Package 'DSAIDE'

August 7, 2018

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 0.7.0

Date 2018-8-6

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), knitr (>= 1.15), rmarkdown (>= 1.10), stats (>= 3.4), tidyr (>= 0.7), utils (>= 3.4), XML

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

Suggests covr, testthat

RoxygenNote 6.1.0

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>), Isaac Fung [ctb], Spencer Hall [ctb], Ben Listyg [ctb],

Brian McKay [ctb],
John Rossow [ctb],
Sina Solaimanpour [ctb],
Henok Woldu [ctb]

Repository CRAN

Date/Publication 2018-08-06 23:00:02 UTC

R topics documented:

DSAIDE 2
dsaideapps 3
dsaidemenu 3
generate_documentation 4
generate_plots 4
generate_text 5
simulate_directtransmission 6
simulate_environmentaltransmission 8
simulate_evolution 9
simulate_heterogeneity 11
simulate_idcharacteristics 13
simulate_idcontrol 14
simulate_idpatterns 16
simulate_introduction 18
simulate_multipathogen 19
simulate_reproductivenumber 21
simulate_stochastic_SEIR 23
simulate_stochastic_SIR 24
simulate_vectortransmission 25

Index **28**

`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

dsaideapps — *A function that lets you run a specific DSAIDE app without going through the main menu*

Description

This function opens the specified DSAIDE Shiny App

Usage

```
dsaideapps(appname = NULL)
```

Arguments

appname	a string (with quotation marks) indicating the name of the app to run. Leave empty to get a list of all available apps.

Details

Run this function with no arguments to list all apps. Specify the name of an app (with quotation marks) to start that app

Author(s)

Andreas Handel

Examples

```
# To see all available apps, run
dsaideapps()
# To start a specific app, call its name, e.g.
## Not run: dsaideapps('IDDynamicsIntro')
```

dsaidemenu — *The main menu for the DSAIDE package*

Description

This function opens a Shiny App menu that will allow the user to run the different simulation apps

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()
```

`generate_documentation` *A helper function for the UI part of the shiny apps*

Description

This function take the documentation provided as html file it then takes the 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()
```

Details

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

Value

A list of tabs for display in a Shiny UI

Author(s)

Andreas Handel

`generate_plots` *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_plots(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 ggplot2 expression, e.g. "identity" or "log10". optional: xmin, xmax, ymin, ymax - manual min and max for axes. optional: legendtitle - legend title, if NULL/not supplied no legend will be plotted. 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.

Details

This function is called by the Shiny server to produce plots returned to the Shiny UI To create plots run the simulation with default parameters, just call a function, e.g.: result <- simulate_basicbacteria()

Value

A plot structure for display in a Shiny UI

Author(s)

Andreas Handel

---

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 imput 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 or missing, 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

---

simulate_directtransmission

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(S0 = 1000, I0 = 1, tmax = 120,
  scenario = 1, bd = 0.01, bf = 0, A = 1, m = 0, n = 0,
  g = 0.1, w = 0)
```

Arguments

S0	initial number of susceptibles
I0	initial number of infected hosts
tmax	maximum simulation time, units of months
scenario	choice between density dependent (=1) and frequency dependent (=2) transmission scenarios

bd	rate of transmission for density-dependent transmission
bf	rate of transmission for frequency-dependent transmission
A	the size of the area in which the hosts are assumed to reside/interact
m	the rate of births
n	the rate of natural deaths
g	the rate at which infected hosts recover
w	the rate of waning immunity

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()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_directtransmission(S0 = 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` *Simulation of a compartmental infectious disease transmission model illustrating environmental transmission*

Description

This model allows for the simulation of an environmentally transmitted infectious disease

Usage

