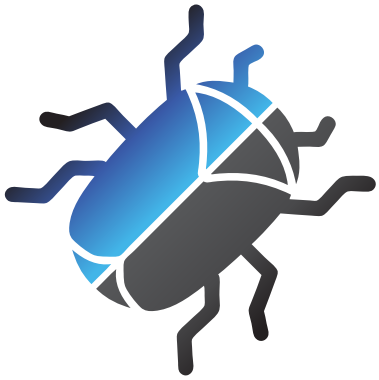
The scaRabee Package: An R-based Tool for Model Simulation and Optimization in Pharmacometrics

Version 1.1-3

Sébastien Bihorel

January 28, 2014



Contents

1 Introduction

scaRabee is a toolkit for modeling and simulation primarily intended for the field of pharmacometrics. It was initially a R port [**?**] of **Scarabee**, a Matlab-based application developed for the simulation and optimization of pharmacokinetic and/or pharmacodynamic models specified with algebraic equations, ordinary or delay differential equations [**?**]. It is now the only version supported and being developed. Since its first release, the **scaRabee** package has been improved and now contains functionnalities, which are not present in the original Matlab version.

1.1 Preliminary notice

This vignette constitutes a user manual for **scaRabee**. This manual assumes that the reader is familiar with the concepts of pharmacokinetic and pharmacodynamic modeling and the underlying statistical theories. It is not the objective of this manual to explain and review those methods and theories. Readers who are new to this field are invited to read the excellent introductory and more advanced books from Gabrielsson and Weiner [**?**], Bonate [**?**] or Ette and Williams [**?**].

The systems analyzed in **scaRabee** must be specified, for most parts, using the R language and all analyses should be executed within the R environment in interactive or batch mode. Presentations of the R language are out of the scope of this manual.

1.2 What's new?

Version 1.1-2 introduces minor changes to **scaRabee** related primarily to the solver of delay differential equation. The `dde` solver from the **PBSddesolve** package was replaced by the `dede` solver from the **deSolve** package, which is more stable for large and complex model and permits easier handling of discontinuities. A bug is the handling of dosing history is also fix in this version.

1.3 How to obtain scaRabee

scaRabee is available at the Comprehensive R Archive Network and also at Pmlab (`http://code.google.com/p/pmlab/`), a repository for open-source software solutions in pharmacometrics. The most recent version of **scaRabee** can be found in the Downloads section. **scaRabee** is distributed under a GNU General Public License version 3. Please, review the terms of this license before using this package. If this license was not distributed with your copy of **scaRabee**, please visit the GNU Project website (`http://www.gnu.org/`).

1.4 Installation and dependencies

This package is available as source archive. You are invited to read Section 6.3 Installing packages from the R Installation and Administration manual [**?**] for more details on how to install a source package from a local .zip or .tar.gz file on your system. Model optimization in **scaRabee** (as described in Section 2) relies on functions from the **neldermead**, **optimsimplex**, and **optimbase** packages, which are also distributed from CRAN and Pmlab as compressed archives of the source packages.

1.5 Credits

scaRabee was written by Sébastien Bihorel, alumni of the Paris 5 - René Descartes University and of the State University of New York (SUNY) at Buffalo, upon suggestions and contributions from:

- Pawel Wiczling, alumni of SUNY at Buffalo, who shared codes at the basis of the first Matlab® versions of the `fitmle` and `fitmle.cov` functions,
- John Harrold, Post-Doctoral Fellow at SUNY at Buffalo, who provided numerous advises during the creation of the Matlab® version of **scaRabee**,
- Sihem Ait-Oudhia, alumni of the Paris 5 - René Descartes University and of SUNY at Buffalo, for her suggestions and support.

The **neldermead**, **optimsimplex**, and **optimbase** packages, used for parameter optimization in **scaRabee**, are R ports of the Scilab modules of the same names which were developed by Michael Baudin at INRIA (Institut National de Recherche en Informatique et en Automatique) and now at the Digitéo consortium. More information on Scilab can be found at `www.scilab.org`.

1.6 Reporting bugs

We welcome bug reports, questions, and suggestions concerning any aspect of **scaRabee** functions, documentation, installation, anything... Please email them to `sb.pmlab@gmail.com`.

For bug reports, please include enough information to allow the maintainer to reproduce the problem. Generally speaking, that means:

- your version of **scaRabee** and the function(s) or manual involved.
- your version of R and package dependencies.
- hardware and operating system names and versions.
- the contents of any data and model files necessary to reproduce the bug.
- a description of the problem and samples of any erroneous output.
- any unusual options you gave to configure the problem.
- anything else that you think would be helpful.

Patches are also most welcome; if possible, please follow the existing coding style, comment our faulty code and clearly marked your editions.

2 Types of analysis performed in scaRabee

scaRabee allows the optimization and simulation of non-linear systems at the population and subject levels but does not implement non-linear mixed effects modeling. For each subject included in an analysis, **scaRabee** allows users to split the analysis problem into subproblems (also referred to as treatments), while still defining a unique model. This feature is especially useful when data obtained from several dose levels or regimens are fitted simultaneously, because it avoids the duplication of algebraic or differential equations usually needed to accommodate the different dose levels or regimens.

scaRabee allows users to perform three types of analysis: simulations, estimations, and direct grid searches.

2.1 Simulation

Simulation runs allow to generate detailed model predictions based upon initial parameter values provided by the user. **scaRabee** also produces default overlay plots of model predictions and actual observations.

2.2 Estimation

Estimation runs allow to optimize model parameters based upon the observations, the structural model and a residual variability model. Model parameters are optimized by the method of the maximum likelihood, more precisely by the minimization of an objective function defined as the exact negative log likelihood of the observed data, given the model structure and a set of parameter values. The minimization algorithm is based upon the Nelder-Mead simplex method, as implemented by the `fminsearch` function from the **neldermead** package. The computation of the data likelihood and the covariance matrices of primary and secondary parameters are performed as described in the Adapt II software user's manual written by D'Argenio and Schumitzky [**?**].

The analysis of population data can be performed by naïve averaging, naïve pooling, or by the standard two-stage approach [**?**]. Note that the standard two-stage approach is automated only since version 1.1-0.

2.3 Direct grid search

Direct grid search runs allow to explore the search space of an optimization problem around the initial point x_0 of parameter estimates. This **scaRabee** feature automatically creates a grid of search points selected around the initial point and evaluates the objective function at each one of these search points. The boundaries of the grid are set either by the lower and upper boundaries set by the users, or by a vector of factors α applied to x_0 as follows: $[x_0/\alpha, x_0 \times \alpha]$. The number $npts$ of points evaluated for each parameter (or dimension of the optimization problem) can also be defined. The total number of points in the grid is $npts^{p_e}$, where p_e is the number of parameters to be estimated. At the end of the search, a table sorted by increasing value of the objective function is created.

This table also reports the feasibility of the objective function at each particular search point because the grid search is actually delegated to the `fmin.gridsearch` function from the **neldermead** package. This function is a wrapper around `optimbase.gridsearch` from the **optimbase** package, which assesses the feasbility of a cost function in addition to calculating its value at each particular search point. Because `fmin.gridsearch` does not accept constraints, the objective function should always be feasible. Additional information is available in `optimbase.gridsearch` and `?fmin.gridsearch`.

3 Getting started

All **scaRabee** analyses are typically conducted in analysis-specific folders and rely on the presence of a given list of files in this working directory. A typical **scaRabee** folder, as one created by the `scarabee.new` function, contains at least the following files:

myanalysis.R the master R script; this file is required to initiate the analysis. Section 3.6 describes which parts of the code must be edited.

data.csv the data file; this file is a comma-separated table containing the dosing and observation data to be used for model simulation or optimization. Section 3.3 describes how this data must be specified.

initials.csv the parameter file; this file is a comma-separated table containing the names and values of the model parameters, used for model optimization or as inputs for model simulation. In the former case, the values provided for each parameters are used as initial estimates for the optimization. Section 3.4 describes how these parameters must be specified.

model.txt the model file ; this file is a text file in which the structural model, residual variability model, and secondary parameter computations are defined. Section 3.5 describes the syntax that must be applied to edit this file.

