Package ‘DSAIDE’

February 2, 2021

Type Package
Title Dynamical Systems Approach to Infectious Disease Epidemiology (Ecology/Evolution)
Description Exploration of simulation models (apps) of various infectious disease transmission dynamics scenarios. The purpose of the package is to help individuals learn about infectious disease epidemiology (ecology/evolution) from a dynamical systems perspective. All apps include explanations of the underlying models and instructions on what to do with the models.
Version 0.9.0
Date 2021-02-01
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, rlang, stats, tidyr, utils (>= 3.4), XML (>= 3.98)
Depends R (>= 3.5), shiny (>= 1.2)
Suggests covr, devtools, knitr, pkgdown, rmarkdown, roxygen2, testthat
RoxygenNote 7.1.1
VignetteBuilder knitr
BugReports https://github.com/ahgroup/DSAIDE/issues
NeedsCompilation no
Author Andreas Handel [aut, cre] (<https://orcid.org/0000-0002-4622-1146>), Cody Dailey [ctb], Isaac Chun-Hai Fung [ctb], Spencer Hall [ctb],
Ben Listyg [ctb],
Brian McKay [ctb],
John Rossow [ctb],
Sina Solaimanpour [ctb],
Alexis Vittengl [ctb],
Henok Woldu [ctb]

Repository CRAN
Date/Publication 2021-02-02 05:20:02 UTC

R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSAIDE</td>
<td>3</td>
</tr>
<tr>
<td>dsaidemenu</td>
<td>3</td>
</tr>
<tr>
<td>flu1918data</td>
<td>4</td>
</tr>
<tr>
<td>generate_documentation</td>
<td>4</td>
</tr>
<tr>
<td>generate_fctcall</td>
<td>5</td>
</tr>
<tr>
<td>generate_ggplot</td>
<td>6</td>
</tr>
<tr>
<td>generate_plotly</td>
<td>7</td>
</tr>
<tr>
<td>generate_shinyinput</td>
<td>8</td>
</tr>
<tr>
<td>generate_text</td>
<td>9</td>
</tr>
<tr>
<td>norodata</td>
<td>10</td>
</tr>
<tr>
<td>run_model</td>
<td>11</td>
</tr>
<tr>
<td>simulate_Characteristics_of_ID_ode</td>
<td>12</td>
</tr>
<tr>
<td>simulate_Complex_ID_Control_ode</td>
<td>14</td>
</tr>
<tr>
<td>simulate_directtransmission_ode</td>
<td>17</td>
</tr>
<tr>
<td>simulate_Drug_Resistance_Evolution_stochastic</td>
<td>19</td>
</tr>
<tr>
<td>simulate_Environmental_Transmission_model_ode</td>
<td>21</td>
</tr>
<tr>
<td>simulate_flu_fit</td>
<td>23</td>
</tr>
<tr>
<td>simulate_Host_Heterogeneity_Model_ode</td>
<td>25</td>
</tr>
<tr>
<td>simulate_idcontrolmultigroup_ode</td>
<td>27</td>
</tr>
<tr>
<td>simulate_idcontrolmultioutbreak_ode</td>
<td>30</td>
</tr>
<tr>
<td>simulate_idpatterns_ode</td>
<td>32</td>
</tr>
<tr>
<td>simulate_idvaccine_ode</td>
<td>34</td>
</tr>
<tr>
<td>simulate_multipathogen_ode</td>
<td>36</td>
</tr>
<tr>
<td>simulate_noro_fit</td>
<td>37</td>
</tr>
<tr>
<td>simulate_SEIRsd_model_stochastic</td>
<td>39</td>
</tr>
<tr>
<td>simulate_SIRsd_model_ode</td>
<td>41</td>
</tr>
<tr>
<td>simulate_SIRsd_model_stochastic</td>
<td>43</td>
</tr>
<tr>
<td>simulate_SIR_modelexploration</td>
<td>45</td>
</tr>
<tr>
<td>simulate_SIR_model_discrete</td>
<td>47</td>
</tr>
<tr>
<td>simulate_SIR_model_ode</td>
<td>49</td>
</tr>
<tr>
<td>simulate_SIR_usanalysis</td>
<td>50</td>
</tr>
<tr>
<td>simulate_Vector_transmission_model_ode</td>
<td>52</td>
</tr>
</tbody>
</table>

Index 55
**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](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**

```r
dsaidemenu()
```

