Package ‘paropt’

October 14, 2022

Type Package

Title Parameter Optimizing of ODE-Systems

Version 0.2.1

Date 2021-05-26

Author Krämer Konrad [aut, cre],
Krämer Johannes [aut],
Heyer Arnd [ths],
University of Stuttgart [uvp],
Institute of Biomaterials and Biomolecular Systems at the University of Stuttgart [his]
| file AUTHORS

Maintainer Krämer Konrad <Konrad_kraemer@yahoo.de>

Copyright file COPYRIGHTS

BugReports https://github.com/Konrad1991/paropt

Description Enable optimization of parameters of ordinary differential equations. Therefore, using 'SUNDIALS' to solve the ODE-System (see Hindmarsh, Alan C., Peter N. Brown, Keith E. Grant, Steven L. Lee, Radu Serban, Dan E. Shumaker, and Carol S. Woodward. (2005) <doi:10.1145/1089014.1089020>). Furthermore, for optimization the particle swarm algorithm is used (see: Akman, Devin, Olcay Akman, and Elsa Schaefer. (2018) <doi:10.1155/2018/9160793> and Sengupta, Saptarshi, Sanchita Basak, and Richard Peters. (2018) <doi:10.3390/make1010010>). The ODE-System has to be passed as 'Rcpp'-function. The information for the parameter boundaries and states are conveyed using data.frames.

License GPL-3 | file LICENSE

Imports Rcpp (>= 1.0.4)

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.1.1

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-06-14 07:20:03 UTC

R topics documented:

`optimizer` *Optimize parameters of ode-systems*

Description

Optimize parameters used in an ode equation in order to match values defined in the state-data.frame

Usage

```
optimizer(
  integration_times,
  ode_system,
  relative_tolerance,
  absolute_tolerances,
  lb,
  ub,
  states,
  npop,
  ngen,
  error,
  solvertype
)
```

Arguments

`integration_times`	a vector containing the time course to solve the ode-system (see Details for more Information)
`ode_system`	the ode-system which will be integrated by the solver (see Details for more Information).
`relative_tolerance`	a number defining the relative tolerance used by the ode-solver.
`absolute_tolerances`	a vector containing the absolute tolerance(s) for each state used by the ode-solver.
`lb`	a data.frame containing the lower bounds for the parameters (see Details for more Information).

`ub`	a data.frame containing the upper bounds for the parameters (see Details for more Information).
`states`	a data.frame containing the predetermined course of the states (see Details for more Information).
`npop`	a number defining the number of particles used by the Particle Swarm Optimizer.
`ngen`	a number defining the number of generations the Particle Swarm Optimizer (PSO) should run.
`error`	a number defining a sufficient small error. When the PSO reach this value optimization is stopped.
`solvertype`	a string defining the type of solver which should be used (bdf, ADAMS, ERK or ARK. see Details for more Information).

Details

The vector containing the time course to solve the ode-system should contain the same entries as the time vector in the state-data.frame (it can be also be a different variable instead of time).

The ode system should be a Rcpp-function with a specific signature Rcpp::NumericVector ode(double time, std::vector<double> parameter, Rcpp::NumericVector states). The first entry defines the time point when the function is called. The second argument defines the parameter which should be optimized. There exist two different types of parameters. Parameters can be either constant or variabel. In order to calculate a variable parameter at a specific timepoint the Catmull-Rom-Spline is used. This vector contains the already interpolated parameters at the specific time-point, in the same order as defined in the data.frames containing the lower- and upper-boundaries. The last argument is a vector containing the states in the same order as defined in the data.frame containing the state-information. Thus, it is obligatory that the state-derivates in the ode-system are in the same order defined as in the data.frame. Furthermore, it is mandatory that the function return a Rcpp::NumericVector with the same dimension as the input vector containing the states. The resulting vector has to contain the right hand side of the ode-system.

For constant parameters use only the first row (below the headers) if other parameters are variable use “NA“ in the following rows for the constant parameters.