While the names of these files correspond to the default assumed by the `scarabee.new` function, they can be modified at the user's discretion. In this manual, the default files names are used for the sake of simplicity. Finally, please note that you can add any file needed or not for your analysis in your analysis folder. The Sections 3.1 through 3.7 offer a step-by-step description of the analysis process.

3.1 Creation of a new analysis folder

If you start a brand new data analysis, it is recommended that you open an interactive R session, and use the `scarabee.new` function to create a new analysis folder that will contain `myanalysis.R`, `data.csv`, `initials.csv`, and `model.txt`. It is recommended that you provide all arguments of `scarabee.new` to better set up the new folder:

`name` controls the name of the analysis, which is used as a base name for the master R script file (in place of the default 'myanalysis') and is also inserted after the $ANALYSIS tag in the model file (see Section 3.5),

`path` defines the (absolute or relative) path to the directory to be created; if the `path` argument is NULL, then it is coerced to `name`, thus causing the (tentative) creation of a new folder named as the `name` argument in the current working directory,

`type` defines whether the analysis is a simulation (default), an estimation, or a direct grid search run,

`method` defines if the analysis is to be performed at the population (default) or subject level,

`template` controls which template will be used for `model.txt`; templates are available for models defined with algebraic ('explicit'), ordinary differential equations ('ode', default), or delay differential equations ('dde').

Here is an example of **scaRabee** folder creation:

```
> require(scaRabee)
> scarabee.new(name='myanalysis',
+              path = 'some/target/directory/',
+              type = 'simulation',
+              method = 'population',
+              template = 'ode')
```

3.2 Creation of new models in an on-going analysis

Alternative models for an on-going analysis might be created in three different ways:

1. Create copies of the master R script and `model.txt` of interest in the current working directory. This method is not recommended but should work as long as the new master R script is updated appropriately and the string following the $ANALYSIS tag in the new model file is different from the one used in the original model.

2. Create a brand new analysis folder using the method described in Section 3.1.

3. Copy an existing analysis folder to a different location, and make the appropriate deletion of analysis subfolders and report files.

Regarless of the chosen method, most analysis files require some form of modification, that are described in Section 3.3 through 3.6. Symbols and notations used in those sections as well as in the **scaRabee** function man pages are summarized in Table 1.

Symbol	Definition
p_e	Number of parameters to be estimated
p_f	Number of fixed parameters
p_s	Number of secondary parameters to be estimated
p_d	Number of derived parameters
c	Number of covariates in the dataset
n	Number of subjects in the dataset
k_i	Number of subproblems for the i^{th} subject
b_{ij}	Number of bolus events in the j^{th} subproblem for the the i^{th} subject
f_{ij}	Number of infusion events in the j^{th} subproblem for the the i^{th} subject
d_{ij}	Number of dosing events in the j^{th} subproblem for the the i^{th} subject (b_{ij}+f_{ij})
m_{ij}	Number of observation times in the j^{th} subproblem for the i^{th} subject
s	Number of system states
o	Number of system outputs
l	Number of delays defined for a solution of a system of delayed differential equations

Table 1: Symbol Definition for Vector and Matrix Dimensions

3.3 Editing of the data file

The data file (named `data.csv` by default) contains the dosing information and endpoint measurements to be modeled or matched against a model simulation. It is required for any type of run. The **scaRabee** data files adopt similar structure and standards as those used in programs commonly used in pharmacometrics, such as NONMEM [**?**], S-SADAPT [**?**], or MONOLIX [**?**].

The data must consist of a time-ordered series of dosing and observations events specific to each subproblem (or treatment; see below) of each subject included in the dataset. Blocks of subject/treatment specific data must simply be stacked one after the other. The dataset must respect a tabulated, comma-separated value, format and can be edited in any text editor or spreadsheet. All **scaRabee** data files can be saved as any user-defined base name; however, the .csv extension is compulsory. The content of the data file must be a full rectangular table, with the following structure:

- All data variables must be stored in specific columns, each having a unique header. A series of variables with reserved names and expected content must be present, but users can add any number of custom (usually numerical) variables. The names and the meanings of the variables

required in any **scaRabee** dataset are provided in the following listing, which also includes one useful but optional variable:

OMIT (optional) omission flag. Only the data records with the OMIT variable set to 0 are included in the analysis. The OMIT variable is coerced to integer numbers by **scaRabee**.

ID subject identifier. A sequence of unique integers starting at 1 is expected to distinguish the subjects in the dataset. The ID variable is coerced to integer numbers by **scaRabee**.

TRT subproblem identifier. This variable must contain integer numbers in **increasing** order from 1 to k_i, the total number of subproblems for the i^{th} subject. If the user decides to define different subproblems for one or more individuals, all subproblems are evaluated separetely, but all contribute to the value of objective function for this(ese) individual(s). This feature typically allows users to define simpler systems when modeling different dose levels/regimens, as it avoids e.g. the duplication of the system equations to accomodate data collected at multiple dose levels, or the need for a system reset between to treatment period. Therefore, the TRT variable is indistinctly referred to as the subproblem or treatment variable in this manual. All records with a similar TRT value will be considered as part of the same subproblem. The TRT variable is coerced to integer numbers by **scaRabee**.

TIME independent variable. It represents the time since the first event; therefore, TIME should be 0 for the first (dosing or observation) record of each unique treatment of each subject. If this is not the case for at least one treatment for one subject, the dataset is processed by **scaRabee** and a new dataset including the calculated time since first event is saved to the working directory and used for the analysis.

AMT amount variable. This variable is used to define dosing events in combination with the RATE, CMT, and TIME variables. For each dosing record, the value set for the AMT variable represents the dose administered at the TIME for the record and assigned to the system state defined in the CMT variable (see below). The content of the AMT variable is ignored for observation records.

RATE rate variable. This variable is used to define dosing events in combination with the AMT, CMT, and TIME variables. For each dosing record, the value set for the RATE variable reflects the rate at which the dose AMT is administered into the system state CMT (see below). The RATE variable can be set to:

0 to indicate an instantenous input into the system,

any value > 0 to define the rate of a zero-order input into the system,

-1 to request the estimation of the rate of a zero-order input into the system, and

-2 to request the estimation of the duration of a zero-order input into the system.

The user cannot request the estimation of the duration for one record, and the estimation of the rate for another: -1 and -2 are mutually exclusive across the dataset.

CMT compartment variable. This variable represents the system state (*i.e.* a compartment in the standard representation of system in pharmacometrics) associated to a dosing record. The CMT variable is ignored for observation records. The CMT variable is coerced to integer numbers by **scaRabee**.

EVID event identifier. This variable is used to define the type of record/event. The EVID variable is set to:

0 for observation records, and to

1 for dosing records.

The EVID variable is coerced to integer numbers by **scaRabee**.

DV dependent variable. This variable represents the observed value associated with the record. This value assigned to this variable is ignored for dosing records.

DVID dependent variable. This variable represents the model output (see Section 3.5) associated to an observation record. Although DVID could be missing for dosing events and is ignored by **scaRabee**, if a value is provided, this value must be 0. The DVID variable is coerced to integer numbers by **scaRabee**.

MDV missing dependent variable. This variable must be set to 1 for dosing records and to 0 for observation records that are to be included in the analysis dataset. Observation records with a DVID value other than 0 are excluded. The DVID variable is coerced to integer numbers by **scaRabee**.

- Any other variable provided in the data file is considered as a covariate. The total number c of covariates are available for use in selected blocks of code in the model file (see Section 3.5).
- Record values set to . or NA are considered missing information by **scaRabee**.
- All data files must contain at least the header line and two records per subproblem for each subject, indicating the beginning and the end of the observation intervals.

NONMEM users must be warned that several data standards and variables are not implemented in **scaRabee**, *e.g.* all records set with EVID of 2, 3, or 4 are ignored by **scaRabee**, and the CONT, ADDL, II, and SS variables are considered as covariates.

3.4 Editing of the parameter file

The parameter file (named `initials.csv` by default) contains the information about the primary model parameters. Derived parameters, *i.e.* parameters that are needed for model computations but do not need to be estimated, can be specified in the $DERIVED or $OUTPUT blocks in the model section. Secondary parameters, *i.e.* parameters that are typically not needed for model computations but fro which precision statistics are required, can also be defined in the model file using the $SECONDARY block of code.

