Package 'DSAIRM'

July 9, 2026

Type Package

Title Dynamical Systems Approach to Immune Response Modeling

Description Simulation models (apps) of various within-host immune response scenarios. The purpose of the package is to help individuals learn about within-host infection and immune response modeling from a dynamical systems perspective. All apps include explanations of the underlying models and instructions on what to do with the models.

Version 1.0.0

Date 2026-07-08

Maintainer Andreas Handel <andreas.handel@gmail.com>

License GPL-3

Encoding UTF-8

LazyData TRUE

Imports adaptivetau, boot, deSolve, dplyr, ggplot2, gridExtra, lhs, nloptr, plotly, rlang, stats, utils, XML

Depends R (>= 4.0.0), shiny (>= 1.2)

Suggests covr, devtools, emoji, knitr, pkgdown, rmarkdown, roxygen2, testthat

VignetteBuilder knitr

URL https://ahgroup.github.io/DSAIRM/, https://github.com/ahgroup/DSAIRM/

BugReports https://github.com/ahgroup/DSAIRM/issues

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Andreas Handel [aut, cre] (ORCID: <https://orcid.org/0000-0002-4622-1146>), Cody Dailey [ctb], Yang Ge [ctb], Spencer Hall [ctb],

Brian McKay [ctb],
Sina Solaimanpour [ctb],
Alexis Vittengl [ctb],
Henok Woldu [ctb]

Repository CRAN

Date/Publication 2026-07-09 16:30:15 UTC

Contents

check_simresults . . . 3
dsairmmenu . . . 3
generate_documentation . . . 4
generate_fctcall . . . 4
generate_ggplot . . . 5
generate_modelsettings . . . 6
generate_plotly . . . 7
generate_shinyinput . . . 9
generate_text . . . 10
hayden96flu . . . 11
run_model . . . 11
schirm20strep . . . 12
simulate_acutevirusir_ode . . . 13
simulate_bacteria_fit . . . 15
simulate_basicbacteria_discrete . . . 17
simulate_basicbacteria_modelexploration . . . 19
simulate_basicbacteria_ode . . . 21
simulate_basicvirus_fit . . . 22
simulate_basicvirus_modelexploration . . . 24
simulate_basicvirus_ode . . . 27
simulate_basicvirus_stochastic . . . 29
simulate_chronicvirusir_ode . . . 31
simulate_confint_fit . . . 33
simulate_drugresistance_stochastic . . . 35
simulate_extendedbacteria_ode . . . 36
simulate_fludrug_fit . . . 38
simulate_modelcomparison_fit . . . 40
simulate_modelvariants_ode . . . 43
simulate_pkpdmodel_ode . . . 45
simulate_usanalysis . . . 48
simulate_virusandir_ode . . . 50
simulate_virusandtx_ode . . . 52

Index **55**

check_simresults *A function that checks if a call to simulation function worked*

Description

This function takes the return from a call to a simulate_ function and checks that no errors occured. If something goes wrong, an error message is returned

Usage

```
check_simresults(simresult)
```

Arguments

simresult	a list of results from a single call to the simulation functions

Details

This function checks simulation results generated from a single call to a simulate_ function.

Value

Ane error message string or NULL

dsairmmenu *The main menu for the DSAIRM package*

Description

This function opens a Shiny app with a menu that will allow the user to run the different simulations.

Usage

```
dsairmmenu()
```

Details

Run this function with no arguments to start the main menu (a Shiny app) for DSAIRM.

Author(s)

Andreas Handel

Examples

```
## Not run: dsairmmenu()
```

`generate_documentation` *A helper function which processes and displays the documentation part for each app*

Description

This function take the documentation provided as html file and extracts sections to generate the tabs with content for each Shiny app. This is a helper function and only useful for this package.

Usage

```
generate_documentation(docfilename)
```

Arguments

`docfilename`	full path and name to html file containing the documentation

Details

This function is called by the Shiny UIs to populate the documentation and information tabs.

Value

tablist A list of tabs for display in a Shiny UI.

Author(s)

Andreas Handel

`generate_fctcall` *A helper function that produces a call to a simulator function for specific settings*

Description

This function takes a modelsettings structure and uses that information to create an unevaluated function call that runs the simulator function with the specified settings

Usage

```
generate_fctcall(modelsettings)
```

Arguments

modelsettings	a list with model settings. Required list elements are: List elements with names and values for all inputs expected by simulation function. modelsettings$simfunction - name of simulation function in variable

Details

This function produces a function call for specific settings.

Value

A string containing an unevaluated function call with the specified settings, or an error message

---

generate_ggplot *A helper function that takes simulation results and produces ggplot plots*

Description

This function generates plots to be displayed in the Shiny UI. This is a helper function. This function processes results returned from the simulation, supplied as a list.

Usage

```
generate_ggplot(res)
```

Arguments

res	A list structure containing all simulation results that are to be plotted. The length of the main list indicates the number of separate plots to make. Each list entry is itself a list, and corresponds to one plot and needs to contain the following information/elements: 1. A data frame list element called "dat" or "ts". If the data frame is "ts" it is assumed to be a time series and by default a line plot will be produced and labeled Time/Numbers. For plotting, the data needs to be in a format with one column called xvals, one column yvals, one column called varnames that contains names for different variables. Varnames needs to be a factor variable or will be converted to one. If a column 'varnames' exist, it is assumed the data is in the right format. Otherwise it will be transformed. An optional column called IDvar can be provided for further grouping (i.e. multiple lines for stochastic simulations). If plottype is 'mixedplot' an additional column called 'style' indicating line or point plot for each variable is needed. 2. Meta-data for the plot, provided in the following variables: optional: plottype - One of "Lineplot" (default if nothing is provided),"Scatterplot","Boxplot", "Mixedplot".

optional: xlab, ylab - Strings to label axes.
optional: xscale, yscale - Scaling of axes, valid ggplot2 expression, e.g. "identity" or "log10".
optional: xmin, xmax, ymin, ymax - Manual min and max for axes.
optional: makelegend - TRUE/FALSE, add legend to plot. Assume true if not provided.
optional: legendtitle - Legend title, if NULL/not supplied, default is used
optional: legendlocation - if "left" is specified, top left. Otherwise top.
optional: linesize - Width of line, numeric, i.e. 1.5, 2, etc. set to 1.5 if not supplied.
optional: pallette - overwrite plot colors by providing a vector of color names or hex numbers to be used for the plot.
optional: title - A title for each plot.
optional: for multiple plots, specify res[[1]]$ncols to define number of columns

Details

This function can be called to produce plots, i.e. those displayed for each app. The input needed by this function is produced by either calling the `run_model` function (as done when going through the UI) or manually transforming the output from a simulate_ function into the correct list structure as explained here.

Value

A ggplot plot structure for display in a Shiny UI.

Author(s)

Andreas Handel

`generate_modelsettings`
A function to parse modelsettings for simulation runs

Description

This function parses the shiny UI and gets input settings and combines those with other settings for a full definition of the model and settings that should be executed. This is a helper function.

Usage

```
generate_modelsettings(app_input, appsettings, appNames)
```

Arguments

app_input	a list of shiny GUI input
appsettings	current appsettings
appNames	character vector of app names, available in shiny server global environment upon app initialization

Details

This function returns specific settings for simulation runs as a list.

Value

modelsettings a list with model settings. List elements are:
modelsettings$apptitle - The name of the app that's being run.
modelsettings$simfunction - name of simulation function(s) as string.
modelsettings$simfunction2 - name of simulation function(s) as string.
modelsettings$is_mbmodel - indicate of simulation function has mbmodel structure modelsettings$modeltype - specify what kind of model should be run. Currently one of: _ode_, _discrete_, _stochastic_, _us-analysis_, _modelexploration_, _fit_.
For more than one model type, place _and_ between them.
modelsettings$plottype - 'Boxplot' or 'Scatterplot' , required for US app
Optinal list elements are:
List elements with names and values for inputs expected by simulation function. If not provided, defaults of simulator function are used.
modelsettings$plotscale - indicate which axis should be on a log scale (x, y or both). If not provided or set to '', no log scales are used.
modelsettings$nplots - indicate number of plots that should be produced (number of top list elements in result). If not provided, a single plot is assumed.
modelsettings$nreps - required for stochastic models to indicate numer of repeat simulations. If not provided, a single run will be done.

---

generate_plotly *A helper function that takes simulation results and produces plotly plots*

Description

This function generates plots to be displayed in the Shiny UI. This is a helper function. This function processes results returned from the simulation, supplied as a list.