For variable parameters at least four points are needed. If a variable parameter is not available at every time point use “NA“ instead.

The two data.frames containg lower and upper-boundaries need the parameter in the same order.

The data.frame containing the state information should hold the time course in the first column. The header-name time is compulsory. The following columns contain the states. Take care that the states are in the same order defined in the ode system. If a state is not available use “NA“. This is possible for every time points except the first one. The ode solver need a start value for each state which is extracted from the first row of this file (below the headers).

The error between the solver output and the measured states is the sum of the absolute differences divided by the number of time points. It is crucial that the states are in the same order in the text file cointaining the state-information and in the ode-system to compare the states correctly!

For solving the ode system the SUNDIALS Software is used (https://computing.llnl.gov/projects/sundials). The last argument defines the solver-type which is used during optimization: “bdf“, “ADAMS“, “ERK“ or “ARK“. bdf = Backward Differentiation Formulas, ADAMS = Adams-Moulton, ERK =

explicite Runge-Kutta and ARK = implicite Runge-Kutta. All solvers are used in the NORMAL-Step method in a for-loop using the time-points defined in the text-file containing the states as output-points. The bdf- and ARK-Solver use the SUNLinSol_Dense as linear solver. Notably here is that for the ARK-Solver the ode system is fully implicit solved (not only part of it).

Examples can be found in the vignette.

`optimizer_pointer` *Optimize parameters of ode-systems*

Description

Optimize parameters used in an ode equation in order to match values defined in the state-data.frame

Usage

```
optimizer_pointer(
  integration_times,
  ode_sys,
  relative_tolerance,
  absolute_tolerances,
  lower,
  upper,
  states,
  npop,
  ngen,
  error,
  solvertype
)
```

Arguments

`integration_times`	a vector containing the time course to solve the ode-system (see Details for more Information)
`ode_sys`	the ode-system which will be integrated by the solver (see Details for more Information).
`relative_tolerance`	a number defining the relative tolerance used by the ode-solver.
`absolute_tolerances`	a vector containing the absolute tolerance(s) for each state used by the ode-solver.
`lower`	a data.frame containing the lower bounds for the parameters (see Details for more Information).
`upper`	a data.frame containing the upper bounds for the parameters (see Details for more Information).

`states`	a data.frame containing the predetermined course of the states (see Details for more Information).
`npop`	a number defining the number of particles used by the Particle Swarm Optimizer.
`ngen`	a number defining the number of generations the Particle Swarm Optimizer (PSO) should run.
`error`	a number defining a sufficient small error. When the PSO reach this value optimization is stopped.
`solvertype`	a string defining the type of solver which should be used (bdf, ADAMS, ERK or ARK. see Details for more Information).

Details

The vector containing the time course to solve the ode-system should contain the same entries as the time vector in the state-data.frame (it can be also be a different variable instead of time).

The ode system should be of type Rcpp::XPtr<OS>. The OS is predefined in the package. The function should possess the following signature: int ode(double &time, std::vector<double> &parameter, std::vector<double> &states). The first entry defines the time point when the function is called. The second argument defines the parameter which should be optimized. There exist two different types of parameters. Parameters can be either constant or variabel. In order to calculate a variable parameter at a specific timepoint the Catmull-Rom-Spline is used. This vector contains the already interpolated parameters at the specific time-point, in the same order as defined in the data.frames containing the lower- and upper-boundaries. The last argument is a vector containing the states in the same order as defined in the data.frame containing the state-information. Thus, it is obligatory that the state-derivates in the ode-system are in the same order defined as in the data.frame. Within the function the new states have to be saved in the states-vector. Please be aware that when using the approach with the Rcpp::XPtr the optimization is run in parallel. Thus, the function has to be thread-safe (among other things don't use any R Code)!

For constant parameters use only the first row (below the headers) if other parameters are variable use "NA" in the following rows for the constant parameters.

