Package 'DSAIDE'

July 4, 2019

Type Package

Title Dynamical Systems Approach to Infectious Disease Epidemiology

Description A collection of 'shiny' apps that allow for the simulation and exploration of various infectious disease transmission dynamics scenarios. The purpose of the package is to help individuals learn about infectious disease epidemiology from a dynamical systems perspective. All apps include explanations of the underlying models and instructions on what to do with the models.

Version 0.8.2

Date 2019-07-04

Maintainer Andreas Handel <ahandel@uga.edu>

License GPL-3

Encoding UTF-8

LazyData TRUE

Imports adaptivetau (>= 2.2), deSolve (>= 1.13), dplyr (>= 0.7.4), ggplot2 (>= 2.2), gridExtra (>= 2.3), lhs (>= 0.15), nloptr, plotly, stats (>= 3.4), utils (>= 3.4), XML (>= 3.98)

Depends R (>= 3.4), shiny (>= 1.0)

Suggests covr, devtools, knitr (>= 1.15), pkgdown, rmarkdown (>= 1.10), roxygen2, testthat

RoxygenNote 6.1.1

VignetteBuilder knitr

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

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

NeedsCompilation no

Author Andreas Handel [aut, cre] (<https://orcid.org/0000-0002-4622-1146>), Christine Casey [ctb] (Parasite Model App), Isaac Chun-Hai Fung [ctb], Ronald Galiwango [ctb] (Surveillance App),

Chase Golden [ctb] (Maternal Immunity App),
Spencer Hall [ctb],
Ben Listyg [ctb],
Brian McKay [ctb],
John Rossow [ctb],
Sina Solaimanpour [ctb],
Kevin Spiegel [ctb] (Global Warming App),
Henok Woldu [ctb]

Repository CRAN

Date/Publication 2019-07-04 16:00:15 UTC

R topics documented:

`DSAIDE` *DSAIDE: A package to learn about Dynamical Systems Approaches to Infectious Disease Epidemiology*

Description

The DSAIDE package provides a number of shiny apps that simulate various infectious disease dynamics models By manipulating the models and working through the instructions provided within the shiny UI, you can learn about some important concepts in infectious disease epidemiology. You will also learn how models can be used to study such concepts.

Details

Start the main menu of the package by calling dsaidemenu().

To learn more about how to use the package, see the vignette or the short introduction on the package github repository. https://github.com/ahgroup/DSAIDE

`dsaidemenu` *The main menu for the DSAIDE package*

Description

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

Usage

```
dsaidemenu()
```

Details

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

Author(s)

Andreas Handel

Examples

```
## Not run: dsaidemenu()
```

flu1918data *1918 Influenza mortality data*

Description

Weekly influenza deaths per 100,000 during the 1918 pandemic for New York City.

Usage

```
flu1918data
```

Format

A data frame with these variables:

Date Week of reporting

Deaths New deaths per week per 100,000

Details

Data is from Supplementary Table 1 of Mills et al 2004 Nature: https://www.nature.com/articles/nature03063

See this article and citations therein for more details on the data. Note that only a subset of the data is present here and the data are meant to be used for illustrative purposes only. For a proper re-analysis of these data, use the source mentioned above or check out data available through project Tycho: https://www.tycho.pitt.edu/

generate_documentation *A helper function for the UI part of the Shiny apps*

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	of 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

`generate_ggplot` *A helper function that takes result from the simulators and produces 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 is called by the Shiny server to produce plots returned to the Shiny UI. Create plots run the simulation with default parameters just call the function: result <- simulate_basicbacteria() plot <- generate_ggplot(result)

Value

A ggplot plot structure for display in a Shiny UI.

Author(s)

Andreas Handel

---

generate_plotly *A helper function that takes result from the simulators and produces 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 list indicates the number of separate plots to make. Each list entry corresponds to one plot and needs to contain the following information/elements: 1. A data frame 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 ggplot 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 is called by the Shiny server to produce plots returned to the Shiny UI. Create plots run the simulation with default parameters just call the function: result <- simulate_basicbacteria() plot <- generate_plotly(result)

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(mbmodel, otherinputs = NULL, packagename)
```

Arguments

mbmodel	a string containing the name of a simulator function for which to build inputs
otherinputs	a list of other shiny inputs to include in the UI
packagename	name of package using this function (DSAIDE or DSAIRM)

Details

This function is called by the Shiny app to produce the Shiny input UI elements. mbmodel is assumed to be the name of a function. The file correpsonding to this function is assumed to live in the simulatorfunctions subfolder and to be an exact copy of the same file in the /R folder of the source package. This R file will be loaded and the documentation parsed to create UI elements. Therefore, all simulator_ R functions/scripts need to follow a specific syntax. Every argument needs to be of the form param X : param explanation : param type example: b : transmission rate : numeric 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

`norodata` *Cases of norovirus during an outbreak*

Description

Norovirus case data from an outbreak among children on a school trip

Usage

```
norodata
```

Format

A data frame with these variables:

Date Day of outbreak, all in December 2007

Cases New cases for the specified date

Details

The data are from Kuo 2009 Wien Klin Woch: "A non-foodborne norovirus outbreak among school children during a skiing holiday, Austria, 2007" Specifically, the data comes from figure 1 of this article. The total number of susceptibles was 284.

See this article and citations therein for more details on the data. Note that only a subset of the data is present here and the data are meant for illustrative purposes only. For a proper re-analysis of these data, use the source mentioned above.

`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$simfunction - name of simulation function(s) as string. modelsettings$modeltype - specify what kind of model should be run. Currently one of: _ode_, _discrete_, _stochastic_, _usanalysis_, _modelexploration_, _fit_ . 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.

---

simulate_directtransmission_ode

Simulation of a compartmental infectious disease transmission model illustrating different types of direct transmission

---

Description

This model allows for the simulation of different direct transmission modes

Usage