Usage

```
generate_plotly(res)
```

Arguments

res	A list structure containing all simulation results that are to be plotted. The length of the main list indicates the number of separate plots to make. Each list entry is itself a list, and corresponds to one plot and needs to contain the following information/elements: 1. A data frame list element called "dat" or "ts". If the data frame is "ts" it is assumed to be a time series and by default a line plot will be produced and labeled Time/Numbers. For plotting, the data needs to be in a format with one column called xvals, one column yvals, one column called varnames that contains names for different variables. Varnames needs to be a factor variable or will be converted to one. If a column 'varnames' exist, it is assumed the data is in the right format. Otherwise it will be transformed. An optional column called IDvar can be provided for further grouping (i.e. multiple lines for stochastic simulations). If plottype is 'mixedplot' an additional column called 'style' indicating line or point plot for each variable is needed. 2. Meta-data for the plot, provided in the following variables: optional: plottype - One of "Lineplot" (default is nothing is provided),"Scatterplot","Boxplot", "Mixedplot". optional: xlab, ylab - Strings to label axes. optional: xscale, yscale - Scaling of axes, valid ggplot2 expression, e.g. "identity" or "log10". optional: xmin, xmax, ymin, ymax - Manual min and max for axes. optional: makelegend - TRUE/FALSE, add legend to plot. Assume true if not provided. optional: legendtitle - Legend title, if NULL/not supplied, default is used optional: legendlocation - if "left" is specified, top left. Otherwise top right. optional: linesize - Width of line, numeric, i.e. 1.5, 2, etc. set to 1.5 if not supplied. optional: title - A title for each plot. optional: for multiple plots, specify res[[1]]$ncols to define number of columns

Details

This function can be called to produce plots, i.e. those displayed for each app. The input needed by this function is produced by either calling the run_model() function (as done when going through the UI) or manually transforming the output from a simulate_ function into the correct list structure explained below.

Value

A plotly plot structure for display in a Shiny UI.

Author(s)

Yang Ge, Andreas Handel

`generate_shinyinput` *A helper function that takes a model and generates shiny UI elements*

Description

This function generates shiny UI inputs for a supplied model. This is a helper function called by the shiny app.

Usage

```
generate_shinyinput(
  use_mbmodel = FALSE,
  mbmodel = NULL,
  use_doc = FALSE,
  model_file = NULL,
  model_function = NULL,
  otherinputs = NULL,
  packagename = NULL
)
```

Arguments

`use_mbmodel`	TRUE/FALSE if mbmodel list should be used to generate UI
`mbmodel`	a valid mbmodel object
`use_doc`	TRUE/FALSE if doc of a model file should be parsed to make UI
`model_file`	name/path to function file for parsing doc
`model_function`	name of function who's formals are parsed to make UI
`otherinputs`	a text string that specifies a list of other shiny inputs to include in the UI
`packagename`	name of package using this function

Details

This function is called by the Shiny app to produce the Shiny input UI elements. It produces UI by 3 different ways. 1. If use_mbmodel is TRUE, an mbmodel list structure, which needs to be provided, is used 2. If use_mbmodel is FALSE and use_doc is TRUE, the documentation header of the function is used. For that approach, model_file needs to contain the name/path to the R script for the function The doc needs to have a specific format for this. 3. If both use_mbmodel and use_doc are FALSE, the function formals are parsed and used as UI. For that approach, model_function needs to specify the name of the model model_function is assumed to be the name of a function. The formals of the function will be parsed to create UI elements. Non-numeric arguments of functions are removed and need to be included in the otherinputs argument.

Value

A renderUI object that can be added to the shiny output object for display in a Shiny UI

`generate_text` *A helper function that takes result from the simulators and produces text output*

Description

This function generates text to be displayed in the Shiny UI. This is a helper function. This function processes results returned from the simulation, supplied as a list.

Usage

```
generate_text(res)
```

Arguments

`res`	A list structure containing all simulation results that are to be processed. This function is meant to be used together with generate_plots() and requires similar input information. See the generate_plots() function for most details. Specific entries for this function are 'maketext', 'showtext' and 'finaltext'. If 'maketext' is set to TRUE (or not provided) the function processes the data corresponding to each plot and reports min/max/final values (lineplots) or correlation coefficient (scatterplot) If 'maketext' is FALSE, no text based on the data is generated. If the entries 'showtext' or 'finaltext' are present, their values will be returned for each plot or for all together. The overall message of finaltext should be in the 1st plot.

Details

This function is called by the Shiny server to produce output returned to the Shiny UI.

Value

HTML formatted text for display in a Shiny UI.

Author(s)

Andreas Handel

hayden96flu *Influenza virus load data*

Description

Daily average virus load of volunteers infected with influenza.

Usage

```
data(hayden96flu)
```

Format

A data frame with these variables:

HoursPI Hours post infection - measurements were taken daily.

txtime Hours post infection when treatment started. The value of 29 is the average of the 2 reported early treatment times. A value of 200, which is later than the last recorded virus load, means no treatment.

LogVirusLoad Average virus load for volunteers in a given treatment condition, in log10 units.

LOD Limit of detection for virus load, in log10 units.

Details

Data is from Hayden et al 1996 JAMA: doi:10.1001/jama.1996.03530280047035

Specifically, data was extracted from Figure 2. See this article and citations therein for more details on the data.

run_model *A function that runs an app for specific settings and processes results for plot and text generation*

Description

This function runs a model based on information provided in the modelsettings list passed into it.

Usage

```
run_model(modelsettings)
```

Arguments

`modelsettings`	a list with model settings. Required list elements are: modelsettings$simfunction - name of simulation function(s) as string. modelsettings$is_mbmodel - indicate of simulation function has mbmodel structure modelsettings$modeltype - specify what kind of model should be run. Currently one of: _ode_, _discrete_, _stochastic_, _usanalysis_, _modelexploration_, _fit_. For more than one model type, place _and_ between them. modelsettings$plottype - 'Boxplot' or 'Scatterplot' , required for US app Optinal list elements are: List elements with names and values for inputs expected by simulation function. If not provided, defaults of simulator function are used. modelsettings$plotscale - indicate which axis should be on a log scale (x, y or both). If not provided or set to '', no log scales are used. modelsettings$nplots - indicate number of plots that should be produced (number of top list elements in result). If not provided, a single plot is assumed. modelsettings$nreps - required for stochastic models to indicate numer of repeat simulations. If not provided, a single run will be done.

Details

This function runs a model for specific settings.

Value

A vectored list named "result" with each main list element containing the simulation results in a dataframe called dat and associated metadata required for generate_plot and generate_text functions. Most often there is only one main list entry (result[[1]]) for a single plot/text.

`schirm20strep` *Streptococcus pneumoniae infection data*

Description

Time series data from Streptococcus pneumoniae infection in mice

Usage

```
data(schirm20strep)
```

Format

A data frame with these variables:

time Hours post infection.

value Amount of bacteria and neutrophils. See original paper for units.

quantity B = Streptococcus pneumoniae bacteria, I = neutrophils.

Details

Data is from Schirm et al 2020 PLoS One: https://doi.org/10.1371/journal.pone.0243147

Specifically, data used here are time-series of bacteria and neutrophils in BAL, as shown in panels a) and c) of Figure 4. The original data uses P for the bacteria, and N for neutrophils. For consistency with other DSAIRM models, we use B for bacteria and I for neutrophils (immune response). See the original article and citations therein for more details on the data.

`simulate_acutevirusir_ode`

Acute Virus Infection and IR Model

Description

A model for an acute virus infection with a simple immune response

Usage

```
simulate_acutevirusir_ode(
  U = 1e+05,
  I = 0,
  V = 10,
  F = 1,
  T = 1,
  dI = 1,
  dV = 2,
  b = 1e-06,
  p = 100,
  g = 1,
  rF = 0.001,
  dF = 1,
  kF = 0.01,
  rT = 0.001,
  dT = 1,
  kT = 0.01,
  tstart = 0,
  tfinal = 50,
  dt = 0.1
)
```

Arguments

`U`	: starting value for Uninfected cells : numeric
`I`	: starting value for Infected cells : numeric
`V`	: starting value for Virus : numeric
`F`	: starting value for Innate Response : numeric

T	: starting value for Adaptive Response : numeric
dI	: rate at which infected cells die : numeric
dV	: rate at which virus is cleared : numeric
b	: rate at which virus infects cells : numeric
p	: rate at which infected cells produce virus : numeric
g	: possible conversion factor for virus units : numeric
rF	: rate of innate response induction : numeric
dF	: rate of innate response decay : numeric
kF	: strength of innate response action : numeric
rT	: rate of adaptive response induction : numeric
dT	: rate of adaptive response decay : numeric
kT	: strength of adaptive response action : numeric
tstart	: Start time of simulation : numeric
tfinal	: Final time of simulation : numeric
dt	: Time step : numeric

Details

The model includes uninfected and infected target cells, virus, and the innate and adaptive immune response. The processes that are modeled are infection, virus production, infected cell and virus death. Immune response dynamics and action are also modeled. See the DSAIRM documentation for model details.

This code was generated by the modelbuilder R package. The model is implemented as a set of ordinary differential equations using the deSolve package. The following R packages need to be loaded for the function to work: deSolve.

Value

The function returns the output as a list. The time-series from the simulation is returned as a dataframe saved as list element `ts`. The `ts` dataframe has one column per compartment/variable. The first column is time.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. have negative values for parameters), the code will likely abort with an error message.

