

Package ‘amei’

April 17, 2009

Type Package

Title Adaptive Management of Epidemiological Interventions

Version 1.0-1

Date 2009-02-15

Author Daniel Merl <dan@stat.duke.edu>, Leah R. Johnson <leah@statslab.cam.ac.uk>, Robert B. Gramacy <bobby@statslab.cam.ac.uk>, and Marc Mangel <msmangel@ams.ucsc.edu>

Maintainer Robert B. Gramacy <bobby@statslab.cam.ac.uk>

Description This package provides a flexible statistical framework for generating optimal epidemiological interventions that are designed to minimize the total expected cost of an emerging epidemic while simultaneously propagating uncertainty regarding underlying disease parameters through to the decision process via Bayesian posterior inference. The strategies produced through this framework are adaptive: vaccination schedules are iteratively adjusted to reflect the anticipated trajectory of the epidemic given the current population state and updated parameter estimates.

License GPL

LazyLoad yes

Repository CRAN

Date/Publication 2009-04-03 09:41:52

R topics documented:

amei-package	2
epistep	2
getcost	4
getpolicy	5
manage	7
MCepi	10
MCmanage	12
optvac	15

plot.epiman	17
plot.optvac	19
print.Rd	20

Index	22
--------------	-----------

amei-package	<i>Adaptive Management of Epidemiological Interventions</i>
--------------	---

Description

This package provides a flexible statistical framework for generating optimal epidemiological interventions. The aim is to minimize the total expected cost of an emerging epidemic, while simultaneously propagating uncertainty regarding underlying disease parameters through to the decision process via Bayesian posterior inference. The strategies produced through this framework are adaptive: vaccination schedules are iteratively adjusted to reflect the anticipated trajectory of the epidemic given the current population state and updated parameter estimates.

Details

Package:	amei
Type:	Package
Version:	1.0
Date:	2008-10-20
License:	GPL
LazyLoad:	yes

Author(s)

Daniel Merl <dan@stat.duke.edu>, Leah R. Johnson <leah@statslab.cam.ac.uk>, Robert B. Gramacy <bobby@statslab.cam.ac.uk>, and Mark S. Mangel <msmangl@ams.ucsc.edu>

Maintainer: Robert B. Gramacy <bobby@statslab.cam.ac.uk>

References

A statistical framework for the adaptive management of epidemiological interventions (2008). Daniel Merl, Leah R. Johnson, Robert B. Gramacy, and Marc S. Mangel. Duke Working Paper 08-29. <http://ftp.stat.duke.edu/WorkingPapers/08-29.html>

epistep	<i>Evolve One Step of an Epidemic in Time</i>
---------	---

Description

This function takes the current state of an epidemic, described by the values of SIR, and evolves the epidemic by one time step, stochastically, according to the parameterization provided

Usage

```
epistep(SIR, last = NULL, true = list(b = 0.00218, k = 10, nu = 0.4, mu = 0))
```

Arguments

SIR	a list with the current scalar values of the number of susceptibles (S), infecteds (I) and recovereds (R)
last	a (dummy) argument used to pass additional information necessary for the sampling the dynamics of the epidemic; the return-value of a user-defined <code>epistep</code> function would be automatically passed in here by the <code>manage</code> function. This is not required by the simple SIR model implemented here in the default version
true	a list containing scalar entries indicating the <i>true</i> parameters according to which the SIR model evolves stochastically: b , k , ν , and μ representing the transmission probability, clumpiness parameter, the recovery probability, and the mortality probability, respectively

Details

This function is intended to be passed as an argument to the `manage` function, to describe the default evolution of an epidemic under the SIR model. Other, user-defined, functions undergoing different disease dynamics should follow the protocol (i.e., inputs and outputs) prototyped by this function. Similarly, this function may be used as input to `MCmanage` which depends on the `manage` function.

The epidemic described by the default parameterization (given by `true`) is an approximation of an influenza epidemic in a British boarding school described by Murray (see references below).

For more details on the parameterization and simulation of the SIR model, etc., see `vignette("amei")`

Value

`epistep` returns a list containing the scalar integer components listed below indicating the number of individuals which are

rem	newly removed
rec	newly recovered
infect	newly infected
dead	newly dead

Author(s)

Daniel Merl <dan@stat.duke.edu>, Leah R. Johnson <leah@statslab.cam.ac.uk>, Robert B. Gramacy <bobby@statslab.cam.ac.uk>, and Mark S. Mangel <msmangl@ams.ucsc.edu>

References

- A statistical framework for the adaptive management of epidemiological interventions* (2008). Daniel Merl, Leah R. Johnson, Robert B. Gramacy, and Marc S. Mangel. Duke Working Paper 08-29. <http://ftp.stat.duke.edu/WorkingPapers/08-29.html>
- Murray, J. D. (2002) *Mathematical Biology I: An Introduction*. Springer Verlag

See Also

[manage](#), [MCmanage](#)