```
simulate_environmentaltransmission(S0 = 1000, I0 = 1, E0 = 0,
  tmax = 120, bd = 0.01, be = 0, m = 0, n = 0, g = 1, p = 0,
  c = 0)
```

Arguments

`S0`	initial number of susceptible hosts
`I0`	initial number of infected hosts
`E0`	initial number of pathogen in environment
`tmax`	maximum simulation time, units of months
`bd`	rate of direct transmission
`be`	rate of environmental transmission
`m`	rate of births of hosts
`n`	the rate of natural death of hosts
`g`	the rate at which infected hosts recover/die
`p`	the rate at which infected host shed pathogen in the environment
`c`	the rate at which pathogen in the environment decays

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. the book "Modeling Infectious Diseases in Humans and Animals" by Keeling and Rohani for information on models of this type see the documentation for the deSolve package for details on ODE solvers

See Also

The UI of the Shiny app 'EnvironmentalTransmission', 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_environmentaltransmission()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_environmentaltransmission(S0 = 100, E0 = 1e5, tmax = 100)
# 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')
# Consider also a case in which we set the birth rate of hosts at 0.2, and
# the rate at which infected hosts recover or die at 0.8.
result <- simulate_environmentaltransmission(S0 = 1000, E0 = 1e5, m = 0.2, g = 0.8)
plot(result$ts[,"Time"],result$ts[,"S"], xlab="Time", ylab = "Number Susceptible",type="l")
# Additionally, consider a case in which we assume that no new hosts are born.
result <- simulate_environmentaltransmission(S0 = 1000, E0 = 1e5, m = 0)
plot(result$ts[,"Time"], result$ts[,"S"], xlab = "Time", ylab = "Number Susceptible")
```

---

simulate_evolution *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)

Usage

```
simulate_evolution(S0 = 1000, Iu0 = 1, It0 = 1, Ir0 = 1,
  tmax = 100, bu = 1/1000, bt = 1/1000, br = 1/1000, cu = 1/1000,
  ct = 1/100, f = 0, gu = 1, gt = 1, gr = 1, rngseed = 100)
```

Arguments

S0	initial number of susceptible hosts
Iu0	initial number of wild-type infected untreated hosts
It0	initial number of wild-type infected treated hosts
Ir0	initial number of resistant infected hosts
tmax	maximum simulation time, units depend on choice of units for your parameters
bu	level/rate of infectiousness for hosts in the Iu compartment
bt	level/rate of infectiousness for hosts in the It compartment
br	level/rate of infectiousness for hosts in the Ir compartment
cu	fraction of resistant mutant infections that an untreated host produces
ct	fraction of resistant mutant infections that a treated host produces
f	fraction of infected receiving treatment
gu	rate at which a host leaves the Iu compartment, which is the inverse of the average time spent in that compartment
gt	rate at which a person leaves the It compartment
gr	rate at which a person leaves the Ir compartment
rngseed	seed for random number generator to allow reproducibility

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 represent the evolution of the model parameters: the number of susceptibles, the number infected by the drug-sensitive strain and not on treatment, the number infected by the drug-sensitive strain and being treated, the number infected by the drug-resistant strain (treatment does not affect these), and the recovered (and therefore immune) individuals.

Warning

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

Author(s)

Andreas Handel

References

See the manual for the adaptivetau package for details on the algorithm. The implemented model is loosely based on: Handel et al 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()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_evolution(S0 = 2000,  tmax = 200, bt = 1/100)
# 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')
# Consider also a case in which the fraction of resistant mutant infections that an
# untreated host produces is high, at 0.9.
result <- simulate_evolution(S0 = 2000, tmax = 200, bt = 1/100, cu = 0.9)
plot(result$ts[,"Time"], result$ts[,"S"], xlab ="Time", ylab = "Number Susceptible",type ="l")
```

---

simulate_heterogeneity

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(S10 = 1000, I10 = 1, S20 = 1000, I20 = 0,
  tmax = 120, b11 = 0.01, b12 = 0, b21 = 0, b22 = 0, g1 = 1,
  g2 = 1, w1 = 0, w2 = 0)
```