For variable parameters at least four points are needed. If a variable parameter is not available at every time point use "NA" instead.

The two data.frames containg lower and upper-boundaries need the parameter in the same order.

The data.frame containing the state information should hold the time course in the first column. The header-name time is compulsory. The following columns contain the states. Take care that the states are in the same order defined in the ode system. If a state is not available use "NA". This is possible for every time points except the first one. The ode solver need a start value for each state which is extracted from the first row of this file (below the headers).

The error between the solver output and the measured states is the sum of the absolute differences divided by the number of time points. It is crucial that the states are in the same order in the text file cointaining the state-information and in the ode-system to compare the states correctly!

For solving the ode system the SUNDIALS Software is used (https://computing.llnl.gov/projects/sundials). The last argument defines the solver-type which is used during optimization: "bdf", "ADAMS", "ERK" or "ARK". bdf = Backward Differentiation Formulas, ADAMS = Adams-Moulton, ERK = explicite Runge-Kutta and ARK = implicite Runge-Kutta. All solvers are used in the NORMAL-Step method in a for-loop using the time-points defined in the text-file containing the states as output-points. The bdf- and ARK-Solver use the SUNLinSol_Dense as linear solver. Notably here is that for the ARK-Solver the ode system is fully implicit solved (not only part of it).

Examples can be found in the vignette.

`solve_ode_system` *Solves ode-system and compare result to measured states*

Description

Solves ode-system and compare result to measured states

Usage

```
solve_ode_system(
  integration_times,
  ode_system,
  relative_tolerance,
  absolute_tolerances,
  start,
  states,
  solvertype
)
```

Arguments

`integration_times`	a vector containing the time course to solve the ode-system (see Details for more Information)
`ode_system`	the ode-system which will be integrated by the solver (see Details for more Information).
`relative_tolerance`	a number defining the relative tolerance used by the ode-solver.
`absolute_tolerances`	a vector containing the absolute tolerance(s) for each state used by the ode-solver.
`start`	a data.frame containing a parameter-set (see Details for more Information).
`states`	a data.frame containing the predetermined course of the states (see Details for more Information).
`solvertype`	a string defining the type of solver which should be used (bdf, ADAMS, ERK or ARK. see Details for more Information).

Details

The vector containing the time course to solve the ode-system should contain the same entries as the time vector in the state-data.frame (it can be also be a different variable instead of time).

The ode system should be a Rcpp-function with a specific signature Rcpp::NumericVector ode(double time, std::vector<double> parameter, Rcpp::NumericVector states). The first entry defines the time point when the function is called. The second argument defines the parameter which should be optimized. There exist two different types of parameters. Parameters can be either constant or variabel. In order to calculate a variable parameter at a specific timepoint the Catmull-Rom-Spline is used. This vector contains the already interpolated parameters at the specific time-point, in the same order as defined in the data.frames containing the lower- and upper-boundaries. The last argument is a vector containing the states in the same order as defined in the data.frame containing the state-information. Thus, it is obligatory that the state-derivates in the ode-system are in the same order defined as in the data.frame. Furthermore, it is mandatory that the function return a Rcpp::NumericVector with the same dimension as the input vector containing the states. The resulting vector has to contain the right hand side of the ode-system.

For constant parameters use only the first row (below the headers) if other parameters are variable use “NA“ in the following rows for the constant parameters.

For variable parameters at least four points are needed. If a variable parameter is not available at every time point use “NA“ instead.

The data.frame containing the state information should hold the time course in the first column. The header-name time is compulsory. The following columns contain the states. Take care that the states are in the same order defined in the ode system. If a state is not available use “NA“. This is possible for every time points except the first one. The ode solver need a start value for each state which is extracted from the first row of this file (below the headers).

The error between the solver output and the measured states is the sum of the absolute differences divided by the number of time points. It is crucial that the states are in the same order in the text file cointaining the state-information and in the ode-system to compare the states correctly!