**Details**

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

**Author(s)**

Andreas Handel

**Examples**

```r
## 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/tibble with these variables:

Date  Week of reporting, date format
Deaths  New deaths in NYC per week per 100,000, numeric

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 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
**generate_fctcall**

**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

---

**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: 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
generate_plotly

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 ggplot plot structure for display in a Shiny UI.

Author(s)
Andreas Handel

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.
generate_shinyinput

optional: xscale,yscale - Scaling of axes, valid ggplot2 expression, e.g. "identity" or "log10".
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**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>use_mbmodel</td>
<td>TRUE/FALSE if mbmodel list should be used to generate UI</td>
</tr>
<tr>
<td>mbmodel</td>
<td>a valid mbmodel object</td>
</tr>
<tr>
<td>use_doc</td>
<td>TRUE/FALSE if doc of a model file should be parsed to make UI</td>
</tr>
<tr>
<td>model_file</td>
<td>name/path to function file for parsing doc</td>
</tr>
<tr>
<td>model_function</td>
<td>name of function who’s formals are parsed to make UI</td>
</tr>
<tr>
<td>otherinputs</td>
<td>a text string that specifies a list of other shiny inputs to include in the UI</td>
</tr>
<tr>
<td>packagename</td>
<td>name of package using this function</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>res</td>
<td>A list structure containing all simulation results that are to be processed. This function is meant to be used together with <code>generate_plots()</code> and requires similar input information. See the <code>generate_plots()</code> 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</td>
</tr>
</tbody>
</table>
norodata

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
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, numeric
- **Cases**: New cases for the specified date, numeric

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$simfunction - name of simulation function(s) as string.
- modelsettings$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
- 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 number 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_Characteristics_of_ID_ode**

*Characteristics of ID*

**Description**

A compartmental model with several different compartments: Susceptibles (S), Infected and Presymptomatic (P), Infected and Asymptomatic (A), Infected and Symptomatic (I), Recovered and Immune (R) and Dead (D)

**Usage**

