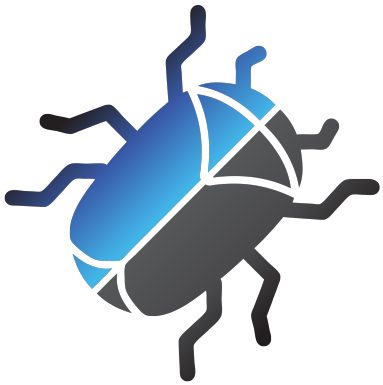
The scaRabee Package: An R-based Tool for Model Simulation and Optimization in Pharmacometrics

Sébastien Bihorel

May 5, 2010



Contents

1 Overview

scaRabee is a toolkit for modeling and simulation primarily intended for the field of pharmacometrics. It is a R port [6] of **Scarabee**, a Matlab-based application developed for the simulation and optimization of pharmacokinetic and/or pharmacodynamic models specified with explicit solutions, ordinary or delay differential equations [1].

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 the field are invited to read the excellent introductory and more advanced books from Gabrielsson and Weiner [4], Bonate [2] or Ette and Williams [3].

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 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 term 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/`).

1.3 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 [5] 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 1.4) relies on functions from the **neldermead**, **optimsimplex**, and **optimbase** packages, which is also distributed from CRAN and Pmlab as compressed archives of the source packages.

1.4 Methodology

scaRabee allows the optimization and simulation of non-linear systems at the individual level and does not yet implement non-linear mixed effects modeling. The analysis of population data must therefore be performed through the naïve averaging and naïve pooling methods [3]. Note that a standard two-stage approach is also possible but must be implemented manually.

Model parameters are optimized by the method of the maximum likelihood, more precisely by the minimization of an objective function defined as minus twice the exact 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 is performed as described in the Adapt II software user's manual written by D'Argenio and Schumitzky [7].

scaRabee allows you to split your analysis problem into as many sub-problems as you need, while still using 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 explicit or differential equations usually needed to accommodate the different dose levels or regimens.

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** function, documentation, installation, anything. Please email them to `sb.pmlab@gmail.com`.

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

- the version number of **scaRabee** and the program(s) or manual involved.
- hardware and operating system names and versions.
- the contents of any input, model, weighting or secondary parameter 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 Getting started

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

`initials.csv`, and `covariates.csv`, as well as the `model.definition` sub-folder. While these file and folder names correspond to the default names assumed by `scarabee.new`, they can be modified at the discretion of the user. However, these names will be used thereafter to refer to those particular files and folders, for the sake of simplicity. Finally, please note that you can add any file needed or not for your analysis in your analysis folder.

myanalysis.R

The master R script that you need to execute to perform your analysis. Section 2.9 describes which parts of the code must be edited.

data.csv

A comma-separated table containing the times and values of observed data to be fitted or matched against a model simulation. Section 2.2 describes how this data must be specified.

dosing.csv

A comma-separated table containing the inputs assigned to your model states – typically expressed as bolus doses or infusions in pharmacokinetic-pharmacodynamic models. Section 2.3 describes how these inputs must be specified.

initials.csv

A comma-separated table containing the names and values of your model parameters, used as initial guesses for model optimization, or as inputs for model simulation. Section 2.4 describes how these parameters must be specified.

covariates.csv

A comma-separated table containing the times and values of the observed covariates to your model. Section 2.5 describes how this data must be specified.

model.definition

A folder containing three R scripts which define the structure of your model (`model.R`), the residual variability model (`weighting.R`) and potential secondary parameters (`secondary.R`). Sections 2.6, 2.7, and 2.8 describe how those files should be edited.

The sections 2.1 through 2.10 offer a step-by-step description of the analysis process.

2.1 Creation of a new analysis folder

Although it is theoretically possible to store multiple analyses within the same directory, users are advised to create a new folder for each analysis. This folder can be located in any directory in your local system where you have read/write permissions. There are two ways to proceed depending at which point of a global analysis the folder is created.

2.1.1 New analysis

If you start a brand new data analysis, it is recommended that you open an interactive R session, change the working directory to the desired location using `setwd` and use `scarabee.new` to create a new folder that will contain `myanalysis.R`, `data.csv`, `dosing.csv`, `initials.csv`, and `covariates.csv`, plus the `model.definition` sub-folder with `model.R`, `weighting.R` and `secondary.R`.

It is recommended that you provide part or all arguments of `scarabee.new` to better set up the new folder:

name controls the name of the new folder and is also used to 'rename' `myanalysis.R`,

path defines where the new folder is created (in case you do not want to use the current working directory),

type defines whether the analysis is a simulation or an estimation run,

template controls which template will be used for `model.R`; templates are available for models defined with closed form solution, ordinary differential equations or delay differential equations,

with.inputs defines whether or not instances of `data.csv`, `dosing.csv`, `initials.csv` and `covariates.csv` should be created in the folder.

```
> require(scaRabee)
> setwd("your/target/directory/")
> scarabee.new(name = "myanalysis", path = getwd(), type = "simulation",
+     template = "ode", with.inputs = TRUE)
```

2.1.2 New model in an on-going analysis

If you want to test an alternative to an exciting model or perform a different type of run, it is recommended that you create a new analysis folder. For this purporse, you can either use `scarabee.new` (with probably the `with.inputs` set to false) or copy/paste the existing folder to a different location. With the latter option, only minor edits to the problem files are needed.

Regarless of the chosen method, part or all analysis files require some form of modifications that are described in section 2.2 through 2.9. Symbols and notation used in those sections are summarized in Table 1.

Symbol	Definition
p	Number of parameters to be estimated
s	Number of secondary parameters to be estimated
k	Number of sub-problems
n	Number of system states or outputs
n_o	Number of states with observations
l	Number of delays evaluated for the solution of a system of delayed differential equations
d_i	Number of dosing events in the i^{th} subproblem
m_i	Number of observation times in the i^{th} subproblem
c	Number of covariates
t_i	Number of covariate observation times in the i^{th} subproblem

Table 1: Symbol Definition for Vector and Matrix Dimensions

2.2 Edition of data.csv

The `data.csv` file contains the data to be modeled or matched against a model simulation. This file is required in both estimation and simulation runs, and can be edited in any text editor or spreadsheet. All modified versions of this file must respect the comma-separated value format but can be saved under any user-defined name (the .csv extension is compulsory though). The content of `data.csv` must be a full rectangular table, with the following structure:

- The first line must contain the headers of each column of your data table.
- The columns must be ordered as follows:

  $$\text{Dose ID, Time, Y(1), Y(2), } \ldots \text{, Y}(n_o)$$

  where Dose ID is the column of sub-problem identifiers (see below), Time is the column of the observation times, Y(i) is the column of observations associated to the i^{th} model output or state, and n_o is the number of system outputs or states with observations.
- All lines should contain n_o elements separated by commas. Dose ID and Time data cannot be missing, while the values in the observation columns could be missing (`NA`).
- The Dose ID column must contain integer numbers, typically in **increasing** order from 1 to k, the total number of different sub-problems (typically dose levels or regimens) included in the analysis. All records with a similar Dose ID will be considered as part of the same sub-problem. It is recommended that you enter all data from the same sub-problem in continuous records.
- The Time column must contain real numbers in the **increasing** order. Failing to sort your data by Dose ID and Time is likely to cause **scaRabee** to return an error message or an erroneous output.
- Y(i) columns must contain real number or missing data indicators (`NA`) corresponding to the observation values for a given model output or state. A column could contain only missing data, although this is not recommended.

ALL `data.csv` FILES MUST CONTAIN AT LEAST A HEADER LINE AND TWO LINES PER SUB-PROBLEM, INDICATING THE BEGINNING AND THE END OF THE OBSERVATION INTERVALS.

2.3 Edition of dosing.csv

The `dosing.csv` file contains the information about the inputs assigned to the analyzed system. In pharmacokinetics and pharmacodynamics, inputs are typically doses of study drug and are specified using instantaneous inputs (thereafter referred as boluses) or constant zero-order rate inputs (thereafter referred as infusions). The structure of dosing.csv respects this convention. This file is required in both estimation and simulation runs and can be edited in any text editor or spreadsheet. All modified versions of this file must respect the comma-separated value format but can be saved under any user-defined name (the .csv extension is compulsory though). The content of `dosing.csv` must be a full rectangular table, 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 `dosing.csv` and should typically not be modified.
- There must be 5 columns, ordered as follows:

Dose ID, Time, State, Bolus, Infusion rate

where Dose ID is the column of sub-problem identifiers (see below), Time is the column of the dosing times, State is the column indicating in which model state (see below) the bolus or infusion enters the system, Bolus is the column of dose amount entering the system as an instantaneous bolus, Infusion rate is the column of zero-order rate at which the drug enters the system.