```
simulate_directtransmission_ode(S = 1000, I = 1, bd = 0.01, bf = 0,
  A = 1, m = 0, n = 0, g = 0.1, w = 0, scenario = 1,
  tmax = 120)
```

Arguments

S	: initial number of susceptibles : numeric
I	: initial number of infected hosts : numeric
bd	: rate of transmission for density-dependent transmission : numeric
bf	: rate of transmission for frequency-dependent transmission : numeric
A	: the size of the area in which the hosts are assumed to reside/interact : numeric
m	: the rate of births : numeric
n	: the rate of natural deaths : numeric
g	: the rate at which infected hosts recover : numeric
w	: the rate of waning immunity : numeric
scenario	: choice between density dependent (=1) and frequency dependent (=2) transmission : numeric
tmax	: maximum simulation time, units of months : numeric

Details

A compartmental ID model with several states/compartments is simulated as a set of ordinary differential equations. The function returns the output from the odesolver as a list, with the element ts, which is a dataframe whose columns represent time, the number of susceptibles, the number of infected, and the number of recovered.

Value

This function returns the simulation result as obtained from a call to the deSolve ode solver.

Warning

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

Author(s)

Andreas Handel

References

See e.g. Keeling and Rohani 2008 for SIR models and the documentation for the deSolve package for details on ODE solvers

See Also

The UI of the Shiny app 'DirectTransmission', which is part of this package, contains more details on the model.

Examples

```
# To run the simulation with default parameters just call this function:
result <- simulate_directtransmission_ode()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_directtransmission_ode(S = 100, tmax = 100, A=10)
# You should then use the simulation result returned from the function, like this:
plot(result$ts[,"time"],result$ts[,"S"],xlab='Time',ylab='Number Susceptible',type='l')
```

`simulate_environmentaltransmission_ode`
Environmental Transmission model

Description

An SIR model including environmental transmission

Usage

```
simulate_environmentaltransmission_ode(S = 10000, I = 1, R = 0,
  E = 0, bd = 1e-04, be = 0, m = 0, n = 0, g = 0.2, p = 0,
  c = 10, tstart = 0, tfinal = 100, dt = 0.1)
```

Arguments

`S`	: starting value for Susceptible : numeric
`I`	: starting value for Infected : numeric
`R`	: starting value for Recovered : numeric
`E`	: starting value for Environmental Pathogen : numeric
`bd`	: direct transmission rate : numeric
`be`	: environmental transmission rate : numeric
`m`	: births : numeric
`n`	: natural deaths : numeric
`g`	: recovery rate : numeric
`p`	: shedding rate : numeric
`c`	: decay rate : numeric
`tstart`	: Start time of simulation : numeric
`tfinal`	: Final time of simulation : numeric
`dt`	: Time step : numeric

Details

The model includes susceptible, infected, recovered and environmental compartments.

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_environmentaltransmission_ode()
```

---

simulate_evolution_stochastic

Stochastic simulation of a compartmental SIR-type model with wild-type and mutant strains and treatment

---

Description

Simulation of a stochastic 2-strain SIR model with the following compartments: Susceptibles (S), Infected with wild-type/sensitive and untreated (Iu), Infected with wild-type and treated (It), infected with resistant (Ir), Recovered and Immune (R). allows exploration of evolutionary dynamics

Usage

```
simulate_evolution_stochastic(S = 1000, Iu = 1, It = 1, Ir = 1,
  bu = 0.001, bt = 0.001, br = 0.001, cu = 0.001, ct = 0.01,
  f = 0, gu = 1, gt = 1, gr = 1, tmax = 100, rngseed = 100)
```

Arguments

`S`	: initial number of susceptible hosts : numeric
`Iu`	: initial number of wild-type infected untreated hosts : numeric
`It`	: initial number of wild-type infected treated hosts : numeric
`Ir`	: initial number of resistant infected hosts : numeric
`bu`	: level/rate of infectiousness for hosts in the Iu compartment : numeric
`bt`	: level/rate of infectiousness for hosts in the It compartment : numeric
`br`	: level/rate of infectiousness for hosts in the Ir compartment : numeric
`cu`	: fraction of resistant mutant infections that an untreated host produces : numeric
`ct`	: fraction of resistant mutant infections that a treated host produces : numeric

f	: fraction of infected receiving treatment : numeric
gu	: rate at which a host leaves the Iu compartment : numeric
gt	: rate at which a person leaves the It compartment : numeric
gr	: rate at which a person leaves the Ir compartment : numeric
tmax	: maximum simulation time : numeric
rngseed	: seed for random number generator to allow reproducibility : 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 adaptivetau package. See the manual of this package for more details.

Value

This function returns the simulation result as obtained from a call to the adaptivetau integrator in list form. The list element ts is a dataframe where the first column is "time," and the remaining columns are the variables

Warning

This function does not perform any error checking. So if you try to do something nonsensical (e.g. have 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 2009 JTB "Antiviral resistance and the control of pandemic influenza: The roles of stochasticity, evolution and model details"

Examples

```
# To run the simulation with default parameters just call the function:
result <- simulate_evolution_stochastic()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_evolution_stochastic(S = 2000,  tmax = 200, bt = 0.01)
# You should then use the simulation result returned from the function, like this:
plot(result$ts[ , "time"],result$ts[ , "S"],xlab='Time',ylab='Number Susceptible',type='l')
```

`simulate_fit_flu` *Fitting a SIR-type model to flu data*

Description

Fitting fitting mortality data from the 1918 influenza pandemic to an SIR-type model to estimate R0. For the data, see 'flu1918data'.

Usage

```
simulate_fit_flu(S = 5e+06, I = 1, D = 0, b = 1e-06,
  blow = 1e-10, bhigh = 0.1, bsim = 1e-04, g = 1, glow = 0.001,
  ghigh = 100, gsim = 1, f = 0.01, flow = 1e-05, fhigh = 0.5,
  fsim = 0.01, noise = 0, iter = 100, solvertype = 1,
  usesimdata = 0, logfit = 0)
```

Arguments

`S`	: starting value for Susceptible : numeric
`I`	: starting value for Infected : numeric
`D`	: starting value for Dead : numeric
`b`	: infection rate : numeric
`blow`	: lower bound for infection rate : numeric
`bhigh`	: upper bound for infection rate : numeric
`bsim`	: infection rate for simulated data : numeric
`g`	: recovery rate : numeric
`glow`	: lower bound for g : numeric
`ghigh`	: upper bound for g : numeric
`gsim`	: recovery rate for simulated data : numeric
`f`	: fraction dying : numeric
`flow`	: lower bound for f : numeric
`fhigh`	: upper bound for f : numeric
`fsim`	: fraction dying 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
`logfit`	: set to 1 if the log of the data should be fitted, 0 otherwise : numeric

