Package ‘EpiEstim’

January 7, 2021

Version 2.2-4
Title Estimate Time Varying Reproduction Numbers from Epidemic Curves
Maintainer Anne Cori <a.cori@imperial.ac.uk>

URL https://github.com/mrc-ide/EpiEstim

BugReports https://github.com/mrc-ide/EpiEstim/issues

Depends R (>= 2.10)
Imports coarseDataTools (>= 0.6-4), stats, graphics, reshape2, ggplot2, gridExtra, fitdistrplus, coda, incidence (>= 1.7.0), scales, grDevices

Suggests testthat, utils, vdiffr, covr, knitr, rmarkdown
License GPL (>= 2)
LazyLoad yes
Encoding UTF-8
RoxygenNote 7.0.2
VignetteBuilder knitr

Author Anne Cori [aut, cre] (<https://orcid.org/0000-0002-8443-9162>), Simon Cauchemez [ctb], Neil M. Ferguson [ctb] (<https://orcid.org/0000-0002-1154-8093>), Christophe Fraser [ctb] (<https://orcid.org/0000-0003-2399-9657>), Elisabeth Dahlqwist [ctb] (<https://orcid.org/0000-0001-5797-6803>), P. Alex Demarsh [ctb], Thibaut Jombart [ctb] (<https://orcid.org/0000-0003-2226-8692>), Zhian N. Kamvar [ctb] (<https://orcid.org/0000-0003-1458-7108>), Justin Lessler [ctb] (<https://orcid.org/0000-0002-9741-8109>), Shikun Li [ctb], Jonathan A. Polonsky [ctb] (<https://orcid.org/0000-0002-8634-4255>),
check_cdt_samples_convergence

Checking convergence of an MCMC chain by using the Gelman-Rubin algorithm

Description

check_cdt_samples_convergence Checking convergence of an MCMC chain by using the Gelman-Rubin algorithm
Usage

`check_cdt_samples_convergence(cdt_samples)`

Arguments

cdt_samples the @sample slot of a `cd.fit.mcmc` S4 object (see package `coarseDataTools`)

Details

This function splits an MCMC chain in two halves and uses the Gelman-Rubin algorithm to assess convergence of the chain by comparing its two halves.

Value

TRUE if the Gelman Rubin test for convergence was successful, FALSE otherwise

Author(s)

Anne Cori

See Also

`estimate_R`

Examples

```r
## Not run:
## Note the following examples use an MCMC routine
## to estimate the serial interval distribution from data,
## so they may take a few minutes to run

## load data on rotavirus
data("MockRotavirus")

## estimate the serial interval from data
SI_fit <- coarseDataTools::dic.fit.mcmc(dat = MockRotavirus$si_data,
  dist="G",
  init_pars=init_mcmc_params(MockRotavirus$si_data, "G"),
  burnin = 1000,
  n.samples = 5000)

## use check_cdt_samples_convergence to check convergence
converg_diag <- check_cdt_samples_convergence(SI_fit@samples)
converg_diag

## End(Not run)
```
coarse2estim

Description

coarse2estim Transforms outputs of \texttt{coarseDataTools::dic.fit.mcmc} to right format for input into \texttt{estimate.R}

Usage

\begin{verbatim}
coarse2estim(x = NULL, dist = x@dist, samples = x@samples, thin = 10)
\end{verbatim}

Arguments

\begin{itemize}
\item \textbf{x} An object generated by function \texttt{coarseDataTools::dic.fit.mcmc}, containing posterior estimates of the serial interval distribution.
\item \textbf{dist} The parametric distribution used when estimating the serial interval. Should be one of "G" (Gamma), "W" (Weibull), "L" (Lognormal), "off1G" (Gamma shifted by 1), "off1W" (Weibull shifted by 1), or "off1L" (Lognormal shifted by 1). If not present, computed automatically from \texttt{x}.
\item \textbf{samples} A dataframe containing the posterior samples of serial interval parameters corresponding to the parametric choice specified in \texttt{dist}. If not present, computed automatically from \texttt{x}.
\item \textbf{thin} A positive integer corresponding to thinning parameter; of the posterior sample of serial interval distributions in \texttt{x}, only 1 in \texttt{thin} will be kept, the rest will be discarded.
\end{itemize}

Value

A list with two elements:

\begin{itemize}
\item \texttt{si_sample}: a matrix where each column gives one distribution of the serial interval to be explored, obtained from \texttt{x} by thinning the MCMC chain.
\item \texttt{si_parametric_distr}: the parametric distribution used when estimating the serial interval stored in \texttt{x}.
\end{itemize}

Author(s)

The Hackout3 Parameter Estimation team.

See Also

\texttt{estimate_R}
Examples

```
## Not run:
## Note the following examples use an MCMC routine
## to estimate the serial interval distribution from data,
## so they may take a few minutes to run

## load data on rotavirus
data("MockRotavirus")

## estimate the serial interval from data
SI.fit <- coarseDataTools::dic.fit.mcmc(dat = MockRotavirus$si_data,
                          dist = "G",
                          init.pars = init_mcmc_params(MockRotavirus$si_data, "G"),
                          burnin = 1000,
                          n.samples = 5000)

## use coarse2estim to turn this in the right format for estimate_R
si_sample <- coarse2estim(SI.fit, thin = 10)$si_sample

## use estimate_R to estimate the reproduction number
## based on these estimates of the serial interval
R_si_from_sample <- estimate_R(MockRotavirus$incidence,
                                method="si_from_sample",
                                si_sample=si_sample,
                                config = make_config(list(n2 = 50)))
plot(R_si_from_sample)

## End(Not run)
```

DiscrSI

Function to ensure compatibility with EpiEstim versions <2.0

Description

Please only use for compatibility; Prefer the new discr_si function instead

Usage

DiscrSI(k, mu, sigma)

Arguments

- **k**: see `k` in discr_si
- **mu**: see `mu` in discr_si
- **sigma**: see `sigma` in discr_si
**discre_Si**

*Discretized Generation Time Distribution Assuming A Shifted Gamma Distribution*

**Description**

`discre_Si` computes the discrete distribution of the serial interval, assuming that the serial interval is shifted Gamma distributed, with shift 1.

**Usage**

`discre_Si(k, mu, sigma)`

**Arguments**

- **k**: Positive integer, or vector of positive integers for which the discrete distribution is desired.
- **mu**: A positive real giving the mean of the Gamma distribution.
- **sigma**: A non-negative real giving the standard deviation of the Gamma distribution.

**Details**

Assuming that the serial interval is shifted Gamma distributed with mean $\mu$, standard deviation $\sigma$ and shift 1, the discrete probability $w_k$ that the serial interval is equal to $k$ is:

$$w_k = k F_{\mu-1, \sigma}(k) + (k-2) F_{\mu-1, \sigma}(k-2) - 2(k-1) F_{\mu-1, \sigma}(k-1) + (\mu-1) (2 F_{\mu-1+\frac{\sigma^2}{\mu-\sigma^2}, \frac{\sigma^2}{\mu-\sigma^2}}(k-1) - F_{\mu-1+\frac{\sigma^2}{\mu-\sigma^2}, \frac{\sigma^2}{\mu-\sigma^2}}(k-2))$$

where $F_{\mu, \sigma}$ is the cumulative density function of a Gamma distribution with mean $\mu$ and standard deviation $\sigma$.

**Value**

Gives the discrete probability $w_k$ that the serial interval is equal to $k$.