- All lines should contain 5 elements separated by commas. There cannot be any missing data in this table: if a bolus or an infusion rate is null at a given time, you should enter 0 in the proper column.
- The Dose ID column must contain integer numbers, typically in **increasing** order from 1 to k, the total number of different sub-problems (typically dose levels or regimens) included in the analysis. All records with a similar Dose ID will be considered as part of the same sub-problem. It is recommended that you enter all data from the same sub-problem in continuous records. Note that the list of unique integers used in `dosing.csv` must match the list of unique integers used in `data.csv`.
- The Time column must contain real numbers in the **increasing** order. Failing to sort your data by Dose ID and Time is likely to cause **scaRabee** to return an error message or an erroneous output.
- The State column must contain integer numbers indicating in which model state the bolus or infusion should enter the system. This information can be used or not, depending on how your model is defined (see 2.6). In some of the available model templates, it is used as a reference to automatically allocate the bolus and/or infusion to the appropriate state. State can be understood in many ways but, in the context of a system defined by ordinary of delayed differential equations, we identify a state to a given differential equation. For instance, if you want that an input enters the system via the 2nd differential equation at time 3.5 in the first sub-problem, you will have to create a line with '1,3.5,2' followed by the appropriate bolus amount or infusion rate. See example 3 in Section 3 for a better illustration of this concept. State loses this meaning when a model is specified with explicit solutions, but the information contained in this column could still be useful, if you want to automate the allocation of the input to a specific variable/equation.
- The Bolus column must contain real numbers, representing the amount of drug entering the system at a given time. More details about how this information could be used in your model can be found in the description of model templates in Section 2.6.
- The Infusion rate column must contain real numbers, representing the zero-order rate at which the drug enters the system at a given time. You have the ability to specify a non-zero bolus and an infusion rate at the same time. However, both inputs will pertain to the same state. More details about how this information could be used in your model can be found in the description of model templates in Section 2.6.

ALL `dosing.csv` FILES MUST CONTAIN AT LEAST CONTAIN A HEADER LINE.

2.4 Edition of initials.csv

The `initials.csv` file contains the information about your model parameters (Note that secondary parameters are defined in `secondary.R`). This file is required in both estimation and simulation

runs, and can be edited in any text editor or spreadsheet. All modified versions of this file 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 `initials.csv` must be a full rectangular table, 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, type and value, 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 (number 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, although an absorption lag-time can also be defined as a 'L' parameter.

  IC: indicates that the parameter is used to define an initial condition of a differential equation.

  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 `initials.csv` FILES MUST CONTAIN AT LEAST A HEADER LINE AND ONE PARAMETER DEFINITION LINE.

2.5 Edition of covariates.csv

The `covariate.csv` file contains the observed covariate data that could be used or not to define a model. This file is required in both estimation and simulation runs, and can be edited in any text editor or spreadsheet. All modified versions of this file 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 `covariate.csv` must be a full rectangular table, with the same following structure as `data.csv`:

- The first line must contain the headers of each column of your data table.
- The columns must be ordered as follows:

  $$\text{Dose ID, Time, COV(1),COV(2),}\ldots\text{, COV}(c)$$

  where Dose ID is the column of sub-problem identifiers, Time is the column of the observation times, COV(i) are the column of values taken by the i^{th} covariate, and c is the number of differentes covariates.
- All lines should contain as many elements as the number of column headers, separated by commas. Dose ID and Time data cannot be missing, while the values in the covariate columns could be missing (`NA`) in one or more columns.
- The Dose ID column must contain integer numbers, typically in **increasing** order from 1 to k, the total number of different sub-problems (typically dose levels or regimens) included in the analysis. All records with a similar Dose ID will be considered as part of the same sub-problem. It is recommended that you enter all data from the same sub-problem in continuous records. Note that the list of unique integers used in `dosing.csv` must match the list of unique integers used in `data.csv`.
- The Time column must contain real numbers in the **increasing** order. Failing to sort your data by Dose ID and Time is likely to cause **scaRabee** to return an error message or an erroneous output.
- The COV(i) column must contain real number or the missing data indicator (`NA`) corresponding to the values taken by the i^{th} covariate. A column could contain only missing data, but it is recommended to delete the column in such case. You can change the header of each covariate data to your convenience.

ALL `covariates.csv` FILES MUST AT LEAST CONTAIN A HEADER LINE.

2.6 Edition of model.R

The `model.R` file, found in the sub-folder model.definition, is a required R script, in which you define the structure of the analyzed model. The purpose of this script is to output a matrix of predictions for each sub-problem of the analysis. The methods used to create this matrix are at your entire discretion, although we recommend that you use the template provided with `scarabee.new`. You can also rename this script if necessary. The structure of the input arguments and function output is described below.

Arguments

x a p x 6 data.frame containing information about the model parameters for the evaluation of the model. The columns `names`, `type`, `value`, `isfix`, `lb`, and `ub` contain the names, the type, the value, the estimation status, and the lower and upper bounds for the p system parameters, respectively. The numbers stored in `value` are the values defined in the `initials.csv` file for a simulation run or the first iteration of an estimation run (see Section 2.9), or the values updated by the optimization algorithm for any other iteration of an estimation run.

dosing a d_i x 4 matrix of dosing history. Dosing event times, assignement states, bolus amounts and infusion rates are stored in the 1^{st}, 2^{nd}, 3^{rd}, and 4^{th} columns, respectively.

xdata a 1 x m_i matrix of observations times, where $m_i \geq 2$

covdata a t_i x $(c+1)$ matrix of covariate data. The observation times are stored in the 1^{st} column of `covdata`, while the actual values of each covariate are stored in the remaining columnds in the order they appeared in `covariates.csv`.
Although this feature is not yet documented in this manual, the value of covariates could be interpolated from `covdata` at times where those metrics were not measured, using linear interpolation (see `?approx`).

issim a scalar, indicating whether the model is evaluated in a simulation (1) or estimation (0) run.

Value

For estimation runs, a n x m_i matrix of model predictions; where n is the number of model outputs and m_i the total number of observations times in the i^{th} sub-problem. For simulation runs, a $(n+1)$ x m_i matrix of prediction times and model predictions (the first row must contain the prediction times).

You can specify the structure of the model using any available R function in order to produce this output. The solvers for ordinary and delayed differential equations from the **deSolve** and **PBSddesolve** packages are especially useful. You should refer to the help of those packages for more information. Model templates provided by `scarabee.new` (see below) and specific application examples (see Section 3) were created to guide you through the process of model creation in **scaRabee**. Although you have the freedom to write your own code from start to finish, we recommend that you base your code on those templates, in order to minimize the risk of incompatibilities between your code and **scaRabee** scripts.

2.6.1 Template for models specified with closed form solutions

This template can be accessed to from an existing script or by creating a new analysis folder with `scarabee.new` and setting the `template` argument to 'explicit':

```
> require(scaRabee)
> setwd("your/target/directory/")
> scarabee.new(name = "myanalysis", path = getwd(), type = "simulation",
+     template = "explicit", with.inputs = TRUE)
```

The section delimited by the `USER CODE STARTS HERE` and `USER CODE ENDS HERE` tags indicates the portion of the script where you can enter your own code. Within this section, you can make use of any function inputs (see Section 2.6) and directly access to model parameters using the names specified in `initials.csv` (see Section 2.4). Please, consider looking at examples 1 and 2 in Section 3 which illustrate the use of this template.

2.6.2 Template for models specified with ordinary differential equations

This template can be accessed to from an existing script or by creating a new analysis folder with `scarabee.new` and setting the `template` argument to 'ode':

```
> require(scaRabee)
> setwd("your/target/directory/")
> scarabee.new(name = "myanalysis", path = getwd(), type = "simulation",
+     template = "ode", with.inputs = TRUE)
```

Provided that you do not alter the code outside the sections delimited by the `USER CODE STARTS HERE` and `USER CODE ENDS HERE` tags, bolus and infusion inputs are automatically allocated to the states defined in the `dosing.csv` file and scaled. Please, consider looking at examples 3 and 4 in Section 3 which illustrate the use of this template.