```r
simulate_Characteristics_of_ID_ode(
  S = 1000,
  P = 1,
  A = 0,
  I = 0,
  R = 0,
  D = 0,
  bP = 0,
  bA = 0,
  bI = 0.001,
  gP = 0.1,
  gA = 0.1,
  gI = 0.1,
  f = 0,
  d = 0,
  tstart = 0,
  tfinal = 200,
  dt = 0.1
)
```

**Arguments**

- **S**: starting value for Susceptible : numeric
- **P**: starting value for Presymptomatic : numeric
- **A**: starting value for Asymptomatic : numeric
- **I**: starting value for Symptomatic : numeric
- **R**: starting value for Recovered : numeric
- **D**: starting value for Dead : numeric Values for model parameters : numeric
- **bP**: rate of transmission from P to S : numeric
- **bA**: rate of transmission from A to S : numeric
- **bI**: rate of transmission from I to S : numeric
- **gP**: rate at which a person leaves the P compartment : numeric
simulate_Characteristics_of_ID_ode

\[ g_A \]: rate at which a person leaves the A compartment : numeric
\[ g_I \]: rate at which a person leaves the I compartment : numeric
\[ f \]: fraction of asymptomatic infections : numeric
\[ d \]: fraction of symptomatic hosts that die : numeric
Values for model times :
\[ t_{\text{start}} \]: Start time of simulation : numeric
\[ t_{\text{final}} \]: Final time of simulation : numeric
\[ dt \]: Time step : numeric

**Details**

The model tracks the dynamics of susceptible, presymptomatic, asymptomatic, symptomatic, recovered, and dead individuals. 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. This model is part of the DSAIDE R package, more information can be found there.

This code is was generated by the modelbuilder package R. 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.

**Starting conditions for model variables**

**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, Alexis Vittengl

**Model creation date**

2020-09-24

**Code Author**

generated by the modelbuilder R package
Examples

# To run the simulation with default parameters:
result <- simulate_Characteristics_of_ID_ode()
# To choose values other than the standard one, specify them like this:
result <- simulate_Characteristics_of_ID_ode(S = 2000,P = 2,A = 0,I = 0,R = 0,D = 0)
# You can display or further process the result, like this:
plot(result$ts[, "time"], result$ts[, "S"], xlab = "Time", ylab = "Numbers", type = "l")
print(paste("Max number of S: ", max(result$ts[, "S"])))

simulate_Complex_ID_Control_ode

Complex ID Control

Description

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.

Usage

simulate_Complex_ID_Control_ode(
  S = 1000,
  P = 1,
  A = 1,
  I = 1,
  R = 0,
  D = 0,
  E = 1,
  Sv = 1000,
  Iv = 1,
  nH = 0.01,
  mH = 0.001,
  nV = 0.01,
  mV = 0.001,
  bP = 0.02,
  bA = 0.02,
  bI = 0.02,
  bE = 0.02,
  bV = 0.02,
  bH = 0.02,
simulate_Complex_ID_Control_ode

gP = 0.7,
gA = 0.1,
gI = 0.1,
pI = 0.04,
pA = 0.03,
c = 0.7,
f = 0.08,
d = 0.01,
w = 0.003,
tstart = 0,
tfinal = 100,
dt = 0.1

Arguments

S : starting value for Susceptible : numeric
P : starting value for Presymptomatic : numeric
A : starting value for Asymptomatic : numeric
I : starting value for Symptomatic : numeric
R : starting value for Recovered : numeric
D : starting value for Dead : numeric
E : starting value for Pathogen in Environment : numeric
Sv : starting value for Susceptible Vector : numeric
Iv : starting value for Infected Vector : numeric

nH : Birth rate of susceptible hosts : numeric
mH : Death rate of hosts : numeric
nV : Birth rate of susceptible vectors : numeric
mV : Death rate of vectors : numeric
bP : rate of transmission from pre-symptomatic hosts : numeric
bA : rate of transmission from asymptomatic hosts : numeric
bI : rate of transmission from symptomatic hosts : numeric
bE : rate of transmission from environment : numeric
bV : rate of transmission from vectors to hosts : numeric
bH : rate of transmission to vectors from hosts : numeric
gP : rate of movement out of P compartment : numeric
gA : rate of movement out of A compartment : numeric
gI : rate of movement out of I compartment : numeric
pI : rate of pathogen shedding into environment by symptomatic hosts : numeric
pA : rate of pathogen shedding into environment by asymptomatic hosts : numeric
simulate_Complex_ID_Control_ode

c
: rate of pathogen decay in environment: numeric

f
: fraction of presymptomatic that move to asymptomatic: numeric

d
: fraction of symptomatic infected hosts that die due to disease: numeric

w
: rate of immunity waning: numeric Values for model times: numeric

tstart
: Start time of simulation: numeric

tfinal
: Final time of simulation: numeric

dt
: Time step: numeric

Details

Susceptible individuals (S) can become infected by pre-symptomatic (P), asymptomatic (A) or symptomatic (I) hosts. The rates at which infections from the different types of infected individuals (P, A, and I) occur are governed by 3 parameters, \( b_P \), \( b_A \), and \( b_I \). Susceptible individuals (S) can also become infected by contact with the environment or infected vectors, at rates \( b_E \) and \( b_v \). Susceptible vectors (S\textsubscript{v}) can become infected by contact with symptomatic hosts at rate \( b_h \). All infected hosts first enter the presymptomatic stage. They remain there for some time (determined by rate \( g_P \), the inverse of which is the average time spent in the presymptomatic stage). A fraction \( f \) of presymptomatic hosts move into the asymptomatic category, and the rest become symptomatic infected hosts. Asymptomatic infected hosts recover after some time (specified by the rate \( g_A \)). Similarly, the rate \( g_I \) determines the duration the symptomatic hosts stay in the symptomatic state. For symptomatic hosts, two outcomes are possible. Either recovery or death. The parameter \( d \) determines the fraction of hosts that die due to disease. Recovered individuals are initially immune to reinfection. They can lose their immunity at rate \( w \) and return to the susceptible compartment. Symptomatic and asymptomatic hosts shed pathogen into the environment at rates \( p_A \) and \( p_I \). The pathogen in the environment decays at rate \( c \). New susceptible hosts and vectors enter the system (are born) at rates \( n_h \) and \( n_v \). Mortality (death unrelated to disease) for hosts and vectors occurs at rates \( m_h \) and \( m_v \).

This code is was generated by the modelbuilder package R. 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.

Starting conditions for model variables

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, Alexis Vittengl
Model creation date
2020-09-24

Code Author
generated by the modelbuilder R package

Code creation date
2020-09-24

Examples

# To run the simulation with default parameters:
result <- simulate_Complex_ID_Control_ode()
# To choose values other than the standard one, specify them like this:
result <- simulate_Complex_ID_Control_ode(S = 2000,P = 2,A = 2,I = 2,R = 0,D = 0,E = 2)
# You can display or further process the result, like this:
plot(result$ts[,"time"],result$ts[,"S"],xlab='Time',ylab='Numbers',type='l')
print(paste('Max number of S: ',max(result$ts[,"S"])))

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 = 999,
  I = 1,
  bd = 0.005,
  bf = 0,
  A = 2,
  n = 0,
  m = 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
- **n**: the rate of births : numeric
- **m**: 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 elements, 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')


describe_Drug_Resistance_Evolution_stochastic

Drug Resistance Evolution

Description

An SIR-type model that includes drug treatment and resistance.

Usage

simulate_Drug_Resistance_Evolution_stochastic(
  S = 1000,
  Iu = 1,
  It = 1,
  Ir = 1,
  R = 0,
  bu = 0.002,
  bt = 0.002,
  br = 0.002,
  gu = 1,
  gt = 1,
  gr = 1,
  f = 0,
  cu = 0,
  ct = 0,
  tstart = 0,
  tfinal = 100,
  dt = 0.1,
  rngseed = 123
)

Arguments

S : starting value for Susceptible : numeric
Iu : starting value for Infected Untreated : numeric
It : starting value for Infected Treated : numeric
Ir : starting value for Infected Resistant : numeric
R : starting value for Recovered : numeric Values for model parameters : numeric
simulate_Drug_Resistance_Evolution_stochastic

bu : untreated infection rate : numeric
bt : treated infection rate : numeric
br : resistant infection rate : numeric
gu : untreated recovery rate : numeric
gt : treated recovery rate : numeric
gr : resistant recovery rate : numeric
f : fraction treated : numeric
cu : resistance emergence untreated : numeric
ct : resistance emergence treated : numeric
tstart : Start time of simulation : numeric
tfinal : Final time of simulation : numeric
dt : Time step : numeric
rngseed : set random number seed for reproducibility : numeric

Details

The model includes susceptible, infected untreated, treated and resistant, and recovered compartments. The processes which are modeled are infection, treatment, resistance generation and recovery.

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: adaptivetau

Starting conditions for model variables

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

2020-10-05
Simulate Environmental Transmission model

**Code Author**

Generated by the modelbuilder R package

**Code creation date**

2020-10-05

**Examples**