Arguments

S10	initial number of susceptible type 1 hosts
I10	initial number of infected type 1 hosts
S20	initial number of susceptible type 2 hosts
I20	initial number of infected type 2 hosts
tmax	maximum simulation time, units of months
b11	rate of transmission from infected type 1 host to susceptible type 1 host
b12	rate of transmission from infected type 1 host to susceptible type 2 host
b21	rate of transmission from infected type 2 host to susceptible type 1 host
b22	rate of transmission from infected type 2 host to susceptible type 2 host
g1	the rate at which infected type 1 hosts recover

g2	the rate at which infected type 2 hosts recover
w1	the rate at which type 1 host immunity wanes
w2	the rate at which type 2 host immunity wanes

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()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_heterogeneity(S10 = 100, S20 = 1e3,  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')
# Consider also if we want to make the rate of transmission from infected type 1
# host to susceptible type 1 host 0.7, and then plot the rate of infection for
# type 1 hosts.
result <- simulate_heterogeneity(S10 = 100, S20 = 1e3, b11 = 0.7, tmax = 100)
plot(result$ts[,"Time"],result$ts[,"I1"],xlab="Time",ylab="Number Infected 1", type = "l")
# We can do the same for the infection of susceptible type 2 hosts from infected
# type 1 hosts, and plot the type 2 rate of infection.
result <- simulate_heterogeneity(S10 = 100, S20 = 1e3, b12 = 0.7, tmax = 100)
plot(result$ts[,"Time"],result$ts[,"I2"],xlab="Time",ylab="Number Infected 2", type = "l")
```

`simulate_idcharacteristics` *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(S0 = 1000, P0 = 1, tmax = 300, bP = 0,
  bA = 0, bI = 1/1000, gP = 0.5, gA = 0.5, gI = 0.5, f = 0,
  d = 0)
```

Arguments

`S0`	specifies the initial number of susceptible hosts
`P0`	initial number of infected, pre-symptomatic hosts
`tmax`	maximum simulation time, units depend on choice of units for your parameters
`bP`	level/rate of infectiousness for hosts in the P compartment
`bA`	level/rate of infectiousness for hosts in the A compartment
`bI`	level/rate of infectiousness for hosts in the I compartment
`gP`	rate at which a person leaves the P compartment, which is the inverse of the average time spent in that compartment
`gA`	rate at which a person leaves the A compartment
`gI`	rate at which a person leaves the A compartment
`f`	fraction of pre-symptomatic individuals that have an asymptomatic infection
`d`	fraction of symptomatic infected hosts that die due to disease

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 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

Examples

```
# To run the simulation with default parameters just call the function:
result <- simulate_idcharacteristics()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_idcharacteristics(S0 = 2000, P0 = 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_idcontrol *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(S0 = 1000, I0 = 1, E0 = 0, Sv0 = 1000,
  Iv0 = 0, tmax = 300, bP = 0, bA = 0, bI = 1/1000, bE = 0,
  bv = 1/1000, bh = 1/1000, 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)
```

Arguments

S0	initial number of susceptible hosts
I0	initial number of infected and symptomatic hosts
E0	initial amount of pathogen in environment
Sv0	initial number of susceptible vectors
Iv0	initial number of infected vectors
tmax	maximum simulation time, in units of months
bP	rate of transmission from pre-symptomatic to susceptible hosts
bA	rate of transmission from asymptomatic to susceptible hosts
bI	rate of transmission from symptomatic to susceptible hosts
bE	rate of transmission from environment to susceptible hosts
bv	rate of transmission from infected vectors to susceptible hosts
bh	rate of transmission from symptomatic hosts to susceptible vectors
gP	rate at which a person leaves the P compartment
gA	rate at which a person leaves the A compartment
gI	rate at which a person leaves the I compartment
pA	rate of pathogen shedding into environment by asymptomatic hosts
pI	rate of pathogen shedding into environment by symptomatic hosts
c	rate of pathogen decay in environment
f	fraction of pre-symptomatic individuals that have an asymptomatic infection
d	fraction of symptomatic infected hosts that die due to disease
w	rate at which recovered persons lose immunity and return to susceptible state
mh	the rate at which new hosts enter the model (are born)
nh	the rate of natural death of hosts (the inverse it the average lifespan)
mv	the rate at which new vectors enter the model (are born)
nv	the rate of natural death of vectors (the inverse it the average lifespan)