Model Author

Andreas Handel

Model creation date

2022-05-09

Code Author

generated by the modelbuilder R package

Code creation date

2022-05-09

Examples

```
# To run the simulation with default parameters:
result <- simulate_acutevirusir_ode()
# To choose values other than the standard one, specify them like this:
result <- simulate_acutevirusir_ode(U = 2e+05,I = 0,V = 2,F = 2,T = 2)
# You can display or further process the result, like this:
plot(result$ts[,'time'],result$ts[,'U'],xlab='Time',ylab='Numbers',type='l')
print(paste('Max number of U: ',max(result$ts[,'U'])))
```

---

simulate_bacteria_fit *Fitting a simple bacteria infection models to Streptococcus pneumoniae infection data*

Description

This function runs a simulation of a compartment model using a set of ordinary differential equations. The model describes a simple bacteria infection system.

Usage

```
simulate_bacteria_fit(
  B = 1,
  I = 1,
  g = 0.1,
  glow = 1e-04,
  ghigh = 100,
  Bmax = 1e+08,
  Bmaxlow = 10000,
  Bmaxhigh = 1e+10,
  dB = 0.01,
  dBlow = 1e-04,
  dBhigh = 100,
  kI = 0.001,
  kIlow = 1e-04,
  kIhigh = 100,
  rI = 0.1,
  rIlow = 1e-04,
  rIhigh = 100,
  Imax = 10000,
```

```
  Imaxlow = 1,
  Imaxhigh = 1e+10,
  dI = 0.1,
  dIlow = 0.001,
  dIhigh = 100,
  iter = 10,
  solvertype = 1
)
```

Arguments

B	: initial number of bacteria : numeric
I	: initial number of neutrophils (immune response) : numeric
g	: maximum rate of bacteria growth : numeric
glow	: lower bound for g : numeric
ghigh	: upper bound for g : numeric
Bmax	: bacteria carrying capacity : numeric
Bmaxlow	: lower bound for Bmax : numeric
Bmaxhigh	: upper bound for Bmax : numeric
dB	: bacteria death rate : numeric
dBlow	: lower bound for dB : numeric
dBhigh	: upper bound for dB : numeric
kI	: rate of bacteria killing by immune response : numeric
kIlow	: lower bound for k : numeric
kIhigh	: upper bound for k : numeric
rI	: immune response growth rate : numeric
rIlow	: lower bound for r : numeric
rIhigh	: upper bound for r : numeric
Imax	: immune response carrying capacity : numeric
Imaxlow	: lower bound for Imax : numeric
Imaxhigh	: upper bound for Imax : numeric
dI	: immune response decay rate : numeric
dIlow	: lower bound for dI : numeric
dIhigh	: upper bound for dI : numeric
iter	: max number of steps to be taken by optimizer : numeric
solvertype	: the type of solver/optimizer to use (1-3) : numeric

Details

A simple compartmental ODE model for a bacterial infection is fitted to data. The fitting is done using solvers/optimizers from the nloptr package (which is a wrapper for the nlopt library). The package provides access to a large number of solvers. Here, we only implement 3 solvers, namely 1 = NLOPT_LN_COBYLA, 2 = NLOPT_LN_NELDERMEAD, 3 = NLOPT_LN_SBPLX For details on what those optimizers are and how they work, see the nlopt/nloptr documentation.

Value

The function returns a list containing as elements the best fit time series data frame, the best fit parameters, the data and the final SSR

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. specify negative parameter or starting values, the code will likely abort with an error message.

Author(s)

Andreas Handel

See Also

See the Shiny app documentation corresponding to this function for more details on this model.

Examples

```
# To run the code with default parameters just call the function:
## Not run: result <- simulate_bacteria_fit()
# To apply different settings, provide them to the simulator function, like such:
result <- simulate_bacteria_fit(iter = 5)
```

---

simulate_basicbacteria_discrete

Basic Bacteria model - discrete

Description

A basic bacteria infection model with 2 compartments, implemented as discrete time simulation. The model tracks bacteria and an immune response dynamics. The processes modeled are bacteria growth, death and killing by the immune response, and immune response activation and decay.

Usage

```
simulate_basicbacteria_discrete(
  B = 10,
  I = 1,
  g = 1,
  Bmax = 1e+06,
  dB = 0.1,
  k = 1e-07,
  r = 0.001,
  dI = 1,
  tstart = 0,
  tfinal = 30,
  dt = 0.01
)
```

Arguments

B	: starting value for bacteria : numeric
I	: starting value for immune response : numeric
g	: maximum rate of bacteria growth : numeric
Bmax	: bacteria carrying capacity : numeric
dB	: bacteria death rate : numeric
k	: rate of bacteria killing by immune reesponse : numeric
r	: immune response growth rate : numeric
dI	: immune response decay rate : numeric
tstart	: start time of simulation : numeric
tfinal	: final time of simulation : numeric
dt	: time step : numeric

Details

The model includes bacteria and an immune response. The processes are bacteria growth, death and killing by the immune response, and immune response activation and decay. This is a predator-prey type model. The model is implemented as a set of discrete-time, deterministic equations, coded as a for-loop. This code is part of the DSAIRM R package. For additional model details, see the corresponding app in the DSAIRM package.

Value

The function returns the output as a list. The time-series from the simulation is returned as a dataframe saved as list element `ts`. The `ts` dataframe has one column per compartment/variable. The first column is time.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. have negative values for parameters), the code will likely abort with an error message.

Examples

```
# To run the simulation with default parameters:
result <- simulate_basicbacteria_discrete()
```

`simulate_basicbacteria_modelexploration` *Simulation to illustrate parameter scan of the basic bacteria model #'*

Description

This function simulates the simple bacteria model ODE for a range of parameters. The function returns a data frame containing the parameter that has been varied and the outcomes (see details).

Usage

```
simulate_basicbacteria_modelexploration(
  B = 100,
  I = 10,
  g = 2,
  Bmax = 1e+05,
  dB = 1,
  k = 1e-04,
  r = 1e-04,
  dI = 2,
  tstart = 0,
  tfinal = 300,
  dt = 0.1,
  samples = 10,
  parmin = 2,
  parmax = 10,
  samplepar = "g",
  pardist = "lin"
)
```

Arguments

`B`	: Starting value for bacteria : numeric
`I`	: Starting value for immune response : numeric
`g`	: Maximum rate of bacteria growth : numeric
`Bmax`	: Bacteria carrying capacity : numeric
`dB`	: Bacteria death rate : numeric
`k`	: Bacteria kill rate : numeric
`r`	: Immune response growth rate : numeric
`dI`	: Immune response decay rate : numeric
`tstart`	: Start time of simulation : numeric
`tfinal`	: Final time of simulation : numeric
`dt`	: Times for which result is returned : numeric
`samples`	: Number of values to run between pmin and pmax : numeric

parmin	: Lower value for varied parameter : numeric
parmax	: Upper value for varied parameter : numeric
samplepar	: Name of parameter to be varied : character
pardist	: spacing of parameter values, can be either 'lin' or 'log' : character

Details

##this code illustrates how to do analyze a simple model. A simple 2 compartment ODE model (the simple bacteria model introduced in the app of that name) is simulated for different parameter values. This function runs the simple bacterial infection model for a range of parameters. The user can specify which parameter is sampled, and the simulation returns for each parameter sample the peak and final value for B and I. Also returned is the varied parameter and an indicator if steady state was reached.

Value

The function returns the output as a list, list element 'dat' contains the data frame with results of interest. The first column is called xvals and contains the values of the parameter that has been varied as specified by 'samplepar'. The remaining columns contain peak and steady state values of bacteria and immune response, Bpeak, Ipeak, Bsteady and Isteady. A final boolean variable 'steady' is returned for each simulation. It is TRUE if the simulation reached steady state, otherwise FALSE.

Notes

The parameter dt only determines for which times the solution is returned, it is not the internal time step. The latter is set automatically by the ODE solver.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. specify negative parameter values or fractions > 1), the code will likely abort with an error message.

Author(s)

Andreas Handel

See Also

See the shiny app documentation corresponding to this simulator function for more details on this model.

Examples

```
# To run the simulation with default parameters just call the function:
## Not run: res <- simulate_basicbacteria_modelexploration()
# To choose parameter values other than the standard one, specify them, like such:
res <- simulate_basicbacteria_modelexploration(samples=5, samplepar='dI', parmin=1, parmax=10)
# You should then use the simulation result returned from the function, like this:
plot(res$dat[,"xvals"],res$data[,"Bpeak"],xlab='Parameter values',ylab='Peak Bacteria',type='l')
```

`simulate_basicbacteria_ode` *Basic Bacteria model - ODE*

Description

A basic bacteria infection model with 2 compartments, implemented as set of ODEs. The model tracks bacteria and an immune response dynamics. The processes modeled are bacteria growth, death and killing by the immune response, and immune response activation and decay.