```r
# To run the simulation with default parameters:
result <- simulate_Drug_Resistance_Evolution_stochastic()
# To choose values other than the standard one, specify them like this:
result <- simulate_Drug_Resistance_Evolution_stochastic(S = 2000, Iu = 2, It = 2, Ir = 2, R = 0)
# You can display or further process the result, like this:
plot(result$ts[, "time"], result$ts[, "S"], xlab = "Time", ylab = "Numbers", type = "l")
print(paste("Max number of S: ", max(result$ts[, "S"])))
```

simulate_Environmental_Transmission_model_ode

**Environmental Transmission model**

**Description**

An SIR model including environmental transmission

**Usage**

```r
simulate_Environmental_Transmission_model_ode(
  S = 1000,
  I = 1,
  R = 0,
  P = 0,
  bI = 0.004,
  bP = 0,
  n = 0,
  m = 0,
  g = 2,
  q = 0,
  c = 0,
  tstart = 0,
  tfinal = 60,
  dt = 0.1
)
```
**Arguments**

- **S**: starting value for Susceptible : numeric
- **I**: starting value for Infected : numeric
- **R**: starting value for Recovered : numeric
- **P**: starting value for Pathogen in environment : numeric
  Values for model parameters : numeric
- **bI**: direct transmission rate : numeric
- **bP**: environmental transmission rate : numeric
- **n**: birth rate : numeric
- **m**: natural death rate : numeric
- **g**: recovery rate : numeric
- **q**: rate at which infected hosts shed pathogen into the environment : numeric
- **c**: rate at which pathogen in the environment decays : numeric
  Values for model times : 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 pathogen compartments. Infection can occur through direct contact with infected or through contact with pathogen in the environment. Infected individuals shed into the environment, pathogen decays there.

This code is was generated by the modelbuilder package R. 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.

Starting conditions for model variables

**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
simulate_flu_fit

Model creation date
2020-09-01

Code Author
generated by the modelbuilder R package

Code creation date
2020-09-01

Examples

# To run the simulation with default parameters:
result <- simulate_Environmental_Transmission_model_ode()
# To choose values other than the standard one, specify them like this:
result <- simulate_Environmental_Transmission_model_ode(S = 2000,I = 2,R = 0,P = 0)
# You can display or further process the result, like this:
plot(result$ts[, 'time'], result$ts[, 'S'], xlab='Time', ylab='Numbers', type='l')
print(paste('Max number of S: ', max(result$ts[, 'S'])))

simulate_flu_fit 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 \( R_0 \). For the data, see 'flu1918data'.

Usage

simulate_flu_fit(
  S = 5e+06,
  I = 1,
  D = 0,
  b = 1e-06,
  blow = 1e-08,
  bhigh = 1e-04,
  g = 1,
  glow = 0.01,
  ghigh = 100,
  f = 0.01,
  flow = 1e-04,
  fhigh = 1,
  usesimdata = 0,
  bsim = 1e-06,
  gsim = 1,
simulate_flu_fit

fsim = 0.01,
noise = 0,
iter = 1,
solvertype = 1,
logfit = 0,
rngseed = 100
)

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
g : recovery rate : numeric
glow : lower bound for g : numeric
ghigh : upper bound for g : numeric
f : fraction dying : numeric
flow : lower bound for f : numeric
fhigh : upper bound for f : numeric
usesimdata : set to 1 if simulated data should be fitted, 0 otherwise : numeric
bsim : infection rate for simulated data : numeric
gsim : recovery rate for simulated data : 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
logfit : set to 1 if the log of the data should be fitted, 0 otherwise : numeric
rngseed : random number seed for reproducibility : 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_flu_fit()
# To apply different settings, provide them to the simulator function, like such:
result <- simulate_flu_fit(iter = 5, logfit = 1, solvertype = 2, usesimdata = 1)

```r
simulate_Host_Heterogeneity_Model_ode
```

Description
An SIR type model stratified for two different types of hosts.

Usage