**Author(s)**

Anne Cori <a.cori@imperial.ac.uk>

**References**


**See Also**

`overall_infectivity`, `estimate_R`
Examples

```r
# Computing the discrete serial interval of influenza
mean_flu_si <- 2.6
sd_flu_si <- 1.5
dicrete_si_distr <- discr_si(seq(0, 20), mean_flu_si, sd_flu_si)
plot(seq(0, 20), dicrete_si_distr, type = "h",
     lwd = 10, lend = 1, xlab = "time (days)", ylab = "frequency")
title(main = "Discrete distribution of the serial interval of influenza")
```

---

EstimateR

*Function to ensure compatibility with EpiEstim versions <2.0*

Description

Please only use for compatibility; Prefer the new estimate_R function instead

Usage

```r
EstimateR(
  I,
  T.Start,
  T.End,
  method = c("NonParametricSI", "ParametricSI", "UncertainSI"),
  n1 = NULL,
  n2 = NULL,
  Mean.SI = NULL,
  Std.SI = NULL,
  Std.Mean.SI = NULL,
  Min.Mean.SI = NULL,
  Max.Mean.SI = NULL,
  Std.Std.SI = NULL,
  Min.Std.SI = NULL,
  Max.Std.SI = NULL,
  SI.Distr = NULL,
  Mean.Prior = 5,
  Std.Prior = 5,
  CV.Posterior = 0.3,
  plot = FALSE,
  leg.pos = "topright"
)
```

Arguments

- **I**: see `incid` in `estimate_R`
- **T.Start**: see `config$t_start` in `estimate_R`
- **T.End**: see `config$t_end` in `estimate_R`
estimate_R

Estimated Instantaneous Reproduction Number estimate_R estimates the reproduction number of an epidemic, given the incidence time series and the serial interval distribution.

Description

Estimated Instantaneous Reproduction Number

estimate_R estimates the reproduction number of an epidemic, given the incidence time series and the serial interval distribution.

Usage

```r
estimate_R(incid,
method = c("non_parametric_si", "parametric_si", "uncertain_si", "si_from_data",
"si_from_sample"),
si_data = NULL,
si_sample = NULL,
config = make_config(incid = incid, method = method))
```
estimate_R

Arguments

- **incid**
  One of the following
  - A vector (or a dataframe with a single column) of non-negative integers containing the incidence time series
  - A dataframe of non-negative integers with either i) `incid$incidence` containing the total incidence, or ii) two columns, so that `incid$local` contains the incidence of cases due to local transmission and `incid$imported` contains the incidence of imported cases (with `incid$local + incid$imported` the total incidence). If the dataframe contains a column `incid$dates`, this is used for plotting. `incid$dates` must contains only dates in a row.
  - An object of class `incidence`

- **method**
  One of "non_parametric_si", "parametric_si", "uncertain_si", "si_from_data" or "si_from_sample" (see details).

- **si_data**
  For method "si_from_data" ; the data on dates of symptoms of pairs of infector/infected individuals to be used to estimate the serial interval distribution (see details).

- **si_sample**
  For method "si_from_sample" ; a matrix where each column gives one distribution of the serial interval to be explored (see details).

- **config**
  An object of class `estimate_R_config`, as returned by function `make_config`.

Details

Analytical estimates of the reproduction number for an epidemic over predefined time windows can be obtained within a Bayesian framework, for a given discrete distribution of the serial interval (see references).

Several methods are available to specify the serial interval distribution.

In short there are five methods to specify the serial interval distribution (see help for function `make_config` for more detail on each method). In the first two methods, a unique serial interval distribution is considered, whereas in the last three, a range of serial interval distributions are integrated over:

- In method "non_parametric_si" the user specifies the discrete distribution of the serial interval
- In method "parametric_si" the user specifies the mean and sd of the serial interval
- In method "uncertain_si" the mean and sd of the serial interval are each drawn from truncated normal distributions, with parameters specified by the user
- In method "si_from_data", the serial interval distribution is directly estimated, using MCMC, from interval censored exposure data, with data provided by the user together with a choice of parametric distribution for the serial interval
- In method "si_from_sample", the user directly provides the sample of serial interval distribution to use for estimation of R. This can be a useful alternative to the previous method, where the MCMC estimation of the serial interval distribution could be run once, and the same estimated SI distribution then used in estimate_R in different contexts, e.g. with different time windows, hence avoiding to rerun the MCMC every time estimate_R is called.
Value

an object of class estimate_R, with components:

- R: a dataframe containing: the times of start and end of each time window considered; the posterior mean, std, and 0.025, 0.05, 0.25, 0.5, 0.75, 0.95, 0.975 quantiles of the reproduction number for each time window.
- method: the method used to estimate R, one of "non_parametric_si", "parametric_si", "uncertain_si", "si_from_data" or "si_from_sample"
- si_distr: a vector or dataframe (depending on the method) containing the discrete serial interval distribution(s) used for estimation
- SI.Moments: a vector or dataframe (depending on the method) containing the mean and std of the discrete serial interval distribution(s) used for estimation
- I: the time series of total incidence
- I_local: the time series of incidence of local cases (so that I_local + I_imported = I)
- I_imported: the time series of incidence of imported cases (so that I_local + I_imported = I)
- dates: a vector of dates corresponding to the incidence time series
- MCMC_converged (only for method "si_from_data"): a boolean showing whether the Gelman-Rubin MCMC convergence diagnostic was successful (TRUE) or not (FALSE)

Author(s)

Anne Cori <a.cori@imperial.ac.uk>

References


See Also

discr_si make_config

Examples

```r
## load data on pandemic flu in a school in 2009
data("Flu2009")

## estimate the reproduction number (method "non_parametric_si")
## when not specifying t_start and t_end in config, they are set to estimate
## the reproduction number on sliding weekly windows
res <- estimate_R(incid = Flu2009$incidence,
                  method = "non_parametric_si",
                  config = make_config(list(si_distr = Flu2009$si_distr)))
plot(res)
```
## the second plot produced shows, at each each day, the estimate of the reproduction number over the 7-day window finishing on that day.

## to specify t_start and t_end in config, e.g. to have biweekly sliding windows

t_start <- seq(2, nrow(Flu2009$incidence)-13)
t_end <- t_start + 13
res <- estimate_R(incid = Flu2009$incidence,
    method = "non_parametric_si",
    config = make_config(list(
        si_distr = Flu2009$si_distr,
        t_start = t_start,
        t_end = t_end)))
plot(res)

## the second plot produced shows, at each each day, the estimate of the reproduction number over the 14-day window finishing on that day.

## example with an incidence object

## create fake data
library(incidence)
data <- c(0,1,1,2,1,3,4,5,5,5,4,4,26,6,7,9)
location <- sample(c("local","imported"), length(data), replace=TRUE)
location[1] <- "imported" # forcing the first case to be imported

## get incidence per group (location)
incid <- incidence(data, groups = location)

## Estimate R with assumptions on serial interval
res <- estimate_R(incid, method = "parametric_si",
    config = make_config(list(
        mean_si = 2.6, std_si = 1.5)))
plot(res)

## estimate the reproduction number (method "parametric_si")
res <- estimate_R(Flu2009$incidence, method = "parametric_si",
    config = make_config(list(mean_si = 2.6, std_si = 1.5)))
plot(res)

## estimate the reproduction number (method "uncertain_si")
res <- estimate_R(Flu2009$incidence, method = "uncertain_si",
    config = make_config(list(mean_si = 2.6, std_mean_si = 1,
        min_mean_si = 1, max_mean_si = 4.2,
std_si = 1.5, std_std_si = 0.5, 
min_std_si = 0.5, max_std_si = 2.5, 
n1 = 100, n2 = 100))

plot(res)

## the bottom left plot produced shows, at each each day, 
## the estimate of the reproduction number over the 7-day window 
## finishing on that day.

