grid` A logical variable, indicating whether the analysis is a direct grid search or not.

`check` An indicator whether checks should be performed to validate function inputs.

Value

Return a list of two elements:

f A vector of model evaluations at all requested time points (all states values are concatenated into a single vector).

weight A vector of residual variability related to the model evaluations.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

See Also

`fitmle`

`residual.report` *Creation of Prediction & Residual Report*

Description

`residual.report` is a secondary function called at the end of the estimations runs. It creates a file containing the predictions, residuals and weighted residuals at all observation time points. `residual.report` is typically not called directly by users.

Usage

```
residual.report(problem = NULL,
                Fit = NULL,
                files = NULL)
```

Arguments

`problem` A list containing the following levels:

code A list of R code extracted from the model file. Depending on content of the model file, the levels of this list could be: template, derived, lags, ode, dde, output, variance, and/or secondary.

method A character string, indicating the scale of the analysis. Should be 'population' or 'subject'.

init A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.

debugmode Logical indicator of debugging mode.

modfun Model function.

data A list which content depends on the scope of the analysis. If the analysis was run at the level of the subject, `data` contains as many levels as the number of subjects in the dataset, plus the `ids` level containing the vector of identification numbers of all subjects included in the analysis population. If the analysis was run at the level of the population, `data` contains only one level of data and `ids` is set to 1.

Each subject-specific level contains as many levels as there are treatment levels for this subject, plus the `trts` level listing all treatments for this subject, and the `id` level giving the identification number of the subject.

Each treatment-specific levels is a list containing the following levels:

- **cov** mij x 3 data.frame containing the times of observations of the dependent variables (extracted from the TIME variable), the indicators of the type of dependent variables (extracted from the CMT variable), and the actual dependent variable observations (extracted from the DV variable) for this particular treatment and this particular subject.
- **cov** mij x c data.frame containing the times of observations of the dependent variables (extracted from the TIME variable) and all the covariates identified for this particular treatment and this particular subject.
- **bolus** bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.
- **infusion** fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.
- **trt** the particular treatment identifier.

`Fit` A list containing the following elements:

estimations The vector of final parameter estimates.

fval The minimal value of the objective function.

cov The matrix of covariance for the parameter estimates.

orderedestimations A data.frame with the same structure as `problem$init` but only containing the sorted estimated estimates. The sorting is performed by `order.param.list`.

cor The upper triangle of the correlation matrix for the parameter estimates.

cv The coefficients of variations for the parameter estimates.

ci The confidence interval for the parameter estimates.

AIC The Akaike Information Criterion.

sec A list of data related to the secondary parameters, containing the following elements:

- **estimates** The vector of secondary parameter estimates calculated using the initial estimates of the primary model parameters.
- **estimates** The vector of secondary parameter estimates calculated using the final estimates of the primary model parameters.
- **names** The vector of names of the secondary parameter estimates.
- **pder** The matrix of partial derivatives for the secondary parameter estimates.
- **cov** The matrix of covariance for the secondary parameter estimates.
- **cv** The coefficients of variations for the secondary parameter estimates.

ci The confidence interval for the secondary parameter estimates.

orderedfixed A data.frame with the same structure as `problem$init` but only containing the sorted fixed estimates. The sorting is performed by `order.param.list`.

orderedinitial A data.frame with the same content as `problem$init` but sorted by `order.param.list`.

`files` A list of input used for the analysis. The following elements are expected and none of them could be null:

data A .csv file located in the working directory, which contains the dosing information, the observations of the dependent variable(s) to be modeled, and possibly covariate information. The expected format of this file is described in details in `vignette('scaRabee', package='scaRabee')`.

param A .csv file located in the working directory, which contains the initial guess(es) for the model parameter(s) to be optimized or used for model simulation. The expected format of this file is described in details in `vignette('scaRabee',package='`

model A text file located in the working directory, which defines the model. Models specified with explicit, ordinary or delay differential equations are expected to respect a certain syntax and organization detailed in `vignette('scaRabee',package='s`

iter A .csv file reporting the values of the objective function and estimates of model parameters at each iteration.

report A text file reporting for each individual in the dataset the final parameter estimates for structural model parameters, residual variability and secondary parameters as well as the related statistics (coefficients of variation, confidence intervals, covariance and correlation matrix).

pred A .csv file reporting the predictions and calculated residuals for each individual in the dataset.

est A .csv file reporting the final parameter estimates for each individual in the dataset.

sim A .csv file reporting the simulated model predictions for each individual in the dataset. (Not used for estimation runs).

Value

Creates the prediction and residual report in the run directory.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

`scarabee.analysis` *Run a scaRabee Analysis*

Description

`scarabee.analysis` is the *de facto* gateway for running any kind of analysis with **scaRabee**. All other functions distributed with this package are secondary functions called directly or indirectly by `scarabee.analysis`.