Bolus inputs

Bolus inputs are allocated by splitting the global integration intervals into several continuous integration intervals based upon the dose event times. The initial conditions of the system are updated for each integration interval using the state values at the end of the previous interval (or the specified initial conditions in the case of the first interval) and the bolus amount and state specified in `dosing.csv`. Therefore, all model predictions made at the time of a bolus assume that this bolus has entered the system. Therefore, the user is 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

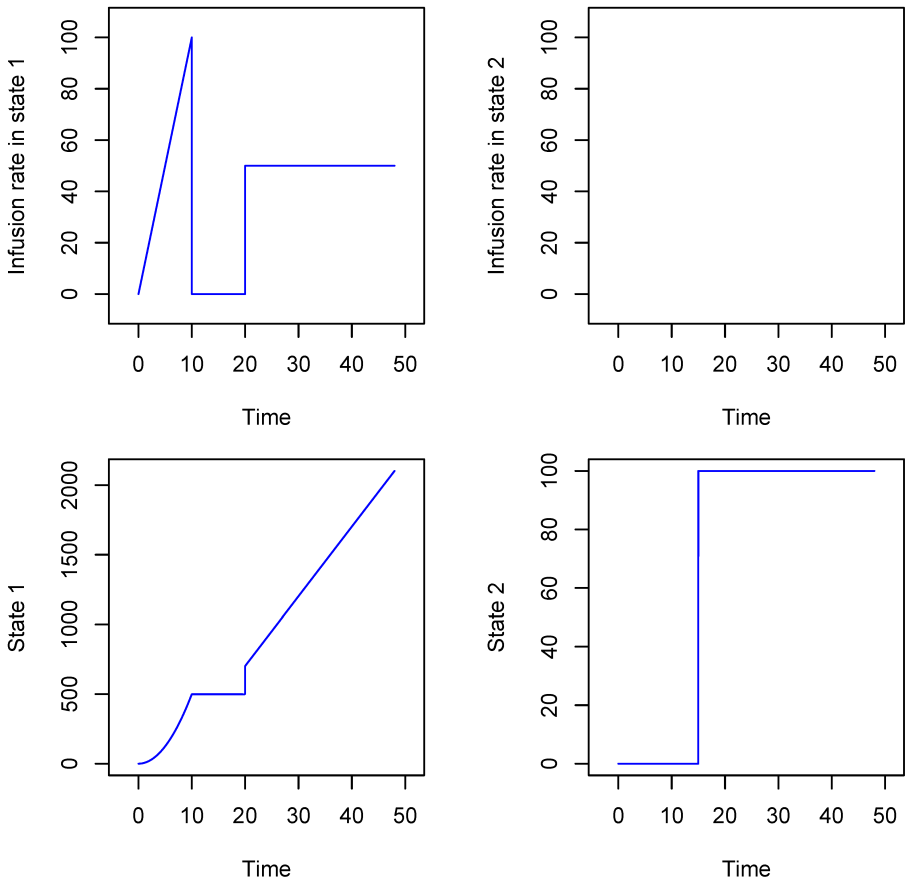
Infusion inputs are allocated into the system of differential equations as zero-order rate to the state defined in the `dosing.csv`. Given that infusion rates are defined within a time window in practice but at specific times in `dosing.csv`, a standard of interpretation of this file is necessary. Here is how the template code interprets the content of dosing.csv for the i^{th} sub-problem, assuming that t is the time at which the system of differential equations is evaluated:

- all dosing events are first split by state; then, for each state,
- if t is lower than the first dose event, the rate is set to zero,
- if t is higher than the last dose event, the rate is set to the last values assigned to the state,
- if t is exactly the time of one or more dose events, the rate is set to the value specified at the last dose event at time t,
- if t is not a exact dose event time, the rate is linearly interpolated between the closest rates specified before and after t.

The following example illustrates this set of rules. Let's assume that the system is specified by two ordinary differential equations, both fixed to zero, and that the tabulated content of `dosing.csv` is as follows:

Dose ID	Time	State	Bolus	Infusion Rate
1	0	1	0	0
1	10	1	0	100
1	10	1	0	0
1	15	2	100	0
1	20	1	0	0
1	20	1	200	50

State 1 receives 1 bolus dose at time 20 and 2 infusions: the first, between 0 and 10, has a rate increasing from 0 to 100, and a second, which is constant, starts at time 20 and does not stop. 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).



Input scaling

All bolus and infusion inputs can be scaled by the user (see how in the next section). This is particularly useful when the dimension 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).

User-defined code

New instances based on the 'ode' template should be edited at 4 different sections of the code. Each section is delimited by the `USER CODE STARTS HERE` and `USER CODE ENDS HERE` tags.

- Lines 174-182: this section is part of the `init` function which defines the initial conditions of the system of differential equations. These initial conditions are specified in the `init` variable which must should be a vector of length n (typically), where n is the number of states (i.e. differential equations). The first argument of this function is `parms` which allows you to access to the model parameters using the names specified in `initials.csv` (see Section 2.4), including the 'IC' parameters.

- Lines 209-221: this section is part of the `inputscaling` function, which defines how bolus and infusion inputs are scaled. Scales are specified in the `scale` variable which must be a scalar or have the same dimension as `dydt`, the output of the `odesyst` function (see below). In the latter case, the user should set `scale[i]` to 0 if there is no input in the i^{th} state. The first argument of this function is `parms` which allows you to access to the model parameters using the names specified in `initials.csv` (see Section 2.4).

- Lines 263-274: this section is part of the `odesyst` function, where the system of differential equations is specified. The system must be defined in `dydt` which should be a vector of length n. The inputs to the `odesyst` function are:

  - `t`, a scalar representing the time at which the system is being evaluated,
  - `y`, a vector of n states values at time `t`,
  - `parms`,
  - `dosing`,
  - `covdata`, and
  - `scale` (see above).

  You can refer to model parameters using the names specified in `initials.csv` (see Section 2.4). State values can be refered to as *e.g.* y1 or y2. Exogenous inputs (boluses and infusions) should NOT be added into the system of differential equations, this is already being taken care of by the code.

- Lines 316-324: this section is part of the `output` function, where you define the structure of the output. The `y` variable is the output of the `output` and model functions, it must therefore be a n x m_i matrix. The inputs to the `output` function are `f`, a n x m_i matrix which gives the state predictions at all observed times, `parms`, `dosing`, and `xdata`.
  For a simulation analysis, you would typically want the complete system to be output and would just specify `y <- f` at line 320.
  For an estimation analysis, you must extract the desire states from the `f` variable or construct a new variable based on `f` (e.g. if one set of observations can be defined using a closed form solution) to match the observations. For instance, if 3 sets of observations are available and correspond to the 1^{st}, 3^{rd}, and 4^{th} state, you would write at line 320:
  `y <- rbind(f[1,],f[3,],f[4,])`

2.6.3 Template for models specified with delayed differential equations

This template can be accessed to from an existing script or by creating a new analysis folder with `scarabee.new` and setting the `template` argument to 'dde':

```
> require(scaRabee)
> setwd("your/target/directory/")
> scarabee.new(name = "myanalysis", path = getwd(), type = "simulation",
+     template = "dde", with.inputs = TRUE)
```

Provided that you do not alter the code outside the sections delimited by the `USER CODE STARTS HERE` and `USER CODE ENDS HERE` tags, bolus and infusion inputs are automatically allocated to the states defined in the `dosing.csv` file and scaled. Please, consider looking at example 5 in Section 3 which illustrates the use of this template.

Bolus inputs

Bolus inputs are considered as discontinuous events in the system as allowed by the `dde` function (see help of **PBSddesolve**) and are actually implemented with the `mySwitch` and `myMap` functions. Briefly, discontinuous events are detected by `mySwitch` and take place at times `t` where:

$$\begin{cases} mySwitch(t) = 0 \\ \dfrac{\partial mySwitch}{\partial t} < 0 \end{cases}$$

Bolus events are detected in the template by implementing a piecewise linear signal for the switch function. The `get.switch.vectors` function allows the creation of a "matrix" of signal values (1 or -1) symmetrically around the known bolus times. Then the `approx` function is used within `mySwitch` to interpolate the signal between those reference points: at bolus times, the interpolated signal is 0 and has a negative slope. The addition of amounts in system states is actually performed in `myMap`. Overall, all model predictions made at or extremelly close to the time of a bolus assume that this bolus has entered the system. The user is 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

Infusion inputs are allocated into the system of differential equations as zero-order rates to the states defined in the `dosing.csv`, using the same rules as defined in Section 2.6.2.

Input scaling

All bolus and infusion inputs can be scaled by the user (see how in the next section).

User-defined code

New instances based on the 'dde' template should be edited at 5 different sections of the code. Each section is delimited by the `USER CODE STARTS HERE` and `USER CODE ENDS HERE` tags.

- Lines 145-153: this section is part of the `init` function which defines the initial conditions of the system of differential equations. These initial conditions are specified as described in 2.6.2.

- Lines 180-191: this section is part of the `inputscaling` function, which defines how bolus and infusion inputs are scaled. Scales are specified in the `scale` variable which must be a scalar or have the same dimension as `dydt`, the output of the `odesyst` function (see below). In the latter case, the user should set `scale[i]` to 0 if there is no input in the i^{th} state. The first argument of this function is `parms` which allows you to access to the model parameters using the names specified in `initials.csv` (see Section 2.4).

- Lines 222-231: this section is part of the `lags` function where the delays used in system of differential equations are defined. The only input variable of this function is `parms`, which allows you to access to the model parameters using the names specified in `initials.csv` (see Section 2.4), including the 'L' parameters. The `delays` output of the `lags` function must be a vector of l elements.
  Note that all elements of `delays` must have names, as illustrated below:

  ```
  delays <- c(lag1=1, lag2=2.5, lag3=myparam)
  ```

- Lines 294-305: this section is part of the `ddesyst` function, where the system of differential equations is specified. The system must be defined in `dydt` which should be a vector of length n. The inputs to the `odesyst` function are:

  - `t`, a scalar representing the time at which the system is being evaluated,
  - `y`, a vector of n states values at time `t`,
  - `parms`, a list containing the following elements: `parms`, the vector of model parameters, `lags`, the vector of delays, `dosing`, the matrix of dosing history, `xdata`, the vector of observation times, `covdata`, the matrix of covariate observations, `scale`, the vector of input scale factors, `times`, the vector of reference times for approximation function, `signal`, the vector of reference signal for approximation function, and `ic`, the vector of initial conditions

  You can refer to model parameters using the names specified in `initials.csv` (see Section 2.4). State values at time `t` can be referred to as *e.g.* y1 or y2; the value of the i^{th} state at time `t-lag3` (as defined above) can be referred to as `ylag.lag3[i]`. Exogenous inputs (boluses and infusions) should NOT be added into the system of differential equations, this is already being taken care of by the code.