Usage

```
simulate_basicbacteria_ode(
  B = 100,
  I = 1,
  g = 1,
  Bmax = 1e+05,
  dB = 0.5,
  k = 1e-04,
  r = 1e-04,
  dI = 2,
  tstart = 0,
  tfinal = 100,
  dt = 0.01
)
```

Arguments

`B`	: starting value for bacteria : numeric
`I`	: starting value for immune response : numeric
`g`	: maximum rate of bacteria growth : numeric
`Bmax`	: bacteria carrying capacity : numeric
`dB`	: bacteria death rate : numeric
`k`	: rate of bacteria killing by immune response : numeric
`r`	: immune response growth rate : numeric
`dI`	: immune response decay rate : numeric
`tstart`	: start time of simulation : numeric
`tfinal`	: final time of simulation : numeric
`dt`	: times for which result is returned : numeric

Details

The model includes bacteria and an immune response. The processes are bacteria growth, death and killing by the immune response, and immune response activation and decay. This is a predator-prey type model. The model is implemented as a set of ordinary differential equations (ODE) using the deSolve package. This code is part of the DSAIRM R package. For additional model details, see the corresponding app in the DSAIRM package.

Value

The function returns the output as a list. The time-series from the simulation is returned as a dataframe saved as list element `ts`. The `ts` dataframe has one column per compartment/variable. The first column is time.

Notes

The parameter dt only determines the times the solution is returned and plotted, it is not the internal time step for the differential equation solver. The latter is set automatically by the ODE solver.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. have negative values for parameters), the code will likely abort with an error message.

Examples

```
# To run the simulation with default parameters:
result <- simulate_basicbacteria_ode()
# To run the simulation with different parameter or starting values,
# supply the ones you want to change.
# all other parameters will be kept at their default values shown in the function call above
result <- simulate_basicbacteria_ode(B = 100, g = 0.5, dI = 2)
```

`simulate_basicvirus_fit`

Fitting a simple viral infection models to influenza data

Description

This function runs a simulation of a compartment model using a set of ordinary differential equations. The model describes a simple viral infection system.

Usage

```
simulate_basicvirus_fit(
  U = 1e+06,
  I = 0,
  V = 1,
  n = 0,
```

```
  dU = 0,
  dI = 2,
  g = 0,
  p = 0.001,
  plow = 1e-04,
  phigh = 100,
  psim = 0.001,
  b = 0.1,
  blow = 0.001,
  bhigh = 10,
  bsim = 0.1,
  dV = 1,
  dVlow = 0.01,
  dVhigh = 100,
  dVsim = 1,
  noise = 0,
  iter = 1,
  solvertype = 1,
  usesimdata = 0
)
```

Arguments

U	: initial number of uninfected target cells : numeric
I	: initial number of infected target cells : numeric
V	: initial number of infectious virions : numeric
n	: rate of uninfected cell production : numeric
dU	: rate at which uninfected cells die : numeric
dI	: rate at which infected cells die : numeric
g	: unit conversion factor : numeric
p	: rate at which infected cells produce virus : numeric
plow	: lower bound for p : numeric
phigh	: upper bound for p : numeric
psim	: rate at which infected cells produce virus for simulated data : numeric
b	: rate at which virus infects cells : numeric
blow	: lower bound for infection rate : numeric
bhigh	: upper bound for infection rate : numeric
bsim	: rate at which virus infects cells for simulated data : numeric
dV	: rate at which infectious virus is cleared : numeric
dVlow	: lower bound for virus clearance rate : numeric
dVhigh	: upper bound for virus clearance rate : numeric
dVsim	: rate at which infectious virus is cleared for simulated data : numeric
noise	: noise to be added to simulated data : numeric

`iter`	: max number of steps to be taken by optimizer : numeric
`solvertype`	: the type of solver/optimizer to use (1-3) : numeric
`usesimdata`	: set to 1 if simulated data should be fitted, 0 otherwise : numeric

Details

A simple compartmental ODE model mimicking acute viral infection is fitted to data. Data can either be real or created by running the model with known parameters and using the simulated data to determine if the model parameters can be identified. The fitting is done using solvers/optimizers from the nloptr package (which is a wrapper for the nlopt library). The package provides access to a large number of solvers. Here, we only implement 3 solvers, namely 1 = NLOPT_LN_COBYLA, 2 = NLOPT_LN_NELDERMEAD, 3 = NLOPT_LN_SBPLX For details on what those optimizers are and how they work, see the nlopt/nloptr documentation.

Value

The function returns a list containing as elements the best fit time series data frame, the best fit parameters, the data and the final SSR

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. specify negative parameter or starting values, the code will likely abort with an error message.

Author(s)

Andreas Handel

See Also

See the Shiny app documentation corresponding to this function for more details on this model.

Examples

```
# To run the code with default parameters just call the function:
## Not run: result <- simulate_basicvirus_fit()
# To apply different settings, provide them to the simulator function, like such:
result <- simulate_basicvirus_fit(iter = 5)
```

`simulate_basicvirus_modelexploration`

Simulation to illustrate parameter scan of the basic virus model #'

Description

This function simulates the basic virus model ODE for a range of parameters. The function returns a data frame containing the parameter that has been varied and the outcomes (see details).

Usage

```
simulate_basicvirus_modelexploration(
  U = 1e+05,
  I = 0,
  V = 1,
  n = 10000,
  dU = 0.1,
  dI = 1,
  dV = 2,
  b = 2e-05,
  p = 5,
  g = 1,
  tstart = 0,
  tfinal = 100,
  dt = 0.1,
  samples = 10,
  parmin = 1,
  parmax = 10,
  samplepar = "p",
  pardist = "lin"
)
```

Arguments

U	: Starting value for uninfected cells : numeric
I	: Starting value for infected cells : numeric
V	: Starting value for virus : numeric
n	: Rate of new uninfected cell replenishment : numeric
dU	: Rate at which uninfected cells die : numeric
dI	: Rate at which infected cells die : numeric
dV	: Rate at which virus is cleared : numeric
b	: Rate at which virus infects cells : numeric
p	: Rate at which infected cells produce virus : numeric
g	: Possible conversion factor for virus units : numeric
tstart	: Start time of simulation : numeric
tfinal	: Final time of simulation : numeric
dt	: Times for which result is returned : numeric
samples	: Number of values to run between pmin and pmax : numeric
parmin	: Lower value for varied parameter : numeric
parmax	: Upper value for varied parameter : numeric
samplepar	: Name of parameter to be varied : character
pardist	: spacing of parameter values, can be either 'lin' or 'log' : character

Details

##this code illustrates how to do analyze a simple model. A simple 3 compartment ODE model (the basic virus model introduced in the app of that name) is simulated for different parameter values. This function runs the model for a range of values for any one parameter, while holding all other paramter values fixed. The user can specify which parameter is sampled, and the simulation returns for each parameter sample the peak and final value for U, I and V. Also returned is the varied parameter and an indicator if steady state was reached.

Value

The function returns the output as a list, list element 'dat' contains the data frame with results of interest. The first column is called xvals and contains the values of the parameter that has been varied as specified by 'samplepar'. The remaining columns contain peak and steady state values of bacteria and immune response, Upeak, Ipeak, Vpeak, Usteady, Isteady and Vsteady. A final boolean variable 'steady' is returned for each simulation. It is TRUE if the simulation reached steady state, otherwise FALSE.

Notes

The parameter dt only determines for which times the solution is returned, it is not the internal time step. The latter is set automatically by the ODE solver.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. specify negative parameter values or fractions > 1), the code will likely abort with an error message.

Author(s)

Andreas Handel

See Also

See the shiny app documentation corresponding to this simulator function for more details on this model.

Examples

```
# To run the simulation with default parameters just call the function:
## Not run: res <- simulate_basicvirus_modelexploration()
# To choose parameter values other than the standard one, specify them, like such:
res <- simulate_basicvirus_modelexploration(samples=5, samplepar='dI', parmin=1, parmax=10)
# You should then use the simulation result returned from the function, like this:
plot(res$dat[,"xvals"],res$data[,"Vpeak"],xlab='Parameter values',ylab='Virus Peak',type='l')
```

`simulate_basicvirus_ode` *Basic Virus Model*

Description

A basic virus infection model with 3 compartments

Usage

```
simulate_basicvirus_ode(
  U = 1e+05,
  I = 0,
  V = 1,
  n = 0,
  dU = 0,
  dI = 1,
  dV = 2,
  b = 2e-05,
  p = 5,
  g = 1,
  tstart = 0,
  tfinal = 30,
  dt = 0.1
)
```

Arguments

`U`	: starting value for Uninfected cells : numeric
`I`	: starting value for Infected cells : numeric
`V`	: starting value for Virus : numeric
`n`	: rate of new uninfected cell replenishment : numeric
`dU`	: rate at which uninfected cells die : numeric
`dI`	: rate at which infected cells die : numeric
`dV`	: rate at which virus is cleared : numeric
`b`	: rate at which virus infects cells : numeric
`p`	: rate at which infected cells produce virus : numeric
`g`	: possible conversion factor for virus units : numeric
`tstart`	: Start time of simulation : numeric
`tfinal`	: Final time of simulation : numeric
`dt`	: Time step : numeric

Details

The model includes uninfected and infected target cells, as well as free virus. The processes that are modeled are infection, virus production, uninfected cell birth and death, infected cell and virus death.