Arguments for `scarabee.analysis` are best defined using the template distributed with the package.

Usage

```
scarabee.analysis(files = NULL,
                  method = 'population',
                  runtype = NULL,
                  debugmode = FALSE,
                  estim.options = NULL,
                  npts = NULL,
                  alpha = NULL,
                  solver.options = list(method='lsoda'))
```

Arguments

`files`	A list of input used for the analysis. The following elements are expected and none of them could be null: **data** A .csv file located in the working directory, which contains the dosing information, the observations of the dependent variable(s) to be modeled, and possibly covariate information. The expected format of this file is described in details in `vignette('scaRabee', package='scaRabee')`. **param** A .csv file located in the working directory, which contains the initial guess(es) for the model parameter(s) to be optimized or used for model simulation. The expected format of this file is described in details in `vignette('scaRabee',package='` **model** A text file located in the working directory, which defines the model. Models specified with explicit, ordinary or delay differential equations are expected to respect a certain syntax and organization detailed in `vignette('scaRabee',package='s`
`method`	A character string, indicating the scale of the analysis. Should be 'population' or 'subject'.
`runtype`	A character string, indicating the type of analysis. Should be 'simulation', 'estimation', or 'gridsearch'.
`debugmode`	A logical value, indicating the debug mode should be turn on (`TRUE`) or off (default). Used only for estimation runs. If turn on, the user could have access to error message returned when the model and residual variability are evaluated in `fitmle` before the likelihood is computed.
`estim.options`	A list of estimation options containing two elements `maxiter` (the maximum number of iterations) and `maxeval` (the maximum number of function evaluations).
`npts`	Only necessary if `runtype` is set to 'gridsearch'; `npts` represents the number of points to be created by dimension of the grid search.

`alpha`	Only necessary if `runtype` is set to 'gridsearch'; `alpha` is a real number, representing the factor applied to the initial estimates of the model parameters to determine the lower and upper bounds to the grid search space.
`solver.options`	A list of differential equation solver options. Currently not modifiable by users.

Value

Run an analysis until completion. See `vignette('scaRabee',package='scaRabee')` for more details about the expected outputs for an estimation, a simulation, or a gird search run.

Author(s)

Sebastien Bihorel (`<sb.pmlab@gmail.com>`)

See Also

`fitmle`

`scarabee.check.model` *Check scaRabee Models*

Description

`scarabee.check.model` is a secondary function called at each **scaRabee** estimation run. It runs a first set of model evaluation at the initial parameter estimates turning all checks on. If all checks are passed, the estimation starts with all checks turned off. `scarabee.check.model` is typically not called directly by users.

Usage

```
scarabee.check.model(problem = NULL,
                     files = NULL)
```

Arguments

`problem` A list containing the following levels:

code A list of R code extracted from the model file. Depending on content of the model file, the levels of this list could be: template, derived, lags, ode, dde, output, variance, and/or secondary.

method A character string, indicating the scale of the analysis. Should be 'population' or 'subject'.

init A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.

debugmode Logical indicator of debugging mode.

modfun Model function.

data A list which content depends on the scope of the analysis. If the analysis was run at the level of the subject, `data` contains as many levels as the number of subjects in the dataset, plus the `ids` level containing the vector of identification numbers of all subjects included in the analysis population. If the analysis was run at the level of the population, `data` contains only one level of data and `ids` is set to 1.

Each subject-specific level contains as many levels as there are treatment levels for this subject, plus the `trts` level listing all treatments for this subject, and the `id` level giving the identification number of the subject.

Each treatment-specific levels is a list containing the following levels:

- **cov** mij x 3 data.frame containing the times of observations of the dependent variables (extracted from the TIME variable), the indicators of the type of dependent variables (extracted from the CMT variable), and the actual dependent variable observations (extracted from the DV variable) for this particular treatment and this particular subject.
- **cov** mij x c data.frame containing the times of observations of the dependent variables (extracted from the TIME variable) and all the covariates identified for this particular treatment and this particular subject.
- **bolus** bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.
- **infusion** fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.
- **trt** the particular treatment identifier.

`files` A list of input used for the analysis. The following elements are expected and none of them could be null:

- **data** A .csv file located in the working directory, which contains the dosing information, the observations of the dependent variable(s) to be modeled, and possibly covariate information. The expected format of this file is described in details in `vignette('scaRabee', package='scaRabee')`.
- **param** A .csv file located in the working directory, which contains the initial guess(es) for the model parameter(s) to be optimized or used for model simulation. The expected format of this file is described in details in `vignette('scaRabee',package='`
- **model** A text file located in the working directory, which defines the model. Models specified with explicit, ordinary or delay differential equations are expected to respect a certain syntax and organization detailed in `vignette('scaRabee',package='s`
- **iter** A .csv file reporting the values of the objective function and estimates of model parameters at each iteration.
- **report** A text file reporting for each individual in the dataset the final parameter estimates for structural model parameters, residual variability and secondary parameters as well as the related statistics (coefficients of variation, confidence intervals, covariance and correlation matrix).
- **pred** A .csv file reporting the predictions and calculated residuals for each individual in the dataset.
- **est** A .csv file reporting the final parameter estimates for each individual in the dataset.
- **sim** A .csv file reporting the simulated model predictions for each individual in the dataset. (Not used for estimation runs).