- Lines 294-305: this section is part of the `output` function, where you define the structure of the output. The `y` variable output should be specified as explained in Section 2.6.2.

2.7 Edition of weighting.R

The `weighting.R` file, found in the sub-folder model.definition, is a required R script, in which you define the structure of the residual variability model. The purpose of the `weighting` function is to provide a matrix of values, defining the residual variability associated to each model prediction. Therefore, the `v` output must have the same dimensions as the output of the `model.R` file, *i.e.* a n x m_i matrix for each sub-problem of the analysis. The function inputs are:

x the same list of data as the one passed to the `model` function (see Section 2.6),

f a n x m_i matrix containing the model predictions,

xdata the same 1 x m_i matrix of observation times, as the one passed to the `model` function (see Section 2.6).

`v[i,j]` must be the variance associated to the model prediction `f[i,j]`. The commented section at the top of the `weighting.R` template details the syntax that should be used for typical residual variability models (additive, proportional, additive + proportional). You can define `v` within a dedicated section between lines 37 and 58.

2.8 Edition of secondary.R

The `secondary.R` file, found in the sub-folder model.definition, is a required R script, in which you can define how secondary parameters are calculated. If no secondary parameter is needed, the `secondary.R` could be left as provided by `scarabee.new`. The `sec` output is a vector of s secondary parameters that must be specified between lines 41 and 54. The only input to the secondary function is x, the same list of data containing information about the model parameters, as the one passed to the model function 2.6.

2.9 Edition of myanalysis.R

The `myanalysis.R` file is the master program that you must execute to perform your analysis. You must modify several lines in a specific section of the template (from line 20 to line 59) to define the particularities of your analysis. Any other line of this file should typically not be modified. Commented lines with the user-editable section explain what and how variable(s) should be defined.

- Lines 37-43: `data`, `param`, `dose`, `cov`, `model`, `var`, and `sec` variables are character variables defining the names of the files where your data, parameters, dosing scheme, covariates, model, residual variability, and secondary parameters are respectively defined. The default values correspond to name of appropriate files created by `scarabee.new`. You can change those names, if necessary.

- Line 46: the `states` variable is a vector of numeric values, representing the indices of the model output corresponding to the various observation columns in `data.csv`, and is only used for simulation runs (see below).

  Example: Let's assume that the output of the `model.R` file is a 5 x 15 matrix, *i.e.* a matrix of 15 predictions for 5 states, and that `data.csv` contains 3 observation columns Y(1), Y(2), and Y(3), with 15 row each. If line 46 is defined as:

  ```
  states <- c(2, 4, 5)
  ```

  scaRabee associates the data in Y(1) to the 2^{th} row of the prediction matrix, data in Y(2) to the 4^{th} row of the prediction matrix, and data in Y(3) to the 5^{th} row of the prediction matrix.

- Line 49: the `runtype` variable is a character variable, defining if the analysis is an estimation or a simulation run. Any other character string than 'simulation' or 'estimation' will cause an early termination of the run and the display of an error message to the console.

- Line 57: the optimization algorithm is designed to return an infinite objective function in case the computation of the objective function at a given point of the multi-dimensional search space returns an error message. This is to prevent R to stop 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 you 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.

- Line 56-57: `estim` is a list a scalar variables `maxiter` and `maxfunc`, defining the maximum number of iterations and function evaluations during an estimation run. They can only contain integer numbers. The default values are 500 and 5000, which should typically allow your problem to converge to a stable point of the search space.

2.10 Execution of myanalysis.R

Once all necessary files have been edited, the analysis can be performed by executing the master `myanalysis.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. 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 `myanalysis.R` file of interest, then run the analysis by typing:

  `R CMD BATCH myanalysis.R`

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

<myanalysis>.<type>.#

where <myanalysis> is the base file name of the master R script, <type> is `sim` for simulation runs and `est` for estimation runs, and # is a two-digit integers.

At the exception of the .Rout file in iteractive mode, all run output are stored in the newly created folder. Additionally, a sub-folder called 'run.config.files' is created to backup all analysis files (`data.csv`, `dosing.csv`, `initials.csv`, `covariates.csv`, `model.R`, `secondary.R`, and `weighting.R`).

In interactive mode, the run progression will be reported on screen, 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 `myanalysis.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 your run, you should be able to see (in interactive mode) as many figures as the number of sub-problems specified in `data.csv` and `dosing.csv`. Those figures represent the predicted changes in all selected outputs overlaid with 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 selected 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 sub-problem.

Estimation runs

Upon successful completion of your run, you should be able to see (in interactive mode) a figure summarizing the changes in the objective function and the estimated parameter values as a function of the iteration, plus, for each sub-problem, 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). All figures are stored in the newly created folder.

A file called `<myanalysis>.report.txt` file is also saved in the same folder and provides a summary of the estimation run, a summary table of final parameter estimates associated with coefficient of variation and confidence intervals (calculated as described in [7]), 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 [7]).

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.