Details

A simple compartmental ODE model is fitted to data. The model includes susceptible, infected, and dead compartments. The two processes that are modeled are infection and recovery. A fraction of recovered can die. 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_fit_flu()
# To apply different settings, provide them to the simulator function, like such:
result <- simulate_fit_flu(iter = 5, logfit = 1, solvertype = 2, usesimdata = 1)
```

---

simulate_fit_noro *Fitting a simple SIR type model to norovirus outbreak data*

Description

This function runs a simulation of a compartment model using a set of ordinary differential equations. The model describes a simple SIR model with an additional environmental source of infection 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_fit_noro(S = 100, I = 1, R = 0, b = 0.001, blow = 1e-10,
  bhigh = 0.1, g = 0.5, glow = 0.001, ghigh = 100, n = 0,
  nlow = 0, nhigh = 1000, t1 = 8, t2 = 15, fitmodel = 1,
  iter = 100, solvertype = 1)
```

Arguments

S	: starting value for Susceptible : numeric
I	: starting value for Infected : numeric
R	: starting value for Recovered : numeric
b	: infection rate : numeric
blow	: lower bound for infection rate : numeric
bhigh	: upper bound for infection rate : numeric
g	: recovery rate : numeric
glow	: lower bound for g : numeric
ghigh	: upper bound for g : numeric
n	: rate of infection from common source : numeric
nlow	: lower bound for n : numeric
nhigh	: upper bound for n : numeric
t1	: start time of infection from common source : numeric
t2	: end time of infection from common source: numeric
fitmodel	: fitting model variant 1, 2 or 3 : numeric
iter	: max number of steps to be taken by optimizer : numeric
solvertype	: the type of solver/optimizer to use (1-3) : numeric

Details

Three versions of a simple SIR type compartmental ODE model are fit to cases of norovirus during an outbreak. #' @section 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.

Value

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

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_fit_noro()
# To apply different settings, provide them to the simulator function, like such:
result <- simulate_fit_noro(iter = 5, fitmodel = 2)
```

`simulate_heterogeneity_ode` *Simulation of a compartmental infectious disease transmission model with 2 types of hosts*

Description

This model allows for the simulation of an ID with 2 types of hosts

Usage

```
simulate_heterogeneity_ode(S1 = 1000, I1 = 1, S2 = 1000, I2 = 0,
  b11 = 0.01, b12 = 0, b21 = 0, b22 = 0, g1 = 1, g2 = 1,
  w1 = 0, w2 = 0, tmax = 120)
```

Arguments

`S1`	: initial number of susceptible type 1 hosts : numeric
`I1`	: initial number of infected type 1 hosts : numeric
`S2`	: initial number of susceptible type 2 hosts : numeric
`I2`	: initial number of infected type 2 hosts : numeric
`b11`	: rate of transmission from infected type 1 host to susceptible type 1 host : numeric
`b12`	: rate of transmission from infected type 1 host to susceptible type 2 host : numeric
`b21`	: rate of transmission from infected type 2 host to susceptible type 1 host : numeric
`b22`	: rate of transmission from infected type 2 host to susceptible type 2 host : numeric
`g1`	: the rate at which infected type 1 hosts recover : numeric
`g2`	: the rate at which infected type 2 hosts recover : numeric
`w1`	: the rate at which type 1 host immunity wanes : numeric
`w2`	: the rate at which type 2 host immunity wanes : numeric
`tmax`	: maximum simulation time, units of months : numeric

Details

A compartmental ID model with several states/compartments is simulated as a set of ordinary differential equations. The function returns the output from the odesolver as a matrix, with one column per compartment/variable. The first column is time.

Value

This function returns the simulation result as obtained from a call to the deSolve ode solver.

Warning

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

Author(s)

Andreas Handel

References

See e.g. Keeling and Rohani 2008 for SIR models and the documentation for the deSolve package for details on ODE solvers

See Also

The UI of the Shiny app 'Host Heterogeneity', which is part of this package, contains more details on the model.

Examples

```
# To run the simulation with default parameters just call the function:
result <- simulate_heterogeneity_ode()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_heterogeneity_ode(S1 = 100, S2 = 1e3, b11 = 0.7, tmax = 100)
# You should then use the simulation result returned from the function, like this:
plot(result$ts[,"time"],result$ts[,"S1"],xlab='Time',ylab='Number Susceptible 1',type='l')
```

---

simulate_idcharacteristics_ode

Simulation of an infectious disease transmission model with multiple compartments

---

Description

Simulation of a compartmental model with several different compartments: Susceptibles (S), Infected and Pre-symptomatic (P), Infected and Asymptomatic (A), Infected and Symptomatic (I), Recovered and Immune (R) and Dead (D)

Usage

```
simulate_idcharacteristics_ode(S = 1000, P = 1, bP = 0, bA = 0,
  bI = 0.001, gP = 0.5, gA = 0.5, gI = 0.5, f = 0, d = 0,
  tmax = 300)
```

Arguments

S	: initial number of susceptible hosts : numeric
P	: initial number of infected, pre-symptomatic hosts : numeric
bP	: level/rate of infectiousness for hosts in the P compartment : numeric
bA	: level/rate of infectiousness for hosts in the A compartment : numeric
bI	: level/rate of infectiousness for hosts in the I compartment : numeric
gP	: rate at which a person leaves the P compartment, which : numeric
gA	: rate at which a person leaves the A compartment : numeric
gI	: rate at which a person leaves the A compartment : numeric
f	: fraction of pre-symptomatic individuals that have an asymptomatic infection : numeric
d	: fraction of symptomatic infected hosts that die due to disease : numeric
tmax	: maximum simulation time : numeric

Details

A compartmental ID model with several states/compartments is simulated as a set of ordinary differential equations. The states are: **S**: Susceptible, uninfected individuals **P**: Presymptomatic individuals who are infected and possibly infectious **A**: Asymptomatic individuals who are infected and possibly infectious **I**: Sympomatic infected individuals, most likely infectious **R**: Removed / recovered individuals, no longer infectious or susceptible **D**: Individuals who have died from the disease The model app contains detailed information on the processes, but briefly, susceptible (S) individuals can become infected by presymptomatic (P), asymptomatic (A), or infected (I) hosts. All infected individuals enter the presymptomatic stage first, from which they can become symptomatic or asymptomatic. Asymptomatic hosts recover within some specified duration of time, while infected hosts either recover or die, thus entering either R or D. Recovered individuals are immune to reinfection.

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. have negative values for parameters), the code will likely abort with an error message.

Author(s)

Andreas Handel

References

See e.g. Keeling and Rohani 2008 for SIR models and the documentation for the deSolve package for details on ODE solvers

Examples