```r
simulate_Host_Heterogeneity_Model_ode(
  S1 = 1000,
  I1 = 1,
  R1 = 0,
  S2 = 200,
  I2 = 1,
  R2 = 0,
  b11 = 0.01,
  b12 = 0,
  b21 = 0,
  b22 = 0,
  g1 = 1,
  g2 = 1,
)```
simulate_Host_Heterogeneity_Model_ode

w1 = 0,
w2 = 0,
tstart = 0,
tfinal = 60,
dt = 0.1
)

Arguments

S1 : starting value for Susceptible type 1 hosts : numeric
I1 : starting value for Infected type 1 hosts : numeric
R1 : starting value for Recovered type 1 hosts : numeric
S2 : starting value for Susceptible type 2 hosts : numeric
I2 : starting value for Infected type 2 hosts : numeric
R2 : starting value for Recovered type 2 hosts : numeric
Values for model parameters
b11 : rate of transmission to susceptible type 1 host from infected type 1 host : numeric
b12 : rate of transmission to susceptible type 1 host from infected type 2 host : numeric
b21 : rate of transmission to susceptible type 2 host from infected type 1 host : numeric
b22 : rate of transmission to susceptible type 2 host from infected 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
Values for model times

Details

This model tracks susceptibles, infected and recovered of 2 different types. Think of those types as e.g. males/females, children/adults, etc. The model includes infection, recovery and waning immunity processes for both hosts.

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.

Starting conditions for model variables
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, Alexis Vittengl

Model creation date

2020-09-24

Code Author

generated by the modelbuilder R package

Code creation date

2020-09-24

Examples

# To run the simulation with default parameters:
result <- simulate_Host_Heterogeneity_Model_ode()
# To choose values other than the standard one, specify them like this:
result <- simulate_Host_Heterogeneity_Model_ode(S1 = 2000,I1 = 2,R1 = 0,S2 = 400,I2 = 2,R2 = 0)
# You can display or further process the result, like this:
plot(result$ts[,"time"],result$ts[,"S1"],xlab='Time',ylab='Numbers',type='l')
print(paste("Max number of S1: ",max(result$ts[,"S1"])))

simulate_idcontrolmultigroup_ode

Simulation of a compartmental infectious disease transmission model with 3 types of hosts and intervention

Description

This model allows for the simulation of an ID with 3 types of hosts. Groups are assumed to be children, adults and elderly. Intervention can be applied to any of the groups for a certain duration.
Usage

simulate_idcontrolmultigroup_ode(
    Sc = 1000,
    Ic = 0,
    Sa = 1000,
    Ia = 1,
    Se = 1000,
    Ie = 0,
    bcc = 3e-04,
    bca = 1e-04,
    bce = 1e-04,
    bac = 1e-04,
    baa = 3e-04,
    bae = 1e-04,
    bca = 1e-04,
    bea = 1e-04,
    bee = 3e-04,
    gc = 0.1,
    ga = 0.1,
    ge = 0.1,
    wc = 0,
    wa = 0,
    we = 0,
    mc = 0.001,
    ma = 0.01,
    me = 0.1,
    f1 = 0,
    T1_start = 50,
    T1_end = 150,
    f2 = 0,
    T2_start = 50,
    T2_end = 150,
    f3 = 0,
    T3_start = 50,
    T3_end = 150,
    tmax = 600
)

Arguments

S: initial number of susceptible children : numeric
Ic: initial number of infected children : numeric
Sa: initial number of susceptible adults : numeric
Ia: initial number of infected adults : numeric
Se: initial number of susceptible elderly : numeric
Ie: initial number of infected elderly : numeric
bcc: rate of transmission to susceptible child from infected child : numeric
simulate_idcontrolmultigroup_ode

- `bca`: rate of transmission to susceptible child from infected adult: numeric
- `bce`: rate of transmission to susceptible child from infected elderly: numeric
- `bac`: rate of transmission to susceptible adult from infected child: numeric
- `baa`: rate of transmission to susceptible adult from infected adult: numeric
- `bae`: rate of transmission to susceptible adult from infected elderly: numeric
- `bec`: rate of transmission to susceptible elderly from infected child: numeric
- `bea`: rate of transmission to susceptible elderly from infected adult: numeric
- `bee`: rate of transmission to susceptible elderly from infected elderly: numeric
- `gc`: rate at which infected children recover or die: numeric
- `ga`: rate at which infected adults recover or die: numeric
- `ge`: rate at which infected elderly recover or die: numeric
- `wc`: rate at which immunity in children wanes: numeric
- `wa`: rate at which immunity in adults wanes: numeric
- `we`: rate at which immunity in elderly wanes: numeric
- `mc`: fraction of infected children who die: numeric
- `ma`: fraction of infected adults who die: numeric
- `me`: fraction of infected elderly who die: numeric
- `f1`: strength of intervention applied to children, between 0 and 1: numeric
- `T1_start`: start of intervention applied to children: numeric
- `T1_end`: end of intervention applied to children: numeric
- `f2`: strength of intervention applied to adults, between 0 and 1: numeric
- `T2_start`: start of intervention applied to adults: numeric
- `T2_end`: end of intervention applied to adults: numeric
- `f3`: strength of intervention applied to elderly, between 0 and 1: numeric
- `T3_start`: start of intervention applied to elderly: numeric
- `T3_end`: end of intervention applied to elderly: 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 implements basic processes of infection, recovery, and death. Waning immunity is also implemented. Control is applied, which reduces transmission by the indicated proportion, during times tstart and tend. Control can be applied at different levels to the different groups.