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

3 Analysis examples

This section is designed to illustrate some selected features of **scaRabee** and show how the provided templates can be used to perform some pharmacokinetic analyses. The five examples described below are available as demos using the following calls:

```
> demo(example1, package = "scaRabee", echo = FALSE)

> demo(example2, package = "scaRabee", echo = FALSE)

> demo(example3, package = "scaRabee", echo = FALSE)

> demo(example4, package = "scaRabee", echo = FALSE)

> demo(example5, 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.

These examples are also intended for validation. The results of examples 1, 2 and 4 can be compared to the same analyses run with the ADAPT V software from D'Argenio and Schumitzky (models & outputs can be downloaded from `http://code.google.com/p/pmlab/`). We refer the reader to the User's manual of this software for any related question [7].

3.1 Example 1

The analysis proposed in example 1 is the simulation of a very simple pharmacokinetic system: a single oral dose is modeled by a 1-compartment model with non-delayed linear absorption and elimination. Dosing and dummy observation data can be read from the `dosing.csv` and `data.csv` files. There is no covariate in this system, so the `covariate.csv` file was left blank (except for

the column headers). The system parameter values used for the simulation can be found in the `initials.csv` file.

This model is implemented using a closed-form solution, i.e. the Bateman function [4]. The noteworthy features of this model are how the dose information is extracted from the dosing input variable on line 84, how model parameters are directly used to compute the model predictions on line 87. Note also that the `weighting.R` was not modified: the residual variability model is not used during a simulation run, it is therefore unnecessary to define it.

The execution of the master `example.1.R` program should produce a single graph output and a .csv report file. For comparison/validation purposes, the analysis was repeated in ADAPT V. The results of this analysis are stored in the adapt5_example1 sub-folder contained the downloaded archive. You are invited to compare the results of the **scaRabee** analysis to those obtained with ADAPT V (`ex1.run` and `ex1.plt`). You should observe that the time versus predicted concentrations curves are exactly superimposed, if plotted together.

3.2 Example 2

The analysis proposed in example 2 is based on the same system as the one used in example 1, except that an optimization of the system parameters is performed. A proportional variability model is defined in `weighting.R`. The execution of the master `example.2.R` program should produce 3 graph outputs, 2 .csv files, and 1 .txt report.

For comparison/validation purposes, the analysis was repeated in ADAPT V. The results of this analysis are stored in the adapt5_example2 sub-folder contained in the downloaded archive. You are invited to compare the results of the **scaRabee** analysis to those obtained with ADAPT V (`ex2.run` and `ex2.plt`). You should observe that the parameter estimates and related statistics are extremely close, showing minor differences in the optimization algorithm between the two programs.

To validate the method of computation of the covariance matrix(ces) and related statistics (coefficient of variation, and confidence intervals) used in **scaRabee** (the same as in ADAPT V), we performed a second analysis in ADAPT V, using the final estimates obtained in **scaRabee** and setting the number of estimation to 0. You are invited to compare the results of the **scaRabee** analysis to those obtained with ADAPT V (`ex2scarabee.run`). You should observe that the statistics of precision associated to all primary and secondary parameters are identical.

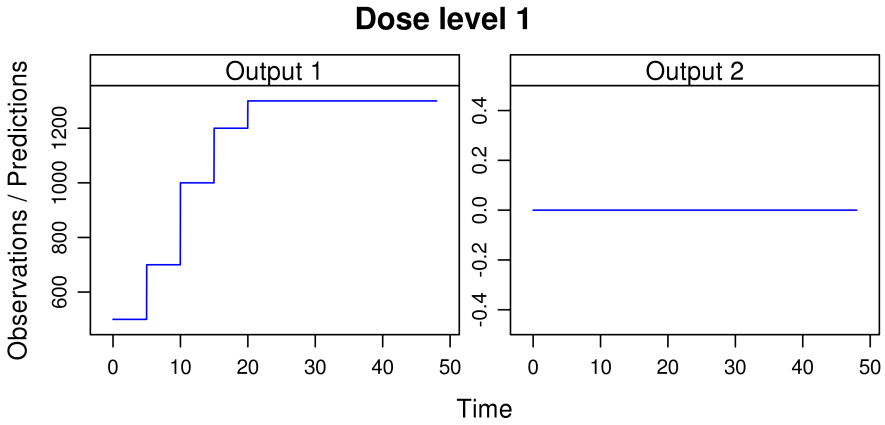
3.3 Example 3

The purpose of the example(s) 3 is to illustrate the automatic allocation of inputs to system states in models specified with the ordinary differential equations template. Three different dosing datasets are available in the directory: `dosing1.csv`, `dosing2.csv`, and `dosing3.csv`. The user is invited to execute the master `example.3.R` script several times, after changing the dosing file at line 39.

The model used in example(s) 3 is a system of 2 differential equations set to zero at all times. Therefore, the accumulation of drug in the 2 states can be easily visualized in the predictions versus time curves. Additionally, scale factors were set to 1 and 2 for the first and second state to illustrate how input scaling is performed in this template (at lines 194-195).

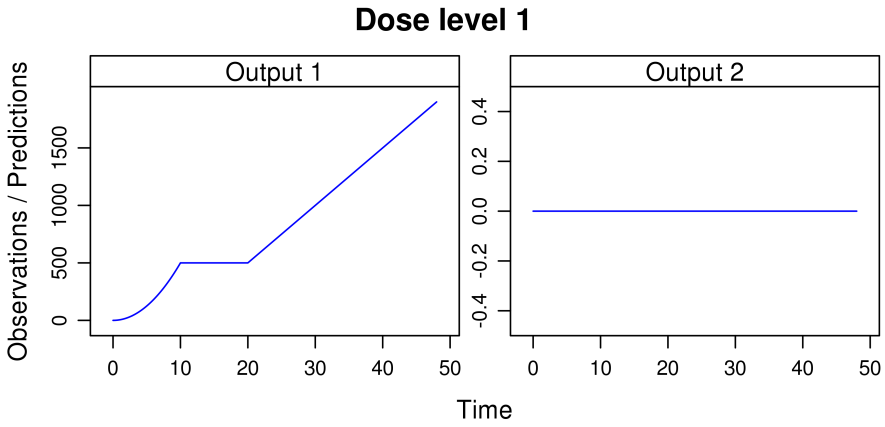
dosing1.csv

In this example, five instantaneous boluses are assigned to the 1^{st} state at different time. The produced time versus predictions curves should look like to following graphs:



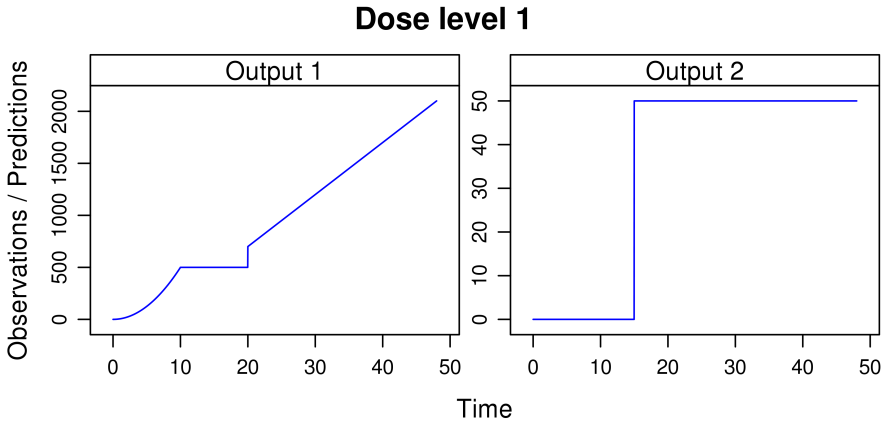
dosing2.csv

In this example, a 1^{st} time-variant rate infusion followed by a 2^{nd} time-invariant rate infusion are assigned to the 1st state. The produced time versus predictions curves should look like to following graphs:



dosing3.csv

This example features several dosing complexities: infusion with time-variant rate, simultaneous assignment of an instantaneous bolus and a time-invariant rate infusion, and allocation of inputs to more than one state. The produced time versus predictions curves should look like to following graphs:



3.4 Example 4

The purpose of the example 4 is to illustrate the estimation of the parameters of a non-linear model specified using ordinary differential equation, and to show how **scaRabee** decomposes a problem into several sub-problems defined by the Dose ID variable in dosing.csv and data.csv. You can see from the `dosing.csv` that three dosing regimens of instantaneous boluses are defined with different doses administered at different frequencies, but all are assigned to the 1^{st} system state. The `data.csv` shows that a rich sampling strategy was applied for the three groups/dose regimens. There was no covariate in this system, so the `covariate.csv` file was left blank (except for the column headers). The system parameter values used for the simulation can be found in the `initials.csv` file.

The model used in example 4 is a 2-compartment model with linear and non-linear elimination, and linear inter-compartmental distribution. The noteworthy features of this model are:

- the definition of the initial conditions using the model parameters on lines 160-161,
- the definition of the input scales on lines 194-195,
- the definition of intermediary variables using the model parameters on lines 233-235,
- the definition of the system of differential equations on lines 241-242, and
- the definition of the single model output at line 285.

The execution of the master `example.4.R` program should produce 7 graph outputs (2 per sub-problems +1), 2 .csv files, and 1 .txt report.

For comparison/validation purposes, the analysis was repeated in ADAPT V. The results of this analysis are stored in the adapt5_example4 sub-folder contained in the downloaded archive. You are invited to compare the results of the **scaRabee** analysis to those obtained with ADAPT V (`ex4.run` and `ex4.plt`). Like in the example 2, the parameter estimates and related statistics are close but not identical. **scaRabee** actually estimates the parameters with a better precision and provides estimates associated with a higher likelihood of the data than in ADAPT 5. Again, this is due to small differences in the optimization algorithms used in the two programs as well as small differences in the differential equation solvers.

3.5 Example 5

The analysis proposed in example 5 is the simulation of a system defined by delayed differential equations receiving 2 instantaneous boluses. The model was defined based upon the delayed differential equations template and consists in a 2-compartment model with linear elimination from the 1^{st} compartment and in which the return from the 2^{nd} to the 1^{st} compartment is delayed. This model is defined with a single output, *i.e.* the concentration in the 1st compartment. Dosing and dummy observation data can be read from the `dosing.csv` and `data.csv` files. There is no covariate in this system, so the `covariate.csv` file was left blank (except for the column headers). The system parameter values used for the simulation can be found in the `initials.csv` file. The main interesting features of this example are:

- the definition of the initial conditions using the model parameters on lines 130-131,
- the definition of input scales on lines 194-195,
- the definition if the unique model delay using the model parameters on line 200,
- how the states values at time t-delay is being used in the second differential equation at line 260, and
- the definition of the single model output at line 303.

The execution of the master `example.5.R` program should produce a single graph output and a .csv report file.