## Not run:
## Note the following examples use an MCMC routine 
## to estimate the serial interval distribution from data, 
## so they may take a few minutes to run

## load data on rotavirus 
data("MockRotavirus")

## estimate the reproduction number (method "si_from_data")
MCMC_seed <- 1
overall_seed <- 2
R_si_from_data <- estimate_R(MockRotavirus$incidence,
   method = "si_from_data",
   si_data = MockRotavirus$si_data,
   config = make_config(list(si_parametric_distr = "G",
                            mcmc_control = make_mcmc_control(list(burnin = 1000,
                                                            thin = 10, seed = MCMC_seed),
                                                            n1 = 500, n2 = 50,
                                                            seed = overall_seed))))

## compare with version with no uncertainty
R_Parametric <- estimate_R(MockRotavirus$incidence,
   method = "parametric_si",
   config = make_config(list(mean_si = mean(R_si_from_data$SI.Moments$Mean),
                            std_si = mean(R_si_from_data$SI.Moments$Std))))

## generate plots
p_uncertainty <- plot(R_si_from_data, "R", options_R=list(ylim=c(0, 1.5)))
p_no_uncertainty <- plot(R_Parametric, "R", options_R=list(ylim=c(0, 1.5)))
gridExtra::grid.arrange(p_uncertainty, p_no_uncertainty,ncol=2)

## the left hand side graph is with uncertainty in the SI distribution, the 
## right hand side without. 
## The credible intervals are wider when accounting for uncertainty in the SI 
## distribution.

## estimate the reproduction number (method "si_from_sample")
MCMC_seed <- 1
overall_seed <- 2
SI.fit <- coarseDataTools::dic.fit.mcmc(dat = MockRotavirus$si_data,
   dist = "G",
   init.pars = init_mcmc_params(MockRotavirus$si_data, "G"),
   burnin = 1000,
   n.samples = 5000,
   seed = MCMC_seed)
si_sample <- coarse2estim(SI.fit, thin = 10)$si_sample
R_si_from_sample <- estimate_R(MockRotavirus$incidence,
    method = "si_from_sample",
    si_sample = si_sample,
    config = make_config(list(n2 = 50,
        seed = overall_seed)))
plot(R_si_from_sample)

## check that R_si_from_sample is the same as R_si_from_data
## since they were generated using the same MCMC algorithm to generate the SI
## sample (either internally to EpiEstim or externally)
all(R_si_from_sample$R$ / Mean(R) == R_si_from_data$R$ / Mean(R))

## End(Not run)

---

**estimate_R_plots**

*Wrapper for plot.estimate_R*

**Description**

This wrapper has been created so that several `estimate_R` objects can be plotted at the same time.

**Usage**

`estimate_R_plots(..., legend = FALSE)`

**Arguments**

- `...` Arguments of `plot.estimate_R`, but in addition, parameter `x` can be an object of class `estimate_R` (obtained as outputs of functions `estimate_R` or `wallinga_teunis`). If `x` is a list, and `what` = 'R' or `what` = 'all', all estimates of `R` are plotted on a single graph. This will only work if all the `estimate_R` objects in the list were computed using the same `config$t_start` and `config$t_end`.

- `legend` A boolean (TRUE by default) governing the presence / absence of legends on the plots.

**Value**

- A plot (if `what` = "incid", "R", or "SI") or a `grob` object (if `what` = "all").

**Author(s)**

Anne Cori, Zhian Kamvar

**See Also**

`plot.estimate_R`
Examples

```r
## load data on pandemic flu in a school in 2009
data("Flu2009")

#### COMPARE THE INSTANTANEOUS AND CASE REPRODUCTION NUMBERS ####

## estimate the instantaneous reproduction number
## (method "non_parametric_si")
R_instantaneous <- estimate_R(Flu2009$incidence,
                               method = "non_parametric_si",
                               config = list(t_start = seq(2, 26),
                                             t_end = seq(8, 32),
                                             si_distr = Flu2009$si_distr)
)

## estimate the case reproduction number
R_case <- wallinga_teunis(Flu2009$incidence,
                           method = "non_parametric_si",
                           config = list(t_start = seq(2, 26),
                                         t_end = seq(8, 32),
                                         si_distr = Flu2009$si_distr)
)

## visualise R estimates on the same plot
estimate_R_plots(list(R_instantaneous, R_case), what = "R",
                 options_R = list(col = c("blue", "red")), legend = TRUE)

#### COMPARE THE INSTANTANEOUS R ON SLIDING WEEKLY OR BIWEEKLY WINDOWS ####

R_weekly <- estimate_R(Flu2009$incidence,
                        method = "non_parametric_si",
                        config = list(t_start = seq(9, 26),
                                      t_end = seq(15, 32),
                                      si_distr = Flu2009$si_distr)
)

R_biweekly <- estimate_R(Flu2009$incidence,
                          method = "non_parametric_si",
                          config = list(t_start = seq(2, 19),
                                        t_end = seq(15, 32),
                                        si_distr = Flu2009$si_distr)
)

## visualise R estimates on the same plot
estimate_R_plots(list(R_weekly, R_biweekly), what = "R",
                 options_R = list(col = c("blue", "red")), legend = TRUE)
```
Data on the 1918 H1N1 influenza pandemic in Baltimore.

Description

This data set gives:

1. the daily incidence of onset of disease in Baltimore during the 1918 H1N1 influenza pandemic (see source and references),
2. the discrete daily distribution of the serial interval for influenza, assuming a shifted Gamma distribution with mean 2.6 days, standard deviation 1.5 days and shift 1 day (see references).

Format

A list of two elements:

- **incidence**: a vector containing 92 days of observation,
- **si_distr**: a vector containing a set of 12 probabilities.

Source


References


Examples

```r
## load data on pandemic flu in Baltimore in 1918
data("Flu1918")

## estimate the reproduction number (method "non_parametric_si")
res <- estimate_R(Flu1918$incidence,
```

```r
```
method = "non_parametric_si",
config = make_config(list(si_distr = Flu1918$si_distr)))

plot(res)
## the second plot produced shows, at each each day,
## the estimate of the reproduction number
## over the 7-day window finishing on that day.

---

**Flu2009**  
*Data on the 2009 H1N1 influenza pandemic in a school in Pennsylvania.*

### Description

This data set gives:

1. the daily incidence of onset of acute respiratory illness (ARI, defined as at least two symptoms among fever, cough, sore throat, and runny nose) among children in a school in Pennsylvania during the 2009 H1N1 influenza pandemic (see source and references),
2. the discrete daily distribution of the serial interval for influenza, assuming a shifted Gamma distribution with mean 2.6 days, standard deviation 1.5 days and shift 1 day (see references).
3. interval-censored serial interval data from the 2009 outbreak of H1N1 influenza in San Antonio, Texas, USA (see references).

### Format

A list of three elements:

- **incidence:** a dataframe with 32 lines containing dates in first column, and daily incidence in second column (Cauchemez et al., 2011),
- **si_distr:** a vector containing a set of 12 probabilities (Ferguson et al, 2005),
- **si_data:** a dataframe with 16 lines giving serial interval patient data collected in a household study in San Antonio, Texas throughout the 2009 H1N1 outbreak (Morgan et al., 2010).

### Source


### References

Examples

```r
## load data on pandemic flu in a school in 2009
data("flu2009")

## estimate the reproduction number (method "non_parametric_si")
res <- estimate_R(flu2009$incidence, method="non_parametric_si",
                 config = make_config(list(si_distr = flu2009$si_distr)))
plot(res)
## the second plot produced shows, at each each day,
## the estimate of the reproduction number
## over the 7-day window finishing on that day.

## Not run:
## Note the following examples use an MCMC routine
## to estimate the serial interval distribution from data,
## so they may take a few minutes to run

## estimate the reproduction number (method "si_from_data")
res <- estimate_R(flu2009$incidence, method="si_from_data",
                 si_data = flu2009$si_data,
                 config = make_config(list(mcmc_control = make_mcmc_control(list(
                   burnin = 1000,
                   thin = 10, seed = 1)),
                   n1 = 1000, n2 = 50,
                   si_parametric_distr = "G")))
plot(res)
## the second plot produced shows, at each each day,
## the estimate of the reproduction number
## over the 7-day window finishing on that day.

## End(Not run)
```

```
flu_2009_NYC_school Data on the 2009 H1N1 influenza pandemic in a school in New York city
```