Examples

```
## parameters to epistep (similar default except mu != 0)
true <- list(b = 0.00218, k = 0.1, nu = 0.4, mu = 0.1)
SIR <- list(S=700, I=200, R=100)

## examine the distribution of the outputs of epistep
T <- 1000
na <- rep(NA, T)
out <- data.frame(rem=na, rec=na, infect=na, dead=na)
for(t in 1:T) {
  out[t,] <- epistep(SIR=SIR, true=true)
}

## make histograms of the output
par(mfrow=c(2,2))
hist(out$rem)
hist(out$rec)
hist(out$infect)
hist(out$dead)
```

getcost

Extract the Final Costs/Number of Vaccinations of the (Managed) Epidemic

Description

Extracts the (summary of the distribution of) final cost(s) or number of vaccinations performed of Monte Carlo epidemic under a static vaccination strategy as implemented by [MCepi](#) or a single adaptively managed epidemic as implemented by [manage](#) or [MCmanage](#)

Usage

```
getcost(obj)
getvac(obj)
```

Arguments

obj a object of class "MCepi", "epiman", or "MCmanage"

Details

`getcost` returns distribution information for the final (cumulative) costs of the (managed) epidemic, and `getvac` returns distribution information for the cumulative number of vaccinations performed.

If `x` is of class "epiman" then the single final cost or number of vaccinations is performed is returned.

For further details of this function please see the documentation for [MCepi](#) and [manage](#)

Value

The return value is a double-precision scalar indicating the median final cost obtained under [manage](#), or a `data.frame` giving the median and quantiles calculated by [MCepi](#) or [MCmanage](#)

Author(s)

Daniel Merl <dan@stat.duke.edu>, Leah R. Johnson <leah@statslab.cam.ac.uk>, Robert B. Gramacy <bobby@statslab.cam.ac.uk>, and Mark S. Mangel <msmangl@ams.ucsc.edu>

References

A statistical framework for the adaptive management of epidemiological interventions (2008). Daniel Merl, Leah R. Johnson, Robert B. Gramacy, and Marc S. Mangel. Duke Working Paper 08-29. <http://ftp.stat.duke.edu/WorkingPapers/08-29.html>

See Also

[MCepi](#), [manage](#), [MCmanage](#)

Examples

```
## for an example of the usage of this function,  
## please see the documentation for MCepi as  
## referenced in the See Also section, above
```

getpolicy

Extract The Optimal Static Vaccination Policy

Description

Extracts the optimal static vaccination policy from an "optvac"-class object

Usage

```
getpolicy(obj, which = c("best", "worst"))
```

Arguments

`obj` and "optvac"-class object
`which` optionally, by supplying `which = "worst"` the worst static vaccination policy can be extracted

Details

This function is designed to work with the output of the `optvac` function, and to provide inputs to the `MCepi` function. It searches the cost grid defined by `vacgrid` for the lowest (or highest) cost, and returns this information

Value

The output is a `data.frame` with scalar entries

<code>row</code>	the row of <code>vacgrid\$stops</code> from the "optvac"-class object corresponding to the lowest (or highest) cost
<code>col</code>	the row of <code>vacgrid\$fracs</code> from the "optvac"-class object corresponding to the lowest (or highest) cost
<code>frac</code>	the actual optimal fraction to vaccinate from the <code>vacgrid</code> table
<code>stop</code>	the actual optimal (stopping) threshold from the <code>vacgrid</code> table
<code>cost</code>	the cost associated with the optimal static vaccination strategy

Author(s)

Daniel Merl <dan@stat.duke.edu>, Leah R. Johnson <leah@statslab.cam.ac.uk>, Robert B. Gramacy <bobby@statslab.cam.ac.uk>, and Mark S. Mangel <msmangl@ams.ucsc.edu>

References

A statistical framework for the adaptive management of epidemiological interventions (2008). Daniel Merl, Leah R. Johnson, Robert B. Gramacy, and Marc S. Mangel. Duke Working Paper 08-29. <http://ftp.stat.duke.edu/WorkingPapers/08-29.html>

See Also

`optvac`, `plot.optvac`, `MCepi`

Examples

```
## for an example of the usage of this function,
## please see the documentation for optvac as
## referenced in the See Also section, above
```

Description

This function adaptively manages an epidemic by learning about (parameters determining) its behavior online and adjusting an optimal vaccination strategy accordingly

Usage

```
manage(init, epistep, vacgrid, costs, T = 40,
       pinit = list(b = 0.1, k = 0.02, nu = 0.2, mu = 0.1),
       hyper = list(bh = c(1,3), kh = c(1,3), nuh = c(1,1), muh = c(1,1)),
       vac0=list(frac=0, stop=0), MCvits = 10, MCMCpits = 1000,
       bkrate = 1, vacsamps = 100, start = 8, ...)
```