This parameter file is required in all types of runs, and can be edited in any text editor or spreadsheet. All parameter files must respect the comma-separated values format but can be saved under any user-defined name (the .csv extension is compulsory though). The content of parameter files must be provided as a full $(p_e + p_f + 1) \times 6$ rectangular table (where p_e and p_e are the numbers of fixed and estimated parameters), with the following structure:

- The first line must contain the headers of each column of your data table. This line is provided in the original `initials.csv` and should typically not be modified.
- There must be 6 columns, ordered as follows:

  Parameter, Type, Value, Fixed, Lower bound, Upper bound

  where Parameter, Type, and Value are the columns of parameter names, types and values, Fixed is the column indicating whether a given parameter should be estimated or fixed in an estimation analysis, and Lower bound and Upper bound are the columns defining the range of values that a given parameter could take.

- Each line must contain 6 elements separated by commas. There cannot be any missing data in this table.
- The Parameter column can contain numbers or strings of characters, representing the name of your model parameters (numbers will be handled as strings of characters).
- The Type column must contain single characters, indicating the type of each single parameter. There is four types of variables in Scarabee, so only four authorized characters:

  P indicates that the parameter is a structural model parameter.

  L indicates that the parameter is a delay. This category exists for the user convenience in the definition of model with delayed differential equations.

  IC indicates that the parameter is used to define an initial condition of a differential equation. This category is a legacy of the original Scarabee Matlab code. It exists for the user convenience in the definition of model with delayed differential equations but is handled exactly the same way as parameters of type 'P'.

  V indicates that the parameter is used to specify the residual variability model.
- The Value column must contain real numbers, representing the values taken by the parameters.
- The Fixed column must contain either 0's or 1's, indicating whether a parameter should be fixed (1) or estimated (0) during an estimation analysis. This column has no impact on simulation runs.
- The Lower and Upper bounds must contain real numbers, representing the range of values that parameters can take. The optimization algorithm implemented in **scaRabee** forces all estimated parameters to remain within these defined ranges.
- All parameter files must contain at least a header line and one parameter definition line.

3.5 Editing of the model file

The model file (named `model.txt` by default) is a text file in which users can specify the structural model, residual variability model, and secondary parameter computations. The model file is required for all types of analysis. It can be modified in any text editor and saved under any user-defined name.

The model file implements a tag-based syntax similar to the one used in NM-TRAN control streams [**?**], S-ADAPT-TRAN [**?**] or MONOLIX [**?**] model files. Tags are defined as strings of characters starting by the $ symbol followed by a keyword. At the exception of $ANALYSIS, each tag of the listing below marks the beginning of a block of R code defining one particular component of the evaluated system. Because of these tags, **scaRabee** model files cannot be interpreted directly by R; their content must first be parsed by **scaRabee**, before each block of R code could be evaluated at relevant stages of the analysis process. Within those blocks of code, users can call any R function that would be available in their workspace.

Upon creation of a new analysis folder, the model file is pre-filled with the tags that are appropriate and required for the specified category of model. The complete list of tags required for each category of model is given in Table 2.

As stated above, users can modify the newly created file in any text editor. Note that any tag keyword could be abbreviated to the first three letters of the keyword, except for $IC. When a analysis is started (see Section 3.7), the model file is read, parsed, and checked by **scaRabee**. If

Model category	explicit	ode	dde
	$ANALYSIS	$ANALYSIS	$ANALYSIS
	$OUTPUT	$DERIVED	$DERIVED
	$VARIANCE	$IC	$IC
	$SECONDARY	$SCALE	$SCALE
		$ODE	$LAGS
		$OUTPUT	$DDE
		$VARIANCE	$OUTPUT
		$SECONDARY	$VARIANCE
			$SECONDARY

Table 2: Required tags for model file

requirements are not met, the analysis stops and users are invited to check the content of the model file. Note that **scaRabee** determines the category of structural model by scanning the content of the file for the $ODE and $DDE tags: if the $ODE tag is detected, the model is assumed to be defined with ordinary differential equations; if the $DDE tag is detected, the model is assumed to be defined with delay differential equations; if both tags are not detected, the model is assumed to be defined with algebraic equations. The $ODE and $DDE tags cannot coexist within the same model file.

3.5.1 $ANALYSIS

The $ANALYSIS tag allows users to provide a name to the analysis, which is used to name the folder created to store the results of the analysis (see Section 3.7) and the analysis report files. The name extracted by **scaRabee** is the first word following the tag.

The $ANALYSIS tag must be present in all model files, regardless of the category of models.

3.5.2 $DERIVED

The $DERIVED tag is specific to and required for structural models specified with ordinary or delay differential equations. It allows users to define derived parameters which could be called later within the $ODE or $DDE blocks of code. Within the $DERIVED block, users can call any primary parameter defined in the parameter file and any covariate name to define new objects. Only the new R objects created in the $DERIVED block will be considered as secondary parameters; in other words, all modifications of a primary parameter will be ignored. Furthermore, users can choose to leave this block of code empty, if no derived parameter is needed.

Although users could choose to define derived parameters within the $ODE or $DDE blocks, it is computationally more efficient to define them in the $DERIVED block, as this block of code is only evaluated once for each model evaluation, while the $ODE or $DDE blocks of code are evaluated up to several thousands of times.

Note that the $DERIVED tag is not required (and actually ignored) for models specified with algebraic equations, because derived parameters could be defined within the $OUTPUT block without loss of computation efficiency.

3.5.3 $IC

The $IC tag is specific to and required for structural models specified with ordinary or delay differential equations. It allows to define the initial conditions of the system of differential equations. Users can call any primary or derived parameters within the $IC block.

scaRabee expects the creation of the `init` object, which must be a vector containing as many elements as there are states in the system of differential equations.

3.5.4 $SCALE

The $SCALE tag is specific to and required for models specified with ordinary or delay differential equations. It allows users to scale any instantaneous or continuous inputs into the system. This is particularly useful when the dimensions of the inputs and the associated stated are different, which is the case when a dose of drug in mass (g) or amount (mol, IU) is assigned to a state modeled as a concentration (g/L, mol/L or IU/L). Users can call any primary or derived parameters within the $SCALE block.

scaRabee expects the creation of the `scale` object, which must be a scalar or a vector containing as many elements as there are states in the system of differential equations. Consequently, all inputs into a given system state will be scaled identically.

3.5.5 $LAGS

The $LAGS tag is specific to and required for structural models specified with delay differential equations. It allows users to define the delays at each the system of differential equations should be evaluated. Users can call any primary parameter and any derived parameter to define delays within the $LAGS block.

All primary parameters of type 'L' and all new `R` objects created in the $LAGS block will be considered as delays. All modifications of a primary or derived parameter will be ignored, so users cannot directly set primary or derived parameters as systems delays. Except for the parameters of type 'L', all delay parameters must be derived from previous parameter and be given new names.

Users must define at least one system delay (either as a primary parameter of type 'L' or as a new `R` object inside the $LAGS block) when the structural model is defined by delay differential equations.

3.5.6 $ODE

The $ODE tag is specific to and required for structural models specified with ordinary differential equations. It allows users to define the system of differential equations. The parameters available to users within the $ODE block are:

- the primary parameters,
- the derived parameters,
- t, the time of evaluations of the system,
- $a1$, ..., a_s, the values of the system states at time t, where s is the total number of states, and
- any covariate name. However, by default, **scaRabee** does not interpolate the covariate data at time t. Users might want to call the `approx` function for this purpose (see `?approx`).

scaRabee expects the creation of the `dadt` object, a $1 \times s$ matrix of system states. Note that it is not necessary to include exogenous inputs (boluses and infusions) into the system of differential equations, this is automatically done by the code.

3.5.7 $DDE

The $DDE tag is specific to and required for structural models specified with delay differential equations. It allows users to define the system of differential equations. The parameters available to users within the $ODE block are:

- the primary parameters,
- the derived parameters,
- t, the time of evaluations of the system,
- $a1$, ..., a_s, the values of the system states at time t, where s is the total number of states,
- $alag.lag_1$, ..., $alag.lag_l$, the vector of system states at time $t - lag_1$, ..., $t - lag_l$, where l is the total number of delays defined in the $LAGS block of code. To access to the value of a particular system state at a particular delay, users must subset the appropriate $alag.lag_i$ vector: *e.g.* $alag.past[3]$ would extract the value of the 3^{rd} system state at a delay named `past`, and
- any covariate name. However, by default, **scaRabee** does not interpolate the covariate data at time t. Users might want to call the `approx` function for this purpose (see `?approx`).

scaRabee expects the creation of the `dadt` object, a $1 \times s$ matrix of system states. Note that it is not necessary to include exogenous inputs (boluses and infusions) into the system of differential equations, this is automatically done by the code.