Description

This data set gives:

1. the daily incidence of self-reported and laboratory-confirmed cases of influenza among children in a school in New York city during the 2009 H1N1 influenza pandemic (see source and references),
2. interval-censored serial interval data from the 2009 outbreak of H1N1 influenza in a New York city school (see references).
Format

A list of two elements:

- **incidence**: a dataframe with 14 lines containing dates in first column, and daily incidence in second column,
- **si_data**: a dataframe containing data on the generation time for 16 pairs of infector/infected individuals (see references and see argument `si_data` of function `estimate_R()` for details on columns)

Source


References


Examples

```r
## Not run:
## Note the following examples use an MCMC routine
## to estimate the serial interval distribution from data,
## so they may take a few minutes to run

## load data on pandemic flu in a New York school in 2009
data("flu_2009_NYC_school")

## estimate the reproduction number (method "si_from_data")
res <- estimate_R(flu_2009_NYC_school$incidence, method="si_from_data",
                 si_data = flu_2009_NYC_school$si_data,
                 config = make_config(list(
                                 t_start = seq(2, 8),
                                 t_end = seq(8, 14),
                                 si_parametric_distr = "G",
                                 mcmc_control = make_mcmc_control(list(burnin = 1000,
                                                            thin = 10, seed = 1)),
                                 n1 = 1000, n2 = 50))

plot(res)
## the second plot produced shows, at each each day,
## the estimate of the reproduction number
## over the 7-day window finishing on that day.

## End(Not run)
```
init_mcmc_params

init_mcmc_params finds clever starting points for the MCMC to be used to estimate the serial interval, e.g., when using option si_from_data in estimate_R.

Description

init_mcmc_params finds values of the serial interval distribution parameters, used to initialise the MCMC estimation of the serial interval distribution. Initial values are computed based on the observed mean and standard deviation of the sample from which the parameters are to be estimated.

Usage

init_mcmc_params(si_data, dist = c("G", "W", "L", "off1G", "off1W", "off1L"))

Arguments

si_data the data on dates of symptoms of pairs of infector/infected individuals to be used to estimate the serial interval distribution. This should be a dataframe with 5 columns:
- EL: the lower bound of the symptom onset date of the infector (given as an integer)
- ER: the upper bound of the symptom onset date of the infector (given as an integer). Should be such that ER>=EL
- SL: the lower bound of the symptom onset date of the infected individual (given as an integer)
- SR: the upper bound of the symptom onset date of the infected individual (given as an integer). Should be such that SR>=SL
- type (optional): can have entries 0, 1, or 2, corresponding to doubly interval-censored, single interval-censored or exact observations, respectively, see Reich et al. Statist. Med. 2009. If not specified, this will be automatically computed from the dates

dist the parametric distribution to use for the serial interval. Should be one of "G" (Gamma), "W" (Weibull), "L" (Lognormal), "off1G" (Gamma shifted by 1), "off1W" (Weibull shifted by 1), or "off1L" (Lognormal shifted by 1).

Value

A vector containing the initial values for the two parameters of the distribution of the serial interval. These are the shape and scale for all but the lognormal distribution, for which it is the meanlog and sdlog.

Author(s)

Anne Cori
make_config

Set and check parameter settings of estimate_R

Description

This function defines settings for estimate_R. It takes a list of named items as input, set defaults where arguments are missing, and return a list of settings.

See Also

estimate_R

Examples

## Not run:
## Note the following examples use an MCMC routine
## to estimate the serial interval distribution from data,
## so they may take a few minutes to run

## load data on rotavirus
data("MockRotavirus")

## get clever initial values for shape and scale of a Gamma distribution
## fitted to the the data MockRotavirus$si_data
clever_init_param <- init_mcmc_params(MockRotavirus$si_data, "G")

## estimate the serial interval from data using a clever starting point for
## the MCMC chain
SI_fit_clever <- coarseDataTools::dic.fit.mcmc(dat = MockRotavirus$si_data,
  dist = "G",
  init.pars = clever_init_param,
  burnin = 1000,
  n.samples = 5000)

## estimate the serial interval from data using a random starting point for
## the MCMC chain
SI_fit_naive <- coarseDataTools::dic.fit.mcmc(dat = MockRotavirus$si_data,
  dist = "G",
  burnin = 1000,
  n.samples = 5000)

## use check_cdt_samples_convergence to check convergence in both situations
converg_diag_clever <- check_cdt_samples_convergence(SI_fit_clever@samples)
converg_diag_naive <- check_cdt_samples_convergence(SI_fit_naive@samples)

## End(Not run)
**Usage**

```r
make_config(
  ..., 
  incid = NULL,
  method = c("non_parametric_si", "parametric_si", "uncertain_si", "si_from_data",
             "si_from_sample")
)
```

**Arguments**

... Acceptable arguments for ... are:

- **t_start** Vector of positive integers giving the starting times of each window over which the reproduction number will be estimated. These must be in ascending order, and so that for all \( i \), \( t_{start}[i] <= t_{end}[i] \). \( t_{start}[1] \) should be strictly after the first day with non null incidence.

- **t_end** Vector of positive integers giving the ending times of each window over which the reproduction number will be estimated. These must be in ascending order, and so that for all \( i \), \( t_{start}[i] <= t_{end}[i] \).

- **n1** For method "uncertain_si" and "si_from_data"; positive integer giving the size of the sample of SI distributions to be drawn (see details).

- **n2** For methods "uncertain_si", "si_from_data" and "si_from_sample"; positive integer giving the size of the sample drawn from the posterior distribution of R for each serial interval distribution considered (see details).

- **mean_si** For method "parametric_si" and "uncertain_si"; positive real giving the mean serial interval (method "parametric_si") or the average mean serial interval (method "uncertain_si", see details).

- **std_si** For method "parametric_si" and "uncertain_si"; non negative real giving the standard deviation of the serial interval (method "parametric_si") or the average standard deviation of the serial interval (method "uncertain_si", see details).

- **std_mean_si** For method "uncertain_si"; standard deviation of the distribution from which mean serial intervals are drawn (see details).

- **min_mean_si** For method "uncertain_si"; lower bound of the distribution from which mean serial intervals are drawn (see details).

- **max_mean_si** For method "uncertain_si"; upper bound of the distribution from which mean serial intervals are drawn (see details).

- **std_std_si** For method "uncertain_si"; standard deviation of the distribution from which standard deviations of the serial interval are drawn (see details).

- **min_std_si** For method "uncertain_si"; lower bound of the distribution from which standard deviations of the serial interval are drawn (see details).

- **max_std_si** For method "uncertain_si"; upper bound of the distribution from which standard deviations of the serial interval are drawn (see details).

- **si_distr** For method "non_parametric_si"; vector of probabilities giving the discrete distribution of the serial interval, starting with \( si_{distr}[1] \) (probability that the serial interval is zero), which should be zero.
si_parametric_distr For method "si_from_data"; the parametric distribution to use when estimating the serial interval from data on dates of symptoms of pairs of infector/infected individuals (see details). Should be one of "G" (Gamma), "W" (Weibull), "L" (Lognormal), "off1G" (Gamma shifted by 1), "off1W" (Weibull shifted by 1), or "off1L" (Lognormal shifted by 1).

mcmc_control An object of class estimate_R_mcmc_control, as returned by function make_mcmc_control.

seed An optional integer used as the seed for the random number generator at the start of the function (then potentially reset within the MCMC for method si_from_data); useful to get reproducible results.

mean_prior A positive number giving the mean of the common prior distribution for all reproduction numbers (see details).

std_prior A positive number giving the standard deviation of the common prior distribution for all reproduction numbers (see details).

cv_posterior A positive number giving the aimed posterior coefficient of variation (see details).

incid As in function estimate_R.

method As in function estimate_R.

Details

Analytical estimates of the reproduction number for an epidemic over predefined time windows can be obtained using function estimate_R, for a given discrete distribution of the serial interval. make_config allows to generate a configuration specifying the way the estimation will be performed.