```
# To run the simulation with default parameters just call the function:
result <- simulate_idcharacteristics_ode()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_idcharacteristics_ode(S = 2000, P = 10, tmax = 100, f = 0.1, d = 0.2)
# You should then use the simulation result returned from the function, like this:
plot(result$ts[,"time"],result$ts[,"S"],xlab='Time',ylab='Number Susceptible',type='l')
```

---

simulate_idcontrolmultioutbreak_ode

Simulation of a compartmental infectious disease transmission model to study the reproductive number

---

Description

Simulation of a basic SIR compartmental model with these compartments: Susceptibles (S), Infected/Infectious (I), Recovered and Immune (R).

The model is assumed to be in units of months when run through the Shiny App. However as long as all parameters are chosen in the same units, one can directly call the simulator assuming any time unit.

Usage

```
simulate_idcontrolmultioutbreak_ode(S = 1000, I = 1, R = 0,
  b = 0.001, g = 1, f = 0.3, tstart = 10, tend = 50, tnew = 50,
  tmax = 100)
```

Arguments

S	: initial number of susceptible hosts : numeric
I	: initial number of infected hosts : numeric
R	: initial number of recovered hosts : numeric
b	: rate of new infections : numeric
g	: rate of recovery : numeric
f	: strength of intervention effort : numeric
tstart	: time at which intervention effort starts : numeric
tend	: time at which intervention effort ends : numeric
tnew	: time at which new infected enter : numeric
tmax	: maximum simulation time : numeric

Details

A compartmental ID model with several states/compartments is simulated as a set of ordinary differential equations. The function returns the output from the odesolver as a matrix, with one column per compartment/variable. The first column is time. The model implement basic processes of infection at rate b and recovery at rate g. Treatment is applied, which reduces b by the indicated proportion, during times tstart and tend. At time intervals given by tnew, a new infected individual enters the population. The simulation also monitors the number of infected and when they drop below 1, they are set to 0.

Value

This function returns the simulation result as obtained from a call to the deSolve ode solver.

Warning

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

Author(s)

Andreas Handel

See Also

The UI of the app 'Multi Outbreak ID Control', which is part of the DSAIDE package, contains more details.

Examples

```
# To run the simulation with default parameters just call the function:
result <- simulate_idcontrolmultioutbreak_ode()
# To choose parameter values other than the standard one,
# specify the parameters you want to change, e.g. like such:
result <- simulate_idcontrolmultioutbreak_ode(S = 2000, I = 10, tmax = 100, g = 0.5)
# You should then use the simulation result returned from the function, like this:
plot(result$ts[ , "time"],result$ts[ , "S"],xlab='Time',ylab='Number Susceptible',type='l')
```

---

simulate_idcontrol_ode

Simulation of a compartmental infectious disease transmission model including different control mechanisms

---

Description

Simulation of a compartmental model with several different compartments: Susceptibles (S), Infected and Pre-symptomatic (P), Infected and Asymptomatic (A), Infected and Symptomatic (I), Recovered and Immune (R) and Dead (D). Also modeled is an environmental pathogen stage (E), and susceptible (Sv) and infected (Iv) vectors.

Any initial conditions not specified below start at 0.

Usage

```
simulate_idcontrol_ode(S = 1000, I = 1, E = 0, Sv = 1000, Iv = 0,
  bP = 0, bA = 0, bI = 0.001, bE = 0, bv = 0.001, bh = 0.001,
  gP = 0.5, gA = 0.5, gI = 0.5, pA = 1, pI = 10, c = 1,
  f = 0, d = 0, w = 0, mh = 0, nh = 0, mv = 0, nv = 0,
  tmax = 300)
```

Arguments

S	: initial number of susceptible hosts : numeric
I	: initial number of infected and symptomatic hosts : numeric
E	: initial amount of pathogen in environment : numeric
Sv	: initial number of susceptible vectors : numeric
Iv	: initial number of infected vectors : numeric
bP	: rate of transmission from pre-symptomatic to susceptible hosts : numeric
bA	: rate of transmission from asymptomatic to susceptible hosts : numeric
bI	: rate of transmission from symptomatic to susceptible hosts : numeric
bE	: rate of transmission from environment to susceptible hosts : numeric
bv	: rate of transmission from infected vectors to susceptible hosts : numeric
bh	: rate of transmission from symptomatic hosts to susceptible vectors : numeric
gP	: rate at which a person leaves the P compartment : numeric
gA	: rate at which a person leaves the A compartment : numeric
gI	: rate at which a person leaves the I compartment : numeric
pA	: rate of pathogen shedding into environment by asymptomatic hosts : numeric
pI	: rate of pathogen shedding into environment by symptomatic hosts : numeric
c	: rate of pathogen decay in environment : numeric
f	: fraction of pre-symptomatic individuals that have an asymptomatic infection : numeric
d	: fraction of symptomatic infected hosts that die due to disease : numeric
w	: rate at which recovered persons lose immunity and return to susceptible state : numeric
mh	: the rate at which new hosts enter the model (are born) : numeric
nh	: the rate of natural death of hosts (the inverse it the average lifespan) : numeric
mv	: the rate at which new vectors enter the model (are born) : numeric
nv	: the rate of natural death of vectors (the inverse it the average lifespan) : numeric
tmax	: maximum simulation time, in units of months : numeric

Details

A compartmental ID model with several states/compartments is simulated as a set of ordinary differential equations. The function returns the output from the odesolver as a matrix, with one column per compartment/variable. The first column is time.

Value

This function returns the simulation result as obtained from a call to the deSolve ode solver.

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 e.g. Keeling and Rohani 2008 for SIR models and the documentation for the deSolve package for details on ODE solvers

See Also

The UI of the Shiny app 'IDPatterns', which is part of this package, contains more details on the model.

Examples