3.5.8 $OUTPUT

The $OUTPUT tag must be present in all model files, regardless of the category of models. It allows users to defined the output(s) of the structural model.

In models defined with algebraic equations, the $OUTPUT block is the place where the derived parameters and the structural model should be defined. The parameters available to users within the $OUTPUT block are:

- the primary parameters,
- *times*, the vector of unique times of observations (or simulated observations),
- *bolus* and *infusion*, the data frames of bolus and infusion dosing records extracted from the data file, and
- any covariate data. Note that it is only necessary to interpolate the covariate data for simulation or direct grid search runs, as covariate data should be available at any observation time in estimation runs.

In models defined with ordinary or delay differential equations, the $OUTPUT block is the place to define the model output using the predictions from the integration of the system of differential equations. The parameters available to users within the $OUTPUT block are:

- the primary parameters,
- the derived parameters,
- *times*, the vector of unique times of observations (or simulated observations), and

- *f*, the $s \times m_{ij}$ matrix of system state predictions, where m_{ij} is the total number of observations in the j^{th} subproblem for the i^{th} subject.

scaRabee expects the creation of the `y` object, which must be a $o \times m_{ij}$ matrix, where o is the number of system outputs. For any type of run, the data records set with a DVID value of *dvid* will be matched against the $dvid^{th}$ system output. Therefore, the maximum value of the DVID variable in the dataset must be o.

3.5.9 $VARIANCE

The $VARIANCE tag must be present in all model files, regardless of the category of structural model. The presence of a $VARIANCE tag is required for types of runs, except for simulations. The $VARIANCE block allows users to define the residual variability models associated with each structural model outputs. The parameters available to users within the $VARIANCE block are:

- the primary parameters,
- the derived parameters,
- y, the $o \times m_{ij}$ matrix of structural model predictions, and
- *ntime*, a scalar which value is set to m_{ij}.

scaRabee expects the creation of the `v` object, which must have exactly the same dimension as the y object created in the $OUTPUT block of code. v represents the matrix of variance associated with each model prediction. Typical residual variability models are (assuming $o = 1$):

- additive variability model with variance 1

```
> v <- rbind(ones(1,ntime))
```

- additive variability model with estimated or fixed standard deviation, *SD*:

```
> v <- rbind((SD^2)*ones(1,ntime))
```

- coefficient of variation model with estimated of fixed standard deviation, *CV*:

```
> v <- rbind((CV^2)*(y[1,]^2))
```

- additive and constant coefficient of variation model with estimated or fixed standard deviations, *SD* and *CV*:

```
> v <- rbind((SD^2)*ones(1,ntime) + (CV^2)*(y[1,]^2))
```

3.5.10 $SECONDARY

The $SECONDARY tag is optional for all model files, regardless of the category of structural model or run type. It allows users to define p_s secondary parameters for which associated statistics must be computed (typically precision and parametric confidence interval). The only parameters available to users within the $SECONDARY block are the primary parameters. Only the new R objects created in this block will be considered as secondary parameters; in other words, all modifications of a primary parameter will be ignored. Furthermore, users can choose to leave this block of code empty, if no secondary parameter should be computed.

3.6 Editing of the master scaRabee script

The master **scaRabee** script (named `myanalysis.R` by default) is the R script that you must execute to perform any analysis. You must edit several lines location in a specific section of the file (from line 21 to line 57, or 60 if 'dde' was selected as a template when `scarabee.new` was called) to define the settings of your analysis. Any other line of this file should typically not be modified. Commented lines within the user-editable section explain what and how variable(s) should be defined.

- Line 25: users can choose to define a working directory by adding a valid path within the `setwd` function. This is optional but recommended if users work in an interactive R session. If provided, the path to the working directory must contain the files specified in the `files` list (see below).

- Lines 34-36: the `data`, `param`, and `model` levels of the `files` list are character variables defining the names of the files where your data, parameters, and model are respectively defined. The default content of these levels matches the name of corresponding files created by `scarabee.new`. Users can change those default names.

- Line 39: the `runtype` variable is a character variable, defining if the analysis is an estimation, a simulation, or a direct grid search. Any other character string than 'estimation', 'simulation', or 'gridsearch' will cause an early termination of the run and the display of an error message to the console or log file.

- Line 42: the `method` variable is a character variable, defining the scope of the analysis. It must be set to 'subject' or 'population'. Any other character string will cause an early termination of the run and the display of an error message to the console or log file.

- Line 44: the optimization algorithm is designed to return an infinite objective function value in case the computation of the objective function at a given point of the multi-dimensional search space returns an error message. This is meant to prevent R from stopping the optimization process. Unfortunately, this will also happen if an execution error occurs during the evaluation of the model or the residual variability functions. The `debug` variable allows users to shut down this feature, and identify potential syntax or variable dimension problems in your model or residual variability files. The `debug` variable is a logical that can only take TRUE or FALSE as value.

- Lines 49-50: `estim` is a list with two levels, `maxiter` and `maxfunc`, defining the maximum number of iterations and function evaluations during an estimation run. Both must be scalar integers. The default values are 500 and 5000, which should typically allow user's problem to converge to a stable point of the search space.

- Lines 54-55: the `npts` and `alpha` are variables specific to direct grid search runs. `npts` must be an integer greater than 2 and defines the number of points that the grid should contain per dimension (*i.e* variable model parameter). `alpha` must be a scalar or a vector of real numbers greater than 1, which give the factor(s) used to calculate the range of evaluation for each dimension of the search grid (see `?scarabee.gridsearch` for more details). If `alpha` is set to NULL, the lower and upper boundaries set in the parameter file are used to define the range of evaluation for each dimension of the grid.

3.7 Execution of the master scaRabee script

Once all necessary files have been edited, the analysis can be performed by executing the master R script. This can be done in two ways:

- from an iteractive R session: we recommend that you set the working directory as the path to the analysis folder both in the R session and in the master script (see Section 3.6). Then, type:

  `source('myanalysis.R').`

  You will be asked whether or not you want to change the working directory, press ENTER if this is not the case. At the end of the run, press ENTER when prompted to display the different plots generated by **scaRabee**.

- from a shell or dos window: navigate to the directory containing the master R file of interest, then run the analysis by typing:

  `R CMD BATCH myanalysis.R`

  You may add any option you see fit.

In both modes, **scaRabee** creates a new folder in the working directory which name has the following structure:

<myanalysis>.<type>.#

where <myanalysis> is the string of character directly following the $ANALYSIS tag in the model file, <type> is `est` for estimation runs, `sim` for simulation runs, `grid` for direct grid search runs, and # is a two-digit integer.

At the exception of the .Rout file, all run outputs are stored in the newly created folder. Additionally, a subfolder called 'run.config.files' is created to backup all original analysis files (`data.csv`, `initials.csv`, `model.R`, and `myanalysis.R`).

In interactive mode, the run progression will be reported to the console, while it is stored to a log file in batch mode. Upon successful completion of the run, a termination message is reported and graphical outputs and ASCII text reports are produced. Most errors happening during the execution of the master R script should stop the run and prevent the creation or the finalization of the graphs and report files. Instead, an informative message should be displayed.

Simulation runs

Upon successful completion of the run, you should be able to see (in interactive mode) as many figures as the number of subject-subproblem combinations (see Section 4 for more details about how the scope of analysis impacts this number). Those overlay figures represent the predicted changes in all selected outputs on top of the observed data. As stated above, all figures are stored in the newly created folder.

A file called `<myanalysis>.simulation.csv` file is also saved in the same folder. This file lists the values taken by the model outputs at >1001 points within the studied time interval (typically from the minimum dose event or observation time to the maximum observation time), for each subproblem of each subject (see Section 4 for more details about the impact of the scope of analysis on this file).