**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

Examples

# To run the simulation with default parameters just call the function:
result <- simulate_idcontrolmultioutbreak_ode()

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
simulate_idcontrolmultioutbreak_ode

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_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**

```r
simulate_idpatterns_ode(
  S = 1000,
  P = 1,
  bP = 0,
  bA = 0,
  bI = 0.002,
  s = 0,
  gP = 1,
  gA = 1,
  gI = 1,
  f = 0,
  d = 0,
  w = 0,
  n = 0,
  m = 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
simulate_idpatterns_ode

\[ g_A \text{ : rate at which a person leaves the A compartment : numeric} \]
\[ g_I \text{ : rate at which a person leaves the I compartment : numeric} \]
\[ f \text{ : fraction of pre-symptomatic individuals that have an asymptomatic infection : numeric} \]
\[ d \text{ : fraction of symptomatic infected hosts that die due to disease : numeric} \]
\[ w \text{ : rate at which recovered persons lose immunity and return to susceptible state : numeric} \]
\[ n \text{ : the rate at which new individuals enter the model (are born) : numeric} \]
\[ m \text{ : the rate of natural death (the inverse it the average lifespan) : numeric} \]
\[ \text{timeunit} \text{ : units of time in which the model should run (1=day, 2=week, 3=month, 4=year) : numeric} \]
\[ t_{max} \text{ : maximum simulation time : numeric} \]

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. 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_idvaccine_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_idvaccine_ode(
    S = 1000,
    I = 1,
    f = 0,
    e = 0,
    b = 0.01,
    g = 10,
    n = 0,
    m = 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
simulate_idvaccine_ode

n : the rate at which new individuals enter the model (are born) : numeric
m : 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 contains more details on the model.

Examples

# To run the simulation with default parameters just call the function:
result <- simulate_idvaccine_ode()
# To choose parameter values other than the standard one,
# specify the parameters you want to change, e.g. like such:
result <- simulate_idvaccine_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_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

```r
simulate_multipathogen_ode(
  S = 1000,
  I1 = 1,
  I2 = 0,
  I12 = 0,
  b1 = 0.001,
  b2 = 0,
  b12 = 0,
  g1 = 0.5,
  g2 = 0.5,
  g12 = 0.5,
  a = 0,
  tmax = 100
)
```

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_noro_fit

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

```r
simulate_noro_fit(
    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,
    solverttype = 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
- **solverttype**: 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_noro_fit()
# To apply different settings, provide them to the simulator function, like such:
result <- simulate_noro_fit(iter = 5, fitmodel = 2)