```
# To run the simulation with default parameters just call the function:
result <- simulate_idcontrol_ode()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_idcontrol_ode(S = 2000, I = 10, tmax = 100, f = 0.1, d = 0.2)
# You should then use the simulation result returned from the function, like this:
plot(result$ts[ , "time"], result$ts[ , "S"],xlab='Time',ylab='Number Susceptible',type='l')
```

---

simulate_idpatterns_ode

Simulation of a compartmental infectious disease transmission model including seasonality

---

Description

Simulation of a compartmental model with several different compartments: Susceptibles (S), Infected and Pre-symptomatic (P), Infected and Asymptomatic (A), Infected and Symptomatic (I), Recovered and Immune (R) and Dead (D).

This model includes natural births and deaths and waning immunity. It also allows for seasonal variation in transmission. The model is assumed to run in units of months. This assumption is hard-coded into the sinusoidally varying transmission coefficient, which is assumed to have a period of a year.

Usage

```
simulate_idpatterns_ode(S = 1000, P = 1, bP = 0, bA = 0,
  bI = 0.001, s = 0, gP = 0.5, gA = 0.5, gI = 0.5, f = 0,
  d = 0, w = 0, m = 0, n = 0, timeunit = 1, tmax = 300)
```

Arguments

S	: initial number of susceptible hosts : numeric
P	: initial number of infected, pre-symptomatic hosts : numeric
bP	: level/rate of infectiousness for hosts in the P compartment : numeric
bA	: level/rate of infectiousness for hosts in the A compartment : numeric
bI	: level/rate of infectiousness for hosts in the I compartment : numeric
s	: strength of seasonal/annual sigmoidal variation of transmission rate : numeric
gP	: rate at which a person leaves the P compartment : numeric
gA	: rate at which a person leaves the A compartment : numeric
gI	: rate at which a person leaves the I compartment : numeric
f	: fraction of pre-symptomatic individuals that have an asymptomatic infection : numeric
d	: fraction of symptomatic infected hosts that die due to disease : numeric
w	: rate at which recovered persons lose immunity and return to susceptible state : numeric
m	: the rate at which new individuals enter the model (are born) : numeric
n	: the rate of natural death (the inverse it the average lifespan) : numeric
timeunit	: units of time in which the model should run (1=day, 2=week, 3=month, 4=year) : numeric
tmax	: maximum simulation time, in units of months : numeric

Details

A compartmental ID model with several states/compartments is simulated as a set of ordinary differential equations. The function returns the output from the odesolver as a matrix, with one column per compartment/variable. The first column is time.

Value

This function returns the simulation result as obtained from a call to the deSolve ode solver.

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 e.g. Keeling and Rohani 2008 for SIR models and the documentation for the deSolve package for details on ODE solvers

See Also

The UI of the app, which is part of this package, contains more details on the model.

Examples

```
# To run the simulation with default parameters just call the function:
result <- simulate_idpatterns_ode()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_idpatterns_ode(S = 2000, P = 10, tmax = 100, f = 0.1, d = 0.2, s = 0.1)
# You should then use the simulation result returned from the function, like this:
plot(result$ts[ , "time"],result$ts[ , "S"],xlab='Time',ylab='Number Susceptible',type='l')
```

---

simulate_idsurveillance_ode

Simulation of a compartmental infectious disease transmission model illustrating the impact of ID surveillance

---

Description

This model allows for the exploration of the impact of ID surveillance on transmission dynamics

Usage

```
simulate_idsurveillance_ode(S = 1000, P = 1, tmax = 200, bP = 0,
  bA = 0, bI = 0.001, gP = 0.5, f = 0, d = 0, w = 0, m = 0,
  n = 0, rP = 0, rA = 0, rI = 0.5)
```

Arguments

S	: initial number of susceptible hosts : numeric
P	: initial number of infected pre-symptomatic hosts : numeric
tmax	: maximum simulation time : numeric
bP	: rate of transmission from presymptomatic to susceptible host : numeric
bA	: rate of transmission from asymptomatic to susceptible host : numeric
bI	: rate of transmission from symptomatic to susceptible host : numeric
gP	: the rate at which presymptomatic hosts move to the next stage : numeric
f	: fraction of asymptomatic hosts : numeric

d	: rate at which infected hosts die : numeric
w	: the rate at which host immunity wanes : numeric
m	: the rate of births : numeric
n	: the rate of natural deaths : numeric
rP	: rate of pre-symptomatic host removal due to surveillance : numeric
rA	: rate of asymptomatic host removal due to surveillance : numeric
rI	: rate of symptomatic host removal due to surveillance : numeric

Details

A compartmental ID model with several states/compartments is simulated as a set of ordinary differential equations. The function returns the output from the odesolver as a matrix, with one column per compartment/variable. The first column is time.

Value

This function returns the simulation result as obtained from a call to the deSolve ode solver.

Warning

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

Author(s)

Andreas Handel, Ronald Galiwango

See Also

The UI of the app 'Parasite Model', which is part of the DSAIDE package, contains more details.

Examples

```
# To run the simulation with default parameters just call the function:
result <- simulate_idsurveillance_ode()
# To choose parameter values other than the standard one,
# specify the parameters you want to change, e.g. like such:
result <- simulate_idsurveillance_ode(S = 2000, tmax = 100, f = 0.5)
# You should then use the simulation result returned from the function, like this:
plot(result$ts[ , "time"],result$ts[ , "S"],xlab='Time',ylab='Number Susceptible',type='l')
```

`simulate_maternalimmunity_ode` *Simulation of a MSEIR model that represents a group of the population that is protected by disease through maternal antibodies*

Description

This function runs a simulation of a MSEIR model using a set of 6 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_maternalimmunity_ode(S = 1000, I = 1, tmax = 1000,
  m = 0.02, n = 0.02, p = 0.005, b = 0.3, gE = 0.1, gI = 0.02,
  w = 0)
```

Arguments

`S`	: initial number of susceptible individuals : numeric
`I`	: initial number of infected hosts : numeric
`tmax`	: maximum simulation time : numeric
`m`	: rate at which individuals are born : numeric
`n`	: rate at which individuals die : numeric
`p`	: rate at which individuals lose passive immunity : numeric
`b`	: rate of new infections : numeric
`gE`	: rate of leaving latent stage : numeric
`gI`	: rate of recovery : numeric
`w`	: rate of waning immunity : numeric

Details

A simple MSEIR 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 values or fractions > 1), the code will likely abort with an error message

Author(s)