Estimation runs

Upon successful completion of the run, a figure summarizing the changes in the objective function and the estimated parameter values as a function of the iteration number is created for each subject and stored in the newly created folder. A overlay figure of model predictions and observed data, and a figure showing 4 goodness-of-fit plots (predictions vs observations, weighted residuals vs time, weighted residuals vs observations, weighted residuals vs predictions) for each subproblem of each subject are also created and stored in the same folder (see Section 4 for more details about the impact of the scope of analysis on these plots). Starting on **scaRabee** version 1.1-0, those figures are not displayed on screen when the analysis is run in interactive mode.

A file called `<myanalysis>.report.txt` file is also saved in the same folder and provides, for each subject in the analysis, a summary of the estimation run, a summary table of final parameter estimates associated with precision statistics expressed as a coefficient of variation and a confidence intervals (calculated as described in [**?**]), the matrices of covariance and correlation for primary parameters, plus a summary table of computed secondary parameters associated with coefficient of variation and confidence intervals (calculated as described in [**?**]).

Moreover, a file called `<myanalysis>.iterations.csv` is saved in the folder and provides, in a tabulated format, the values of objective function and estimated parameters obtained at all iterations for each subject.

A file called `<myanalysis>.predictions.csv` is also saved in the folder and provides the values of observations, predictions, residuals, variances, and weighted residuals for each non-missing observation time, stacked by subject, subproblem, and model output.

Finally, a file called `<myanalysis>.estimates.csv` is also saved in the folder and summarizes the final parameter estimates for each subject included in the analysis. This file could be helpful to calculate statistics of distribution of the different parameters in the analysis population.

Direct grid search runs

Direct grid search runs include two main steps: the actual grid search, followed by a simulation step that is based upon the combination of parameter values that provided the lowest objective function value during the grid search. Direct grid search runs coerce the scope of the analysis to the population, even though the `method` variable in the master **scaRabee** script is set to 'subject'. Therefore, the computation of the objective function during the grid search and the model predictions obtained during the simulation step are performed at the population level (see Section 4 for more details about the impact of the scope of analysis)

The progression of the grid search step is reported on the console in interactive mode or in the log file in batch mode. Upon completion of this step, no graph is created. Instead, a regular simulation run starts and results in the creation of the standard diagnostic plots mentionned above.

The same files created by standard simulation runs are generated by a direct grid search run in the newly created folder. Furthermore, the results of the grid search are reported in a text file called `<myanalysis>.report.txt` that is also saved in the newly created folder.

4 Scope of analysis

scaRabee analysis can be conducted at the subject or population level. Users can set this scope of analysis by modifying the `method` variable in the master **scaRabee** script, as described in Section 3.6.

When `method` is set to 'subject', **scaRabee** processes and stratifies the content of the data file assuming that all dosing and observation records with specific ID values were obtained from different individuals. In this case, estimation runs optimize the model parameter separately for each individual, starting at the same search point provided by the initial parameter estimates. This corresponds to the standard two-stage approach, when the data file actually contains data from multiple subjects (*i.e.* multiple unique ID variable values can be found in the data file), or to the naïve pooling approach, when the data file only contains data from a single individual (*i.e.* the ID variable is set to 1 for all records) [**?**]. Simulation runs performed at the subject level evaluate the model for each subproblem/treatment of each subject using the same initial parameter estimates. Grid search runs are not performed at the individual level, as the `method` variable is coerced to 'population' for this type of analysis.

When `method` is set to 'population', **scaRabee** processes the content of the data file assuming that all observation records were obtained from a single individual. The dosing history is extracted from the dosing records with an ID variable set to 1. In this case, estimation runs optimize the model parameter on the global data, starting at the search point provided by the initial parameter estimates. This corresponds to the naïve pooling approach [**?**]. Simulation runs performed at the population level evaluate the model for all detected subproblems/treatments found in the dataset, using the initial parameter estimates. Finally, all grid search runs are performed at the population level.

5 Design information

5.1 Solvers of differential equations

Structural models defined using systems of differential equations require those systems to be integrated before model outputs could be generated. This step of integration is performed using solvers of differential equations, which are the `lsoda` solver from the **deSolve** package for systems of ordinary differential equations and the `dde` solver from the **PBSddesolve** package for systems of delay differential equations. Users are invited to refer to the documentation of those packages for more information.

5.2 Implementation of dosing history for model defined with differential equations

Instantaneous (*i.e.* bolus) and zero-order (*i.e.* infusion) inputs are automatically allocated to the appropriate system state by the functions evaluating the systems of differential equations (see the source code of `ode.model` and `dde.model` for more details). General rules for the implementation of dosing history are provided below.

Input scaling

All bolus and infusion input amounts (provided in the AMT variable) must be scaled by users. Input scaling is implemented in the R code provided in the $SCALE block of the model file as explained in Section 3.5). Scaling is particularly useful when the dimensions of the inputs and the associated stated are different, which is the case when a dose of drug in mass (g) or amount (mol, IU) is assigned to a state modeled as a concentration (g/L, mol/L or IU/L).

Bolus inputs

The `lsoda` solver used for models defined with ordinary differential equations does not include any handler of discontinuities. Because bolus inputs represents discontinuous events, their implementation require the integration of the system of differential equations to be performed by steps. When bolus inputs are detected in the data file, **scaRabee** splits the global integration interval into several continuous integration intervals based upon the dose event times. The initial conditions of the system are updated for each integration interval by adding the scaled bolus amount (AMT) specified in the data file to the value of the state (CMT) at the end of the previous interval (or the specified initial conditions in the case of the first interval). Therefore, all model predictions made at the time of a bolus assume that this bolus has entered the system. Users are thus advised to set the time of pre-dose samples slightly before the time of the boluses, to ensure that those samples are handled as pre-dose and not post-dose samples.

Infusion inputs

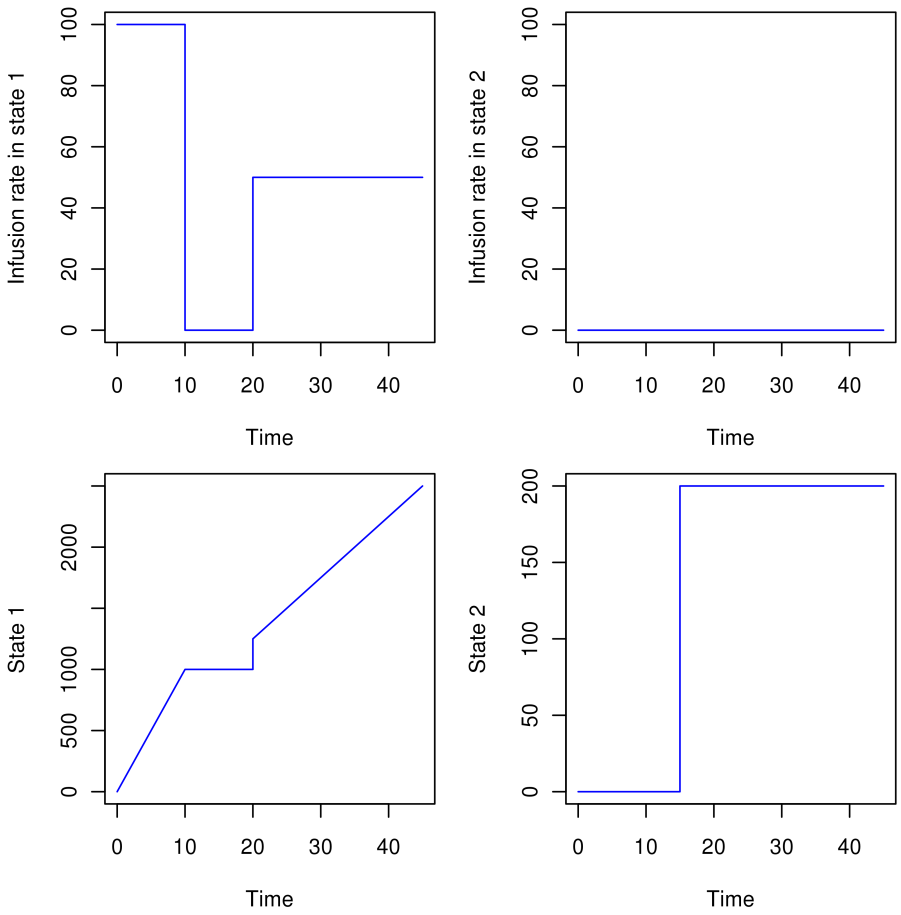
The reduction from multiple data files to a single one introduced in the version 1.1-0 of **scaRabee** resulted in major modifications in the automated processing and assignement of infusions to system states.