The more incident cases are observed over a time window, the smallest the posterior coefficient of variation (CV, ratio of standard deviation over mean) of the reproduction number. An aimed CV can be specified in the argument cv_posterior (default is 0.3), and a warning will be produced if the incidence within one of the time windows considered is too low to get this CV.

The methods vary in the way the serial interval distribution is specified.

In short there are five methods to specify the serial interval distribution (see below for details on each method). In the first two methods, a unique serial interval distribution is considered, whereas in the last three, a range of serial interval distributions are integrated over:

- In method "non_parametric_si" the user specifies the discrete distribution of the serial interval
- In method "parametric_si" the user specifies the mean and sd of the serial interval
- In method "uncertain_si" the mean and sd of the serial interval are each drawn from truncated normal distributions, with parameters specified by the user
- In method "si_from_data", the serial interval distribution is directly estimated, using MCMC, from interval censored exposure data, with data provided by the user together with a choice of parametric distribution for the serial interval
- In method "si_from_sample", the user directly provides the sample of serial interval distribution to use for estimation of R. This can be a useful alternative to the previous method, where the MCMC estimation of the serial interval distribution could be run once, and the same estimated SI distribution then used in estimate_R in different contexts, e.g. with different time windows, hence avoiding to rerun the MCMC everytime estimate_R is called.
The discrete distribution of the serial interval is directly specified in the argument `si_distr`.

The mean and standard deviation of the continuous distribution of the serial interval are given in the arguments `mean_si` and `std_si`. The discrete distribution of the serial interval is derived automatically using `discr_si`

Method "uncertain_si" allows accounting for uncertainty on the serial interval distribution as described in Cori et al. AJE 2013. We allow the mean $\mu$ and standard deviation $\sigma$ of the serial interval to vary according to truncated normal distributions. We sample $n_1$ pairs of mean and standard deviations, $(\mu^{(1)}_1, \sigma^{(1)}_1), \ldots, (\mu^{(n_2)}_1, \sigma^{(n_2)}_1)$, by first sampling the mean $\mu^{(k)}$ from its truncated normal distribution (with mean `mean_si`, standard deviation `std_mean_si`, minimum `min_mean_si` and maximum `max_mean_si`), and then sampling the standard deviation $\sigma^{(k)}$ from its truncated normal distribution (with mean `std_si`, standard deviation `std_std_si`, minimum `min_std_si` and maximum `max_std_si`), but imposing that $\sigma^{(k)} < \mu^{(k)}$. This constraint ensures that the Gamma probability density function of the serial interval is null at $t = 0$. Warnings are produced when the truncated normal distributions are not symmetric around the mean. For each pair $(\mu^{(k)}, \sigma^{(k)})$, we then draw a sample of size $n_2$ in the posterior distribution of the reproduction number over each time window, conditionally on this serial interval distribution. After pooling, a sample of size $n_1 \times n_2$ of the joint posterior distribution of the reproduction number over each time window is obtained. The posterior mean, standard deviation, and 0.025, 0.05, 0.25, 0.5, 0.75, 0.95, 0.975 quantiles of the reproduction number for each time window are obtained from this sample.

Method "si_from_data" allows accounting for uncertainty on the serial interval distribution. Unlike method "uncertain_si", where we arbitrarily vary the mean and std of the SI in truncated normal distributions, here, the scope of serial interval distributions considered is directly informed by data on the (potentially censored) dates of symptoms of pairs of infector/infected individuals. This data, specified in argument `si_data`, should be a dataframe with 5 columns:

- EL: the lower bound of the symptom onset date of the infector (given as an integer)
- ER: the upper bound of the symptom onset date of the infector (given as an integer). Should be such that ER >= EL
- SL: the lower bound of the symptom onset date of the infected individual (given as an integer)
- SR: the upper bound of the symptom onset date of the infected individual (given as an integer). Should be such that SR >= SL
- type (optional): can have entries 0, 1, or 2, corresponding to doubly interval-censored, single interval-censored or exact observations, respectively, see Reich et al. Statist. Med. 2009. If not specified, this will be automatically computed from the dates

Assuming a given parametric distribution for the serial interval distribution (specified in `si_parametric_distr`), the posterior distribution of the serial interval is estimated directly from these data using MCMC methods implemented in the package `coarsedatatools`. The argument `mcmc_control` is a list of characteristics which control the MCMC. The MCMC is run for a total number of iterations of `mcmc_control$burnin + n1*mcmc_control$thin`; but the output is only recorded after the burnin, and only 1 in every `mcmc_control$thin` iterations, so that the posterior sample size is $n_1$. For each element in the posterior sample of serial interval distribution, we then draw a sample of size $n_2$ in
the posterior distribution of the reproduction number over each time window, conditionally on this
serial interval distribution. After pooling, a sample of size $n1 \times n2$ of the joint posterior distribution
of the reproduction number over each time window is obtained. The posterior mean, standard de-
viation, and 0.025, 0.05, 0.25, 0.5, 0.75, 0.95, 0.975 quantiles of the reproduction number for each
time window are obtained from this sample.

Method "si_from_sample" also allows accounting for uncertainty on the serial interval distribu-
tion. Unlike methods "uncertain_si" and "si_from_data", the user directly provides (in argument
si_sample) a sample of serial interval distribution to be explored.

Value

An object of class estimate_R_config with components t_start, t_end, n1, n2, mean_si, std_si,
std_mean_si, min_mean_si, max_mean_si, std_std_si, min_std_si, max_std_si, si_distr, si_parametric_distr,
mcmc_control, seed, mean_prior, std_prior, cv_posterior, which can be used as an argument of
function estimate_R.

Examples

## Not run:
## Note the following examples use an MCMC routine
## to estimate the serial interval distribution from data,
## so they may take a few minutes to run

## load data on rotavirus
data("MockRotavirus")

## estimate the reproduction number (method "si_from_data")
## we are not specifying the time windows, so by defaults this will estimate
## R on sliding weekly windows
incid <- MockRotavirus$incidence
method <- "si_from_data"
config <- make_config(incid = incid,
                      method = method,
                      list(si_parametric_distr = "G",
                           mcmc_control = make_mcmc_control(burnin = 1000,
                           thin = 10, seed = 1),
                           n1 = 500,
                           n2 = 50,
                           seed = 2))

R_si_from_data <- estimate_R(incid,
                             method = method,
                             si_data = MockRotavirus$si_data,
                             config = config)
plot(R_si_from_data)

## you can also create the config straight within the estimate_R call,
## in that case incid and method are automatically used from the estimate_R
## arguments:
R_si_from_data <- estimate_R(incid,
make_mcmc_control

make_mcmc_control Creates a list of mcmc control parameters to be used in config$mcmc_control, where config is an argument of the estimate_R function. This is used to configure the MCMC chain used to estimate the serial interval within estimate_R (with method "si_from_data").

Description

make_mcmc_control Creates a list of mcmc control parameters to be used in config$mcmc_control, where config is an argument of the estimate_R function. This is used to configure the MCMC chain used to estimate the serial interval within estimate_R (with method "si_from_data").

Usage

make_mcmc_control(
  burnin = 3000,
  thin = 10,
  seed = as.integer(Sys.time()),
  init_pars = NULL
)

Arguments

burnin A positive integer giving the burnin used in the MCMC when estimating the serial interval distribution.
thin A positive integer corresponding to thinning parameter; the MCMC will be run for burnin+n1*thin iterations; 1 in thin iterations will be recorded, after the burnin phase, so the posterior sample size is n1.
seed An integer used as the seed for the random number generator at the start of the MCMC estimation; useful to get reproducible results.
init_pars vector of size 2 corresponding to the initial values of parameters to use for the SI distribution. This is the shape and scale for all but the lognormal distribution, for which it is the meanlog and sdlog.
### Details

The argument `si_data`, should be a dataframe with 5 columns:

- **EL**: the lower bound of the symptom onset date of the infector (given as an integer)
- **ER**: the upper bound of the symptom onset date of the infector (given as an integer). Should be such that ER>=EL
- **SL**: the lower bound of the symptom onset date of the infected individual (given as an integer)
- **SR**: the upper bound of the symptom onset date of the infected individual (given as an integer). Should be such that SR>=SL
- **type** (optional): can have entries 0, 1, or 2, corresponding to doubly interval-censored, single interval-censored or exact observations, respectively, see Reich et al. Statist. Med. 2009. If not specified, this will be automatically computed from the dates.

