Package: EpiEstim (via r-universe)

August 30, 2024

Version 2.4

Title Estimate Time Varying Reproduction Numbers from Epidemic Curves

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

Description Tools to quantify transmissibility throughout an epidemic from the analysis of time series of incidence as described in Cori et al. (2013) <doi:10.1093/aje/kwt133> and Wallinga and Teunis (2004) <doi:10.1093/aje/kwh255>.

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

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

Depends R (>= 3.3.0)

Imports coarseDataTools (>= 0.6-4), stats, graphics, reshape2, ggplot2, gridExtra, fitdistrplus, coda, incidence (>= 1.7.0), scales, grDevices, abind, epitrix, distcrete, patchwork (>= 1.2.0)

Suggests testthat, utils, vdiffr, covr, knitr, rmarkdown, kableExtra, hrbrthemes, MCMCglmm, projections, dplyr, readxl, gghighlight

License GPL (>=2)

LazyLoad yes

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

Repository https://mrc-ide.r-universe.dev

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

RemoteRef master

RemoteSha 238d43548dfb70374089c2a541ce822d218093dc

Contents

aggregate_inc . . . 3
backimpute_I . . . 3
check_cdt_samples_convergence . . . 4
coarse2estim . . . 5
compute_lambda . . . 7
compute_si_cutoff . . . 8
compute_t_min . . . 9
covid_deaths_2020_uk . . . 9
default_mcmc_controls . . . 10
default_priors . . . 11
DiscrSI . . . 12
discr_si . . . 12
draw_epsilon . . . 13
draw_R . . . 15
EstimateR . . . 16
estimate_advantage . . . 17
estimate_R . . . 20
estimate_R_agg . . . 26
estimate_R_plots . . . 30
first_nonzero_incid . . . 32
Flu1918 . . . 33
Flu2009 . . . 34
flu_2009_NYC_school . . . 35
get_shape_epsilon . . . 37
get_shape_R_flat . . . 38
init_mcmc_params . . . 39
make_config . . . 40
make_mcmc_control . . . 45
Measles1861 . . . 47
mers_2014_15 . . . 48
MockRotavirus . . . 49
OverallInfectivity . . . 50
overall_infectivity . . . 50
plot.estimate_R . . . 52
process_I_multivariant . . . 54
sample_posterior_R . . . 55
SARS2003 . . . 56
Smallpox1972 . . . 58
wallinga_teunis . . . 59
WT . . . 61

Index **63**

aggregate_inc *Aggregating daily incidence to longer time windows*

Description

Aggregating daily incidence to longer time windows

Usage

```
aggregate_inc(incid, dt = 7L)
```

Arguments

incid	a vector of daily incidence values
dt	a positive integer, or vector thereof, indicating the length(s) of the desired aggregation window(s). If a vector, this will be recycled. For example, `dt = c(3L, 4L)` would correspond to alternating aggregation windows of 3 and 4 days

Value

a vector of incidence values, aggregated to dt

Examples

```
## Constant aggregation e.g. weekly reporting
data("SARS2003")
incid <- SARS2003$incidence
dt <- 7L
aggregate_inc(incid, dt)

## Non-constant aggregation e.g. reporting 3x week
#' data("SARS2003")
incid <- SARS2003$incidence
dt <- c(2L,2L,3L)
aggregate_inc(incid, dt)
```

backimpute_I *Impute unobserved generations of infection*

Description

This function imputes incidence prior the first date of reported cases to address early bias in R_t estimates. A simple linear model is fitted on shifted, logged-incidence cases, based on an initial observation window. Log-incidence is computed as log(local + 0.5) to avoid -infinite logs. Currently, no cases are assumed to be imported.