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

`scarabee.check.reserved` *Check for Reserved Variable Names*

Description

`scarabee.check.reserved` is a secondary function called at each **scaRabee** run. It determined whether user-defined parameter names use reserved names and if some user-defined parameters are related to the dosing history. `scarabee.check.reserved` is typically not called directly by users. `scarabee.check.reserved` is typically not called directly by users.

Usage

```
scarabee.check.reserved(names = NULL, covnames = NULL)
```

Arguments

`names`	A vector of parameter names, typically extracted from the file of parameter definition.
`covnames`	A vector of covariate names, typically extracted from the data file.

Details

If one or more user-defined parameters are found to use reserved names, the run is stopped and the user is ask to update the name(s) of this(ese) parameter(s).

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

`scarabee.clean` *Cleaning of the Run Directory*

Description

`scarabee.clean` is a secondary function called at each **scaRabee** run. It cleans the run directory from unwanted files. `scarabee.clean` is typically not called directly by users.

Usage

```
scarabee.clean(files = NULL,
               analysis = NULL)
```

Arguments

`files` A list of input used for the analysis. The following elements are expected and none of them could be null:

- **data** A .csv file located in the working directory, which contains the dosing information, the observations of the dependent variable(s) to be modeled, and possibly covariate information. The expected format of this file is described in details in `vignette('scaRabee', package='scaRabee')`.
- **param** A .csv file located in the working directory, which contains the initial guess(es) for the model parameter(s) to be optimized or used for model simulation. The expected format of this file is described in details in `vignette('scaRabee',package='`
- **model** A text file located in the working directory, which defines the model. Models specified with explicit, ordinary or delay differential equations are expected to respect a certain syntax and organization detailed in `vignette('scaRabee',package='s`
- **iter** A .csv file reporting the values of the objective function and estimates of model parameters at each iteration.
- **report** A text file reporting for each individual in the dataset the final parameter estimates for structural model parameters, residual variability and secondary parameters as well as the related statistics (coefficients of variation, confidence intervals, covariance and correlation matrix).
- **pred** A .csv file reporting the predictions and calculated residuals for each individual in the dataset.
- **est** A .csv file reporting the final parameter estimates for each individual in the dataset.
- **sim** A .csv file reporting the simulated model predictions for each individual in the dataset. (Not used for estimation runs).

`analysis` A character string directly following the $ANALYSIS tag in the model file.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

See Also

`scarabee.analysis`

`scarabee.directory` *Creation of the Run Directory*

Description

`scarabee.directory` is a secondary function called at each **scaRabee** run. It creates a directory to store the results of the run and a sub-directory to backup all files used for the run. This directory is referred to as the 'run directory' in all **scaRabee** documentation and help. `scarabee.directory` is typically not called directly by users.

Usage

```
scarabee.directory(curwd = getwd(),
                   files = NULL,
                   runtype = NULL,
                   analysis = NULL)
```

Arguments

`curwd`	The current working directory.
`files`	A list of input used for the analysis. The following elements are expected and none of them could be null: **data** A .csv file located in the working directory, which contains the dosing information, the observations of the dependent variable(s) to be modeled, and possibly covariate information. The expected format of this file is described in details in `vignette('scaRabee', package='scaRabee')`. **param** A .csv file located in the working directory, which contains the initial guess(es) for the model parameter(s) to be optimized or used for model simulation. The expected format of this file is described in details in `vignette('scaRabee',package='` **model** A text file located in the working directory, which defines the model. Models specified with explicit, ordinary or delay differential equations are expected to respect a certain syntax and organization detailed in `vignette('scaRabee',package='s` **iter** A .csv file reporting the values of the objective function and estimates of model parameters at each iteration. (Not used for simulation runs). **report** A text file reporting for each individual in the dataset the final parameter estimates for structural model parameters, residual variability and secondary parameters as well as the related statistics (coefficients of variation, confidence intervals, covariance and correlation matrix). (Not used for simulation runs). **pred** A .csv file reporting the predictions and calculated residuals for each individual in the dataset. (Not used for simulation runs). **est** A .csv file reporting the final parameter estimates for each individual in the dataset. (Not used for simulation runs). **sim** A .csv file reporting the simulated model predictions for each individual in the dataset. (Not used for estimation runs).
`runtype`	A character string, indicating the type of analysis. Should be 'simulation', 'estimation', or 'gridsearch'.
`analysis`	A character string directly following the $ANALYSIS tag in the model file.