Chase Golden, 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 this function
result <- simulate_maternalimmunity_ode()
# To choose parameter values other than the standard one, specify them e.g. like such
result <- simulate_maternalimmunity_ode(S = 2000, I = 10, tmax = 100, b = 0.2)
# You should then use the simulation result returned from the function, e.g. like this:
plot(result$ts[ , "time"], result$ts[ , "S"],xlab='Time',ylab='Number Susceptible',type='l')
```

---

simulate_modelexploration_sir

Simulation to illustrate parameter scan of the basic SIR model with births and deaths #'

Description

This function simulates the SIR 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_modelexploration_sir(S = 1000, I = 1, R = 0, b = 0.002,
  g = 1, m = 0, n = 0, tstart = 0, tfinal = 100, dt = 0.1,
  samples = 10, parmin = 5e-04, parmax = 0.005, samplepar = "b",
  pardist = "lin")
```

Arguments

S	: starting value for Susceptible : numeric
I	: starting value for Infected : numeric
R	: starting value for Recovered : numeric
b	: infection rate : numeric
g	: recovery rate : numeric
m	: the rate at which new individuals enter the model (are born) : numeric
n	: the rate of natural death (the inverse is the average lifespan) : 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 systematically analyze the impact of a specific parameter. The SIR ODE model with births and deaths is simulated for different parameter values. The user can specify which parameter is sampled, and the simulation returns for each parameter sample the max and final value for the variables. 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 maximum and final state numbers of susceptible, infected and recovered Smax, Imax, Rmax and Sfinal, Ifinal, Rfinal. 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_modelexploration_sir()
# To choose parameter values other than the standard one, specify them, like such:
res <- simulate_modelexploration_sir(tfinal=100, samples=5, samplepar='g', parmin=0.1, parmax=1)
# You should then use the simulation result returned from the function, like this:
plot(res$dat[,"xvals"],res$data[,"Imax"],xlab='Parameter values',ylab='Max Infected',type='l')
```

`simulate_multipathogen_ode` *Simulation of a compartmental infectious disease transmission model with 2 types of pathogens*

Description

This model allows for the simulation of 2 IDs in a single host

Usage

```
simulate_multipathogen_ode(S = 1000, I1 = 1, I2 = 0, I12 = 0,
  b1 = 0.001, b2 = 0, b12 = 0, g1 = 1, g2 = 1, g12 = 1,
  a = 0, tmax = 120)
```

Arguments

`S`	: initial number of susceptible hosts : numeric
`I1`	: initial number of hosts infected with type 1 : numeric
`I2`	: initial number of hosts infected with type 2 : numeric
`I12`	: initial number of double infected hosts : numeric
`b1`	: rate at which type 1 infected hosts transmit : numeric
`b2`	: rate at which type 2 infected hosts transmit : numeric
`b12`	: rate at which double infected hosts transmit : numeric
`g1`	: the rate at which infected type 1 hosts recover : numeric
`g2`	: the rate at which infected type 2 hosts recover : numeric
`g12`	: the rate at which double infected hosts recover : numeric
`a`	: fraction of type 1 infections produced by double infected hosts : numeric
`tmax`	: maximum simulation time, units of months : numeric

Details

A compartmental ID model with several states/compartments is simulated as a set of ordinary differential equations. The function returns the output from the odesolver as a matrix, with one column per compartment/variable. The first column is time.

Value

This function returns the simulation result as obtained from a call to the deSolve ode solver.

Warning

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

Author(s)

Andreas Handel and Spencer Hall

References

See e.g. Keeling and Rohani 2008 for SIR models and the documentation for the deSolve package for details on ODE solvers

See Also

The UI of the Shiny app 'Multi-Pathogen Dynamics', which is part of this package, contains more details on the model

Examples

```
# To run the simulation with default parameters just call the function:
result <- simulate_multipathogen_ode()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_multipathogen_ode(S = 100, I2 = 10, tmax = 100, b1 = 2.5)
# You should then use the simulation result returned from the function, like this:
plot(result$ts[,"time"],result$ts[,"I1"], xlab="Time",ylab="Number Infected Type 1",type="l")
```

---

simulate_parasites_ode

Simulation of a compartmental infectious disease transmission model illustrating parasite infection dynamics with intermediate hosts

---

Description

This model allows for the simulation of a parasitic infection that requires an intermediate host for transmission

Usage

```
simulate_parasites_ode(Sh = 1000, Ih = 1, E = 1, Si = 0, Ii = 0,
  tmax = 120, bi = 0.01, be = 0.01, m = 0, n = 0, g = 0,
  w = 0, p = 0.01, c = 0.001)
```

Arguments

Sh	: initial number of susceptible definitive hosts : numeric
Ih	: initial number of infected definitive hosts : numeric
E	: initial number of pathogens in the environment : numeric
Si	: initial number of susceptible intermediate hosts : numeric
Ii	: initial number of infected intermediate hosts : numeric
tmax	: maximum simulation time : numeric
bi	: rate of transmission from infected intermediate host to susceptible host : numeric
be	: rate of transmission from environment to susceptible intermediate host : numeric
m	: the rate of births of intermediate hosts : numeric
n	: the rate of natural intermediate hosts : numeric
g	: the rate at which infected hosts recover/die : numeric
w	: the rate at which host immunity wanes in host : numeric
p	: rate at which infected host shed the pathogen in the environment : numeric
c	: rate at which the pathogen decays in the environment : numeric

Details

A compartmental ID model with several states/compartments is simulated as a set of ordinary differential equations. The function returns the output from the odesolver as a matrix, with one column per compartment/variable. The first column is time.

Value

This function returns the simulation result as obtained from a call to the deSolve ode solver.

Warning

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

Author(s)

Andreas Handel, Christine Casey

See Also

The UI of the app 'Parasite Model', which is part of the DSAIDE package, contains more details.

Examples

```
# To run the simulation with default parameters just call the function:
result <- simulate_parasites_ode()
# To choose parameter values other than the standard one,
# specify the parameters you want to change, e.g. like such:
result <- simulate_parasites_ode(Sh = 2000, Ih = 10, tmax = 100, g = 0.5)
# You should then use the simulation result returned from the function, like this:
plot(result$ts[ , "time"],result$ts[ , "Sh"],xlab='Time',ylab='Number Susceptible',type='l')
```

`simulate_reproductivenumber1_ode` *Basic SIR model*

Description

A basic SIR model with 3 compartments and infection and recovery processes

Usage

```
simulate_reproductivenumber1_ode(S = 1000, I = 1, R = 0, b = 0.002,
  g = 1, tstart = 0, tfinal = 100, dt = 0.1)
```

Arguments

`S`	: starting value for Susceptible : numeric
`I`	: starting value for Infected : numeric
`R`	: starting value for Recovered : numeric
`b`	: infection rate : numeric
`g`	: recovery rate : numeric
`tstart`	: Start time of simulation : numeric
`tfinal`	: Final time of simulation : numeric
`dt`	: Time step : numeric

Details

The model includes susceptible, infected, and recovered compartments. The two processes that are modeled are infection and recovery. The simulation also monitors the number of infected and when they drop below 1, they are set to 0.

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_reproductivenumber1_ode()
```

---

simulate_reproductivenumber2_ode

Simulation of a compartmental infectious disease transmission model to study the reproductive number

---

Description

Simulation of a basic SIR compartmental model with these compartments: Susceptibles (S), Infected/Infectious (I), Recovered and Immune (R).