4 References

[1] Sébastien Bihorel. *Scarabee User's Manual*, 2009.

[2] Peter L. Bonate. *Pharmacokinetic pharmacodynamic modeling and simulation.* Springer, New York, NY, 2006.

[3] Ene I. (ed.) Ette and Paul J. (ed.) Williams. *Pharmacometrics: The science of quantitative pharmacology.* Hoboken, NJ: John Wiley & Sons. xix, 1205 p., 2007.

[4] Johan Gabrielsson and Daniel Weiner. *Pharmacokinetic/Pharmacodynamic data analysis: concepts and applications.* ApoteKarsocieteten, 4th edition, 2007.

[5] R Development Core Team. *R Installation and Administration.* R Foundation for Statistical Computing.

[6] R Development Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2009. ISBN 3-900051-07-0.

[7] Alan Schumitzky and David Z. D'Argenio. *ADAPT II Users'Guide: Pharmacokinetic/Pharmacodyanmic System Analysis Software.* Biomedical Simulations Resource, Los Angeles, CA, 1997.

5 Network of scaRabee functions

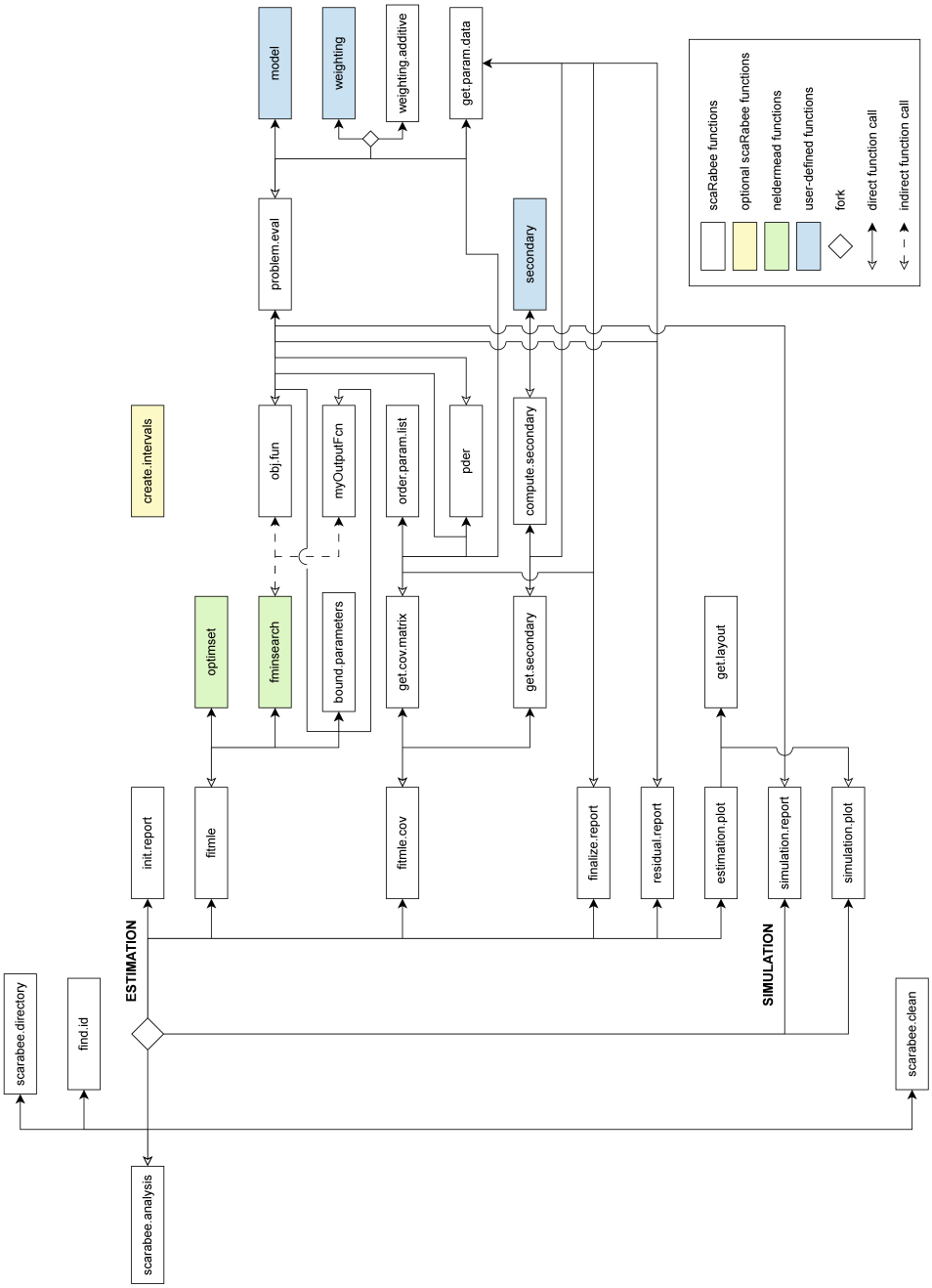


Figure 1: Map of the functions distributed with scaRabee

6 Help on scaRabee functions

`scaRabee-package` *scaRabee Toolkit*

Description

Framework for Pharmacokinetic-Pharmacodynamic Model Simulation and Optimization

Details

Package:	scaRabee
Type:	Package
Version:	1.0
Date:	2010-05-01
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 delayed 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.

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 function defined as the `subproblem$sec` argument at the initial and the final estimates of the model parameters.

Usage

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

Arguments

`subproblem` A list typically provided by `get.secondary` containing the following levels:

data A list containing the following levels:

xdata 1 x m matrix of independent variable.

ydata n x m matrix of observations from model states.

ids Data.frame of indices for data subsetting (output from `find.id`).

dosing A list containing the following levels:

history d x 4 data.frame of dosing history.

ids data.frame of indices for dosing subsetting (output from `find.id`).

cov A list containing the following levels:

data c x t data.frame of covariate history.

ids Data.frame of indices for cov subsetting (output from `find.id`).

states Indices of the states to be output by the model.

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.

varfun Variance function; if empty `weighting.additive` is used.

secfun Secondary parameter function.

`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.additive`, `fitmle`, `get.secondary`

`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>)

`estimation.plot` *Create Summary Estimation Plots*

Description

`estimation.plot` is a secondary function called at the end of the estimations runs. It generates plots from the iteration log file and the predictions & 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 sub-problem, 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.

Usage

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

Arguments

`problem` A list containing the following levels:

data A list containing the following levels:

- **xdata** 1 x m matrix of independent variable.
- **ydata** n x m matrix of observations from model states.
- **ids** Data.frame of indices for data subsetting (output from `find.id`).

dosing A list containing the following levels:

- **history** d x 4 data.frame of dosing history.
- **ids** data.frame of indices for dosing subsetting (output from `find.id`).

cov A list containing the following levels:

- **data** c x t data.frame of covariate history.
- **ids** Data.frame of indices for cov subsetting (output from `find.id`).

states Indices of the states to be output by the model.

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.

varfun Variance function; if empty `weighting.additive` is used.

secfun Secondary parameter 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.

delta The intervals used for the computation of confidence intervals.

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.

`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 observations of the dependent variable(s) to be modeled. 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')`.

dose A .csv file located in the working directory, which contains the dosing information. The expected format of this file is described in details in `vignette('scaRabee',package='scaRabee')`.

cov A .csv file located in the working directory, which contains the values of one or more covariates that may or may or be used within the model. The expected format of this file is described in details in `vignette('scaRabee',package='scaR`

model A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model. Models specified with explicit, ordinary or delayed differential equations should be preferentially defined using the provided templates. More details about the expected structure of this file is provided in `vignette('scaRabee',package='scaRabee')`, in case the user would want to develop her/his own template.

var A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model of residual variability. More details about the expected content of this file is provided in `vignette('scaRabee',package='scaRabee'`

sec A .R file located in the 'model.definition' sub-directory in the working directory, which defines the method of computation of secondary parameters (derived from the fixed or estimated model parameters). More details about the expected content of this file is provided in `vignette('scaRabee',package='scaR`

Author(s)

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

`finalize.report` *Finalize Estimation Report*

Description

`finalize.report` is a secondary function called at the end of the estimations runs. It outputs to the report file the final parameter estimates for strutural model parameters, residual variability and secondary parametes as well as the related statistics (coefficients of variation, confidence intervals, covariance and correlation matrix)

Usage

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

Arguments

`problem` A list containing the following levels:

data A list containing the following levels:

xdata 1 x m matrix of independent variable.

ydata n x m matrix of observations from model states.

ids Data.frame of indices for data subsetting (output from `find.id`).

dosing A list containing the following levels:

history d x 4 data.frame of dosing history.

ids data.frame of indices for dosing subsetting (output from `find.id`).

cov A list containing the following levels:

data c x t data.frame of covariate history.

ids Data.frame of indices for cov subsetting (output from `find.id`).

states Indices of the states to be output by the model.

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.

varfun Variance function; if empty `weighting.additive` is used.

secfun Secondary parameter 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.

delta The intervals used for the computation of confidence intervals.

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.

`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 observations of the dependent variable(s) to be modeled. 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')`.

dose A .csv file located in the working directory, which contains the dosing information. The expected format of this file is described in details in `vignette('scaRabee',package='scaRabee')`.

cov A .csv file located in the working directory, which contains the values of one or more covariates that may or may or be used within the model. The expected format of this file is described in details in `vignette('scaRabee',package='scaR`

model A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model. Models specified with explicit, ordinary or delayed differential equations should be preferentially defined using the provided templates. More details about the expected structure of this file is provided in `vignette('scaRabee',package='scaRabee')`, in case the user would want to develop her/his own template.

var A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model of residual variability. More details about the expected content of this file is provided in `vignette('scaRabee',package='scaRabee'`

sec A .R file located in the 'model.definition' sub-directory in the working directory, which defines the method of computation of secondary parameters (derived from the fixed or estimated model parameters). More details about the expected content of this file is provided in `vignette('scaRabee',package='scaR`

Author(s)

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

See Also

`weighting.additive`, `find.id`

find.id — *Scan for Integers in a Sorted Vector and Return Table of Min and Max Indices*

Description

`find.id` is a secondary function, which main purpose is to split the problem into sub-problems. It scans a vector of sorted integers (i.e. the `DoseID` variable from the data, dosing and covariate input files) and returns a table indicating between indices a particular integer is used.

Usage