Value

When `scarabee.directory` is called, a new folder is created in the working directory. The name of the new folder is a combination of the string directly following the $ANALYSIS tag in the model file, an abbreviation of the type of run ('est' for estimation, 'sim' for simulation, or 'grid' for grid search) and an incremental integer, e.g. 'test.est.01'. This directory contains the text and graph outputs of the run.

Additionally, a sub-directory called `run.config.files` is created into the new folder and all the files defining the run, i.e. the dataset, the file of initial model parameters, the model file and the master R script), are stored.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

See Also

`scarabee.analysis`

`scarabee.gridsearch` *Direct Grid Search Utility*

Description

`scarabee.gridsearch` is a secondary function called during direct grid search runs. It creates a matrix made of unique vectors of parameter estimates set around the vector of initial estimates and evaluates the objective function (i.e. minus twice the log of the exact likelihood of the observed data, given the structural model, the model of residual variability, and the vector of parameter estimates) at each of those vectors at the population level. The grid of objective function values is then sorted and the best vector is used to simulate the model at the population level. `scarabee.gridsearch` is typically not called directly by users.

Usage

```
scarabee.gridsearch(problem = NULL,
                    npts = NULL,
                    alpha = NULL,
                    files = NULL)
```

Arguments

`problem` A list containing the following levels:

data A list which content depends on the scope of the analysis. If the analysis was run at the level of the subject, `data` contains as many levels as the number of subjects in the dataset, plus the `ids` level containing the vector of identification numbers of all subjects included in the analysis population. If the analysis was run at the level of the population, `data` contains only one level of data and `ids` is set to 1.
Each subject-specific level contains as many levels as there are treatment levels for this subject, plus the `trts` level listing all treatments for this subject, and the `id` level giving the identification number of the subject.
Each treatment-specific levels is a list containing the following levels:

cov mij x 3 data.frame containing the times of observations of the dependent variables (extracted from the TIME variable), the indicators of the type of dependent variables (extracted from the CMT variable), and the actual dependent variable observations (extracted from the DV variable) for this particular treatment and this particular subject.

cov mij x c data.frame containing the times of observations of the dependent variables (extracted from the TIME variable) and all the covariates identified for this particular treatment and this particular subject.

bolus bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.

infusion fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.

trt the particular treatment identifier.

method A character string, indicating the scale of the analysis. Should be 'population' or 'subject'.

init A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.

debugmode Logical indicator of debugging mode.

modfun Model function.

`npts` An integer greater than 2, defining the number of points that the grid should contain per dimension (i.e variable model parameter).

`alpha` A vector of numbers greater than 1, which give the factor(s) used to calculate the evaluation range of each dimension of the search grid (see Details). If `alpha` length is lower than the number of variable parameters, elements of `alpha` are recycled. If its length is higher than number of variable parameters, `alpha` is truncated.

`files` A list of input used for the analysis. The following elements are expected and none of them could be null:

data A .csv file located in the working directory, which contains the dosing information, the observations of the dependent variable(s) to be modeled, and possibly covariate information. The expected format of this file is described in details in `vignette('scaRabee', package='scaRabee')`.

param A .csv file located in the working directory, which contains the initial guess(es) for the model parameter(s) to be optimized or used for model simulation. The expected format of this file is described in details in `vignette('scaRabee',package='`

model A text file located in the working directory, which defines the model. Models specified with explicit, ordinary or delay differential equations are expected to respect a certain syntax and organization detailed in `vignette('scaRabee',package='s`

iter A .csv file reporting the values of the objective function and estimates of model parameters at each iteration. (Not used for direct grid search runs).

report A text file reporting the summary tables of ordered objective function values for the various tested vectors of model parameters.

pred A .csv file reporting the predictions and calculated residuals for each individual in the dataset. (Not used for direct grid search runs).

est A .csv file reporting the final parameter estimates for each individual in the dataset. (Not used for direct grid search runs).

sim A .csv file reporting the simulated model predictions for each individual in the dataset. (Not used for direct grid search runs).

Details

The actual creation of the grid and the evaluation of the objective function is delegated by `scarabee.gridsearch` to the `fmin.gridsearch` function of the **neldermead** package.

This function evaluates the cost function - that is, in the present case, the objective function - at each point of a grid of `npts^length(x0)` points, where `x0` is the vector of model parameters set as variable. If `alpha` is NULL, the range of the evaluation points is limited by the lower and upper bounds of each parameter of `x0` provided in the parameter file. If `alpha` is not NULL, the range of the evaluation points is defined as `[x0/alpha,x0*alpha]`.