Usage

```
backimpute_I(incid, window_b)
```

Arguments

incid	the raw, reported incidence cases.
window_b	length of the observation window to fit the exponential growth model for back-imputation

Value

an incidence data.frame, combining back-imputed cases for a maximum of 100 time points (with rows indexed by a negative integer rowname) and cases (with rows #' indexed by a non-negative integer)

Examples

```
incid_all <- ceiling(exp(.3 * 0:20))
incid_trunc <- tail(incid_all, 10)
x <- backimpute_I(incid=incid_trunc, window_b=6)
idx <- as.integer(rownames(x)) > -10
x[idx, ]$local
incid_all
```

---

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

```
## 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` *Link coarseDataTools and EpiEstim*

Description

`coarse2estim` Transforms outputs of `coarseDataTools::dic.fit.mcmc` to right format for input into `estimate_R`

Usage

```
coarse2estim(x = NULL, dist = x@dist, samples = x@samples, thin = 10)
```

Arguments

x	An object generated by function coarseDataTools::dic.fit.mcmc, containing posterior estimates of the serial interval distribution.
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 x.
samples	A dataframe containing the posterior samples of serial interval parameters corresponding to the parametric choice specified in dist. If not present, computed automatically from x.
thin	A positive integer corresponding to thinning parameter; of the posterior sample of serial interval distributions in x, only 1 in thin will be kept, the rest will be discarded.

Value

A list with two elements:

- si_sample: a matrix where each column gives one distribution of the serial interval to be explored, obtained from x by thinning the MCMC chain.
- si_parametric_distr: the parametric distribution used when estimating the serial interval stored in x.

Author(s)

The Hackout3 Parameter Estimation team.

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")

## 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)
```

`compute_lambda` *Compute the overall infectivity*

Description

Compute the overall infectivity

Usage

```
compute_lambda(incid, si_distr)
```

Arguments

`incid`	a list (as obtained from function 'process_I_multivariant') of two multidimensional arrays ("local" and "imported") containing values of the incidence for each time step (1st dimension), location (2nd dimension) and pathogen/strain/variant (3rd dimension)
`si_distr`	a matrix where each column contains the probability mass function for the discrete serial interval for each of the pathogen/strain/variants, starting with the probability mass function for day 0 in the first row, which should be 0. Each column in the matrix should sum to 1

Value

a multidimensional array containing values of the overall infectivity for each time step (1st dimension), location (2nd dimension) and pathogen/strain/variant (3rd dimension). The overall infectivity for a given location and pathogen/strain/variant represents the sum of the incidence for that location and that pathogen/strain/variant at all previous time steps, weighted by the current infectivity of those past incident cases. Pre-calculating the overall infectivity makes the algorithm much faster

Examples

```
n_v <- 2
n_loc <- 3 # 3 locations
T <- 100 # 100 time steps
priors <- default_priors()
# constant incidence 10 per day everywhere
incid <- array(10, dim = c(T, n_loc, n_v))
incid <- process_I_multivariant(incid)
# arbitrary serial interval, same for both variants
w_v <- c(0, 0.2, 0.5, 0.3)
si_distr <- cbind(w_v, w_v)
lambda <- compute_lambda(incid, si_distr)
```

---

compute_si_cutoff *Index before which at most a given probability mass is captured*

Description

Across a matrix of discretised probability distributions (see `estimate_advantage` this function returns the largest index (across all columns) such that the cumulative probability mass before index is 1 - `miss_at_most`.

Usage

```
compute_si_cutoff(si_distr, miss_at_most = 0.05)
```

Arguments

`si_distr`	a matrix with two columns, each containing the probability mass function for the discrete serial interval for each of the two pathogen/strain/variants, starting with the probability mass function for day 0 in the first row, which should be 0. each column in the matrix should sum to 1
`miss_at_most`	numeric. probability mass in the tail of the SI distribution

Value

integer

Author(s)

Sangeeta Bhatia

compute_t_min *Compute the smallest index at which joint estimation should start*

Description

Unless specified by the user, t_min in `estimate_advantage` is computed as the sum of two indices: (i) the first day of non-zero incidence across all locations, computed using `first_nonzero_incid` and (ii) the 95th percentile of the probability mass function of the SI distribution across all variants computed using `compute_si_cutoff`