```
find.id(x)
```

Arguments

`x`	A vector of sorted integers starting at 1 and increasing to n.

Value

Returns a data.frame with the following columns:

ID the unique integers of `x`.

starting first indice of `x` where the corresponding integer is used.

ending last indice of `x` where the corresponding integer is used.

Author(s)

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

Examples

```
x <- rep(1:3,each=5)
x
find.id(x)
```

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.

Usage

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

Arguments

`problem` A list containing the following levels:

data A list containing the following levels:

xdata 1 x m matrix of independent variable.

ydata n x m matrix of observations from model states.

ids Data.frame of indices for data subsetting (output from `find.id`).

dosing A list containing the following levels:

history d x 4 data.frame of dosing history.

ids data.frame of indices for dosing subsetting (output from `find.id`).

cov A list containing the following levels:

data c x t data.frame of covariate history.

ids Data.frame of indices for cov subsetting (output from `find.id`).

states Indices of the states to be output by the model.

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.

varfun Variance function; if empty `weighting.additive` is used.

secfun Secondary parameter 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.

delta The intervals used for the computation of confidence intervals.

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.

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.param.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 twice the log of the exact 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.

Usage

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

Arguments

`problem` A list containing the following levels:

data A list containing the following levels:

xdata 1 x m matrix of independent variable.

ydata n x m matrix of observations from model states.

ids Data.frame of indices for data subsetting (output from `find.id`).

dosing A list containing the following levels:

history d x 4 data.frame of dosing history.

ids data.frame of indices for dosing subsetting (output from `find.id`).

cov A list containing the following levels:

data c x t data.frame of covariate history.

ids Data.frame of indices for cov subsetting (output from `find.id`).

states Indices of the states to be output by the model.

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.

varfun Variance function; if empty `weighting.additive` is used.

secfun Secondary parameter 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 observations of the dependent variable(s) to be modeled. 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')`.

dose A .csv file located in the working directory, which contains the dosing information. The expected format of this file is described in details in `vignette('scaRabee',package='scaRabee')`.

cov A .csv file located in the working directory, which contains the values of one or more covariates that may or may or be used within the model. The expected format of this file is described in details in `vignette('scaRabee',package='scaR`

model A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model. Models specified with explicit, ordinary or delayed differential equations should be preferentially defined using the provided templates. More details about the expected structure of this file is provided in `vignette('scaRabee',package='scaRabee')`, in case the user would want to develop her/his own template.

var A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model of residual variability. More details about the expected content of this file is provided in `vignette('scaRabee',package='scaRabee'`

sec A .R file located in the 'model.definition' sub-directory in the working directory, which defines the method of computation of secondary parameters (derived from the fixed or estimated model parameters). More details about the expected content of this file is provided in `vignette('scaRabee',package='scaR`

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

`fitmle.cov` is a secondary function called during estimation runs by `fitmle.cov`. It computes the covariance matrix for the parameter estimates.

Usage

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

Arguments

`problem` A list containing the following levels:

data A list containing the following levels:

xdata 1 x m matrix of independent variable.

ydata n x m matrix of observations from model states.

ids Data.frame of indices for data subsetting (output from `find.id`).

dosing A list containing the following levels:

history d x 4 data.frame of dosing history.

ids data.frame of indices for dosing subsetting (output from `find.id`).

cov A list containing the following levels:

data c x t data.frame of covariate history.

ids Data.frame of indices for cov subsetting (output from `find.id`).

states Indices of the states to be output by the model.

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.

varfun Variance function; if empty `weighting.additive` is used.

secfun Secondary parameter 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.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.

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`.

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.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).

Usage

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

Arguments

`subproblem` A list containing the following levels:

data A list containing the following levels:

xdata 1 x m matrix of independent variable.

ydata n x m matrix of observations from model states.

ids Data.frame of indices for data subsetting (output from `find.id`).

dosing A list containing the following levels:

history d x 4 data.frame of dosing history.

ids data.frame of indices for dosing subsetting (output from `find.id`).

cov A list containing the following levels:

data c x t data.frame of covariate history.

ids Data.frame of indices for cov subsetting (output from `find.id`).

states Indices of the states to be output by the model.

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.

varfun Variance function; if empty `weighting.additive` is used.

secfun Secondary parameter 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`

`init.report` *Initiliaze Iteration Log and Report Files*

Description

`init.report` is a secondary function called during estimation runs. It creates the log file and the report file in the results & backup directory. The log file stores 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.

Usage

```
init.report(param = NULL,
            files = NULL)
```

Arguments

`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 observations of the dependent variable(s) to be modeled. 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')`.

dose A .csv file located in the working directory, which contains the dosing information. The expected format of this file is described in details in `vignette('scaRabee',package='scaRabee')`.

cov A .csv file located in the working directory, which contains the values of one or more covariates that may or may or be used within the model. The expected format of this file is described in details in `vignette('scaRabee',package='scaR`

model A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model. Models specified with explicit, ordinary or delayed differential equations should be preferentially defined using the provided templates. More details about the expected structure of this file is provided in `vignette('scaRabee',package='scaRabee')`, in case the user would want to develop her/his own template.

var A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model of residual variability. More details about the expected content of this file is provided in `vignette('scaRabee',package='scaRabee'`

sec A .R file located in the 'model.definition' sub-directory in the working directory, which defines the method of computation of secondary parameters (derived from the fixed or estimated model parameters). More details about the expected content of this file is provided in `vignette('scaRabee',package='scaR`

Author(s)

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

`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').

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>)

`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.

Usage

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

Arguments

`subproblem`	A list containing the following levels: **data** A list containing the following levels: **xdata** 1 x m matrix of independent variable. **ydata** n x m matrix of observations from model states. **ids** Data.frame of indices for data subsetting (output from `find.id`). **dosing** A list containing the following levels: **history** d x 4 data.frame of dosing history. **ids** data.frame of indices for dosing subsetting (output from `find.id`). **cov** A list containing the following levels: **data** c x t data.frame of covariate history. **ids** Data.frame of indices for cov subsetting (output from `find.id`). **states** Indices of the states to be output by the model. **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. **varfun** Variance function; if empty `weighting.additive` is used. **secfun** Secondary parameter function.
`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 independent variable (typically time).

Usage

```
problem.eval(subproblem = NULL,
             x = NULL)
```

Arguments

`subproblem` A list containing the following levels:

data A list containing the following levels:

xdata 1 x m matrix of independent variable.

ydata n x m matrix of observations from model states.

ids Data.frame of indices for data subsetting (output from `find.id`).

dosing A list containing the following levels:

history d x 4 data.frame of dosing history.

ids data.frame of indices for dosing subsetting (output from `find.id`).

cov A list containing the following levels:

data c x t data.frame of covariate history.

ids Data.frame of indices for cov subsetting (output from `find.id`).

states Indices of the states to be output by the model.

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.

varfun Variance function; if empty `weighting.additive` is used.

secfun Secondary parameter function.

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

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` *Create of Predictions & Residuals 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.

Usage

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

Arguments

`problem` A list containing the following levels:

data A list containing the following levels:

xdata 1 x m matrix of independent variable.

ydata n x m matrix of observations from model states.

ids Data.frame of indices for data subsetting (output from `find.id`).

dosing A list containing the following levels:

history d x 4 data.frame of dosing history.

ids data.frame of indices for dosing subsetting (output from `find.id`).

cov A list containing the following levels:

data c x t data.frame of covariate history.

ids Data.frame of indices for cov subsetting (output from `find.id`).

states Indices of the states to be output by the model.

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.

varfun Variance function; if empty `weighting.additive` is used.

secfun Secondary parameter 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.

delta The intervals used for the computation of confidence intervals.

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.

`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 observations of the dependent variable(s) to be modeled. 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')`.

dose A .csv file located in the working directory, which contains the dosing information. The expected format of this file is described in details in `vignette('scaRabee',package='scaRabee')`.

cov A .csv file located in the working directory, which contains the values of one or more covariates that may or may or be used within the model. The expected format of this file is described in details in `vignette('scaRabee',package='scaR`

model A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model. Models specified with explicit, ordinary or delayed differential equations should be preferentially defined using the provided templates. More details about the expected structure of this file is provided in `vignette('scaRabee',package='scaRabee')`, in case the user would want to develop her/his own template.

var A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model of residual variability. More details about the expected content of this file is provided in `vignette('scaRabee',package='scaRabee'`

sec A .R file located in the 'model.definition' sub-directory in the working directory, which defines the method of computation of secondary parameters (derived from the fixed or estimated model parameters). More details about the expected content of this file is provided in `vignette('scaRabee',package='scaR`

Value

Creates the predictions and residuals report (extension .predictions.csv) in the run directory.

Author(s)

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

See Also

weighting.additive, find.id

---

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,
                  states = NULL,
                  runtype = NULL,
                  debugmode = FALSE,
                  estim.options = NULL,
                  analysis = 'myanalysis')
```

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 observations of the dependent variable(s) to be modeled. 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').
- **dose** A .csv file located in the working directory, which contains the dosing information. The expected format of this file is described in details in vignette('scaRabee',package='scaRabee').
- **cov** A .csv file located in the working directory, which contains the values of one or more covariates that may or may or be used within the model. The expected format of this file is described in details in vignette('scaRabee',package='scaR

model A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model. Models specified with explicit, ordinary or delayed differential equations should be preferentially defined using the provided templates. More details about the expected structure of this file is provided in `vignette('scaRabee',package='scaRabee')`, in case the user would want to develop her/his own template.

var A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model of residual variability. More details about the expected content of this file is provided in `vignette('scaRabee',package='scaRabee'`

sec A .R file located in the 'model.definition' sub-directory in the working directory, which defines the method of computation of secondary parameters (derived from the fixed or estimated model parameters). More details about the expected content of this file is provided in `vignette('scaRabee',package='scaR`

`states` A vector of integers, indicating which model states should be associated with model predictions. Only used in simulation runs.