This code was generated by the modelbuilder R package. The model is implemented as a set of ordinary differential equations using the deSolve package. The following R packages need to be loaded for the function to work: deSolve.

Value

The function returns the output as a list. The time-series from the simulation is returned as a dataframe saved as list element `ts`. The `ts` dataframe has one column per compartment/variable. The first column is time.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. have negative values for parameters), the code will likely abort with an error message.

Model Author

Andreas Handel

Model creation date

2021-07-19

Code Author

generated by the `modelbuilder` R package

Code creation date

2021-07-19

Examples

```
# To run the simulation with default parameters:
result <- simulate_basicvirus_ode()
# To choose values other than the standard one, specify them like this:
result <- simulate_basicvirus_ode(U = 2e+05,I = 0,V = 2)
# You can display or further process the result, like this:
plot(result$ts[,'time'],result$ts[,'U'],xlab='Time',ylab='Numbers',type='l')
print(paste('Max number of U: ',max(result$ts[,'U'])))
```

`simulate_basicvirus_stochastic` *Basic Virus Model*

Description

A basic virus infection model with 3 compartments

Usage

```
simulate_basicvirus_stochastic(
  U = 1e+05,
  I = 0,
  V = 1,
  n = 0,
  dU = 0,
  dI = 1,
  dV = 2,
  b = 2e-05,
  p = 5,
  g = 1,
  tfinal = 30,
  rngseed = 123
)
```

Arguments

`U`	: starting value for Uninfected cells : numeric
`I`	: starting value for Infected cells : numeric
`V`	: starting value for Virus : numeric
`n`	: rate of new uninfected cell replenishment : numeric
`dU`	: rate at which uninfected cells die : numeric
`dI`	: rate at which infected cells die : numeric
`dV`	: rate at which virus is cleared : numeric
`b`	: rate at which virus infects cells : numeric
`p`	: rate at which infected cells produce virus : numeric
`g`	: possible conversion factor for virus units : numeric
`tfinal`	: Final time of simulation : numeric
`rngseed`	: set random number seed for reproducibility : numeric

Details

The model includes uninfected and infected target cells, as well as free virus. The processes that are modeled are infection, virus production, uninfected cell birth and death, infected cell and virus death.

This code was generated by the modelbuilder R package. The model is implemented as a set of stochastic equations using the adaptivetau package. The following R packages need to be loaded for the function to work: adpativetau

Value

The function returns the output as a list. The time-series from the simulation is returned as a dataframe saved as list element `ts`. The `ts` dataframe has one column per compartment/variable. The first column is time.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. have negative values for parameters), the code will likely abort with an error message.

Model Author

Andreas Handel

Model creation date

2021-07-19

Code Author

generated by the `modelbuilder` R package

Code creation date

2021-07-19

Examples

```
# To run the simulation with default parameters:
result <- simulate_basicvirus_stochastic()
# To choose values other than the standard one, specify them like this:
result <- simulate_basicvirus_stochastic(U = 2e+05,I = 0,V = 2)
# You can display or further process the result, like this:
plot(result$ts[,'time'],result$ts[,'U'],xlab='Time',ylab='Numbers',type='l')
print(paste('Max number of U: ',max(result$ts[,'U'])))
```

simulate_chronicvirusir_ode

Chronic Virus Infection and IR Model

Description

A model for a chronic virus infection with a simple immune response

Usage

```
simulate_chronicvirusir_ode(
  U = 1e+05,
  I = 0,
  V = 10,
  F = 1,
  T = 1,
  n = 10000,
  dU = 0.1,
  dI = 1,
  dV = 4,
  b = 1e-06,
  p = 100,
  g = 1,
  rF = 0.001,
  dF = 1,
  kF = 0.01,
  rT = 0.001,
  dT = 1,
  kT = 0.01,
  tstart = 0,
  tfinal = 200,
  dt = 0.1
)
```

Arguments

U	: starting value for Uninfected cells : numeric
I	: starting value for Infected cells : numeric
V	: starting value for Virus : numeric
F	: starting value for Innate Response : numeric
T	: starting value for Adaptive Response : numeric
n	: rate of new uninfected cell replenishment : numeric
dU	: rate at which uninfected cells die : numeric
dI	: rate at which infected cells die : numeric

dV	: rate at which virus is cleared : numeric
b	: rate at which virus infects cells : numeric
p	: rate at which infected cells produce virus : numeric
g	: possible conversion factor for virus units : numeric
rF	: rate of innate response induction : numeric
dF	: rate of innate response decay : numeric
kF	: strength of innate response action : numeric
rT	: rate of adaptive response induction : numeric
dT	: rate of adaptive response decay : numeric
kT	: strength of adaptive response action : numeric
tstart	: Start time of simulation : numeric
tfinal	: Final time of simulation : numeric
dt	: Time step : numeric

Details

The model includes uninfected and infected target cells, virus, and the innate and adaptive immune response. The processes that are modeled are cell birth and death, infection, virus production, infected cell and virus death. Immune response dynamics and action are also modeled. See the DSAIRM documentation for model details.

This code was generated by the modelbuilder R package. The model is implemented as a set of ordinary differential equations using the deSolve package. The following R packages need to be loaded for the function to work: deSolve.

Value

The function returns the output as a list. The time-series from the simulation is returned as a dataframe saved as list element `ts`. The `ts` dataframe has one column per compartment/variable. The first column is time.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. have negative values for parameters), the code will likely abort with an error message.

Model Author

Andreas Handel

Model creation date

2022-05-09

Code Author

generated by the `modelbuilder` R package

Code creation date

2022-05-09

Examples

```
# To run the simulation with default parameters:
result <- simulate_chronicvirusir_ode()
# To choose values other than the standard one, specify them like this:
result <- simulate_chronicvirusir_ode(U = 2e+05,I = 0,V = 2,F = 2,T = 2)
# You can display or further process the result, like this:
plot(result$ts[,'time'],result$ts[,'U'],xlab='Time',ylab='Numbers',type='l')
print(paste('Max number of U: ',max(result$ts[,'U'])))
```

---

simulate_confint_fit *Fit a simple viral infection model and compute confidence intervals*

Description

This function runs a simulation of a compartment model using a set of ordinary differential equations. The model describes a simple viral infection system.

Usage

```
simulate_confint_fit(
  U = 1e+05,
  I = 0,
  V = 10,
  n = 0,
  dU = 0,
  dI = 2,
  p = 0.01,
  g = 0,
  b = 0.01,
  blow = 1e-06,
  bhigh = 1000,
  dV = 2,
  dVlow = 0.001,
  dVhigh = 1000,
  iter = 20,
  nsample = 10,
  rngseed = 100,
  parscale = 1
)
```

Arguments

U	: initial number of uninfected target cells : numeric
I	: initial number of infected target cells : numeric
V	: initial number of infectious virions : numeric
n	: rate of uninfected cell production : numeric
dU	: rate at which uninfected cells die : numeric
dI	: rate at which infected cells die : numeric
p	: rate at which infected cells produce virus : numeric
g	: unit conversion factor : numeric
b	: rate at which virus infects cells : numeric
blow	: lower bound for infection rate : numeric
bhigh	: upper bound for infection rate : numeric
dV	: rate at which infectious virus is cleared : numeric
dVlow	: lower bound for virus clearance rate : numeric
dVhigh	: upper bound for virus clearance rate : numeric
iter	: max number of steps to be taken by optimizer : numeric
nsample	: number of samples for conf int determination : numeric
rngseed	: seed for random number generator to allow reproducibility : numeric
parscale	: 1 for linear, 2 for log space parameter fitting : numeric

Details

A simple compartmental ODE model mimicking acute viral infection is fitted to data. Confidence intervals are computed by simple bootstrapping of the data using the boot R package. Confidence intervals are computed using the percentage method in boot.ci. See the boot package for more information. This code is part of the DSAIRM R package. For additional model details, see the corresponding app in the DSAIRM package.

Value

The function returns a list containing the best fit time series, the best fit parameters for the data, the final SSR, and the bootstrapped 95 percent confidence intervals.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. specify negative parameter or starting values), the code will likely abort with an error message.

Examples

```
# To run the code with default parameters just call the function:
## Not run: result <- simulate_confint_fit()
# To apply different settings, provide them to the simulator function, like such:
result <- simulate_confint_fit(iter = 5, nsample = 5)
```

`simulate_drugresistance_stochastic` *Stochastic simulation of a compartmental acute virus infection model with treatment and drug resistant strain*

Description

Simulation of a stochastic model with the following compartments: Uninfected target cells (U), Infected with wild-type/sensitive and untreated (Is), infected with resistant (Ir), wild-type virus (Vs), resistant virus (Vr).