For solving the ode system the SUNDIALS Software is used (https://computing.llnl.gov/projects/sundials). The last argument defines the solver-type which is used during optimization: “bdf“, “ADAMS“, “ERK“ or “ARK“. bdf = Backward Differentiation Formulas, ADAMS = Adams-Moulton, ERK = explicite Runge-Kutta and ARK = implicite Runge-Kutta. All solvers are used in the NORMAL-Step method in a for-loop using the time-points defined in the text-file containing the states as output-points. The bdf- and ARK-Solver use the SUNLinSol_Dense as linear solver. Notably here is that for the ARK-Solver the ode system is fully implicit solved (not only part of it).

Examples can be found in the vignette.

`solve_ode_system_pointer`

Solves ode-system and compare result to measured states

Description

Solves ode-system and compare result to measured states

Usage

```
solve_ode_system_pointer(
  integration_times,
  fctptr,
  relative_tolerance,
  absolute_tolerances,
  start,
  states,
  solvertype
)
```

Arguments

`integration_times`	a vector containing the time course to solve the ode-system (see Details for more Information)
`fctptr`	is a pointer to the ode-system which will be integrated by the solver (see Details for more Information).
`relative_tolerance`	a number defining the relative tolerance used by the ode-solver.
`absolute_tolerances`	a vector containing the absolute tolerance(s) for each state used by the ode-solver.
`start`	a data.frame containing a parameter-set (see Details for more Information).
`states`	a data.frame containing the predetermined course of the states (see Details for more Information).
`solvertype`	a string defining the type of solver which should be used (bdf, ADAMS, ERK or ARK. see Details for more Information).

Details

The vector containing the time course to solve the ode-system should contain the same entries as the time vector in the state-data.frame (it can be also be a different variable instead of time).

The ode system should be of type Rcpp::XPtr<OS>. The OS is predefined in the package. The function should possess the following signature: int ode(double &time, std::vector<double> &parameter, std::vector<double> &states). The first entry defines the time point when the function is called. The second argument defines the parameter which should be optimized. There exist two different types of parameters. Parameters can be either constant or variabel. In order to calculate a variable parameter at a specific timepoint the Catmull-Rom-Spline is used. This vector contains the already interpolated parameters at the specific time-point, in the same order as defined in the data.frames containing the lower- and upper-boundaries. The last argument is a vector containing the states in the same order as defined in the data.frame containing the state-information. Thus, it is obligatory that the state-derivates in the ode-system are in the same order defined as in the data.frame. Within the function the new states have to be saved in the states-vector.

For constant parameters use only the first row (below the headers) if other parameters are variable use “NA“ in the following rows for the constant parameters.

For variable parameters at least four points are needed. If a variable parameter is not available at every time point use “NA“ instead.

The data.frame containing the state information should hold the time course in the first column. The header-name time is compulsory. The following columns contain the states. Take care that the states are in the same order defined in the ode system. If a state is not available use “NA“. This is possible for every time points except the first one. The ode solver need a start value for each state which is extracted from the first row of this file (below the headers).

The error between the solver output and the measured states is the sum of the absolute differences divided by the number of time points. It is crucial that the states are in the same order in the text file cointaining the state-information and in the ode-system to compare the states correctly!

For solving the ode system the SUNDIALS Software is used (https://computing.llnl.gov/projects/sundials). The last argument defines the solver-type which is used during optimization: “bdf“, “ADAMS“, “ERK“ or “ARK“. bdf = Backward Differentiation Formulas, ADAMS = Adams-Moulton, ERK = explicite Runge-Kutta and ARK = implicite Runge-Kutta. All solvers are used in the NORMAL-Step method in a for-loop using the time-points defined in the text-file containing the states as output-points. The bdf- and ARK-Solver use the SUNLinSol_Dense as linear solver. Notably here is that for the ARK-Solver the ode system is fully implicit solved (not only part of it).

Examples can be found in the vignette.

Index