Assuming a given parametric distribution for the serial interval distribution (specified in `si_parametric_distr`), the posterior distribution of the serial interval is estimated directly from these data using MCMC methods implemented in the package.

### Value

An object of class `estimate_R_mcmc_control` with components `burnin`, `thin`, `seed`, `init_pars`. This can be used as an argument of function `make_config`.

### Examples

```r
## Not run:  
## Note the following examples use an MCMC routine  
## to estimate the serial interval distribution from data,  
## so they may take a few minutes to run  

## load data on rotavirus  
data("MockRotavirus")

## estimate the reproduction number (method "si_from_data")  
mcmc_seed <- 1  
burnin <- 1000  
thin <- 10  
mcmc_control <- make_mcmc_control(burnin = burnin, thin = thin,  
                                   seed = mcmc_seed)

incid <- MockRotavirus$incidence  
method <- "si_from_data"  
overall_seed <- 2  
config <- make_config(incid = incid,  
                      method = method,  
                      si_parametric_distr = "G",  
                      mcmc_control = mcmc_control,  
                      n1 = 500  
                      n2 = 50,  
                      seed = overall_seed)
```

R_si_from_data <- estimate_R(incid,
    method = method,
    si_data = MockRotavirus$si_data,
    config = config)

## End(Not run)

Measles1861

Data on the 1861 measles epidemic in Hagelloch, Germany.

Description

This data set gives:

1. the daily incidence of onset of symptoms in Hallegoch (Germany) during the 1861 measles epidemic (see source and references),
2. the discrete daily distribution of the serial interval for measles, assuming a shifted Gamma distribution with mean 14.9 days, standard deviation 3.9 days and shift 1 day (see references).

Format

A list of two elements:

• incidence: a vector containing 48 days of observation,
• si_distr: a vector containing a set of 38 probabilities.

Source


References


Examples

## load data on measles in Hallegoch in 1861
data("Measles1861")

## estimate the reproduction number (method "non_parametric_si")
res <- estimate_R(Measles1861$incidence, method="non_parametric_si",
    config = make_config(list(
        t_start = seq(17, 42),
        t_end = seq(23, 48),
        si_distr = Measles1861$si_distr)))

plot(res)

## the second plot produced shows, at each each day,
## the estimate of the reproduction number
## over the 7-day window finishing on that day.
Description

This data set gives:

1. the daily incidence of onset of symptoms of laboratory confirmed human infections with MERS-CoV in Saudi Arabia between the beginning of July 2014 and the end of December 2015, and
2. estimates of the mean and standard deviation of the serial interval for MERS.

Format

A list of two elements:

- **incidence**: a dataframe containing 495 days of observations with dates in the first column, and number of local (2nd column) and imported (3rd column) cases of MERS,
- **si**: a list of estimates of the mean (mean_si) and standard deviation (std_si) of the serial interval for MERS.

Source

The incidence data was extracted from the EMPRES I system from FAO (Global Animal Disease Information System - Food and Agriculture Organization of the United Nations, 2017). Note incidence on the first day was originally made of one local case and zero imported cases; this has been modified to zero local cases and one imported case in the dataset shown here so the reproduction number can be estimated from the start using the function `estimate_R()`. The serial interval parameters were those estimated by Cauchemez et al. (2016).

References

Global Animal Disease Information System - Food and Agriculture Organization of the United Nations, 2017


Examples

```r
## load data
data("mers_2014_15")

## estimate the reproduction number (method "parametric_si")
bimonthly_R <- estimate_R(mers_2014_15$incidence[,c("local", "imported")],
                        method = "parametric_si",
                        config = make_config(
```

mean_si = mers_2014_15$si$mean_si,
std_si = mers_2014_15$si$std_si,
t_start = 2:(nrow(mers_2014_15$incidence)-8*7),
t_end = (2:(nrow(mers_2014_15$incidence)-8*7)) + 8*7)

plot(bimonthly_R, legend = FALSE, add_imported_cases = TRUE,
     options_I = list(col = c("local" = "black",
       "imported" = "red"),
       interval = 7, # show weekly incidence
       ylab = "Weekly incidence"),
     options_R = list(ylab = "Bimonthly R"))

# The first plot shows the weekly incidence,
# with imported cases shown in red and local cases in black

MockRotavirus

Mock data on a rotavirus epidemic.

Description

This data set gives:

1. the daily incidence of onset of symptoms in a mock outbreak of rotavirus,
2. mock observations of symptom onset dates for 19 pairs of infector/infected individuals.

Format

A list of two elements:

- **incidence**: a vector containing 53 days of observation,
- **si_distr**: a dataframe containing a set of 19 observations; each observation corresponds to a pair of infector/infected individuals. EL and ER columns contain the lower an upper bounds of the dates of symptoms onset in the infectors. SL and SR columns contain the lower an upper bounds of the dates of symptoms onset in the infected individuals. The type column has entries 0, 1, or 2, corresponding to doubly interval-censored, single interval-censored or exact observations, respectively, see Reich et al. Statist. Med. 2009

Examples

```r
## Not run:
## Note the following example uses an MCMC routine
## to estimate the serial interval distribution from data,
## so may take a few minutes to run

## load data
data("MockRotavirus")

## estimate the reproduction number (method "si_from_data")
res <- estimate_R(MockRotavirus$incidence,
```
method = "si_from_data",
    si_data = MockRotavirus$si_data,
    config = make_config(list(
        si_parametric_distr = "G",
        mcmc_control = make_mcmc_control(list(burnin = 3000, thin = 10)),
        n1 = 500, n2 = 50))

plot(res)
## the second plot produced shows, at each each day,
## the estimate of the reproduction number
## over the 7-day window finishing on that day.
## End(Not run)

---

**OverallInfectivity**  
*Function to ensure compatibility with EpiEstim versions <2.0*

**Description**

Please only use for compatibility; Prefer the new overall_infectivity function instead

**Usage**

OverallInfectivity(I, SI.Distr)

**Arguments**

- I: see incid in overall_infectivity
- SI.Distr: see si_distr in overall_infectivity

---

**overall_infectivity**  
*Overall Infectivity Due To Previously Infected Individuals*

**Description**

overall_infectivity computes the overall infectivity due to previously infected individuals.

**Usage**

overall_infectivity(incid, si_distr)
overall_infectivity

Arguments

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>incid</td>
<td>One of the following</td>
</tr>
<tr>
<td></td>
<td>• A vector (or a dataframe with a single column) of non-negative integers</td>
</tr>
<tr>
<td></td>
<td>containing an incidence time series</td>
</tr>
<tr>
<td></td>
<td>• A dataframe of non-negative integers with two columns, so that incid$local</td>
</tr>
<tr>
<td></td>
<td>contains the incidence of cases due to local transmission and incid$imported</td>
</tr>
<tr>
<td></td>
<td>contains the incidence of imported cases (with incid$local + incid$imported</td>
</tr>
<tr>
<td></td>
<td>the total incidence).</td>
</tr>
<tr>
<td>si_distr</td>
<td>Vector of probabilities giving the discrete distribution of the serial interval.</td>
</tr>
</tbody>
</table>

Note that the cases from the first time step are always assumed to be imported cases.