Because `fmin.gridsearch` can be applied to the evaluation of constrained systems, it also assesses the feasibility of the cost function at each point of the grid (i.e. whether or not the points satisfy the defined constraints). In the context of scaRabee, the objective function is always feasible.

Value

Return a data.frame with pe+2 columns. The last 2 columns report the value and the feasibility of the objective function at each specific vector of parameter estimates which is documented in the first pe columns. This data.frame is ordered by feasibility and increasing value of the objective function.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

See Also

`fmin.gridsearch`

`scarabee.new` *Create a scaRabee Analysis Folder*

Description

`scarabee.new` creates a new **scaRabee** analysis folder.

Usage

```
scarabee.new(name = 'myanalysis',
             path = NULL,
             type = 'simulation',
             method = 'population',
             template = 'ode')
```

Arguments

`name`	A string of characters defining the name of the master analysis script; `name` is also appended to the $ANALYSIS tag in the model file. Default is 'myanalysis'.
`path`	A path where the analysis files must be created. The path must not yet exist.
`type`	A string of characters, either 'simulation', 'estimation', or 'gridsearch'. Default is 'simulation'.
`method`	A string of characters, either 'population' or 'subject'. Default is 'population'.
`template`	A string of characters, either 'explicit', 'ode' or 'dde'. Default is 'ode'.

Details

The content of new **scaRabee** analysis folder `path/` is:

`name`**.R** The template-based **scaRabee** analysis script.

model.txt A template-based txt file for the definition of the structural model. Depending on `template`, this text file contains various tags which delimit blocks of R code needed when models are defined with closed form solution ('explicit'), ordinary differential equations ('ode') or delay differential equations ('dde').

data.csv (optional) An empty comma-separated file for dosing, observations, and covariate information; contains the following default headers: OMIT, TRT, ID, TIME, AMT, RATE, CMT, EVID, DV, DVID, and MDV.

initials.csv (optional) An empty comma-separated file for initial guesses of model parameter estimates; contains the following default headers: Parameter, Type, Value, Fixed, Lower bound, Upper bound.

If the `path` argument is NULL, then it is coerced to `name`, thus creating a new folder in the current working directory.

See `vignette('scaRabee',package='scaRabee')` to learn about how to specify your model based on those template files.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

`scarabee.read.data` *Read scaRabee Data File*

Description

`scarabee.read.data` is a secondary function called at each **scaRabee** run. It reads and processes the data contained in the specified data file.`scarabee.read.data` is typically not called directly by users.

Usage

```
scarabee.read.data(files = NULL,
                   method = NULL)
```

Arguments

`files` A list of input used for the analysis. The following elements are expected and none of them could be null:

data A .csv file located in the working directory, which contains the dosing information, the observations of the dependent variable(s) to be modeled, and possibly covariate information. The expected format of this file is described in details in `vignette('scaRabee', package='scaRabee')`.

param A .csv file located in the working directory, which contains the initial guess(es) for the model parameter(s) to be optimized or used for model simulation. The expected format of this file is described in details in `vignette('scaRabee',package='`

model A text file located in the working directory, which defines the model. Models specified with explicit, ordinary or delay differential equations are expected to respect a certain syntax and organization detailed in `vignette('scaRabee',package='s`

iter A .csv file reporting the values of the objective function and estimates of model parameters at each iteration. (Not used for simulation runs).

report A text file reporting for each individual in the dataset the final parameter estimates for structural model parameters, residual variability and secondary parameters as well as the related statistics (coefficients of variation, confidence intervals, covariance and correlation matrix). (Not used for simulation runs).

pred A .csv file reporting the predictions and calculated residuals for each individual in the dataset. (Not used for simulation runs).

est A .csv file reporting the final parameter estimates for each individual in the dataset. (Not used for simulation runs).

sim A .csv file reporting the simulated model predictions for each individual in the dataset. (Not used for estimation runs).

`method` A character string, indicating the scale of the analysis. Should be 'population' or 'subject'.

Value

Return a list with 2 levels:

data A list which content depends on the scope of the analysis. If the analysis was run at the level of the subject, `data` contains as many levels as the number of subjects in the dataset, plus the `ids` level containing the vector of identification numbers of all subjects included in the analysis population. If the analysis was run at the level of the population, `data` contains only one level of data and `ids` is set to 1.

Each subject-specific level contains as many levels as there are treatment levels for this subject, plus the `trts` level listing all treatments for this subject, and the `id` level giving the identification number of the subject.