Arguments

<code>init</code>	a list containing scalar entries $\$S0$, $\$I0$, $\$R0$, $\$D0$ depicting the initial number of susceptibles, infecteds, recovereds, and dead individuals in the outbreak
<code>epistep</code>	a function which moves the epidemic ahead one time-step; see epistep
<code>T</code>	the maximum number of time steps during which the epidemic is allowed to evolve
<code>vacgrid</code>	a list containing vector entries $\$fracs$ and $\$stops$ indicating the permissible fractions (in $[0,1]$) of the population to be vaccinated and the (positive integer) of stopping thresholds having a maximum of <code>init$\\$S0$</code> ; can be NULL, see details below
<code>costs</code>	a list containing scalar entries $\$vac$, $\$death$ and $\$infect$ depicting the costs associated with a single vaccination or death, or the daily cost of maintaining an infected individual, respectively; can be NULL, but only when <code>vacgrid = NULL</code> also
<code>pinit</code>	a list containing scalar entries $\$b$, $\$k$, $\$nu$, and $\$mu$ depicting the <i>initial values</i> of parameters of the SIR model representing the transmission probability, clumpiness parameter, the recovery probability, and the mortality probability, respectively, which are subsequently sampled by MCMC from the posterior
<code>hyper</code>	a list containing 2-vector entries describing parameters to the prior distribution of the parameters listed in the <code>pinit</code> argument. The prior for b follows a gamma distribution with parameters $\$bh$ where the shape is given by <code>bh[1]</code> and scale by <code>bh[2]</code> . The prior for k specified by parameters $\$kh$ is similar. The prior(s) for ν and μ are specified through p_r and p_d , respectively, which follow Beta distributions and the default specification is uniform. See <code>vignette("amei")</code> for more details
<code>vac0</code>	the initial (static) vaccination policy to be used before estimation of parameters begins (at <code>start</code>). This is a list with scalar entries $\$frac$ and $\$stop$ depicting the fraction to be vaccinated at each time step, and the vaccination

	(stopping) threshold, respectively. The default corresponds to no initial vaccination
<code>MCvits</code>	scalar number of Monte Carlo iterations of forward epidemic evolution used at each time step to determine the optimal vaccination policy
<code>MCMCpits</code>	scalar number of Markov chain Monte Carlo iterations used at each step to estimate the SIR model parameters
<code>bkrate</code>	number of samples of <code>b</code> and <code>k</code> , relative to <code>mu</code> and <code>nu</code> before a sample of all four parameters is saved; this acknowledges that <code>b</code> and <code>k</code> are correlated and thus mix slower than <code>mu</code> and <code>nu</code>
<code>vacsamp</code>	used to thin the MCMC samples of the parameters sampled from the posterior that are used to calculate optimal vaccination policies; this should be an integer scalar such that $0 < \text{vacsamp} \leq \text{MCMCpits}$
<code>start</code>	at what time, after time 1 where the state is given by <code>init</code> , should vaccinations be allowed to start
<code>...</code>	additional arguments passed to a user-defined <code>epistep</code> function

Details

At each time step of the epidemic – evolving stochastically according to the initialization in `init` and progression described by `epistep` – the parameters are inferred by sampling from their posterior distribution conditioned on the available data via MCMC. These samples are fed into the Monte Carlo method for determining the optimal vaccination strategy for (the remainder) of the epidemic. That policy is then enacted, and then time is incremented.

Parameter estimation (alone) can be performed by specifying the “null” vaccination grid `vacgrid = NULL`, i.e., with `vac0`.

For more details on the parameterization and simulation of the SIR model, etc., and the calculation of the optimal vaccination strategy, see `vignette("amei")`

Value

`management` returns an object of class "epiman", which is a list containing the components listed below.

<code>soln</code>	a <code>data.frame</code> describing the evolution of the epidemic giving the following for each time step (in columns): a time index (<code>\$TIME</code>); the total number of susceptibles (<code>\$S</code>), infecteds (<code>\$I</code>), recovereds (<code>\$R</code>), and deads (<code>\$D</code>); the number of people who became infected (<code>\$itilde</code>), recovered (<code>\$rtilde</code>), or died (<code>\$dtilde</code>), in that particular time step; the total number vaccinated (<code>\$V</code>), or culled (<code>\$QC</code> , not supported in this version); and the corresponding cumulative cost of the epidemic (<code>\$C</code> so far)
<code>vacgrid</code>	a copy of the input <code>vacgrid</code>
<code>pols</code>	a <code>data.frame</code> describing the policy enacted at each time step with columns giving the fraction of the population vaccinated (<code>\$frac</code>), and the (stopping) threshold (<code>\$stop</code>)
<code>vactimes</code>	a scalar integer vector indicating the times at which vaccinations actually occurred; this coincides with changes to <code>soln\$V</code>

samp data.frame containing samples from the posterior distribution of the parameters β , k , ν , and μ collected during the final time step

Note

It may be important to plot the epidemic trajectory, with the generic method `plot.epiman`, or inspect the output `$soln[T,]`, to check that the full dynamics of the epidemic have played out in the number of time steps, `T`, allotted

Author(s)

Daniel Merl <dan@stat.duke.edu>, Leah R. Johnson <leah@statslab.cam.ac.uk>, Robert B. Gramacy <bobby@statslab.cam.ac.uk>, and Mark S. Mangel <msmangl@ams.ucsc.edu>

References

A statistical framework for the adaptive management of epidemiological interventions (2008). Daniel Merl, Leah R. Johnson, Robert B. Gramacy, and Marc S. Mangel. Duke Working Paper 08-29. <http://ftp.stat.duke.edu/WorkingPapers/08-29.html>