Description

A SEIRS model with 4 compartments

Usage

```r
simulate_SEIRSd_model_stochastic(
  S = 1000,
  E = 1,
  I = 1,
  R = 0,
  bE = 0,
  bI = 0.001,
  gE = 1,
  gI = 1,
  w = 1,
  n = 0,
  m = 0,
)```
simulate_SEIRSd_model_stochastic

tstart = 0,
tfinal = 100,
dt = 0.1,
rngseed = 123
)

Arguments

S : starting value for Susceptible : numeric
E : starting value for Exposed : numeric
I : starting value for Infected and Symptomatic : numeric
R : starting value for Recovered : numeric
bE : infection by exposed : numeric
bI : infection by symptomatic : numeric
gE : progression rate : numeric
gI : recovery rate : numeric
w : waning immunity : numeric
n : births : numeric
m : deaths : numeric
tstart : Start time of simulation : numeric
tfinal : Final time of simulation : numeric
dt : Time step : numeric
rngseed : set random number seed for reproducibility : numeric

Details

The model includes susceptible, exposed/asymptomatic, infected/symptomatic, and recovered compartments. The processes that are modeled are infection, progression to infectiousness, recovery and waning immunity. Demographics through natural births and deaths are also included.

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: adaptivetau

Starting conditions for model variables

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.
simulate_SIRSd_model_ode

Model Author
Andreas Handel

Model creation date
2020-09-28

Code Author
generated by the modelbuilder R package

Code creation date
2020-09-28

Examples

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

# To choose values other than the standard one, specify them like this:
result <- simulate_SEIRSd_model_stochastic(S = 2000,E = 2,I = 2,R = 0)

# You can display or further process the result, like this:
plot(result$ts[,1],result$ts[,2],xlab='Time',ylab='Numbers',type='l')
print(paste('Max number of S: ',max(result$ts[,1])))

simulate_SIRSd_model_ode

SIRSd model

Description
A SIRSd model with 3 compartments. Processes are infection, recovery, births deaths and waning immunity.

Usage

simulate_SIRSd_model_ode(
  S = 1000,
  I = 1,
  R = 0,
  b = 0.002,
  g = 1,
  n = 0,
  m = 0,
  w = 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
- **n**: birth rate : numeric
- **m**: death rate : numeric
- **w**: waning immunity 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 processes which are modeled are infection, recovery, natural births and deaths and waning immunity.

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.

Starting conditions for model variables

**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**

2020-09-01

**Code Author**

generated by the modelbuilder R package
simulate_SIRSd_model_stochastic

Description

A SIRSd model with 3 compartments. Processes are infection, recovery, births deaths and waning immunity.

Usage

simulate_SIRSd_model_stochastic(
  S = 1000,
  I = 1,
  R = 0,
  b = 0.002,
  g = 1,
  w = 1,
  n = 0,
  m = 0,
  tstart = 0,
  tfinal = 100,
  dt = 0.1,
  rngseed = 123
)

Arguments

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

\[ w \] : waning immunity rate : numeric
\[ n \] : birth rate : numeric
\[ m \] : death rate : numeric
\[ t_{\text{start}} \] : Start time of simulation : numeric
\[ t_{\text{final}} \] : Final time of simulation : numeric
\[ dt \] : Time step : numeric
\[ \text{rngseed} \] : set random number seed for reproducibility : numeric

Values for model times

Details

The model includes susceptible, infected, and recovered compartments. The processes which are modeled are infection, recovery, natural births and deaths and waning immunity.

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: adaptivetau

Starting conditions for model variables

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

2020-09-01

Code Author

generated by the modelbuilder R package

Code creation date

2020-09-28
Examples

# To run the simulation with default parameters:
result <- simulate_SIRSd_model_stochastic()
# To choose values other than the standard one, specify them like this:
result <- simulate_SIRSd_model_stochastic(S = 2000, I = 2, R = 0)
# You can display or further process the result, like this:
plot(result$ts[, "time"], result$ts[, "S"], xlab = "Time", ylab = "Numbers", type = "l")
print(paste("Max number of S: ", max(result$ts[, "S"])))

simulate_SIR_modelexploration

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

Description

This function simulates the SIRS 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_SIR_modelexploration(
  S = 1000,
  I = 1,
  R = 0,
  b = 0.002,
  g = 1,
  w = 0,
  n = 0,
  m = 0,
  tstart = 0,
  tfinal = 1000,
  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
simulate_SIR_modelexploration

\[ g \text{ : recovery rate : numeric} \]
\[ w \text{ : rate of waning immunity : numeric} \]
\[ n \text{ : the rate at which new individuals enter the model (are born) : numeric} \]
\[ m \text{ : the rate of natural death (the inverse is the average lifespan) : numeric} \]
\[ t_{\text{start}} \text{ : Start time of simulation : numeric} \]
\[ t_{\text{final}} \text{ : Final time of simulation : numeric} \]
\[ dt \text{ : Times for which result is returned : numeric} \]
\[ \text{samples} \text{ : Number of values to run between p_{\text{min}} and p_{\text{max}} : numeric} \]
\[ \text{par}_{\text{min}} \text{ : Lower value for varied parameter : numeric} \]
\[ \text{par}_{\text{max}} \text{ : Upper value for varied parameter : numeric} \]
\[ \text{samplepar} \text{ : Name of parameter to be varied : character} \]
\[ \text{pardist} \text{ : spacing of parameter values, can be either 'lin' or 'log' : character} \]

**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 $S_{\text{max}}$, $I_{\text{max}}$, $R_{\text{max}}$ and $S_{\text{final}}$, $I_{\text{final}}$, $R_{\text{final}}$. 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_SIR_modelexploration()
# To choose parameter values other than the standard one, specify them, like such:
res <- simulate_SIR_modelexploration(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_SIR_model_discrete

SIR model

Description

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

Usage

simulate_SIR_model_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 Values for model parameters : numeric
b : infection rate : numeric
g : recovery rate : numeric Values for model times : 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.

This code is was generated by the modelbuilder package R. The model is implemented as a set of discrete time equations using a for loop. The following R packages need to be loaded for the function to work: none

Starting conditions for model variables

**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**

2020-09-01

**Code Author**

generated by the modelbuilder R package

**Code creation date**

2020-09-01

**Examples**

```r
# To run the simulation with default parameters:
result <- simulate_SIR_model_discrete()
# To choose values other than the standard one, specify them like this:
result <- simulate_SIR_model_discrete(S = 2000,I = 2,R = 0)
# You can display or further process the result, like this:
plot(result$ts[,"time"],result$ts[,"S"],xlab="Time",ylab="Numbers",type="l")
print(paste("Max number of S: ",max(result$ts[,"S"])))
```
simulate_SIR_model_ode

SIR model

Description

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

Usage

```r
simulate_SIR_model_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`: values for model parameters : numeric
- `g`: infection rate : numeric
- `tstart`: recovery rate : numeric
- `tstart`: Start time of simulation : numeric
- `tfinal`: Values for model times : numeric
- `dt`: 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.