Usage

```
simulate_drugresistance_stochastic(
  U = 1e+05,
  Is = 0,
  Ir = 0,
  Vs = 10,
  Vr = 0,
  b = 1e-05,
  dI = 1,
  e = 0,
  m = 0.001,
  p = 100,
  c = 4,
  f = 0.1,
  rngseed = 100,
  tfinal = 30
)
```

Arguments

`U`	: initial number of target cells : numeric
`Is`	: initial number of wild-type infected cells : numeric
`Ir`	: initial number of resistant infected cells : numeric
`Vs`	: initial number of wild-type virus : numeric
`Vr`	: initial number of resistant virus : numeric
`b`	: level/rate of infection of cells : numeric
`dI`	: rate of infected cell death : numeric
`e`	: efficacy of drug : numeric
`m`	: fraction of resistant mutants created : numeric
`p`	: virus production rate : numeric
`c`	: virus removal rate : numeric
`f`	: fitness cost of resistant virus : numeric

`rngseed` : seed for random number generator to allow reproducibility : numeric
`tfinal` : maximum simulation time : numeric

Details

A compartmental ID model with several states/compartments is simulated as a stochastic model using the adaptive tau algorithm as implemented by ssa.adaptivetau in the adpativetau package. See the manual of this package for more details. The function returns the time series of the simulated disease as output matrix, with one column per compartment/variable. The first column is time.

Value

A list. The list has only one element called ts. ts contains the time-series of the simulation. The 1st column of ts is Time, the other columns are the model variables.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. have I0 > PopSize or any negative values or fractions > 1), the code will likely abort with an error message.

Author(s)

Andreas Handel

References

See the manual for the adaptivetau package for details on the algorithm. The implemented model is loosely based on: Handel et al 2007 PLoS Comp Bio "Neuraminidase Inhibitor Resistance in Influenza: Assessing the Danger of Its Generation and Spread"

Examples

```
# To run the simulation with default parameters just call the function:
result <- simulate_drugresistance_stochastic()
# To choose parameter values other than the standard one, specify them, like such:
result <- simulate_drugresistance_stochastic(tfinal = 100, e = 0.5)
# You should then use the simulation result returned from the function, like this:
plot(result$ts[,"time"],result$ts[,"Vs"],xlab='Time',ylab='Uninfected cells',type='l')
```

---

simulate_extendedbacteria_ode

Extended Bacteria model - ODE

Description

A bacteria infection model with 3 compartments, implemented as set of ODEs. The model tracks bacteria, and one compartment each for innate and adaptive immune response. The processes modeled are bacteria growth, death and killing by the immune response, and growth and decay of each immune response component.

Usage

```
simulate_extendedbacteria_ode(
  B = 100,
  I = 1,
  A = 1,
  g = 1,
  Bmax = 1e+05,
  dB = 0.5,
  kI = 1e-04,
  kA = 1e-04,
  rI = 1,
  Imax = 1e+05,
  dI = 1,
  rA = 1,
  h = 1000,
  dA = 0.1,
  tstart = 0,
  tfinal = 100,
  dt = 0.01
)
```

Arguments

B	: starting value for bacteria : numeric
I	: starting value for innate immune response : numeric
A	: starting value for adaptive immune response : numeric
g	: maximum rate of bacteria growth : numeric
Bmax	: bacteria carrying capacity : numeric
dB	: bacteria death rate : numeric
kI	: rate of bacteria killing by innate response : numeric
kA	: rate of bacteria killing by adaptive response : numeric
rI	: innate response growth rate : numeric
Imax	: innate response carrying capacity : numeric
dI	: innate response decay rate : numeric
rA	: adaptive response growth rate : numeric
h	: adaptive response half-growth : numeric
dA	: adaptive response decay rate : numeric
tstart	: start time of simulation : numeric
tfinal	: final time of simulation : numeric
dt	: times for which result is returned : numeric

Details

The model tracks bacteria, and one compartment each for innate and adaptive immune response. The processes modeled are bacteria growth, death and killing by the immune response, and growth and decay of each immune response component. The model is implemented as a set of ordinary differential equations (ODE) using the deSolve package. This code is part of the DSAIRM R package. For additional model details, see the corresponding app in the DSAIRM package.

Value

The function returns the output as a list. The time-series from the simulation is returned as a dataframe saved as list element `ts`. The `ts` dataframe has one column per compartment/variable. The first column is time.

Notes

The parameter dt only determines the times the solution is returned and plotted, it is not the internal time step for the differential equation solver. The latter is set automatically by the ODE solver.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. have negative values for parameters), the code will likely abort with an error message.

Examples

```
# To run the simulation with default parameters:
result <- simulate_extendedbacteria_ode()
# To run the simulation with different parameter or starting values,
# supply the ones you want to change.
# all other parameters will be kept at their default values shown in the function call above
result <- simulate_extendedbacteria_ode(B = 100, g = 0.5, dI = 2)
```

---

simulate_fludrug_fit *Fitting a simple viral infection model with 2 types of drug mechanisms to influenza data*

Description

This function fits the simulate_virusandtx_ode model, which is a compartment model using a set of ordinary differential equations. The model describes a simple viral infection system in the presence of drug treatment. The user provides initial conditions and parameter values for the system. The function simulates the ODE using an ODE solver from the deSolve package.

Usage

```
simulate_fludrug_fit(
  U = 1e+07,
  I = 0,
  V = 1,
  dI = 1,
  dV = 2,
  b = 0.002,
  blow = 0,
  bhigh = 100,
  p = 0.002,
  plow = 0,
  phigh = 100,
  g = 0,
  glow = 0,
  ghigh = 100,
  k = 0,
  fitmodel = 1,
  iter = 1
)
```

Arguments

U	: initial number of uninfected target cells : numeric
I	: initial number of infected target cells : numeric
V	: initial number of infectious virions : numeric
dI	: rate at which infected cells die : numeric
dV	: rate at which infectious virus is cleared : numeric
b	: rate at which virus infects cells : numeric
blow	: lower bound for infection rate : numeric
bhigh	: upper bound for infection rate : numeric
p	: rate at which infected cells produce virus : numeric
plow	: lower bound for virus production rate : numeric
phigh	: upper bound for virus production rate : numeric
g	: unit conversion factor : numeric
glow	: lower bound for unit conversion factor : numeric
ghigh	: upper bound for unit conversion factor : numeric
k	: drug efficacy (between 0-1) : numeric
fitmodel	: fitting model 1 or 2 : numeric
iter	: max number of steps to be taken by optimizer : numeric

Details

A simple compartmental ODE models describing an acute viral infection with drug treatment mechanism/model 1 assumes that drug treatment reduces rate of new cell infection. mechanism/model 2 assumes that drug treatment reduces rate of new virus production.

Value

The function returns a list containing the best fit timeseries, the best fit parameters, the data and the AICc for the model.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. specify negative parameter or starting values), the code will likely abort with an error message.

Author(s)

Andreas Handel

See Also

See the Shiny app documentation corresponding to this function for more details on this model.

Examples

```
# To run the code with default parameters just call the function:
result <- simulate_fludrug_fit()
# To apply different settings, provide them to the simulator function, like such:
result <- simulate_fludrug_fit(k = 0.5, iter = 5, fitmodel = 2)
```

---

simulate_modelcomparison_fit

Fitting 2 simple viral infection models to influenza data

Description

This function runs a simulation of a compartment model using a set of ordinary differential equations. The model describes a simple viral infection system in the presence of drug treatment. The user provides initial conditions and parameter values for the system. The function simulates the ODE using an ODE solver from the deSolve package. The function returns a matrix containing time-series of each variable and time.

Usage

```
simulate_modelcomparison_fit(
  U = 1e+06,
  I = 0,
  V = 1,
  X = 1,
  dI = 2,
  dV = 4,
  p = 0.1,
  g = 0,
  k = 0.001,
  a = 0.001,
  alow = 1e-05,
  ahigh = 10,
  b = 0.001,
  blow = 1e-06,
  bhigh = 10,
  r = 0.1,
  rlow = 0.001,
  rhigh = 10,
  dX = 1,
  dXlow = 0.1,
  dXhigh = 10,
  fitmodel = 1,
  iter = 1
)
```

Arguments

U	: initial number of uninfected target cells : numeric
I	: initial number of infected target cells : numeric
V	: initial number of infectious virions : numeric
X	: initial level of immune response : numeric
dI	: rate at which infected cells die : numeric
dV	: rate at which infectious virus is cleared : numeric
p	: rate at which infected cells produce virus : numeric
g	: unit conversion factor : numeric
k	: rate of killing of infected cells by T-cells (model 1) or virus by Ab (model 2) : numeric
a	: activation of T-cells (model 1) or growth of antibodies (model 2) : numeric
alow	: lower bound for activation rate : numeric
ahigh	: upper bound for activation rate : numeric
b	: rate at which virus infects cells : numeric
blow	: lower bound for infection rate : numeric

`bhigh`	: upper bound for infection rate : numeric
`r`	: rate of T-cell expansion (model 1) : numeric
`rlow`	: lower bound for expansion rate : numeric
`rhigh`	: upper bound for expansion rate : numeric
`dX`	: rate at which antibodies decay (model 2) : numeric
`dXlow`	: lower bound for decay rate : numeric
`dXhigh`	: upper bound for decay rate : numeric
`fitmodel`	: fitting model 1 or 2 : numeric
`iter`	: max number of steps to be taken by optimizer : numeric

Details

Two simple compartmental ODE models mimicking acute viral infection with T-cells (model 1) or antibodies (model 2) are fitted to data.