See Also

[epistep](#), [MCmanage](#), [plot.epiman](#)

Examples

```
## manage an epidemic evolving according to epistep with
## with the following initial population
init <- list(S0=762, I0=1, R0=0, D0=0)

## construct a grid of valid vaccination strategies
## and specify costs
vacgrid <- list(frac=seq(0,1.0,0.1), stops=seq(2,init$S0-75,75))
costs <- list(vac=2, death=4, infect=1)

## adaptively manage the epidemic
out.man <- manage(init, epistep, vacgrid, costs)

## plot the trajectories of SIR and the associated costs
plot(out.man)
plot(out.man, type="cost")
getcost(out.man)

## plot the samples from the posterior distribution of
## the parameters obtained during the last time step
true <- as.list(formals(epistep)$true)
plot(out.man, type="params", true=true)

## Bobby isnt really sure what this is plotting
plot(out.man, type="frac")
plot(out.man, type="stops")
```

Description

This function performs a Monte Carlo simulation of epidemic trajectories under a static vaccination strategy. Statistics are collected for a collection of characteristics including the state of the epidemic (evolution of susceptibles (S), infecteds (I), recovered (R), and deads) and the cost of the vaccination strategy employed

Usage

```
MCepi(init, params, vac, costs, T = 40, MCreps = 1000,
      quant = c(0.025, 0.975), midepi = FALSE, start = 7, ...)
```

Arguments

<code>init</code>	a list containing scalar entries <code>\$S0</code> , <code>\$I0</code> , <code>\$R0</code> , <code>\$D0</code> depicting the initial number of susceptibles, infecteds, recovered, and deads in the outbreak
<code>params</code>	a list containing scalar entries <code>\$b</code> , <code>\$k</code> , <code>\$nu</code> , and <code>\$mu</code> depicting the <i>true</i> parameters of the SIR model representing the transmission probability, clumpiness parameter, the recovery probability, and the mortality probability, respectively; optionally <code>params</code> may be an <code>epistep</code> function
<code>vac</code>	a list containing scalar entries <code>\$frac</code> and <code>\$stop</code> depicting the fraction to be vaccinated at each time step, and the vaccination (stopping) threshold, respectively
<code>costs</code>	a list containing scalar entries <code>\$vac</code> , <code>\$death</code> , and <code>\$infect</code> depicting the costs associated with a single vaccination, death, or daily cost of maintaining an infected individual, respectively
<code>T</code>	the maximum number of time steps during which the epidemic is allowed to evolve
<code>MCreps</code>	the number of times to repeat the Monte Carlo experiment, each time starting with the state in <code>init</code> and collecting characteristics of the epidemic trajectories and (vaccination/death) costs
<code>quant</code>	a 2-vector of quantiles to use in order to capture the spread in the density of characteristics of the epidemic trajectory and costs
<code>midepi</code>	a debugging logical indicating whether to signal that a trajectory is unlikely to be an epidemic
<code>start</code>	at what time, after time 1 where the state is given by <code>init</code> , should vaccinations be allowed to start
<code>...</code>	additional arguments passed to a user-defined <code>epistep</code> function that may be passed in via <code>params</code>

Details

This function simulates many (`MCreps`) trajectories of an epidemic starting out in a particular state (`init`) and evolving according to a particular (`true`) parameterization under a fixed vaccination strategy (`vac`). It returns a summary of characteristics of the state trajectory(s) and the associated costs. The output can be visualized with the generic `plot.MCepi` method and costs can be extracted with `getcost`.

For more details on the parameterization and simulation of the SIR model, etc., and the calculation of the optimal vaccination strategy, please see `vignette("amei")`

Value

`MCepi` returns an object of class "MCepi", which is a `list` containing the components listed below.

<code>Q1</code>	a <code>data.frame</code> containing 6 columns (S, I, R, D, V, C) depicting the first quantile (<code>quant[1]</code>) of the distribution of the evolution of the state of the epidemic (SIRD), the number of vaccinations (V), and the cost (C), at each time point
<code>Mean</code>	same as <code>Q1</code> except the mean rather than a quantile
<code>Q3</code>	same as <code>Q1</code> except the third quantile (<code>quant[2]</code>)

These quantities can be visually inspected using the `plot.MCepi` method

Author(s)

Daniel Merl <dan@stat.duke.edu>, Leah R. Johnson <leah@statslab.cam.ac.uk>, Robert B. Gramacy <bobby@statslab.cam.ac.uk>, and Mark S. Mangel <msmangl@ams.ucsc.edu>

References

A statistical framework for the adaptive management of epidemiological interventions (2008). Daniel Merl, Leah R. Johnson, Robert B. Gramacy, and Marc S. Mangel. Duke Working Paper 08-29. <http://ftp.stat.duke.edu/WorkingPapers/08-29.html>