Each treatment-specific levels is a list containing the following levels:

- **cov** mij x 3 data.frame containing the times of observations of the dependent variables (extracted from the TIME variable), the indicators of the type of dependent variables (extracted from the CMT variable), and the actual dependent variable observations (extracted from the DV variable) for this particular treatment and this particular subject.
- **cov** mij x c data.frame containing the times of observations of the dependent variables (extracted from the TIME variable) and all the covariates identified for this particular treatment and this particular subject.
- **bolus** bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.
- **infusion** fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.
- **trt** the particular treatment identifier.

new A logical indicator defining whether or not a modified data file has been created based upon the original file. This is the case if and if only the time of first data record for one or more individuals in the original data file is not 0. The new data file is created such as the TIME variable is modified so that time of the first data record for all individuals is 0; the time of later records is modified accordingly.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

See Also

`scarabee.analysis`

`scarabee.read.model` *Read scaRabee Model File*

Description

`scarabee.read.model` is a secondary function called at each **scaRabee** run. It reads and processes the data contained in the specified model file. This function performs numerous checks to ensure that the model file contains the expected information. `scarabee.read.model` is typically not be called directly by users.

Usage

```
scarabee.read.model(files = NULL,
                    runtype = NULL)
```

Arguments

`files` A list of input used for the analysis. The following elements are expected and none of them could be null:

- **data** A .csv file located in the working directory, which contains the dosing information, the observations of the dependent variable(s) to be modeled, and possibly covariate information. The expected format of this file is described in details in `vignette('scaRabee', package='scaRabee')`.

- **param** A .csv file located in the working directory, which contains the initial guess(es) for the model parameter(s) to be optimized or used for model simulation. The expected format of this file is described in details in `vignette('scaRabee',package='`
- **model** A text file located in the working directory, which defines the model. Models specified with explicit, ordinary or delay differential equations are expected to respect a certain syntax and organization detailed in `vignette('scaRabee',package='s`
- **iter** A .csv file reporting the values of the objective function and estimates of model parameters at each iteration. (Not used for simulation runs).
- **report** A text file reporting for each individual in the dataset the final parameter estimates for structural model parameters, residual variability and secondary parameters as well as the related statistics (coefficients of variation, confidence intervals, covariance and correlation matrix). (Not used for simulation runs).
- **pred** A .csv file reporting the predictions and calculated residuals for each individual in the dataset. (Not used for simulation runs).
- **est** A .csv file reporting the final parameter estimates for each individual in the dataset. (Not used for simulation runs).
- **sim** A .csv file reporting the simulated model predictions for each individual in the dataset. (Not used for estimation runs).

`runtype`	A single character string, indicating the type of run; either 'estimation', 'simulation', or 'gridsearch'.

Value

Return a list with various levels, depending on the content of the model file and the type of run:

- **name** The character string which follows the $ANALYSIS tag.
- **template** A character string, indicating which scaRabee function needs to be called to evaluate the model. This level is set to 'ode.model' if the $ODE tag is detected, to 'dde.model' if the $DDE tag is detected, and to 'explicit.model' otherwise.
- **derived** A character string containing the R code provided within the $DERIVED block. (Only for models defined with $ODE or $DDE).
- **ic** A character string containing the R code provided within the $IC block. (Only for models defined with $ODE or $DDE).
- **scale** A character string containing the R code provided within the $SCALE block. (Only for models defined with $ODE or $DDE).
- **de** A character string containing the R code provided within the $ODE or $DDE block. (Only for models defined with $ODE or $DDE).
- **lags** A character string containing the R code provided within the $LAGS block. (Only for models defined with $DDE).
- **output** A character string containing the R code provided within the $OUTPUT block.
- **var** A character string containing the R code provided within the $VARIANCE block.
- **sec** A character string containing the R code provided within the $SECONDARY block.

Author(s)

Sebastien Bihorel (<sb.pmlab@gmail.com>)

See Also

`scarabee.analysis`

`scarabee.read.parms` *Read scaRabee Parameter File*

Description

`scarabee.read.parms` is a secondary function called at each **scaRabee** run. It reads and checks the information contained in the specified parameter file.`scarabee.read.parms` is typically not called directly by users.

Usage

```
scarabee.read.parms(files = NULL)
```

Arguments

`files` A list of input used for the analysis. The following elements are expected and none of them could be null:

data A .csv file located in the working directory, which contains the dosing information, the observations of the dependent variable(s) to be modeled, and possibly covariate information. The expected format of this file is described in details in `vignette('scaRabee', package='scaRabee')`.

param A .csv file located in the working directory, which contains the initial guess(es) for the model parameter(s) to be optimized or used for model simulation. The expected format of this file is described in details in `vignette('scaRabee',package='`

model A text file located in the working directory, which defines the model. Models specified with explicit, ordinary or delay differential equations are expected to respect a certain syntax and organization detailed in `vignette('scaRabee',package='s`

iter A .csv file reporting the values of the objective function and estimates of model parameters at each iteration. (Not used for simulation runs).

report A text file reporting for each individual in the dataset the final parameter estimates for structural model parameters, residual variability and secondary parameters as well as the related statistics (coefficients of variation, confidence intervals, covariance and correlation matrix). (Not used for simulation runs).

pred A .csv file reporting the predictions and calculated residuals for each individual in the dataset. (Not used for simulation runs).

est A .csv file reporting the final parameter estimates for each individual in the dataset. (Not used for simulation runs).

sim A .csv file reporting the simulated model predictions for each individual in the dataset. (Not used for estimation runs).