Value

The function returns a list containing the best fit timeseries, the best fit parameters, the data and the AICc for the model.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. specify negative parameter or starting values), the code will likely abort with an error message.

Author(s)

Andreas Handel

See Also

See the Shiny app documentation corresponding to this function for more details on this model.

Examples

```
# To run the code with default parameters just call the function:
## Not run: result <- simulate_modelcomparison_fit()
# To apply different settings, provide them to the simulator function, like such:
result <- simulate_modelcomparison_fit(iter = 5, fitmodel = 1)
```

simulate_modelvariants_ode *Simulation of a viral infection model with immune response The simulation illustrates different alternative models.*

Description

This function runs a simulation of a compartment model using a set of ordinary differential equations. The user provides initial conditions and parameter values for the system. The function simulates the ODE using an ODE solver from the deSolve package. The function returns a matrix containing time-series of each variable and time.

Usage

```
simulate_modelvariants_ode(
  U = 1e+05,
  I = 0,
  V = 10,
  F = 0,
  A = 0,
  n = 0,
  dU = 0,
  dI = 1,
  dV = 4,
  b = 1e-05,
  p = 100,
  pF = 1,
  dF = 1,
  f1 = 1e-04,
  f2 = 0,
  f3 = 0,
  Fmax = 1000,
  sV = 1e-10,
  k1 = 0.001,
  k2 = 0,
  k3 = 0,
  a1 = 1000,
  a2 = 0,
  a3 = 0,
  hV = 1e-10,
  k4 = 0.001,
  k5 = 0,
  k6 = 0,
  sA = 1e-10,
  dA = 0.1,
  tstart = 0,
  tfinal = 20,
```

```
  dt = 0.01
)
```

Arguments

U	: initial number of uninfected target cells : numeric
I	: initial number of infected target cells : numeric
V	: initial number of infectious virions : numeric
F	: initial level of innate response : numeric
A	: initial level of adaptive response : numeric
n	: rate of uninfected cell production : numeric
dU	: rate of natural death of uninfected cells : numeric
dI	: rate at which infected cells die : numeric
dV	: rate at which infectious virus is cleared : numeric
b	: rate at which virus infects cells : numeric
p	: rate at which infected cells produce virus : numeric
pF	: rate of innate response production in absence of infection : numeric
dF	: rate of innate response removal in absence of infection : numeric
f1	: growth of innate response alternative 1 : numeric
f2	: growth of innate response alternative 2 : numeric
f3	: growth of innate response alternative 3 : numeric
Fmax	: maximum level of innate response in alternative 1 : numeric
sV	: saturation of innate response growth in alternative 2 and 3 : numeric
k1	: action of innate response alternative 1 : numeric
k2	: action of innate response alternative 2 : numeric
k3	: action of innate response alternative 3 : numeric
a1	: growth of adaptive response alternative 1 : numeric
a2	: growth of adaptive response alternative 2 : numeric
a3	: growth of adaptive response alternative 3 : numeric
hV	: saturation of adaptive response growth in alternative 2 and 3 : numeric
k4	: action of adaptive response alternative 1 : numeric
k5	: action of adaptive response alternative 2 : numeric
k6	: action of adaptive response alternative 3 : numeric
sA	: saturation of adaptive response killing for alternative action 2 : numeric
dA	: adaptive immune response decay : numeric
tstart	: Start time of simulation : numeric
tfinal	: Final time of simulation : numeric
dt	: Times for which result is returned : numeric

Details

A compartmental infection model is simulated as a set of ordinary differential equations, using an ode solver from the deSolve package.

Value

The function returns the output from the odesolver as a matrix, with one column per compartment/variable. The first column is time.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. specify negative parameter or starting values), the code will likely abort with an error message.

Author(s)

Andreas Handel

See Also

See the Shiny app documentation corresponding to this simulator function for more details on this model. See the manual for the deSolve package for details on the underlying ODE simulator algorithm.

Examples

```
# To run the simulation with default parameters just call the function:
result <- simulate_modelvariants_ode()
# To choose parameter values other than the standard one, specify them, like such:
result <- simulate_modelvariants_ode(V = 100, k1 = 0 , k2 = 0, k3 = 1e-4)
# You should then use the simulation result returned from the function, like this:
plot(result$ts[,"time"],result$ts[,"V"],xlab='Time',ylab='Virus',type='l',log='y')
```

---

simulate_pkpdmodel_ode

PkPd Virus model

Description

This function runs a simulation of the basic 3 compartment virus infection model including the pharmacokinetics and pharmacodynamics of a drug. The user provides initial conditions and parameter values for the system. The function simulates the ODE using an ODE solver from the deSolve package.

Usage

```
simulate_pkpdmodel_ode(
  U = 1e+05,
  I = 0,
  V = 10,
  n = 0,
  dU = 0,
  dI = 1,
  dV = 2,
  b = 1e-05,
  g = 1,
  p = 10,
  C0 = 1,
  dC = 1,
  C50 = 1,
  k = 1,
  Emax = 0,
  txstart = 10,
  txinterval = 1,
  tstart = 0,
  tfinal = 20,
  dt = 0.01
)
```

Arguments

U	: initial number of uninfected target cells : numeric
I	: initial number of infected target cells : numeric
V	: initial number of infectious virions : numeric
n	: rate of new uninfected cell replenishment : numeric
dU	: rate at which uninfected cells die : numeric
dI	: rate at which infected cells die : numeric
dV	: rate at which infectious virus is cleared : numeric
b	: rate at which virus infects cells : numeric
g	: unit conversion factor : numeric
p	: rate at which infected cells produce virus : numeric
C0	: drug dose given at specified times : numeric
dC	: drug concentration decay rate : numeric
C50	: drug concentration at which effect is half maximum : numeric
k	: steepness of concentration-dependent drug effect : numeric
Emax	: maximum drug efficacy (0-1) : numeric
txstart	: time of drug treatment start : numeric
txinterval	: time between drug doses : numeric

`tstart` : Start time of simulation : numeric

`tfinal` : Final time of simulation : numeric

`dt` : Times for which result is returned : numeric

Details

A basic virus infection model with drug PkPd

A simple compartmental model is simulated as a set of ordinary differential equations, using an ode solver from the deSolve package. This code is part of the DSAIRM R package. For additional model details, see the corresponding app in the DSAIRM package.

Value

A list. The list has only one element called ts. ts contains the time-series of the simulation. The 1st column of ts is Time, the other columns are the model variables.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. specify negative parameter or starting values), the code will likely abort with an error message.

Author(s)

Andreas Handel

See Also

See the Shiny app documentation corresponding to this simulator function for more details on this model. See the manual for the deSolve package for details on the underlying ODE simulator algorithm.

Examples

```
# To run the simulation with default parameters just call the function:
result <- simulate_pkpdmodel_ode()
# To choose parameter values other than the standard one, specify them, like such:
result <- simulate_pkpdmodel_ode(V = 100, txstart = 10, n = 1e5, dU = 1e-2)
# You should then use the simulation result returned from the function, like this:
plot(result$ts[,"time"],result$ts[,"V"],xlab='Time',ylab='Virus',type='l',log='y')
```

`simulate_usanalysis` *Simulation to illustrate uncertainty and sensitivity analysis*

Description

This function performs uncertainty and sensitivity analysis using the simple, continuous-time basic bacteria model.

Usage

```
simulate_usanalysis(
  Bmin = 100,
  Bmax = 200,
  Imin = 1,
  Imax = 2,
  Bmaxmin = 1e+05,
  Bmaxmax = 2e+05,
  dBmin = 0.5,
  dBmax = 1,
  kmin = 1e-04,
  kmax = 2e-04,
  rmin = 1e-04,
  rmax = 2e-04,
  dImin = 1,
  dImax = 2,
  gmean = 2,
  gvar = 0.5,
  samples = 10,
  rngseed = 100,
  tstart = 0,
  tfinal = 300,
  dt = 0.5
)
```

Arguments

`Bmin`	: lower bound for initial bacteria numbers : numeric
`Bmax`	: upper bound for initial bacteria numbers : numeric
`Imin`	: lower bound for initial immune response : numeric
`Imax`	: upper bound for initial immune response : numeric
`Bmaxmin`	: lower bound for maximum bacteria load : numeric
`Bmaxmax`	: upper bound for maximum bacteria load : numeric
`dBmin`	: lower bound for bacteria death rate : numeric
`dBmax`	: upper bound for bacteria death rate : numeric
`kmin`	: lower bound for immune response kill rate : numeric

kmax	: upper bound for immune response kill rate : numeric
rmin	: lower bound for immune response growth rate : numeric
rmax	: upper bound for immune response growth rate : numeric
dImin	: lower bound for immune response death rate : numeric
dImax	: upper bound for immune response death rate : numeric
gmean	: mean for bacteria growth rate : numeric
gvar	: variance for bacteria growth rate : numeric
samples	: number of LHS samples to run : numeric
rngseed	: seed for random number generator : numeric
tstart	: Start time of simulation : numeric
tfinal	: Final time of simulation : numeric
dt	: times for which result is returned : numeric

Details

A simple 2 compartment ODE model (the simple bacteria model introduced in the app of that name) is simulated for different parameter values. The user provides ranges for the initial conditions and parameter values and the number of samples. The function does Latin Hypercube Sampling (LHS) of the parameters and runs the basic bacteria ODE model for each sample. Distribution for all parameters is assumed to be uniform between the min and max values. The only exception is the bacteria growth parameter, which is assumed to be gamma distributed with the specified mean and variance. This code is part of the DSAIRM R package. For additional model details, see the corresponding app in the DSAIRM package.