See Also

`plot.MCepi`, `optvac`

Examples

```
## true epidemic parameters, initial values, and
## vaccination costs
truth <- list(b=0.00218, k=10, nu=0.4, mu=0)
init <- list(S0=762, I0=1, R0=0, D0=0)
costs <- list(vac=2, death=4, infect=1)

## trivial vaccination strategy -- dont vaccinate
vac <- list(frac=0, stop=0)

## simulate the resulting trajectories
```

```

init.MCepi <- MCepi(init, truth, vac, costs)

## plot the distribution of trajectories and costs
## under no vaccination
plot(init.MCepi, main="no vaccination")
plot(init.MCepi, type="costs")

## Now try the optimal strategy.
## See the optvac function for more info
vac.opt <- list(frac=0.7, stop=502)
opt.MCepi <- MCepi(init, truth, vac.opt, costs)

## plot the distribution of trajectories and costs
## under the optimal (static) vaccination
plot(opt.MCepi, main="optimal static vaccination")
plot(opt.MCepi, type="costs")

## show the total number of vaccinations
## median and quantiles
getvac(opt.MCepi)

## compare the median costs of the the initial
## (no vaccination) strategy versus the optimal
## (static) policy
T <- length(opt.MCepi$Median$C)
optC <- getcost(opt.MCepi)
initC <- getcost(init.MCepi)
rbind(initC, optC)

```

MCmanage

Monte Carlo Epidemic Simulation with Adaptive Vaccination

Description

This function performs a Monte Carlo simulation of epidemic trajectories under an adaptive vaccination strategy as implemented by the `manage` function. Statistics are tallied for a collection of characteristics including the state of the epidemic (evolution of susceptibles (S), infecteds (I), recovered (R), and deads) and the cost of the vaccination strategy employed

Usage

```

MCmanage(init, epistep, vacgrid, costs,
  pinit = list(b = 0.1, k = 0.02, nu = 0.2, mu = 0.1),
  hyper = list(bh = c(1,3), kh = c(1,3), nuh = c(1,1), muh = c(1,1)),
  vac0 = list(frac = 0, stop = 0), T = 40, MCreps = 30,
  MCvits = 50, MCMCpits = 1000, bkrate = 1, vacsamps = 50,
  quant = c(0.025, 0.975), start = 7, ...)

```

Arguments

<code>init</code>	a list containing scalar entries $\$S0$, $\$I0$, $\$R0$, $\$D0$ depicting the initial number of susceptibles, infecteds, recovered, and deads in the outbreak
<code>epistep</code>	a function which moves the epidemic ahead one time-step; see epistep
<code>vacgrid</code>	a list containing vector entries $\$fracs$, $\$stops$ indicating the permissible fractions (in $[0,1]$) of the population to be vaccinated and the (positive integer) of stopping thresholds having a maximum of <code>init$\\$S0$</code>
<code>costs</code>	a list containing scalar entries $\$vac$, $\$death$, and $\$infect$, depicting the costs associated with a single vaccination, death, or the daily cost of maintaining an infected individual, respectively
<code>pinit</code>	a list containing scalar entries $\$b$, $\$k$, $\$nu$, and $\$mu$ depicting the <i>initial values</i> of parameters of the SIR model representing the transmission probability, clumpiness parameter, the recovery probability, and the mortality probability, respectively, which are subsequently sampled by MCMC from the posterior
<code>hyper</code>	a list containing 2-vector entries describing parameters to the prior distribution of the parameters listed in the <code>pinit</code> argument. The prior for b follows a gamma distribution with parameters $\$bh$ where the shape is given by $bh[1]$ and scale by $bh[2]$. The prior for k specified by parameters $\$kh$ is similar. The prior(s) for ν and μ are specified through p_r and p_d , respectively, which follow Beta distributions and the default specification is uniform. See <code>vignette("amei")</code> for more details
<code>vac0</code>	the initial (static) vaccination policy to be used before estimation of parameters begins (at <code>start</code>). This is a list with scalar entries $\$frac$ and $\$stop$ depicting the fraction to be vaccinated at each time step, and the vaccination (stopping) threshold, respectively
<code>T</code>	the maximum number of time steps during which the epidemic is allowed to evolve
<code>MCreps</code>	number of times to repeat the Monte Carlo experiment, each time starting with the state in <code>init</code> and collecting characteristics of the epidemic trajectory and (vaccination/death) costs
<code>MCvits</code>	scalar number of Monte Carlo iterations of forward epidemic evolution used at each time step in manage to determine the optimal vaccination policy
<code>MCMCpits</code>	scalar number of Markov chain Monte Carlo iterations used at each step to estimate the SIR model parameters in manage
<code>bkrate</code>	number of samples of b and k , relative to μ and ν before a sample of all four parameters is saved in manage
<code>vacamps</code>	used to thin the MCMC samples of the parameters sampled from the posterior that are used to calculate optimal vaccination policies; this should be an integer scalar such that $0 < vacamps \leq MCMCpits$
<code>quant</code>	a 2-vector of quantiles to use in order to capture the spread in the density of characteristics of the epidemic trajectory and costs
<code>start</code>	at what time, after time 1 where the state is given by <code>init</code> , should vaccinations be allowed to start
<code>...</code>	additional arguments passed to a user-defined epistep function

Details

This function simulates many (MCreps) trajectories of an epidemic starting out in a particular state (`init`) and evolving according the dynamics encoded in `epistep` (or some other user-defined function) under an adaptive vaccination strategy as implemented by `manage`. Many of the arguments to this function are simply passed to `manage`.