Value

If the content of the parameter file is valid, `scarabee.read.parms` returns a data.frame with the same content. The names of the data.frame columns are: names, type, value, isfix, lb, and ub.

Author(s)

Sebastien Bihorel (<`sb.pmlab@gmail.com`>)

See Also

`scarabee.analysis`

`simulation.plot` *Create Simulation Plots*

Description

`simulation.plot` is a secondary function called at the end of the simulation runs. It generates overlay plots of model predictions and observations for all the output system states and for each subject in the analysis. If the analysis is run at the population level, only one set of plots is generated. See `vignette('scaRabee',package='scaRabee')` for more details. `simulation.plot` is typically not called directly by users.

Usage

```
simulation.plot(problem = NULL,
                simdf = NULL,
                files = NULL)
```

Arguments

`problem` A list containing the following levels:

code A list of R code extracted from the model file. Depending on content of the model file, the levels of this list could be: template, derived, lags, ode, dde, output, variance, and/or secondary.

data A list which content depends on the scope of the analysis. If the analysis was run at the level of the subject, `data` contains as many levels as the number of subjects in the dataset, plus the `ids` level containing the vector of identification numbers of all subjects included in the analysis population. If the analysis was run at the level of the population, `data` contains only one level of data and `ids` is set to 1.
Each subject-specific level contains as many levels as there are treatment levels for this subject, plus the `trts` level listing all treatments for this subject, and the `id` level giving the identification number of the subject.
Each treatment-specific levels is a list containing the following levels:

cov mij x 3 data.frame containing the times of observations of the dependent variables (extracted from the TIME variable), the indicators of the type of dependent variables (extracted from the CMT variable), and the actual dependent variable observations (extracted from the DV variable) for this particular treatment and this particular subject.

cov mij x c data.frame containing the times of observations of the dependent variables (extracted from the TIME variable) and all the covariates identified for this particular treatment and this particular subject.

bolus bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.

infusion fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.

trt the particular treatment identifier.

method A character string, indicating the scale of the analysis. Should be 'population' or 'subject'.

init A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.

debugmode Logical indicator of debugging mode.

modfun Model function.

`simdf` A data.frame of simulated and observed data typically created by `simulation.report` and containing the following columns:

ID Subject Identifier. If the analysis is run at the population level and if the original dataset contained multiple subjects distinguished by a different ID number, please note that the original ID is lost and replaced by 1 so that all available data is considered to come from the same subject.

TRT Indicator of treatment level (defining the sub-problems).

CMT Indicator of system state to which the simulated or observed value is associated.

TIME Time of the observation or model prediction.

SIM Value of the simulated state. NA if DV is not NA.

DV Value of the observed state. NA if SIM is not NA.

`files` A list of input used for the analysis. The following elements are expected and none of them could be null:

data A .csv file located in the working directory, which contains the dosing information, the observations of the dependent variable(s) to be modeled, and possibly covariate information. The expected format of this file is described in details in `vignette('scaRabee', package='scaRabee')`.

param A .csv file located in the working directory, which contains the initial guess(es) for the model parameter(s) to be optimized or used for model simulation. The expected format of this file is described in details in `vignette('scaRabee',package='`

model A text file located in the working directory, which defines the model. Models specified with explicit, ordinary or delay differential equations are expected to respect a certain syntax and organization detailed in `vignette('scaRabee',package='s`

iter A .csv file reporting the values of the objective function and estimates of model parameters at each iteration. (Not used for simulation runs).

report A text file reporting for each individual in the dataset the final parameter estimates for structural model parameters, residual variability and secondary parameters as well as the related statistics (coefficients of variation, confidence intervals, covariance and correlation matrix). (Not used for simulation runs).

pred A .csv file reporting the predictions and calculated residuals for each individual in the dataset. (Not used for simulation runs).

est A .csv file reporting the final parameter estimates for each individual in the dataset. (Not used for simulation runs).

sim A .csv file reporting the simulated model predictions for each individual in the dataset. (Not used for estimation runs).

Author(s)

Sebastien Bihorel (`<sb.pmlab@gmail.com>`)

See Also

`simulation.report`

`simulation.report` *Simulations*

Description

`simulation.report` is a secondary function called to initiate a simulation run in **scaRabee**. It evaluates the structural model using the initial estimates of model parameters and outputs the results to a report file stored in the run directory. See `vignette('scaRabee',package='scaRabee')` for more details. `simulation.report` is typically not called directly by users.