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()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_idcontrol(S0 = 2000, I0 = 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')
# Consider also a case where recoverd persons become susceptible again at a
# rate of 1.5.
result <- simulate_idcontrol(S0 = 2000, I0 = 10, tmax = 100, w = 1.5)
plot(result$ts[,"Time"], result$ts[,"S"],xlab = "Time",ylab = "Number Susceptible",type="l")
# Make death rate of natural hosts 0.5 and rate of pathogen decay 0.1.
result <- simulate_idcontrol(S0 = 2000, I0 = 10, tmax = 100, nh = 0.5, c = 0.1)
plot(result$ts[,"Time"], result$ts[,"S"],xlab = "Time",ylab = "Number Susceptible",type="l")
```

---

simulate_idpatterns *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(S0 = 1000, P0 = 1, timeunit = 1, tmax = 300,
  bP = 0, bA = 0, bI = 1/1000, gP = 0.5, gA = 0.5, gI = 0.5,
  f = 0, d = 0, w = 0, m = 0, n = 0, s = 0)
```

Arguments

S0	specifies the initial number of susceptible hosts
P0	initial number of infected, pre-symptomatic hosts
timeunit	units of time in which the model should run, needs to be one of (1=day,2=week,3=month,4=year)
tmax	maximum simulation time, in units of months
bP	level/rate of infectiousness for hosts in the P compartment
bA	level/rate of infectiousness for hosts in the A compartment
bI	level/rate of infectiousness for hosts in the I compartment
gP	rate at which a person leaves the P compartment, which is the inverse of the average time spent in that compartment
gA	rate at which a person leaves the A compartment
gI	rate at which a person leaves the I compartment
f	fraction of pre-symptomatic individuals that have an asymptomatic infection
d	fraction of symptomatic infected hosts that die due to disease
w	rate at which recovered persons lose immunity and return to susceptible state
m	the rate at which new individuals enter the model (are born)
n	the rate of natural death (the inverse it the average lifespan)
s	strength of seasonal/annual sigmoidal variation of transmission rate

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_idpatterns()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_idpatterns(S0 = 2000, P0 = 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_introduction *Simulation of a basic SIR model illustrating a single infectious disease outbreak*

Description

This function runs a simulation of a basic SIR model using a set of 3 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_introduction(S0 = 1000, I0 = 1, tmax = 300, g = 0.5,
  b = 1/1000)
```

Arguments

S0	initial number of susceptible individuals
I0	initial number of infected hosts
tmax	maximum simulation time, units depend on choice of units for your parameters
g	rate at which a person leaves the infectious compartment, which is the inverse of the average duration of the infectious period
b	level of infectiousness, i.e. rate of transmission of pathogen from infected to susceptible host

Details

: A simple SIR (Susceptible, Infected, Recovered) model is simulated as a set of ordinary differential equations, using an ode solver from the deSolve package. The S, I, and R parameters are the compartments of the model, representing people who are uninfected but susceptible to infection, people who are infected **and** infectious to others, and people who were infected but are now removed from the model (from recovery or death), respectively. There are two processes in the

model. First, a susceptible individual (S) can become infected by an infected individual (I), at rate *b*. This is represented by the individual leaving compartment S and moving to compartment I. Second, an infected individual dies or recovers and moves to the R component, at rate *g*.

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)

Andreas Handel

See Also

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

Examples