It returns a summary of characteristics of the state trajectory(y/ies) and the associated costs. The output can be visualized with the generic `plot.MCepi` method and costs can be extracted with `getcost`.

For more details on the parameterization and simulation of the SIR model, etc., and the calculation of the optimal vaccination strategy, please see `vignette("amei")`

Value

MCmanage returns an object of class "MCepi", which is a list containing the following components.

Q1	a <code>data.frame</code> containing 8 columns (<code>S</code> , <code>I</code> , <code>R</code> , <code>D</code> , <code>V</code> , <code>C</code> , <code>frac</code> , <code>stop</code>) depicting the first quantile (<code>quant[1]</code>) of the distribution of the evolution of the state of the epidemic (SIRD), the number of vaccinations (<code>V</code>), the cost (<code>C</code>), the fraction vaccinated (<code>F</code>), and the stopping threshold (<code>S</code>) at each time point
Mean	same as Q1 except the mean rather than a quantile
Median	same as Median except the median rather than mean
Q3	same as Q1 except the third quantile (<code>quant[2]</code>)

These quantities can be visually inspected using the `plot.MCepi` method

Author(s)

Daniel Merl <dan@stat.duke.edu>, Leah R. Johnson <leah@statslab.cam.ac.uk>, Robert B. Gramacy <bobby@statslab.cam.ac.uk>, and Mark S. Mangel <msmangl@ams.ucsc.edu>

References

A statistical framework for the adaptive management of epidemiological interventions (2008). Daniel Merl, Leah R. Johnson, Robert B. Gramacy, and Marc S. Mangel. Duke Working Paper 08-29. <http://ftp.stat.duke.edu/WorkingPapers/08-29.html>

See Also

`manage`, `MCepi`, `plot.MCepi`

Examples

```
## adaptively manage the epidemic with the following
## initial population
init <- list(S0=762, I0=1, R0=0, D0=0)

## construct a grid of valid vaccination strategies
```

```

## and specify costs
vacgrid <- list(frac=seq(0,1.0,0.1), stops=seq(2,init$S0-75,75))
costs <- list(vac=2, death=4, infect=1)

## run the Monte Carlo management experiment
out.MCmanage <- MCmanage(init, epistep, vacgrid, costs)

## plot the trajectories of SIR and the associated costs
plot(out.MCmanage, main="optimal adaptive vaccination")
plot(out.MCmanage, type="costs")

## extract the distribution of the number of
## cumulative vaccinations via median and quantiles
getvac(out.MCmanage)

## plot the distribution fractions vaccinated and
## stopping times
plot(out.MCmanage, type="frac")
plot(out.MCmanage, type="stops")

## get the final median cost and quantiles --
## these can be compared with the static ones
## calculated by MCepi
getcost(out.MCmanage)

```

optvac

Optimal (Static) Vaccination Policy

Description

A Monte Carlo method is used to calculate the expected costs of a range of static vaccination policies for an epidemic with a known parameterization and initialization

Usage

```

optvac(init, params, vacgrid, costs, T = 40, MCvits = 100,
       midepi = FALSE, start = 7)

```

Arguments

init	a list containing scalar entries \$S0, \$I0, \$R0, \$D0 depicting the initial number of susceptibles, infecteds, recovereds, and deads in the outbreak
params	a list containing scalar entries \$b, \$k, \$nu, and \$mu depicting the <i>true</i> parameters of the SIR model representing the transmission probability, clumpiness parameter, the recovery probability, and the mortality probability, respectively
vacgrid	a list containing vector entries \$frac and \$stops indicating the permissible fractions (in [0,1]) of the population to be vaccinated and the (positive integer) of stopping thresholds having a maximum of init\$S0

<code>costs</code>	a <code>list</code> containing scalar entries <code>\$vac</code> , <code>\$death</code> , and <code>\$infect</code> depicting the costs associated with a single vaccination, death, or the daily cost of maintaining an infected individual, respectively
<code>T</code>	the maximum number of time steps during which the epidemic is allowed to evolve
<code>MCvits</code>	the number of Monte Carlo iterations of forward epidemic evolution used to determine the optimal vaccination policy
<code>midepi</code>	a debugging <code>logical</code> indicating whether to signal that a trajectory is unlikely to be an epidemic
<code>start</code>	at what time, after time 1 where the state is given by <code>init</code> , should vaccinations be allowed to start

Details

This function use a Monte Carlo experiment to calculate the expected costs over a range of vaccination policies specified by permissible fractions of individuals to be vaccinated and stopping thresholds. These policies are constructed by simulating SIR-modeled epidemics that evolve according to `params` starting in the `initial` configuration provided. The output is an object of class "optvac", so the cost grid, or matrix, can be visualized with the `plot.optvac` generic method. The `getpolicy` function can be used to select out the best (and worst) one(s).

For more details on the parameterization and simulation of the SIR model, etc., see `vignette("amei")`

Value

`optvac` returns an object of class "optvac", which is a `list` containing the following components.