Value

The function returns the output as a list. The list element 'dat' contains a data frame. The simulation returns for each parameter sample the peak and final value for B and I. Also returned are all parameter values as individual columns and an indicator stating if steady state was reached. A final variable 'steady' is returned for each simulation. It is TRUE if the simulation did reach steady state, otherwise FALSE.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. specify negative parameter values or fractions > 1), the code will likely abort with an error message.

Author(s)

Andreas Handel

See Also

See the Shiny app documentation corresponding to this simulator function for more details on this model.

Examples

```
# To run the simulation with default parameters just call the function:
## Not run: result <- simulate_usanalysis()
# To choose parameter values other than the standard one, specify them, like such:
result <- simulate_usanalysis(dImin = 0.1, dImax = 10, samples = 5, tfinal = 50)
# You should then use the simulation result returned from the function, like this:
plot(result$dat[,"dI"],result$dat[,"Bpeak"],xlab='values for d',ylab='Peak Bacteria',type='l')
```

`simulate_virusandir_ode` *Simulation of a viral infection model with an immune response*

Description

This function runs a simulation of a compartment model which tracks uninfected and infected cells, virus, innate immune response, T-cells, B-cells and antibodies. The model is implemented as set of ordinary differential equations using the deSolve package.

Usage

```
simulate_virusandir_ode(
  U = 1e+05,
  I = 0,
  V = 10,
  T = 0,
  B = 0,
  A = 0,
  n = 0,
  dU = 0,
  dI = 1,
  dV = 4,
  b = 1e-05,
  p = 1000,
  sF = 0.01,
  kA = 1e-05,
  kT = 1e-05,
  pF = 1,
  dF = 1,
  gF = 1,
  Fmax = 1000,
  hV = 1e-06,
  hF = 1e-05,
  gB = 1,
  gT = 1e-04,
  rT = 0.5,
  rA = 10,
  dA = 0.2,
```

```
  tstart = 0,
  tfinal = 30,
  dt = 0.05
)
```

Arguments

U	: initial number of uninfected target cells : numeric
I	: initial number of infected target cells : numeric
V	: initial number of infectious virions : numeric
T	: initial number of T cells : numeric
B	: initial number of B cells : numeric
A	: initial number of antibodies : numeric
n	: rate of new uninfected cell replenishment : numeric
dU	: rate at which uninfected cells die : numeric
dI	: rate at which infected cells die : numeric
dV	: rate at which infectious virus is cleared : numeric
b	: rate at which virus infects cells : numeric
p	: rate at which infected cells produce virus : numeric
sF	: strength of innate response at reducing virus production : numeric
kA	: rate of virus removal by antibodies : numeric
kT	: rate of infected cell killing by T cells : numeric
pF	: rate of innate response production in absence of infection : numeric
dF	: rate of innate response removal in absence of infection : numeric
gF	: rate of innate response growth during infection : numeric
Fmax	: maximum level of innate response : numeric
hV	: innate growth saturation constant : numeric
hF	: B-cell growth saturation constant : numeric
gB	: maximum growth rate of B cells : numeric
gT	: T-cell induction rate : numeric
rT	: T-cell expansion rate : numeric
rA	: rate of antibody production by B cells : numeric
dA	: rate of antibody decay : numeric
tstart	: start time of simulation : numeric
tfinal	: final time of simulation : numeric
dt	: times for which result is returned : numeric

Details

A compartmental infection model is simulated as a set of ordinary differential equations, using an ode solver from the deSolve package. This code is part of the DSAIRM R package. For additional model details, see the corresponding app in the DSAIRM package.

Value

A list. The list has only one element, called ts. ts contains the time-series of the simulation. The 1st column of ts is time, the other columns are the model variables.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. specify negative parameter or starting values), the code will likely abort with an error message.

Author(s)

Andreas Handel

See Also

See the Shiny app documentation corresponding to this simulator function for more details on this model. See the manual for the deSolve package for details on the underlying ODE simulator algorithm.

Examples

```
# To run the simulation with default parameters just call the function:
result <- simulate_virusandir_ode()
# To choose parameter values other than the standard one, specify them, like such:
result <- simulate_virusandir_ode(V = 100, tfinal = 50, n = 1e5, dU = 1e-2, kT=1e-7)
# You should then use the simulation result returned from the function, like this:
plot(result$ts[,"time"],result$ts[,"V"],xlab='Time',ylab='Virus',type='l',log='y')
```

---

simulate_virusandtx_ode

Simulation of a basic viral infection model with drug treatment

---

Description

This function runs a simulation of a compartment model using a set of ordinary differential equations. The model describes a simple viral infection system in the presence of drug treatment. The user provides initial conditions and parameter values for the system. The function simulates the ODE using an ODE solver from the deSolve package. The function returns a list containing time-series of each variable and time. inspired by a study on HCV and IFN treatment (Neumann et al. 1998, Science)

Usage

```
simulate_virusandtx_ode(
  U = 1e+05,
  I = 0,
  V = 10,
  n = 10000,
```

```
  dU = 0.001,
  dI = 1,
  dV = 2,
  b = 1e-05,
  p = 10,
  g = 1,
  f = 0,
  e = 0,
  tstart = 0,
  tfinal = 30,
  dt = 0.1,
  steadystate = FALSE,
  txstart = 0
)
```

Arguments

U	: initial number of uninfected target cells : numeric
I	: initial number of infected target cells : numeric
V	: initial number of infectious virions : numeric
n	: rate of uninfected cell replenishment : numeric
dU	: rate at which uninfected cells die : numeric
dI	: rate at which infected cells die : numeric
dV	: rate at which infectious virus is cleared : numeric
b	: rate at which virus infects cells : numeric
p	: rate at which infected cells produce virus : numeric
g	: conversion between experimental and model virus units : numeric
f	: strength of cell infection reduction by drug : numeric
e	: strength of virus production reduction by drug : numeric
tstart	: Start time of simulation : numeric
tfinal	: Final time of simulation : numeric
dt	: times for which result is returned : numeric
steadystate	: start simulation at steady state : logical
txstart	: time at which treatment starts : numeric

Details

A simple compartmental model is simulated as a set of ordinary differential equations, using an ode solver from the deSolve package. if the steadystate input is set to TRUE, the starting values for U, I and V are set to their steady state values. Those steady state values are computed from the parameter values. See the Basic Virus Model To-do section for an explanation. In this case, user supplied values for U0, I0, V0 are ignored. This code is part of the DSAIRM R package. For additional model details, see the corresponding app in the DSAIRM package.

Value

A list. The list has only one element called ts. ts contains the time-series of the simulation. The 1st column of ts is Time, the other columns are the model variables.

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. specify negative parameter or starting values), the code will likely abort with an error message.

Examples

```
# To run the simulation with default parameters just call the function:
result <- simulate_virusandtx_ode()
# To choose parameter values other than the standard one, specify them, like such:
result <- simulate_virusandtx_ode(V = 100, tfinal = 100, n = 1e5, dU = 1e-2)
# You should then use the simulation result returned from the function, like this:
plot(result$ts[,"time"],result$ts[,"V"],xlab='Time',ylab='Virus',type='l',log='y')
```

Index

∗ **datasets**
    hayden96flu, 11
    schirm20strep, 12

check_simresults, 3

dsairmmenu, 3

generate_documentation, 4
generate_fctcall, 4
generate_ggplot, 5
generate_modelsettings, 6
generate_plotly, 7
generate_shinyinput, 9
generate_text, 10

hayden96flu, 11

run_model, *6*, 11

schirm20strep, 12
simulate_acutevirusir_ode, 13
simulate_bacteria_fit, 15
simulate_basicbacteria_discrete, 17
simulate_basicbacteria_modelexploration, 19
simulate_basicbacteria_ode, 21
simulate_basicvirus_fit, 22
simulate_basicvirus_modelexploration, 24
simulate_basicvirus_ode, 27
simulate_basicvirus_stochastic, 29
simulate_chronicvirusir_ode, 31
simulate_confint_fit, 33
simulate_drugresistance_stochastic, 35
simulate_extendedbacteria_ode, 36
simulate_fludrug_fit, 38
simulate_modelcomparison_fit, 40
simulate_modelvariants_ode, 43
simulate_pkpdmodel_ode, 45
simulate_usanalysis, 48
simulate_virusandir_ode, 50
simulate_virusandtx_ode, 52