```
# To run the simulation with default parameters just call the function:
result <- simulate_introduction()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_introduction(S0 = 2000, I0 = 10, tmax = 100, g = 1, b = 1/100)
# 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')
# Suppose we want to model the average duration of the infectious period as 4 days;
# the inverse of this is 0.25, which is the rate at which the person leaves
# the infectious stage.
result <- simulate_introduction(S0 = 2000, I0 = 10, tmax = 100, g = 0.25)
plot(result$ts[,"Time"],result$ts[,"S"], xlab = "Time", ylab = "Number Susceptible",type="l")
# We could also set the rate of infectiousness very low.
result <- simulate_introduction(S0 = 2000, I0 = 10, tmax = 100, b = 0.0001)
plot(result$ts[,"Time"],result$ts[,"S"], xlab = "Time", ylab = "Number Susceptible",type="l")
```

---

simulate_multipathogen

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(S0 = 1000, I10 = 1, I20 = 0, I120 = 0,
  tmax = 120, b1 = 0.001, b2 = 0, b12 = 0, g1 = 1, g2 = 1,
  g12 = 1, a = 0)
```

Arguments

S0	initial number of susceptible hosts
I10	initial number of hosts infected with type 1
I20	initial number of hosts infected with type 2
I120	initial number of double infected hosts
tmax	maximum simulation time, units of months
b1	rate at which type 1 infected hosts transmit
b2	rate at which type 2 infected hosts transmit
b12	rate at which double infected hosts transmit
g1	the rate at which infected type 1 hosts recover
g2	the rate at which infected type 2 hosts recover
g12	the rate at which double infected hosts recover
a	fraction of type 1 infections produced by double infected hosts

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()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_multipathogen(S0 = 100, I20 = 10,  tmax = 100, a = 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')
# We could set the infected type 1 host recovery rate at a high level, e.g., 1.2, and
# examine the infected 1 curve.
result <- simulate_multipathogen(S0 = 100, I20 = 10, tmax = 100, g1 = 1.2)
plot(result$ts[,"Time"],result$ts[,"I1"], xlab="Time",ylab="Number Infected Type 1",type="l")
# Additionally, consider making type 1 hosts transmit at a high rate.
result <- simulate_multipathogen(S0 = 100, I20 = 10, tmax = 100, b1 = 2.5)
plot(result$ts[,"Time"],result$ts[,"I1"], xlab="Time",ylab="Number Infected Type 1",type="l")
```

---

simulate_reproductivenumber

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_reproductivenumber(S0 = 1000, I0 = 1, f = 0, e = 0,
  tmax = 300, g = 10, b = 0.01, m = 0, n = 0, w = 0)
```

Arguments

S0	initial number of susceptible hosts
I0	initial number of infected hosts
f	fraction of vaccinated individuals. Those individuals are moved from S to R at the beginning of the simulation
e	efficacy of vaccine, given as fraction between 0 and 1
tmax	maximum simulation time, units depend on choice of units for your parameters

g	rate at which a person leaves the I compartment
b	level/rate of infectiousness for hosts in the I compartment
m	the rate at which new individuals enter the model (are born)
n	the rate of natural death (the inverse it the average lifespan)
w	rate at which recovered persons lose immunity and return to susceptible state

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 Shiny app 'ReproductiveNumber', 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_reproductivenumber()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_reproductivenumber(S0 = 2000, I0 = 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')
# We might also want to have infectiousness rate for hosts in the I compartment
# of 0.4.
result <- simulate_reproductivenumber(S0 = 2000, I0 = 10, tmax = 100, b = 0.4)
plot(result$ts[,"Time"],result$ts[,"S"],xlab = "Time", ylab = "Number Susceptible",type="l")
# We could also have infectiousness rate of 0.6 and recovery rate of 0.2.
result <- simulate_reproductivenumber(S0 = 2000, I0 = 10, tmax = 100, b = 0.6, g = 0.2)
plot(result$ts[,"Time"],result$ts[,"S"],xlab = "Time", ylab = "Number Susceptible",type="l")
```

`simulate_stochastic_SEIR` *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_stochastic_SEIR(S0 = 1000, I0 = 10, tmax = 100, bE = 0,
  bI = 1/1000, gE = 0.5, gI = 0.5, w = 0, m = 0, n = 0,
  rngseed = 100)
```