<code>vacgrid</code>	a copy of the input <code>vacgrid</code>
<code>C</code>	a matrix of expected costs estimated for each combination of <code>vacgrid\$fracs</code> and <code>vacgrid\$stops</code>

Author(s)

Daniel Merl <dan@stat.duke.edu>, Leah R. Johnson <leah@statslab.cam.ac.uk>, Robert B. Gramacy <bobby@statslab.cam.ac.uk>, and Mark S. Mangel <msmangl@ams.ucsc.edu>

References

A statistical framework for the adaptive management of epidemiological interventions (2008). Daniel Merl, Leah R. Johnson, Robert B. Gramacy, and Marc S. Mangel. Duke Working Paper 08-29. <http://ftp.stat.duke.edu/WorkingPapers/08-29.html>

See Also

[plot.optvac](#), [getpolicy](#), [MCepi](#)

Examples

```
## same inputs as in the MCepi example
truth <- list(b=0.00218, k=10, nu=0.4, mu=0)
init <- list(S0=762, I0=1, R0=0, D0=0)
costs <- list(vac=2, death=4, infect=1)

## construct a grid of valid vaccination strategies
vacgrid <- list(frac=seq(0,1.0,0.1), stops=seq(2,init$S0-50,50))

## calculate the cost surface of all combinations in vacgrid
out.optvac <- optvac(init, truth, vacgrid, costs)

## extract the best and worst (static) policy
best <- getpolicy(out.optvac)
worst <- getpolicy(out.optvac, which="worst")
rbind(best, worst)

## plot the cost surface along with the best and worst policy
plot(out.optvac)

## now return to MCepi for a cost comparison to no vaccination
## using these values
vac.opt <- best[3:4]
vac.opt
```

plot.epiman

Plotting Epidemic Trajectories and Costs

Description

These functions provide a visualization of the evolution of an epidemic, or multiple epidemics obtained via Monte Carlo, and the associated costs of the vaccination strategy employed

Usage

```
## S3 method for class 'epiman':
plot(x, type = c("epi", "costs", "params", "frac",
               "stops"), showd = FALSE, main = NULL, true = NULL, ...)
## S3 method for class 'MCepi':
plot(x, type = c("epi", "costs", "frac", "stops"),
     showd = FALSE, showv = FALSE, main = NULL, ...)
```

Arguments

x	the object to be plotted, either of class "MCepi" or "epiman"
type	indicates the type of plot to be produced, including the epidemic trajectory/ies ("epi", the default), cost(s), estimated distribution(s) of parameters ("params", only in the case of "epiman"-class objects), the fraction vaccinated at each time step ("frac"), and the vaccination (stopping) threshold ("stops")

main	optional title argument for the plot. If not specified, then an automatically generated default is used depending on the type of plot specified
true	this argument only applies to <code>plot.epiman</code> with <code>type = "params"</code> where it should be a list with scalar entries <code>\$b</code> , <code>\$k</code> , <code>\$nu</code> , and <code>\$mu</code> indicating the true parameter entries for the evolution of the epidemic to be added (for comparison) to the posterior density plots
showd	logical indicating if deaths should be shown in the trajectory plots when <code>type = "epi"</code>
showv	this argument only applies to <code>plot.MCepi</code> with <code>type = "epi"</code> where it should be a logical indicating if vaccinations should be shown in the trajectory plot
...	additional arguments passed to <code>plot</code>

Details

The functions documented here support visualization of "MCepi"-class objects which are generated by the `MCepi` and `MCmanage` function, and "epiman"-class objects are generated by the `manage` function. In both cases they enable a visualization of the evolution of the resulting epidemic(s) and costs associated with deaths, vaccinations, etc.

Value

The only output of this function is beautiful plots

Author(s)

Daniel Merl <dan@stat.duke.edu>, Leah R. Johnson <leah@statslab.cam.ac.uk>, Robert B. Gramacy <bobby@statslab.cam.ac.uk>, and Mark S. Mangel <msmangl@ams.ucsc.edu>

References

A statistical framework for the adaptive management of epidemiological interventions (2008). Daniel Merl, Leah R. Johnson, Robert B. Gramacy, and Marc S. Mangel. Duke Working Paper 08-29. <http://ftp.stat.duke.edu/WorkingPapers/08-29.html>

See Also

`MCepi`, `manage`, `MCmanage`

Examples

```
## for examples of the usage of these functions,
## please see the documentation for the functions
## listed in the See Also section, above
```

plot.optvac *Plotting Vaccination Cost Surfaces*

Description

This function provides a visualization of the cost surface return by the `optvac` function and can show the best and worst vaccination strategy

Usage

```
## S3 method for class 'optvac':  
plot(x, main = NULL, ...)
```

Arguments

<code>x</code>	an "optvac"-class object
<code>main</code>	and optional title for the plot; if not specified then a sensible default is generated automatically
<code>...</code>	additional arguments passed to <code>plot</code>

Details

The function documented here supports visualization of "optvac"-class objects which are generated by the `optvac` function

Value

The only output of this function is beautiful plots

Author(s)