Previous versions of **scaRabee** required infusions to be 'constructed' by multiple records in a dosing-specific input files. Input rates were then linealy interpolated between two consecutive time points, allowing for an infusion rate to change over time. In version 1.1-0 of **scaRabee**, infusions are documented as single dosing records in the data file, providing the time of infusion start, the amount and rate of dosing. The rate is assumed to be constant for the whole duration of the infusion. If RATE>0 for the dosing record, the duration is calculated as RATE/AMT. If RATE=-1, the rate of infusion is estimated and the duration is calculated as R#/AMT, where R# is a derived parameter expected to be defined in the $DERIVED block (# represents the value of the CMT variable set for the dosing record). If RATE=-2, the duration of infusion is estimated and the rate is calculated as AMT/D#, where D# is a derived parameter expected to be defined in the $DERIVED block (# represents the value of the CMT variable set for the dosing record).

The following example illustrates the automated dosing allocation in **scaRabee**. Let's assume that the system is specified by two ordinary differential equations, both fixed to zero, and that the data in provided as follows in the dataset:

OMIT	ID	TRT	TIME	AMT	RATE	CMT	EVID	DV	DVID	MDV
0	1	0	0	0	0	1	0	0	1	0
0	1	0	0	1000	100	1	1	0	0	1
0	1	0	0	0	0	2	0	0	1	0
0	1	0	15	100	0	2	1	0	0	1
0	1	0	20	10000	50	1	1	0	0	1
0	1	0	20	250	0	1	1	0	0	1
0	1	0	45	0	0	1	0	0	1	0
0	1	0	45	0	0	2	0	0	1	0

State 1 receives 1 bolus dose at time 20 and 2 infusions: the first, between 0 and 10, has a constant rate of 100, and a second starts at time 20 and does not stop before the last observation. State 2 only receives a bolus dose at time 15. The following graphs show the changes in the infusion rate entering both states (top graphs), as well as the accumulation of the drug in both states (bottom graphs).



5.3 Implementation of dosing history for model defined with algebraic equations

Dosing history cannot be automatically assigned to a model defined with algebraic equations. However, users can use dosing information in the $OUTPUT block by calling the `bolus` and `infusion` variable, which each contain the TIME, CMT, AMT, RATE variable extracted from the d_{ij} dosing records identified as instantaneous (RATE=0) or zero-order (RATE≠0) inputs. Relevant data extraction would need to be performed by user-specific code.

It might also be convenient to carry dosing information in a covariate (*e.g.* DOSE) which could be used in a explicit solution of a specific pharmacokinetic model.

As such, it might have occured to users familiar with NONMEM that the implementation of models defined with algebraic equations in **scaRabee** is not too different from what NONMEM allows via to the $ERROR record.

6 Analysis examples

This section is designed to illustrate some selected features of **scaRabee**. Eight examples are available as demos using calls such as the following (replacing `ex` by `example1` to `example8`):

```
> demo(ex,package='scaRabee',echo=FALSE)
```

Running these examples will create analysis folders in your working directory. We recommend that you review their content after their creation.

6.1 Example 1: Simulation of a model defined with algebraic equations at the population level

A simple PK/PD model defined with algebraic equatons is simulated at the population in this example . The PK model describes the drug concentration C_p using a one-compartment model with linear elimination after a single 2h infusion. The response E is related to C_p by a direct effect Imax model:

$$C_p(t) = \frac{infusion\,rate}{CL} \cdot \left(1 - H(t-2) \cdot \left(1 - e^{-\frac{CL}{V_c} \cdot (t-2)}\right) - e^{-\frac{CL}{V_c} \cdot t}\right)$$
$$E(t) = E_0 \cdot \left(1 - \frac{I_{max} \cdot C_p(t)}{IC_{50} + C_p(t)}\right)$$

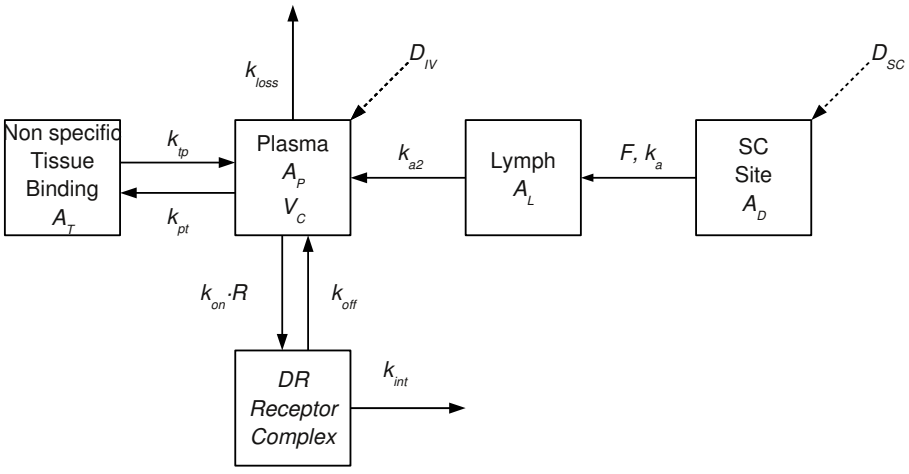
where H is the Heaviside function, CL the elimination clearance, V_c the volume of distribution, E_0 the baseline response, I_{max} the maximum inhibition factor, and IC_{50} the half-inhibitory concentration.

Note that because this is a simulation, there is no need for a $VARIANCE block.

6.2 Example 2: Simulation of inputs into a model defined with ordinary differential equations

This example uses a model defined by a system of 2 ordinary differential equations to illustrate how inputs are automatically assigned and scaled to system states. Because both states have null initial conditions and gradients, the output of the model represents the cumulative scaled amount of drug assigned to each state based upon the information provided in the dataset.

6.3 Example 3: Simulation of a model defined with ordinary differential equations at the population level



A target-mediated disposition model for interferon-β1a pharmacokinetics in monkey was described by Mager and colleagues [**?**]. This model is simulated at the population level in example 3 using the following system of differential equations:

	SC	IV
$\frac{dA_D}{dt} = -k_a \cdot A_D$	$A_D(0) = F \cdot D_{SC}$	$A_D(0) = 0$
$\frac{dA_L}{dt} = k_a \cdot A_D - k_{a2} \cdot A_L$	$A_L(0) = 0$	$A_L(0) = 0$
$\frac{dA_P}{dt} = k_{a2} \cdot A_L + k_{tp} \cdot A_T + k_{off} \cdot DR - (k_{on}/V_c) \cdot A_P \cdot R - (k_{pt} + k_{loss}) \cdot A_P$	$A_P(0) = 0$	$A_P(0) = D_{IV}$
$\frac{dA_T}{dt} = k_{pt} \cdot A_P - k_{tp} \cdot A_T$	$A_T(0) = 0$	$A_T(0) = 0$
$\frac{dDR}{dt} = (k_{on}/V_c) \cdot A_P \cdot R - (k_{off} + k_{int}) \cdot DR$	$DR(0) = 0$	$DR(0) = 0$
$R = R_{max} - DR$		

where A_D, A_L, A_P, and A_T are the amounts of drug in the subcutaneous depot, lymph, central, and peripheral compartments and DR is the concentration of drug-receptor complex. The noteworthy features of this example are:

- how dose information is extracted from the vector of covariate DOSE to define the scaling bioavailability factor F in the $DERIVED block,
- how the TRT variable is used in the dataset to define the different dosing regimens (3 different dose levels administered by single sub-cutaneous or intravenous dosing), and to avoid the duplication of the model equations, and
- how the output of the system of differential equations is subset and transformed to just extract the predicted concentration in the central compartment in the $OUTPUT block.