`runtype` A character string, indicating the type of analysis. Should be 'simulation' or 'estimation'.

`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).

`analysis` A character string, defining the 'title' of the run that will be used to name the output files of the analysis. If `scarabee.analysis` is called within `script.R`, a file created from the provided template , `analysis` will be 'script'.

Value

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

Author(s)

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

See Also

`fitmle`

`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.

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 observations of the dependent variable(s) to be modeled. 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')`.

dose A .csv file located in the working directory, which contains the dosing information. The expected format of this file is described in details in `vignette('scaRabee',package='scaRabee')`.

cov A .csv file located in the working directory, which contains the values of one or more covariates that may or may or be used within the model. The expected format of this file is described in details in `vignette('scaRabee',package='scaR`

model A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model. Models specified with explicit, ordinary or delayed differential equations should be preferentially defined using the provided templates. More details about the expected structure of this file is provided in `vignette('scaRabee',package='scaRabee')`, in case the user would want to develop her/his own template.

var A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model of residual variability. More details about the expected content of this file is provided in `vignette('scaRabee',package='scaRabee'`

sec A .R file located in the 'model.definition' sub-directory in the working directory, which defines the method of computation of secondary parameters (derived from the fixed or estimated model parameters). More details about the expected content of this file is provided in `vignette('scaRabee',package='scaR`

`analysis` A character string, defining the 'title' of the run that will be used to name the output files of the analysis. If `scarabee.analysis` is called within `script.R`, a file created from the provided template , `analysis` will be 'script'.

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.

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 observations of the dependent variable(s) to be modeled. 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')`.

dose A .csv file located in the working directory, which contains the dosing information. The expected format of this file is described in details in `vignette('scaRabee',package='scaRabee')`.

cov A .csv file located in the working directory, which contains the values of one or more covariates that may or may or be used within the model. The expected format of this file is described in details in `vignette('scaRabee',package='scaR`

model A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model. Models specified with explicit, ordinary or delayed differential equations should be preferentially defined using the provided templates. More details about the expected structure of this file is provided in `vignette('scaRabee',package='scaRabee')`, in case the user would want to develop her/his own template.

var A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model of residual variability. More details about the expected content of this file is provided in `vignette('scaRabee',package='scaRabee'`

sec A .R file located in the 'model.definition' sub-directory in the working directory, which defines the method of computation of secondary parameters (derived from the fixed or estimated model parameters). More details about the expected content of this file is provided in `vignette('scaRabee',package='scaR`

`runtype` A character string, indicating the type of analysis. Should be 'simulation' or 'estimation'.

`analysis` A character string, defining the 'title' of the run that will be used to name the output files of the analysis. If `scarabee.analysis` is called within `script.R`, a file created from the provided template , `analysis` will be 'script'.

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 contained in `analysis`, an abbreviation of the type of run ('est' for estimation or 'sim' for simulation) and an incremental integer, e.g. test.est.01. This directory will contain 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 (`covariates.csv`, `data.csv`, `dosing.csv`, `initials`, `model.R`, `secondary.R`, `weighting.R`, and the master R script) are stored.

Author(s)

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

See Also

`scarabee.analysis`

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

Description

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

Usage

```
scarabee.new(name = 'myanalysis',
             path = getwd(),
             type = 'simulation',
             template = 'ode',
             with.inputs = TRUE)
```

Arguments

`name`	A string of characters defining the name of the new folder; `name` is also used to name the **scaRabee** analysis script. Default is 'myanalysis'.
`path`	A path where the new folder is created. Default is the current working directory.
`type`	A string of characters, either 'simulation' or 'estimation'. Default is 'simulation'.
`template`	A string of characters, either 'explicit', 'ode' or 'dde'. Default is 'ode'.
`with.inputs`	A logical indicator whether template input files should be created or not. Default is TRUE.

Details

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

model.definition A sub-folder containing:

- **model.R** A template-based R script for the definition of the structural model. Depending on `template`, it is either based on a template for model defined with closed form solution ('explicit'), ordinary differential equations ('ode') or delay differential equations ('dde').
- **weighting.R** A template-based R script for the definition of the residual variability model.
- **secondary.R** A template-based R script for the definition of secondary parameters.

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

data.csv (optional) An empty comma-separated file for system observations; contains the following default headers: Dose ID, Time, Y(1).

dosing.csv (optional) An empty comma-separated file for dosing events; contains the following default headers: Dose ID, Time, State, Bolus, Infusion Rate.

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.

covariates.csv (optional) An empty comma-separated file for covariate observations; contains the following default headers: Dose ID, Time, Cov(1).

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>)

`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. See `vignette('scaRabee',package='scaRabee')` for more details.

Usage

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

Arguments

`problem` A list containing the following levels:

data A list containing the following levels:

xdata 1 x m matrix of independent variable.

ydata n x m matrix of observations from model states.

ids Data.frame of indices for data subsetting (output from `find.id`).

dosing A list containing the following levels:

history d x 4 data.frame of dosing history.

ids data.frame of indices for dosing subsetting (output from `find.id`).

cov A list containing the following levels:

data c x t data.frame of covariate history.

ids Data.frame of indices for cov subsetting (output from `find.id`).

states Indices of the states to be output by the model.

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.

varfun Variance function; if empty `weighting.additive` is used.

secfun Secondary parameter function.

`Fsim` A data.frame of simulated and observed data containing the following columns:

doseID Indicator of sub-problem.

output Indicator ofsystem state.

time Evaluation time.

value Value of the system state.

type Category of system state value, either 'pred' for system prediction or 'obs' for system observationsl

`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 observations of the dependent variable(s) to be modeled. 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')`.

dose A .csv file located in the working directory, which contains the dosing information. The expected format of this file is described in details in `vignette('scaRabee',package='scaRabee')`.

cov A .csv file located in the working directory, which contains the values of one or more covariates that may or may or be used within the model. The expected format of this file is described in details in `vignette('scaRabee',package='scaR`

model A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model. Models specified with explicit, ordinary or delayed differential equations should be preferentially defined using the provided templates. More details about the expected structure of this file is provided in `vignette('scaRabee',package='scaRabee')`, in case the user would want to develop her/his own template.

var A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model of residual variability. More details about the expected content of this file is provided in `vignette('scaRabee',package='scaRabee'`

sec A .R file located in the 'model.definition' sub-directory in the working directory, which defines the method of computation of secondary parameters (derived from the fixed or estimated model parameters). More details about the expected content of this file is provided in `vignette('scaRabee',package='scaR`

Author(s)

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

`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.

Usage

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

Arguments

`problem` A list containing the following levels:

data A list containing the following levels:

xdata 1 x m matrix of independent variable.

ydata n x m matrix of observations from model states.

ids Data.frame of indices for data subsetting (output from `find.id`).

dosing A list containing the following levels:

history d x 4 data.frame of dosing history.

ids data.frame of indices for dosing subsetting (output from `find.id`).

cov A list containing the following levels:

data c x t data.frame of covariate history.

ids Data.frame of indices for cov subsetting (output from `find.id`).

states Indices of the states to be output by the model.

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.

varfun Variance function; if empty `weighting.additive` is used.

secfun Secondary parameter 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 observations of the dependent variable(s) to be modeled. 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')`.

dose A .csv file located in the working directory, which contains the dosing information. The expected format of this file is described in details in `vignette('scaRabee',package='scaRabee')`.

cov A .csv file located in the working directory, which contains the values of one or more covariates that may or may or be used within the model. The expected format of this file is described in details in `vignette('scaRabee',package='scaR`

model A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model. Models specified with explicit, ordinary or delayed differential equations should be preferentially defined using the provided templates. More details about the expected structure of this file is provided in `vignette('scaRabee',package='scaRabee')`, in case the user would want to develop her/his own template.

var A .R file located in the 'model.definition' sub-directory in the working directory, which defines the model of residual variability. More details about the expected content of this file is provided in `vignette('scaRabee',package='scaRabee'`

sec A .R file located in the 'model.definition' sub-directory in the working directory, which defines the method of computation of secondary parameters (derived from the fixed or estimated model parameters). More details about the expected content of this file is provided in `vignette('scaRabee',package='scaR`

Value

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

doseID Indicator of sub-problem.

output Indicator ofsystem state.

time Evaluation time.

value Value of the system state.

type Category of system state value, either 'pred' for system prediction or 'obs' for system observations.

Author(s)

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

`weighting.additive` *Default Residual Variability Model*

Description

`weighting additive` is a secondary function called during estimation run for which no residual variability model was provided

Usage

```
weighting.additive(f = NULL)
```

Arguments

`f`	matrix of structural model predictions.

Value

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

Author(s)

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

7 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.
```