Daniel Merl <dan@stat.duke.edu>, Leah R. Johnson <leah@statslab.cam.ac.uk>, Robert B. Gramacy <bobby@statslab.cam.ac.uk>, and Mark S. Mangel <msmangl@ams.ucsc.edu>

References

A statistical framework for the adaptive management of epidemiological interventions (2008). Daniel Merl, Leah R. Johnson, Robert B. Gramacy, and Marc S. Mangel. Duke Working Paper 08-29. <http://ftp.stat.duke.edu/WorkingPapers/08-29.html>

See Also

`optvac`, `getpolicy`

Examples

```
## for an example of the usage of this function,  
## please see the documentation for optvac as  
## referenced in the See Also section, above
```

 print.Rd

 Summarizing and printing amei output

Description

Summarizing, printing, and plotting the contents of a the following objects "MCepi", "optvac", "epiman". In the current version the printing and summary commands are very similar

Usage

```
## S3 method for class 'MCepi':
print(x, ...)
## S3 method for class 'optvac':
print(x, ...)
## S3 method for class 'epiman':
print(x, ...)
## S3 method for class 'MCepi':
summary(object, ...)
## S3 method for class 'optvac':
summary(object, ...)
## S3 method for class 'epiman':
summary(object, ...)
## S3 method for class 'summary.MCepi':
print(x, ...)
## S3 method for class 'summary.optvac':
print(x, ...)
## S3 method for class 'summary.epiman':
print(x, ...)
```

Arguments

object	an object of class "MCepi", "optvac", or "epiman" that must be named object for the generic methods <code>summary</code>
x	an object of class "MCepi", "optvac", or "epiman" that must be named x for generic printing and plotting via <code>print</code>
...	passed to the generic <code>print</code> or <code>summary</code> commands

Details

The printing and summaries provided by these functions are essentially identical except that `summary` returns a `list` that allows the information to be extracted for external use (in code) whereas the printing adds some additional text for human consumption including information about the call

Other ways of extracting information contained in these objects include the functions `getvac`, `getcost`, and `getpolicy`. The plotting functions `plot.MCepi`, `plot.optvac`, and `plot.epiman` are also helpful

The `list(s)` returned by the `summary` command are detailed in the value section below

Value

The `summary` commands documented here return a list containing (a subset of) the items below. The other functions do not return values.

<code>obj</code>	a copy of the input object
<code>final</code>	in the case of "MCepi"-class object this is a <code>data.frame</code> containing a summary of the distribution of the total number of vaccinations and the final cost; for "epiman"-class objects the distributional information is replaced by a scalar
<code>params</code>	in the case of "epiman"-class objects this <code>data.frame</code> containing a summary of the distributional information of SIR model parameters obtained during the final time step of the epidemic
<code>best</code>	the best policy contained in the "optvac"-class object
<code>worst</code>	the worst policy contained in the "optvac"-class object

Author(s)

Daniel Merl <dan@stat.duke.edu>, Leah R. Johnson <leah@statslab.cam.ac.uk>, Robert B. Gramacy <bobby@statslab.cam.ac.uk>, and Mark S. Mangel <msmangl@ams.ucsc.edu>

References

A statistical framework for the adaptive management of epidemiological interventions (2008). Daniel Merl, Leah R. Johnson, Robert B. Gramacy, and Marc S. Mangel. Duke Working Paper 08-29. <http://ftp.stat.duke.edu/WorkingPapers/08-29.html>

See Also

`MCepi`, `optvac`, `manage`, `MCmanage`, `getvac`, `getcost`, `getpolicy`, `plot.MCepi`, `plot.optvac`, `plot.epiman`

Index

*Topic **hplot**

plot.epiman, 17
plot.optvac, 19

*Topic **iteration**

MCepi, 10
MCmanage, 12
optvac, 15

*Topic **methods**

epistep, 2
manage, 7
MCepi, 10
MCmanage, 12
optvac, 15
print.Rd, 20

*Topic **misc**

getcost, 4
getpolicy, 5

*Topic **optimize**

manage, 7
MCmanage, 12
optvac, 15

*Topic **package**

amei-package, 2

amei (amei-package), 2
amei-package, 2

epistep, 2, 7–10, 13, 14

getcost, 4, 11, 14, 20, 21
getpolicy, 5, 16, 19–21
getvac, 20, 21
getvac (getcost), 4

manage, 3–5, 7, 13, 14, 18, 21
MCepi, 4–6, 10, 14, 16, 18, 21
MCmanage, 3–5, 9, 12, 18, 21

optvac, 6, 11, 15, 19, 21

plot, 18, 19

plot.epiman, 9, 17, 20, 21

plot.MCepi, 11, 14, 20, 21

plot.MCepi (plot.epiman), 17

plot.optvac, 6, 16, 19, 20, 21

print, 20

print.epiman (print.Rd), 20

print.MCepi (print.Rd), 20

print.optvac (print.Rd), 20

print.Rd, 20

print.summary.epiman (print.Rd),
20

print.summary.MCepi (print.Rd), 20

print.summary.optvac (print.Rd),
20

summary, 20, 21

summary.epiman (print.Rd), 20

summary.MCepi (print.Rd), 20

summary.optvac (print.Rd), 20