Details

The overall infectivity $\lambda_t$ at time step $t$ is equal to the sum of the previously infected individuals (given by the incidence vector $I$, with $I = incid$local + incid$imported$ if $I$ is a matrix), weighted by their infectivity at time $t$ (given by the discrete serial interval distribution $w_k$). In mathematical terms:

$$\lambda_t = \sum_{k=1}^{t-1} I_{t-k} w_k$$

Value

A vector which contains the overall infectivity $\lambda_t$ at each time step.

Author(s)

Anne Cori <a.cori@imperial.ac.uk>

References


See Also

discr_si, estimate_R

Examples

```r
## load data on pandemic flu in a school in 2009
data("Flu2009")

## compute overall infectivity
lambda <- overall_infectivity(Flu2009$incidence, Flu2009$si_distr)
par(mfrow=c(2,1))
plot(Flu2009$incidence, type = "s", xlab = "time (days)", ylab = "incidence")
title(main = "Epidemic curve")
plot(lambda, type = "s", xlab = "time (days)", ylab = "Infectivity")
title(main = "Overall infectivity")
```
plot.estimate_R

Plot outputs of estimate_r

Description

The plot method of estimate_r objects can be used to visualise three types of information. The first one shows the epidemic curve. The second one shows the posterior mean and 95% credible interval of the reproduction number. The estimate for a time window is plotted at the end of the time window. The third plot shows the discrete distribution(s) of the serial interval.

Usage

```r
## S3 method for class 'estimate_R'
plot(
  x,
  what = c("all", "incid", "R", "SI"),
  add_imported_cases = FALSE,
  options_I = list(col = palette(), transp = 0.7, xlim = NULL, ylim = NULL, interval = 1L, xlab = "Time", ylab = "Incidence"),
  options_R = list(col = palette(), transp = 0.2, xlim = NULL, ylim = NULL, xlab = "Time", ylab = "R"),
  options_SI = list(prob_min = 0.001, col = "black", transp = 0.25, xlim = NULL, ylim = NULL, xlab = "Time", ylab = "Frequency"),
  legend = TRUE,
  ...
)
```

Arguments

- `x` The output of function `estimate_R` or function `wallinga_teunis`. To plot simultaneous outputs on the same plot use `estimate_R_plots` function
- `what` A string specifying what to plot, namely the incidence time series (`what = 'incid'`), the estimated reproduction number (`what = 'R'`), the serial interval distribution (`what = 'SI'`), or all three (`what = 'all'`).
- `add_imported_cases` A boolean to specify whether, on the incidence time series plot, to add the incidence of imported cases.
- `options_I` For `what = "incid"` or "all". A list of graphical options:
  - `col` A color or vector of colors used for plotting incid. By default uses the default R colors.
  - `transp` A numeric value between 0 and 1 used to monitor transparency of the bars plotted. Defaults to 0.7.
  - `xlim` A parameter similar to that in `par`, to monitor the limits of the horizontal axis
  - `ylim` A parameter similar to that in `par`, to monitor the limits of the vertical axis
plot.estimate_R

interval An integer or character indicating the (fixed) size of the time interval used for plotting the incidence; defaults to 1 day.

xlab, ylab Labels for the axes of the incidence plot

options_R For what = "R" or "all". A list of graphical options:

col A color or vector of colors used for plotting R. By default uses the default R colors.

transp A numeric value between 0 and 1 used to monitor transparency of the 95% CrI. Defaults to 0.2.
xlim A parameter similar to that in par, to monitor the limits of the horizontal axis

ylim A parameter similar to that in par, to monitor the limits of the vertical axis

xlab, ylab Labels for the axes of the R plot

options_SI For what = "SI" or "all". A list of graphical options:

prob_min A numeric value between 0 and 1. The SI distributions explored are only shown from time 0 up to the time t so that each distribution explored has probability < prob_min to be on any time step after t. Defaults to 0.001.
col A color or vector of colors used for plotting the SI. Defaults to black.
transp A numeric value between 0 and 1 used to monitor transparency of the lines. Defaults to 0.25
xlim A parameter similar to that in par, to monitor the limits of the horizontal axis

ylim A parameter similar to that in par, to monitor the limits of the vertical axis

xlab, ylab Labels for the axes of the serial interval distribution plot

legend A boolean (TRUE by default) governing the presence / absence of legends on the plots

... further arguments passed to other methods (currently unused).

Value

a plot (if what = "incid", "R", or "SI") or a grob object (if what = "all").

Author(s)

Rolina van Gaalen <rolina.van.gaalen@rivm.nl> and Anne Cori <a.cori@imperial.ac.uk>; S3 method by Thibaut Jombart

See Also

estimate_R, wallinga_teunis and estimate_R_plots

Examples

## load data on pandemic flu in a school in 2009
data("Flu2009")

## estimate the instantaneous reproduction number
```r
## (method "non_parametric_si")
R_i <- estimate_R(Flu2009$incidence,
    method = "non_parametric_si",
    config = list(t_start = seq(2, 26),
                 t_end = seq(8, 32),
                 si_distr = Flu2009$si_distr
    )
)

## visualise results
plot(R_i, legend = FALSE)

## estimate the instantaneous reproduction number
## (method "non_parametric_si")
R_c <- wallinga_teunis(Flu2009$incidence,
    method = "non_parametric_si",
    config = list(t_start = seq(2, 26),
                 t_end = seq(8, 32),
                 si_distr = Flu2009$si_distr
    )
)

## produce plot of the incidence
## (with, on top of total incidence, the incidence of imported cases),
## estimated instantaneous and case reproduction numbers
## and serial interval distribution used
p_I <- plot(R_i, "incid", add_imported_cases=TRUE) # plots the incidence
p_SI <- plot(R_i, "SI") # plots the serial interval distribution
p_Ri <- plot(R_i, "R",
             options_R = list(ylim = c(0, 4)))
    # plots the estimated instantaneous reproduction number
p_Rc <- plot(R_c, "R",
             options_R = list(ylim = c(0, 4)))
    # plots the estimated case reproduction number
gridExtra::grid.arrange(p_I, p_SI, p_Ri, p_Rc, ncol = 2)
```

---

**Description**

sample from the posterior R distribution

**Usage**

`sample_posterior_R(R, n = 1000, window = 1L)`
Arguments

R  
an estimate_R object from the estimate_r function function.

n  
an integer specifying the number of samples to be taken from the gamma distribution.

window  
an integer (or sequence of integers) specifying the window(s) from which to estimate R. Defaults to the first window. If multiple windows are specified, the resulting samples will be drawn from several distributions.

Value

n values of R from the posterior R distribution

Author(s)

Anne Cori

Examples

```r
## load data on pandemic flu in a school in 2009
data("Flu2009")

## estimate the reproduction number (method "non_parametric_si")
## when not specifying t_start and t_end in config, they are set to estimate
## the reproduction number on sliding weekly windows
res <- estimate_R(incid = Flu2009$incidence,
                   method = "non_parametric_si",
                   config = make_config(list(si_distr = Flu2009$si_distr)))

## Sample R from the first weekly window
win <- 1L
R_median <- res$R$[win]
R_CrI <- c(res$R$Quantile.0.025(R)[win], res$R$Quantile.0.975(R)[win])

set.seed(2019-06-06) # fixing the random seed for reproducibility
R_sample <- sample_posterior_R(res, n = 1000, window = win)
hist(R_sample, col = "grey", main = "R sampled from the first weekly window")
abline(v = R_median, col = "red")  # show the median estimated R
abline(v = R_CrI, col = "red", lty = 2)  # show the 95%CrI of R
```

Data on the 2003 SARS epidemic in Hong Kong.
Description

This data set gives:

1. the daily incidence of onset of symptoms in Hong Kong during the 2003 severe acute respiratory syndrome (SARS) epidemic (see source and references),
2. the discrete daily distribution of the serial interval for SARS, assuming a shifted Gamma distribution with mean 8.4 days, standard deviation 3.8 days and shift 1 day (see references).