Arguments

`S0`	initial number of susceptible hosts
`I0`	initial number of infected, symptomatic hosts
`tmax`	maximum simulation time, units depend on choice of units for your parameters
`bE`	level/rate of infectiousness for hosts in the E compartment
`bI`	level/rate of infectiousness for hosts in the I compartment
`gE`	rate at which a person leaves the E compartment, which is the inverse of the average time spent in that compartment
`gI`	rate at which a person leaves the I compartment
`w`	rate at which recovered persons lose immunity and return to susceptible state
`m`	the rate at which new individuals enter the model (are born)
`n`	the rate of natural death (the inverse is the average lifespan)
`rngseed`	seed for random number generator to allow reproducibility

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_stochastic_SEIR()
# To choose parameter values other than the standard one, specify them like this:
result <- simulate_stochastic_SEIR(S0 = 2000,  tmax = 200, bE = 1/100)
# 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_stochastic_SIR

Stochastic simulation of an SIR-type model

Description

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

Usage

```
simulate_stochastic_SIR(S0 = 1000, I0 = 10, tmax = 100, b = 1/1000,
  g = 0.5, m = 0, n = 0, rngseed = 100)
```

Arguments

S0	initial number of susceptible hosts
I0	initial number of infected, symptomatic hosts
tmax	maximum simulation time, units depend on choice of units for your parameters
b	level/rate of infectiousness for hosts in the I compartment
g	rate at which a person leaves the I compartment
m	the rate at which new individuals enter the model (are born)
n	the rate of natural death (the inverse is the average lifespan)
rngseed	seed for random number generator to allow reproducibility

Details

A compartmental SIR model is simulated. Initial conditions for the R variable is 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. 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_stochastic_SIR()
# To choose parameter values other than the standard one, specify them like this:
result <- simulate_stochastic_SIR(S0 = 2000,  tmax = 200, b = 1/100)
# 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_vectortransmission

Simulation of a compartmental infectious disease transmission model illustrating vector-borne transmission

---

Description

This model allows for the simulation of a vector-borne infectious disease

Usage

```
simulate_vectortransmission(Sh0 = 1000, Ih0 = 1, Sv0 = 0, Iv0 = 0,
  tmax = 120, b1 = 0.01, b2 = 0, m = 0, n = 0, g = 1, w = 0)
```

Arguments

Sh0	initial number of susceptible hosts
Ih0	initial number of infected hosts
Sv0	initial number of susceptible vectors
Iv0	initial number of infected vectors
tmax	maximum simulation time, units of months
b1	rate of transmission from infected vector to susceptible host
b2	rate of transmission from infected host to susceptible vector
m	the rate of births of vectors
n	the rate of natural death of vectors
g	the rate at which infected hosts recover/die
w	the rate at which host immunity wanes

Details

A compartmental ID model with several states/compartments is simulated as a set of ordinary differential equations. The compartments are Sh, Ih, Rh, and Sv, Iv. 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 the information in the corresponding Shiny app for model details. See the documentation for the deSolve package for details on ODE solvers.

See Also

The UI of the Shiny app 'VectorTransmission', 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_vectortransmission()
# To choose parameter values other than the standard one, specify them like such:
result <- simulate_vectortransmission(Sh0 = 100, Sv0 = 1e5,  tmax = 100)
# 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')
```

Index

DSAIDE, 2
DSAIDE-package (DSAIDE), 2
dsaideapps, 3
dsaidemenu, 3

generate_documentation, 4
generate_plots, 4
generate_text, 5

simulate_directtransmission, 6
simulate_environmentaltransmission, 8
simulate_evolution, 9
simulate_heterogeneity, 11
simulate_idcharacteristics, 13
simulate_idcontrol, 14
simulate_idpatterns, 16
simulate_introduction, 18
simulate_multipathogen, 19
simulate_reproductivenumber, 21
simulate_stochastic_SEIR, 23
simulate_stochastic_SIR, 24
simulate_vectortransmission, 25