Usage

```
simulation.report(problem = NULL,
                  files = NULL)
```

Arguments

`problem` A list containing the following levels:

code A list of R code extracted from the model file. Depending on content of the model file, the levels of this list could be: template, derived, lags, ode, dde, output, variance, and/or secondary.

data A list which content depends on the scope of the analysis. If the analysis was run at the level of the subject, `data` contains as many levels as the number of subjects in the dataset, plus the `ids` level containing the vector of identification numbers of all subjects included in the analysis population. If the analysis was run at the level of the population, `data` contains only one level of data and `ids` is set to 1.

Each subject-specific level contains as many levels as there are treatment levels for this subject, plus the `trts` level listing all treatments for this subject, and the `id` level giving the identification number of the subject.
Each treatment-specific levels is a list containing the following levels:

- **cov** mij x 3 data.frame containing the times of observations of the dependent variables (extracted from the TIME variable), the indicators of the type of dependent variables (extracted from the CMT variable), and the actual dependent variable observations (extracted from the DV variable) for this particular treatment and this particular subject.
- **cov** mij x c data.frame containing the times of observations of the dependent variables (extracted from the TIME variable) and all the covariates identified for this particular treatment and this particular subject.
- **bolus** bij x 4 data.frame providing the instantaneous inputs for a treatment and individual.
- **infusion** fij x (4+c) data.frame providing the zero-order inputs for a treatment and individual.
- **trt** the particular treatment identifier.

method A character string, indicating the scale of the analysis. Should be 'population' or 'subject'.

init A data.frame of parameter data with the following columns: 'names', 'type', 'value', 'isfix', 'lb', and 'ub'.

debugmode Logical indicator of debugging mode.

modfun Model function.

`files` A list of input used for the analysis. The following elements are expected and none of them could be null:

data A .csv file located in the working directory, which contains the dosing information, the observations of the dependent variable(s) to be modeled, and possibly covariate information. The expected format of this file is described in details in `vignette('scaRabee', package='scaRabee')`.

param A .csv file located in the working directory, which contains the initial guess(es) for the model parameter(s) to be optimized or used for model simulation. The expected format of this file is described in details in `vignette('scaRabee',package='`

model A text file located in the working directory, which defines the model. Models specified with explicit, ordinary or delay differential equations are expected to respect a certain syntax and organization detailed in `vignette('scaRabee',package='s`

iter A .csv file reporting the values of the objective function and estimates of model parameters at each iteration. (Not used for simulation runs).

report A text file reporting for each individual in the dataset the final parameter estimates for structural model parameters, residual variability and secondary parameters as well as the related statistics (coefficients of variation, confidence intervals, covariance and correlation matrix). (Not used for simulation runs).

pred A .csv file reporting the predictions and calculated residuals for each individual in the dataset. (Not used for simulation runs).

est A .csv file reporting the final parameter estimates for each individual in the dataset. (Not used for simulation runs).

sim A .csv file reporting the simulated model predictions for each individual in the dataset. (Not used for estimation runs).

Value

Creates a simulation report and returns a data.frame of simulated and observed data containing the following columns:

ID Subject Identifier. If the analysis is run at the population level and if the original dataset contained multiple subjects distinguished by a different ID number, please note that the original ID is lost and replaced by 1 so that all available data is considered to come from the same subject.

TRT Indicator of treatment level (defining the sub-problems).

CMT Indicator of system state to which the simulated or observed value is associated.

TIME Time of the observation or model prediction.

SIM Value of the simulated state. NA if DV is not NA.

DV Value of the observed state. NA if SIM is not NA.

Author(s)

Sebstien Bihorel (<sb.pmlab@gmail.com>)

`weighting` *Residual Variability*

Description

`weighting` is a secondary function called during estimation run to evaluate the model(s) of residual variability specified by the code provided in the $VARIANCE block. `weighting` is typically not called directly by users.

Usage

```
weighting(parms = NULL,
          derparms = NULL,
          codevar=NULL,
          y=NULL,
          xdata=NULL,
          check=FALSE)
```

Arguments

`parms`	A vector of primary parameters.
`derparms`	A list of derived parameters, specified in the $DERIVED block of code.
`codevar`	The content of the R code specified within the $VARIANCE block in the model file.
`y`	The matrix of structural model predictions.
`xdata`	A vector of times at which the system is being evaluated.
`check`	An indicator whether checks should be performed to validate function inputs

Value

Return a matrix of numeric values of the same dimension as `f`.

Author(s)

Sebastien Bihorel (`<sb.pmlab@gmail.com>`)

Index