This code is was generated by the modelbuilder package R. 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.

Starting conditions for model variables

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

2020-09-01

Code Author

generated by the modelbuilder R package

Code creation date

2020-09-01

Examples

# To run the simulation with default parameters:
result <- simulate_SIR_model_ode()
# To choose values other than the standard one, specify them like this:
result <- simulate_SIR_model_ode(S = 2000, I = 2, R = 0)
# You can display or further process the result, like this:
plot(result$ts[, 'time'], result$ts[, 'S'], xlab = 'Time', ylab = 'Numbers', type = 'l')
print(paste('Max number of S: ', max(result$ts[, 'S'])))

simulate_SIR_usanalysis

Simulation to illustrate uncertainty and sensitivity analysis

Description

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

Usage

simulate_SIR_usanalysis(
  Smin = 1000,
  Smax = 1000,
  Imin = 10,
  Imax = 10,
  bmin = 0.005,
  bmax = 0.01,
simulate_SIR_usanalysis


gmean = 0.5, 
gvar = 0.01, 
nmin = 0, 
nmax = 0, 
mmin = 0, 
mmax = 0, 
wmin = 0, 
wmax = 0, 
samples = 5, 
rngseed = 100, 
tstart = 0, 
tfinal = 500, 
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
nmin         : lower bound for birth rate : numeric
nmax         : upper bound for birth rate : numeric
mmin         : lower bound for death rate : numeric
mmax         : upper bound for death rate : numeric
wmin         : lower bound for waning immunity rate : numeric
wmax         : upper bound for waning immunity 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 SIRS 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_SIR_usanalysis()
# To choose parameter values other than the standard one, specify them, like such:
result <- simulate_SIR_usanalysis(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_Vector_transmission_model_ode

Vector transmission model

Description

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

Usage

simulate_Vector_transmission_model_ode(
  Sh = 1000,
  Ih = 1,
  Rh = 0,
  Sv = 1000,
  Iv = 1,
  b1 = 0.003,
  b2 = 0.003,
simulate_Vector_transmission_model_ode

```r
  g = 2,
  w = 0,
  n = 200,
  m = 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`: rate at which susceptible hosts are infected by vectors: numeric
- `b2`: rate at which susceptible vectors are infected by hosts: numeric
- `g`: recovery rate of hosts: numeric
- `w`: waning immunity rate: numeric
- `n`: vector birth rate: numeric
- `m`: 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 was generated by the modelbuilder package R. 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.

Starting conditions for model variables

**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

2020-09-01

Code Author

generated by the modelbuilder R package

Code creation date

2020-09-01

Examples

# To run the simulation with default parameters:
result <- simulate_Vector_transmission_model_ode()
# To choose values other than the standard one, specify them like this:
result <- simulate_Vector_transmission_model_ode(Sh = 2000,Ih = 2,Rh = 0,Sv = 2000,Iv = 2)
# You can display or further process the result, like this:
plot(result$ts[,"time"],result$ts[,"Sh"],xlab='Time',ylab='Numbers',type='l')
print(paste('Max number of Sh: ',max(result$ts[,"Sh"])))
Index

* datasets
  - flu1918data, 4
  - norodata, 10
  - DSAIDE, 3
  - dsaidemenu, 3
  - flu1918data, 4

generate_documentation, 4
generate_fctcall, 5
generate_ggplot, 6
generate_plotly, 7
generate_shinyinput, 8
generate_text, 9

norodata, 10

run_model, 11

simulate_Characteristics_of_ID_ode, 12
simulate_Complex_ID_Control_ode, 14
simulate_directtransmission_ode, 17
simulate_Drug_Resistance_Evolution_stochastic, 19
simulate_Environmental_Transmission_model_ode, 21
simulate_flu_fit, 23
simulate_Host_Heterogeneity_Model_ode, 25
simulate_idcontrolmultigroup_ode, 27
simulate_idcontrolmultioutbreak_ode, 30
simulate_idpatterns_ode, 32
simulate_idvaccine_ode, 34
simulate_multipathogen_ode, 36
simulate_noro_fit, 37
simulate_SEIRSd_model_stochastic, 39
simulate_SIR_model_discrete, 47
simulate_SIR_model_ode, 49
simulate_SIR_modelexploration, 45
simulate_SIR_usanalysis, 50
simulate_SIRSD_model_ode, 41
simulate_SIRSD_model_stochastic, 43
simulate_Vector_transmission_model_ode, 52

55