Format

A list of two elements:

- **incidence**: a vector containing 107 days of observation,
- **si_distr**: a vector containing a set of 25 probabilities.

Source


References


Examples

```r
## load data on SARS in Hong Kong in 2003
data("SARS2003")

## estimate the reproduction number (method "non_parametric_si")
res <- estimate_R(SARS2003$incidence, method="non_parametric_si",
                 config = make_config(list(
                   t_start = seq(14, 101),
                   t_end = seq(20, 107),
                   si_distr = SARS2003$si_distr)))
plot(res)
```

### the second plot produced shows, at each each day,
### the estimate of the reproduction number
### over the 7-day window finishing on that day.
**Description**

This data set gives:

1. the daily incidence of onset of symptoms in Kosovo during the 1972 smallpox epidemic (see source and references),
2. the discrete daily distribution of the serial interval for smallpox, assuming a shifted Gamma distribution with mean 22.4 days, standard deviation 6.1 days and shift 1 day (see references).

**Format**

A list of two elements:

- **incidence**: a vector containing 57 days of observation,
- **si_distr**: a vector containing a set of 46 probabilities.

**Source**


**References**


**Examples**

```r
## load data on smallpox in Kosovo in 1972
data("Smallpox1972")

## estimate the reproduction number (method "non_parametric_si")
res <- estimate_R(Smallpox1972$incidence, method="non_parametric_si",
                 config = make_config(list(
                     t_start = seq(27, 51),
                     t_end = seq(33, 57),
                     si_distr = Smallpox1972$si_distr)))

plot(res)
## the second plot produced shows, at each each day,
## the estimate of the reproduction number
## over the 7-day window finishing on that day.
```
**Estimation of the case reproduction number using the Wallinga and Teunis method**

**Description**

`wallinga_teunis` estimates the case reproduction number of an epidemic, given the incidence time series and the serial interval distribution.

**Usage**

```r
wallinga_teunis(
  incid,
  method = c("non_parametric_si", "parametric_si"),
  config
)
```

**Arguments**

- **incid**: One of the following
  - Vector (or a dataframe with a column named 'incid') of non-negative integers containing an incidence time series. If the dataframe contains a column `incid$dates`, this is used for plotting. `incid$dates` must contain only dates in a row.
  - An object of class `incidence`.

- **method**: the method used to estimate R, one of "non_parametric_si", "parametric_si", "uncertain_si", "si_from_data" or "si_from_sample".

- **config**: a list with the following elements:
  - `t_start`: Vector of positive integers giving the starting times of each window over which the reproduction number will be estimated. These must be in ascending order, and so that for all `i`, `t_start[i] <= t_end[i]`. `t_start[1]` should be strictly after the first day with non null incidence.
  - `t_end`: Vector of positive integers giving the ending times of each window over which the reproduction number will be estimated. These must be in ascending order, and so that for all `i`, `t_start[i] <= t_end[i]`.
  - `method`: One of "non_parametric_si" or "parametric_si" (see details).
  - `mean_si`: For method "parametric_si"; positive real giving the mean serial interval.
  - `std_si`: For method "parametric_si"; non negative real giving the standard deviation of the serial interval.
  - `si_distr`: For method "non_parametric_si"; vector of probabilities giving the discrete distribution of the serial interval, starting with `si_distr[1]` (probability that the serial interval is zero), which should be zero.
  - `n_sim`: A positive integer giving the number of simulated epidemic trees used for computation of the confidence intervals of the case reproduction number (see details).
Details

Estimates of the case reproduction number for an epidemic over predefined time windows can be obtained, for a given discrete distribution of the serial interval, as proposed by Wallinga and Teunis (AJE, 2004). Confidence intervals are obtained by simulating a number (config$n_sim) of possible transmission trees (only done if config$n_sim > 0).

The methods vary in the way the serial interval distribution is specified.

______________ method "non_parametric_si" _____________

The discrete distribution of the serial interval is directly specified in the argument config$si_distr.

______________ method "parametric_si" ________________

The mean and standard deviation of the continuous distribution of the serial interval are given in the arguments config$mean_si and config$std_si. The discrete distribution of the serial interval is derived automatically using discr_si.

Value

a list with components:

- R: a dataframe containing: the times of start and end of each time window considered; the estimated mean, std, and 0.025 and 0.975 quantiles of the reproduction number for each time window.
- si_distr: a vector containing the discrete serial interval distribution used for estimation
- SI.Moments: a vector containing the mean and std of the discrete serial interval distribution(s) used for estimation
- I: the time series of total incidence
- I_local: the time series of incidence of local cases (so that I_local + I_imported = I)
- I_imported: the time series of incidence of imported cases (so that I_local + I_imported = I)
- dates: a vector of dates corresponding to the incidence time series

Author(s)

Anne Cori <a.cori@imperial.ac.uk>

References


See Also

discr_si, estimate_R
Examples

```r
## load data on pandemic flu in a school in 2009
data("Flu2009")

## estimate the case reproduction number (method "non_parametric_si")
res <- wallinga_teunis(Flu2009$incidence,
    method="non_parametric_si",
    config = list(t_start = seq(2, 26), t_end = seq(8, 32),
                  si_distr = Flu2009$si_distr,
                  n_sim = 100))
plot(res)
## the second plot produced shows, at each each day,
## the estimate of the case reproduction number over the 7-day window
## finishing on that day.

## estimate the case reproduction number (method "parametric_si")
res <- wallinga_teunis(Flu2009$incidence, method="parametric_si",
    config = list(t_start = seq(2, 26), t_end = seq(8, 32),
                  mean_si = 2.6, std_si = 1.5,
                  n_sim = 100))
plot(res)
## the second plot produced shows, at each each day,
## the estimate of the case reproduction number over the 7-day window
## finishing on that day.
```

---

**WT**

*Function to ensure compatibility with EpiEstim versions <2.0*

**Description**

Please only use for compatibility; Prefer the new wallinga_teunis function instead

**Usage**

```r
WT(
  I, 
  T.Start, 
  T.End, 
  method = c("NonParametricSI", "ParametricSI"), 
  Mean.SI = NULL, 
  Std.SI = NULL, 
  SI.Distr = NULL, 
  nSim = 10, 
  plot = FALSE, 
  leg.pos = "topright" 
)
```
Arguments

- **I**  
  see `incid` in `wallinga_teunis`
- **T.Start**  
  see `config$t_start` in `wallinga_teunis`
- **T.End**  
  see `config$t_end` in `wallinga_teunis`
- **method**  
  see method in `wallinga_teunis` (but WT uses CamelCase where `wallinga_teunis` uses `snake_case` for the method names)
- **Mean.SI**  
  see `config$mean_si` in `wallinga_teunis`
- **Std.SI**  
  see `config$std_si` in `wallinga_teunis`
- **SI.Distr**  
  see `config$si_distr` in `wallinga_teunis`
- **nSim**  
  see `config$n_sim` in `wallinga_teunis`
- **plot**  
  Not used anymore, only there for compatibility
- **leg.pos**  
  Not used anymore, only there for compatibility
Index

check_cdt_samples_convergence, 2
coarse2estim, 4
discr_si, 6, 10, 23, 31, 39
DiscrSI, 5
estimate_R, 3, 4, 6, 8, 13, 20, 31–33, 39
estimate_R(), 18, 28
estimate_R_plots, 13, 32, 33
EstimateR, 7
Flu1918, 15
Flu2009, 16
flu_2009_NYC_school, 17
grob, 13, 33
incidence, 9, 38
init_mcmc_params, 19
make_config, 10, 20
make_mcmc_control, 25
Measles1861, 27
mers_2014_15, 28
MockRotavirus, 29
overall_infectivity, 6, 30
OverallInfectivity, 30
plot.estimate_R, 13, 32
sample_posterior_R, 34
SARS2003, 35
Smallpox1972, 37
wallinga_teunis, 13, 32, 33, 38
WT, 40