The model is assumed to be in units of months when run through the Shiny App. However as long as all parameters are chosen in the same units, one can directly call the simulator assuming any time unit.

Usage

```
simulate_reproductivenumber2_ode(S = 1000, I = 1, f = 0, e = 0,
  b = 0.01, g = 10, m = 0, n = 0, w = 0, tmax = 300)
```

Arguments

S	: initial number of susceptible hosts : numeric
I	: initial number of infected hosts : numeric
f	: fraction of vaccinated individuals. Those individuals are moved from S to R at the beginning of the simulation : numeric
e	: efficacy of vaccine, given as fraction between 0 and 1 : numeric
b	: level/rate of infectiousness for hosts in the I compartment : numeric
g	: rate at which a person leaves the I compartment : numeric
m	: the rate at which new individuals enter the model (are born) : numeric
n	: the rate of natural death (the inverse it the average lifespan) : numeric
w	: rate at which recovered persons lose immunity and return to susceptible state : numeric
tmax	: maximum simulation time : numeric

Details

A compartmental ID model with several states/compartments is simulated as a set of ordinary differential equations. The function returns the output from the odesolver as a matrix, with one column per compartment/variable. The first column is time.

Value

This function returns the simulation result as obtained from a call to the deSolve ode solver.

Warning

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

Author(s)

Andreas Handel

References

See e.g. Keeling and Rohani 2008 for SIR models and the documentation for the deSolve package for details on ODE solvers

See Also

The UI of the app 'ReproductiveNumber 2', which is part of this package, contains more details on the model.

Examples

```
# To run the simulation with default parameters just call the function:
result <- simulate_reproductivenumber2_ode()
# To choose parameter values other than the standard one,
# specify the parameters you want to change, e.g. like such:
result <- simulate_reproductivenumber2_ode(S = 2000, I = 10, tmax = 100, g = 0.5, n = 0.1)
# You should then use the simulation result returned from the function, like this:
plot(result$ts[ , "time"],result$ts[ , "S"],xlab='Time',ylab='Number Susceptible',type='l')
```

---

simulate_seir_stochastic

Stochastic simulation of an SEIR-type model

Description

Simulation of a stochastic SEIR type model with the following compartments: Susceptibles (S), Infected and pre-symptomatic/exposed (E), Infected and Symptomatic (I), Recovered and Immune (R)

Usage

```
simulate_seir_stochastic(S = 1000, I = 10, bE = 0, bI = 0.001,
  gE = 0.5, gI = 0.5, w = 0, m = 0, n = 0, tmax = 100,
  rngseed = 100)
```

Arguments

S	: initial number of susceptible hosts : numeric
I	: initial number of infected, symptomatic hosts : numeric
bE	: level/rate of infectiousness for hosts in the E compartment : numeric
bI	: level/rate of infectiousness for hosts in the I compartment : numeric
gE	: rate at which a person leaves the E compartment : numeric
gI	: rate at which a person leaves the I compartment : numeric
w	: rate at which recovered lose immunity and return to susceptible : numeric
m	: the rate at which new individuals enter the model (are born) : numeric
n	: the rate of natural death (the inverse is the average lifespan) : numeric
tmax	: maximum simulation time : numeric
rngseed	: seed for random number generator to allow reproducibility : numeric

Details

A compartmental ID model with several states/compartments is simulated. Initial conditions for the E and R variables are 0. Units of time depend on the time units chosen for model parameters. The simulation runs 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.

Value

The function returns a list. The list has one element, a data frame ts which contains the time series of the simulated model, 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 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. See the manual for the adaptivetau package for details on the stochastic algorithm.

Examples

```
# To run the simulation with default parameters, just call the function:
result <- simulate_seir_stochastic()
# To choose parameter values other than the standard one, specify them like this:
result <- simulate_seir_stochastic(S = 2000,  tmax = 200, bE = 0.01)
# You can display or further process the result, like this:
plot(result$ts[,'time'],result$ts[,'S'],xlab='Time',ylab='Number Susceptible',type='l')
print(paste('Max number of infected: ',max(result$ts[,'I'])))
```

`simulate_sirdemographic_ode`

Basic SIR model with births and deaths

Description

A basic SIR model with 3 compartments, infection and recovery and birth and death processes

Usage

```
simulate_sirdemographic_ode(S = 1000, I = 1, R = 0, b = 0.002,
  g = 1, m = 0, n = 0, tstart = 0, tfinal = 100, dt = 0.1)
```

Arguments

`S`	: starting value for Susceptible : numeric
`I`	: starting value for Infected : numeric
`R`	: starting value for Recovered : numeric
`b`	: infection rate : numeric
`g`	: recovery rate : numeric
`m`	: the rate at which new individuals enter the model (are born) : numeric
`n`	: the rate of natural death (the inverse is the average lifespan) : numeric
`tstart`	: Start time of simulation : numeric
`tfinal`	: Final time of simulation : numeric
`dt`	: Time step : numeric

Details

The model includes susceptible, infected, and recovered compartments. The two infection related processes that are modeled are infection and recovery. Natural births and deaths are also included.