6.4 Example 4: Simulation of a model defined with delay differential equations at the population level

This example features a 2-compartment model with linear inter-compartment distribution but with a delayed entry of the drug into the peripheral compartment. The system can described by the following equations:

$$\begin{aligned} \frac{dA_P}{dt} &= -(k_e + k_{pt}) \cdot A_P(t) + k_{tp} \cdot A_T(t) & A_P(0) = 0 \\ \frac{dA_T}{dt} &= k_{pt} \cdot A_P(t - xyz) - k_{tp} \cdot A_T & A_T(0) = 0 \end{aligned}$$

where A_P and A_T are the amounts of drug in the central and peripheral compartments and xyz is the delay of entry into the peripheral compartment.

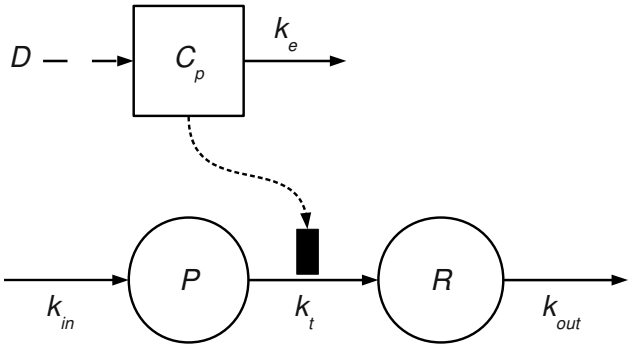
This system is simulated at the population level assuming repeated bolus administrations is the central compartment. The noteworthy features of this example are:

- how derived rate constants are computed in the $DERIVED block using the clearance and volume parameters defined in the parameter file,
- how the delay xyz is directly available in the system of delay differential equations, because it was defined as a 'L' parameter in the parameter file, and
- how a variance model is defined in the $VARIANCE block but not used for the simulation (this could be useful, if the same model is then used in an estimation analysis).

6.5 Example 5: Estimation of a model defined with algebraic equations at the population level

This example estimates the parameters of the Example 1 model using the observations provided in the Example 1 dataset and the naïve pooling approach. The model file was however modified to include the variance models of the concentrations and responses. Note that the concentrations were initially log-transformed in the dataset to fit the original data with log residual variability model. Consistently, the predicted C_P concentrations are log-transformed before assigned as the first raw of y.

6.6 Example 6: Simulation of a model defined with ordinary differential equations at the subject level



In this example, a precursor turn-over model is simulated at the subject level. The rate of transformation of the precursor P into response R is inhibited by the drug concentration C_p. The changes in drug concentration, precursor and response are described by the following equations:

$$\begin{array}{ll} \frac{dC_P}{dt} = -k_e \cdot C_P(t) & C_P(0) = D/V_c \\ \frac{dP}{dt} = k_{in} - k_t \cdot (1 - \frac{I_{max} \cdot C_p}{IC_{50} + C_p}) \cdot P & P(0) = R_0 \\ \frac{dR}{dt} = k_t \cdot (1 - \frac{I_{max} \cdot C_p}{IC_{50} + C_p}) \cdot P - k_{out} \cdot R & R(0) = R_0 \end{array}$$

6.7 Example 7: Estimation of a model defined with algebraic equations at the subject level

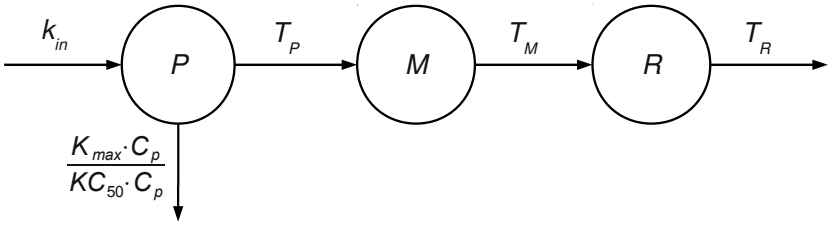
WARNING: this example can be time consuming.

This example estimates the parameters of the Example 1 model using the observations provided in the Example 1 dataset and the standard two-stage approach. The model file was however modified to include the variance models of the concentrations and responses. Note that the concentrations were initially log-transformed in the dataset to fit the original data with log residual variability model. Consistently, the predicted C_P concentrations are log-transformed before assigned as the first raw of y.

6.8 Example 8: Direct grid search for a model defined with delay differential equations

WARNING: this example can be time consuming.

This example illustrates how direct grid search can be performed using a life-span model for paclitaxel (C_P) on leukocytes counts in cancer patient [**?**].



Normalized leukocyte counts ($R_{\%}$) collected in one patient were digitized and a direct grid search run is performed to improve the estimates roughly chosen for the PD parameters (C_P is assumed to be accurately described by the parameter estimates obtained for a 3-compartment model). The paclitaxel effect is modeled with the following delay differential equation:

$$\frac{dR_{\%}}{dt} = k_{in\%} \cdot \left(e^{-\int_{t-T_P-T_M}^{t-T_M} f\left(C_P(z)\right)dz} - e^{-\int_{t-T_P-T_M-T_R}^{t-T_M-T_R} f\left(C_P(z)\right)dz} \right)$$

$$f\left(C_P\right) = \frac{K_{max} \cdot C_P}{KC_{50} + C_P}$$

The grid is formed by combining 3 grid points per variable parameters (T_P, T_M, K_{max} and KC_{50}) and by setting the scaling factor to 2 for all parameters. T_R was fixed as described in [**?**]. The best solutions found by direct grid search is finally compared to the reported estimates.

7 References

8 Network of scaRabee functions

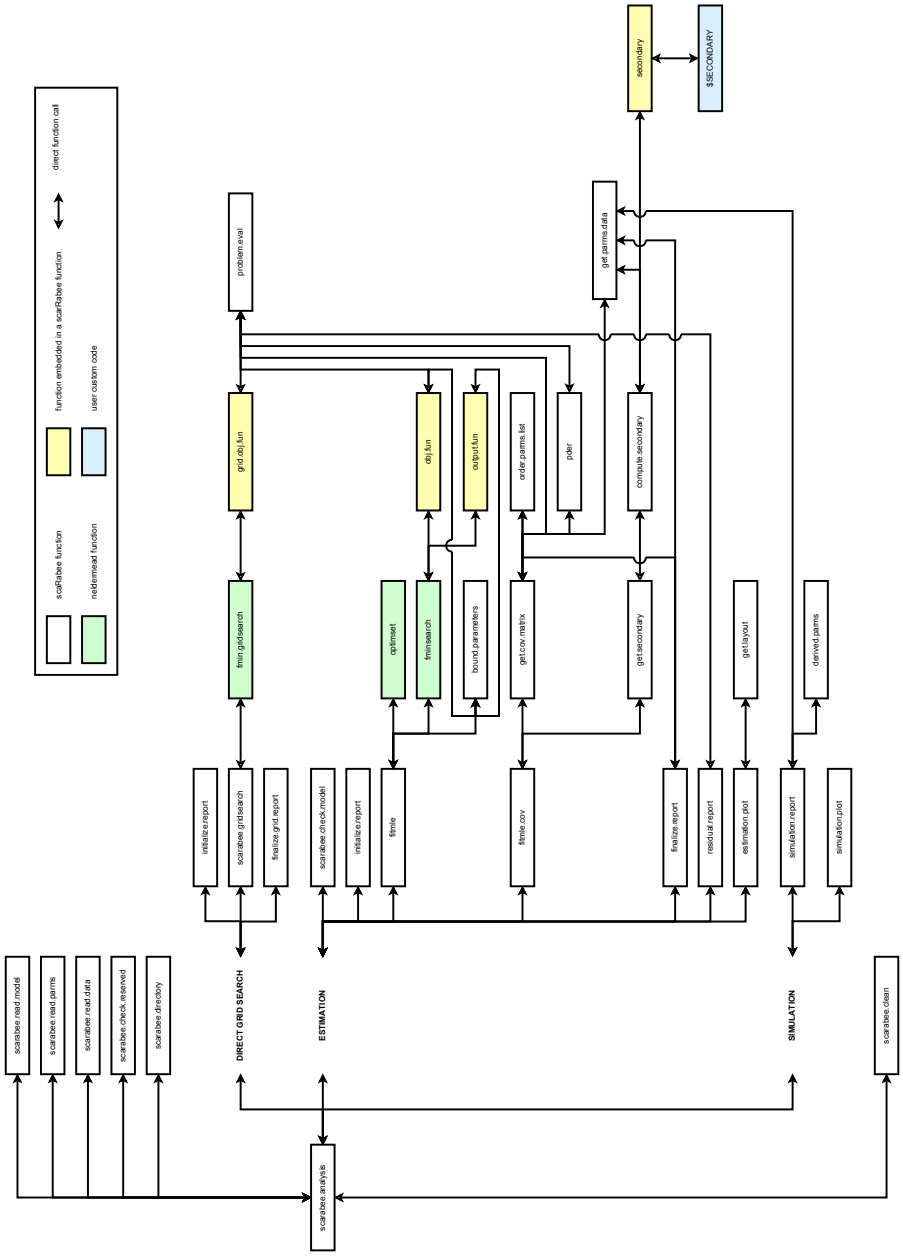


Figure 1: Map of the functions distributed with scaRabee (1/2)