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_sirdemographic_ode()
```

`simulate_sirdemographic_stochastic`
Stochastic simulation of an SIR-type model with births and deaths

---

Description

Simulation of a stochastic SIR type model with the following compartments: Susceptibles (S), Infected and Infectious (I), Recovered and Immune (R)

Usage

```
simulate_sirdemographic_stochastic(S = 1000, I = 10, R = 0,
  b = 0.001, g = 0.5, m = 0, n = 0, tmax = 100, rngseed = 100)
```

Arguments

`S`	: initial number of susceptible hosts : numeric
`I`	: initial number of infected, symptomatic hosts : numeric
`R`	: initial number of recovered hosts : numeric
`b`	: level/rate of infectiousness for hosts in the I compartment : numeric
`g`	: rate at which a person leaves the I compartment : numeric
`m`	: the rate at which new individuals enter the model (are born) : numeric
`n`	: the rate of natural death (the inverse is the average lifespan) : numeric
`tmax`	: maximum simulation time : numeric
`rngseed`	: seed for random number generator to allow reproducibility : numeric

Details

A compartmental SIR model is simulated. Units of time depend on the time units chosen for model parameters. The simulation runs 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 a list, with the time series of the simulated disease as list element ts, with one column per compartment/variable. The first column is time.

Value

The function returns a list. The list has one element, a data frame ts which contains the time series of the simulated model, 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 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. See the manual for the adaptivetau package for details on the stochastic algorithm.

Examples

```
# To run the simulation with default parameters, just call the function:
result <- simulate_sirdemographic_stochastic()
# To choose parameter values other than the standard one, specify them like this:
result <- simulate_sirdemographic_stochastic(S = 2000,  tmax = 200, b = 0.01)
# You can display or further process the result, like this:
plot(result$ts[,'time'],result$ts[,'S'],xlab='Time',ylab='Number Susceptible',type='l')
print(paste('Max number of infected: ',max(result$ts[,'I'])))
```

---

simulate_sir_discrete *Basic discrete time SIR model*

Description

A basic SIR model with 3 compartments and infection and recovery processes

Usage

```
simulate_sir_discrete(S = 1000, I = 1, R = 0, b = 0.002, g = 1,
  tstart = 0, tfinal = 100, dt = 0.1)
```

Arguments

S	: starting value for Susceptible : numeric
I	: starting value for Infected : numeric
R	: starting value for Recovered : numeric
b	: infection rate : numeric
g	: recovery rate : numeric

tstart	: Start time of simulation : numeric
tfinal	: Final time of simulation : numeric
dt	: Time step : numeric

Details

The model includes susceptible, infected, and recovered compartments. The two processes that are modeled are infection and recovery. The model is implemented as a discrete-time simulation.

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_sir_discrete()
```

---

simulate_sir_ode *Basic SIR model*

Description

A basic SIR model with 3 compartments and infection and recovery processes

Usage

```
simulate_sir_ode(S = 1000, I = 1, R = 0, b = 0.002, g = 1,
  tstart = 0, tfinal = 100, dt = 0.1)
```

Arguments

S	: starting value for Susceptible : numeric
I	: starting value for Infected : numeric
R	: starting value for Recovered : numeric
b	: infection rate : numeric
g	: recovery rate : numeric
tstart	: Start time of simulation : numeric
tfinal	: Final time of simulation : numeric
dt	: Time step : numeric

Details

The model includes susceptible, infected, and recovered compartments. The two processes that are modeled are infection and recovery.

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_sir_ode()
```

---

simulate_usanalysis_sir

Simulation to illustrate uncertainty and sensitivity analysis

---

Description

This function performs uncertainty and sensitivity analysis using the SIR model.

Usage

```
simulate_usanalysis_sir(Smin = 1000, Smax = 1500, Imin = 1,
  Imax = 10, bmin = 1e-04, bmax = 0.01, gmean = 1, gvar = 0.1,
  mmin = 0, mmax = 10, nmin = 0, nmax = 0.1, samples = 10,
  rngseed = 100, tstart = 0, tfinal = 200, dt = 0.1)
```

Arguments

`Smin`	: lower bound for initial susceptible : numeric
`Smax`	: upper bound for initial susceptible : numeric
`Imin`	: lower bound for initial infected : numeric
`Imax`	: upper bound for initial infected : numeric
`bmin`	: lower bound for infection rate : numeric
`bmax`	: upper bound for infection rate : numeric
`gmean`	: mean for recovery rate : numeric

`gvar` : variance for recovery rate : numeric

`mmin` : lower bound for birth rate : numeric

`mmax` : upper bound for birth rate : numeric

`nmin` : lower bound for death rate : numeric

`nmax` : upper bound for death 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

The SIR model with demographics 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 model for each sample. Distribution for all parameters is assumed to be uniform between the min and max values. The only exception is the recovery parameter, which (for illustrative purposes) is assumed to be gamma distributed with the specified mean and variance. This code is part of the DSAIDE R package. For additional model details, see the corresponding app in the DSAIDE 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 I and final for S. 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_sir()
# To choose parameter values other than the standard one, specify them, like such:
result <- simulate_usanalysis_sir(gmean = 2, gvar = 0.2, samples = 5, tfinal = 50)
# You should then use the simulation result returned from the function, like this:
plot(result$dat[,"g"],result$dat[,"Ipeak"],xlab='values for g',ylab='Peak Bacteria',type='l')
```

`simulate_vectortransmission_ode` *Vector transmission model*

Description

A basic model with several compartments to model vector-borne transmission

Usage

```
simulate_vectortransmission_ode(Sh = 1000, Ih = 1, Rh = 0,
  Sv = 1000, Iv = 1, b1 = 0.002, b2 = 0.002, g = 1, w = 0.1,
  m = 100, n = 0.1, tstart = 0, tfinal = 100, dt = 0.1)
```

Arguments

`Sh`	: starting value for Susceptible hosts : numeric
`Ih`	: starting value for Infected hosts : numeric
`Rh`	: starting value for Recovered hosts : numeric
`Sv`	: starting value for Susceptible Vectors : numeric
`Iv`	: starting value for Infected Vectors : numeric
`b1`	: infection rate of hosts : numeric
`b2`	: infection rate of vectors : numeric
`g`	: recovery rate of hosts : numeric
`w`	: wanning immunity rate : numeric
`m`	: vector birth rate : numeric
`n`	: vector death rate : numeric
`tstart`	: Start time of simulation : numeric
`tfinal`	: Final time of simulation : numeric
`dt`	: Time step : numeric

Details

The model tracks the dynamics of susceptible, infected, and recovered hosts, and susceptible and infected vectors. Infection, recovery, and waning immunity processes are implemented for hosts. Births and deaths and infection processes are implemented for vectors. This code is based on a dynamical systems model created by the modelbuilder package. The model is implemented here as a set of ordinary differential equations, using the deSolve 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_vectortransmission_ode()
```

Index