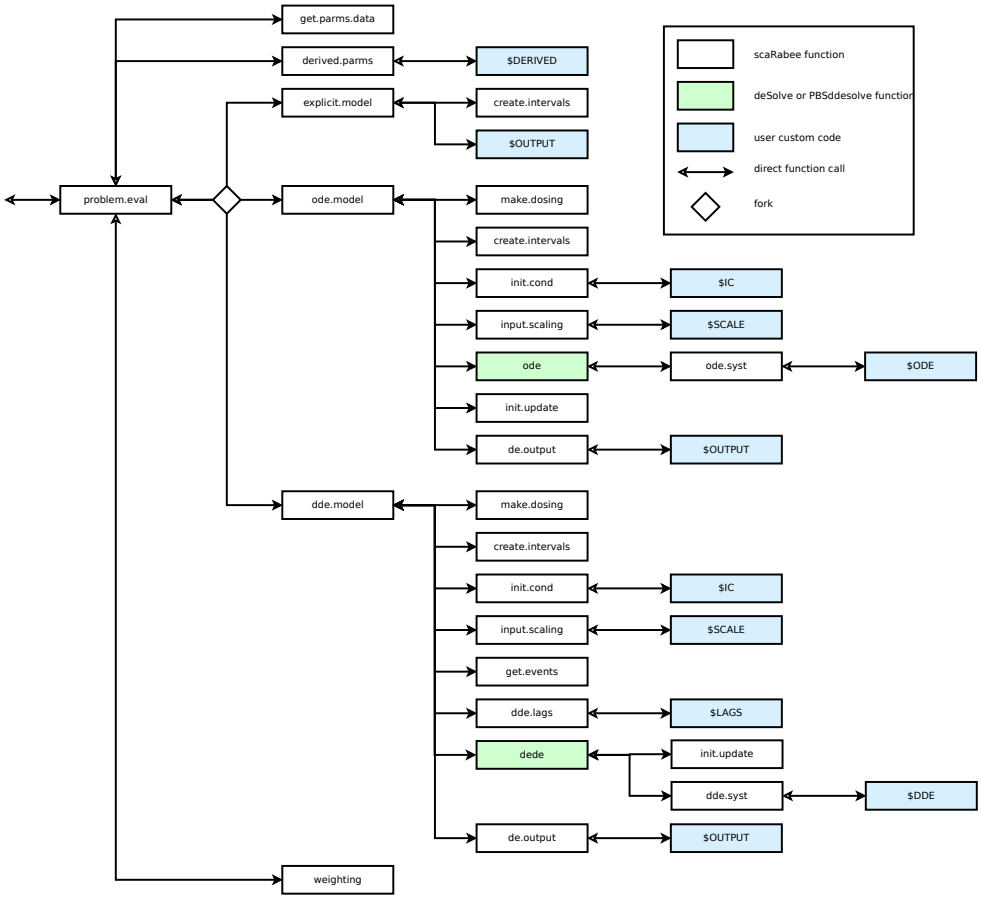


Figure 2: Map of the functions distributed with scaRabee (2/2)

9 Help on scaRabee functions

`scaRabee-package` *scaRabee toolkit*

Description

Framework for Pharmacokinetic-Pharmacodynamic Model Simulation and Optimization

Details

Package:	scaRabee
Type:	Package
Version:	1.1-3
Date:	2014-01-29
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: `http://code.google.com/p/pmlab/`

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

---

create.intervals *Create Integration Intervals Based on Dosing History*

Description

create.intervals is a utility function that is called by the model template based on ordinary differential equations. It 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.

Usage

```
create.intervals(xdata = NULL,
                 dosing = NULL)
```

Arguments

`xdata`	A vector of numerical observations times.
`dosing`	A d x 4 data.frame of dosing history containing the following columns: **Time** Dosing event times. **State** State where the input should be assigned to. **Bolus** Amount that should be assigned to system `state` at the corresponding `Time`. **Infusion.Rate** Rate of input that should be assigned to system `state` 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`.

Details

`create.intervals` 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.

Value

Returns a 2 x v matrix of numerical values, giving the beginning and the end of the integration intervals.

Author(s)

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

`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.

`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='scaRabee')`.

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='scaRabee')`.

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='scaRabee')`.

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='scaRabee')`.

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='scaRabee')`.

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='scaRabee')`.

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.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`

`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='scaRabee')`.

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='scaRabee')`.

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`

`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.param.data` *Extract data from scaRabee parameter table*

Description

`get.param.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.param.data` is typically not directly called by users.

Usage

```
get.param.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.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='scaRabee')`.

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='scaRabee')`.

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,
         xdata = 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.
`xdata`	A vector of times at which the system is being evaluated.
`check`	An indicator whether checks should be performed to validate function inputs
`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.

`covdata`	A matrix of covariate data extracted from the dataset.
`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`
`scale`	A vector of system scale, typically returned by `input.scaling`
`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.param.list *Sort a scaRabee parameter table*

Description

`order.param.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.param.list` is typically not directly called by users.

Usage

```
order.param.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>)

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='scaRabee')`.

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='scaRabee')`.

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='scaRabee')`.
- **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='scaRabee')`.

`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='scaRabee')`.

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='scaRabee')`.

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='scaRabee')`.
- **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='scaRabee')`.
- **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='scaRabee')`.

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='scaRabee')`.

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='scaRabee')`.

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='scaRabee')`.

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='scaRabee')`.

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='scaRabee')`.

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='scaRabee')`.
- **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='scaRabee')`.
- **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='scaRabee')`.

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='scaRabee')`.

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='scaRabee')`.

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='scaRabee')`.

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='scaRabee')`.

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='scaRabee')`.

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>)

10 GNU Free Documentation License

GNU Free Documentation License
Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with

modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word

processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the

copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities
   responsible for authorship of the modifications in the Modified
   Version, together with at least five of the principal authors of the
   Document (all of its principal authors, if it has fewer than five),
   unless they release you from this requirement.
C. State on the Title page the name of the publisher of the
   Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications
   adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice
   giving the public permission to use the Modified Version under the
   terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections
   and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add
   to it an item stating at least the title, year, new authors, and
   publisher of the Modified Version as given on the Title Page. If
   there is no section Entitled "History" in the Document, create one
   stating the title, year, authors, and publisher of the Document as
   given on its Title Page, then add an item describing the Modified
   Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for
   public access to a Transparent copy of the Document, and likewise
   the network locations given in the Document for previous versions
   it was based on. These may be placed in the "History" section.
   You may omit a network location for a work that was published at
   least four years before the Document itself, or if the original
   publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications",
   Preserve the Title of the section, and preserve in the section all
   the substance and tone of each of the contributor acknowledgements
   and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document,
   unaltered in their text and in their titles. Section numbers
   or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section
   may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements"
   or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or
appendices that qualify as Secondary Sections and contain no material
copied from the Document, you may at your option designate some or all
of these sections as invariant. To do this, add their titles to the
list of Invariant Sections in the Modified Version's license notice.

These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between

the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of

following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME.
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3

```
    or any later version published by the Free Software Foundation;
    with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
    A copy of the license is included in the section entitled "GNU
    Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts,
replace the "with...Texts." line with this:

    with the Invariant Sections being LIST THEIR TITLES, with the
    Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other
combination of the three, merge those two alternatives to suit the
situation.

If your document contains nontrivial examples of program code, we
recommend releasing these examples in parallel under your choice of
free software license, such as the GNU General Public License,
to permit their use in free software.
```