Usage

```
compute_t_min(incid, si_distr, miss_at_most)
```

Arguments

`incid`	a multidimensional array containing values of the incidence for each time step (1st dimension), location (2nd dimension) and pathogen/strain/variant (3rd dimension)
`si_distr`	a matrix with two columns, each containing the probability mass function for the discrete serial interval for each of the two pathogen/strain/variants, starting with the probability mass function for day 0 in the first row, which should be 0. each column in the matrix should sum to 1
`miss_at_most`	numeric. probability mass in the tail of the SI distribution

Value

integer

Author(s)

Sangeeta Bhatia

covid_deaths_2020_uk *Data on the 2020-2022 SARS-CoV-2 epidemic in the UK.*

Description

This data set gives:

1. the UK daily incidence of deaths within 28 days of a positive COVID test (see source and references),
2. the discrete daily distribution of the serial interval for SARS-CoV-2, assuming a shifted Gamma distribution with mean 6.1 days, standard deviation 4.2 days and shift 1 day (see references).

Format

A list of two elements:

- **incidence:** a data.frame containing 672 days of observation,
- **si_distr:** a vector containing a set of 30 probabilities.

Source

Bi Q, Wu Y, Mei S, Ye C, Zou X, Zhang Z, et al. Epidemiology and transmission of COVID-19 in 391 cases and 1286 of their close contacts in Shenzhen, China: a retrospective cohort study. The Lancet Infectious Diseases. 2020 Aug 1;20(8):911–9.

https://coronavirus.data.gov.uk/details/cases

References

Nash RK, Cori A, Nouvellet P. Estimating the epidemic reproduction number from temporally aggregated incidence data: a statistical modelling approach and software tool. (medRxiv pre-print)

Bi Q, Wu Y, Mei S, Ye C, Zou X, Zhang Z, et al. Epidemiology and transmission of COVID-19 in 391 cases and 1286 of their close contacts in Shenzhen, China: a retrospective cohort study. The Lancet Infectious Diseases. 2020 Aug 1;20(8):911–9.

Examples

```
data("covid_deaths_2020_uk")

## estimate the reproduction number (method "non_parametric_si")
res <- estimate_R(covid_deaths_2020_uk$incidence$Incidence,
method="non_parametric_si",
          config = make_config(list(si_distr = covid_deaths_2020_uk$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.
```

---

default_mcmc_controls *Set default for MCMC control*

Description

Set default for MCMC control

Usage

```
default_mcmc_controls()
```

Value

a list of default MCMC control parameters, containing:

- n_iter: the number if iterations of the MCMC to perform

- burnin: the burnin to use; MCMC iterations will only be recorded after the burnin

- thin: MCMC iterations will only be recorded after the burnin and every ‘thin‘ iteration

Values can then be manually be edited as in the examples below.

Examples

```
mcmc_control<- default_mcmc_controls()
# change to run for 10 times longer
mcmc_control$n_iter <- mcmc_control$n_iter * 10
```

---

default_priors *Set default for Gamma priors*

Description

Set default for Gamma priors

Usage

```
default_priors()
```

Value

a list of default parameters for the priors. Values can then be manually be edited as in the examples below. Users could use functions ‘epitrix::gamma_shapescale2mucv‘ and ‘epitrix::gamma_mucv2shapescale‘ to set the shape and scale corresponding to the desired prior mean and coefficient of variation.

Examples

```
priors <- default_priors()
# change the prior for R to have a mean of 3
priors$R$shape <- 3
```

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

discr_si — *Discretized Generation Time Distribution Assuming A Shifted Gamma Distribution*

Description

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

Usage

```
discr_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 μ, standard deviation σ and shift 1, the discrete probability w_k that the serial interval is equal to k is:

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

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

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

Cori, A. et al. A new framework and software to estimate time-varying reproduction numbers during epidemics (AJE 2013).

See Also

overall_infectivity, estimate_R

Examples

```
## 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")
```

---

draw_epsilon *Draw epsilon from marginal posterior distribution*

Description

Draw epsilon from marginal posterior distribution

Usage

```
draw_epsilon(
  R,
  incid,
  lambda,
  priors,
  shape_epsilon = NULL,
  t_min = 2L,
  t_max = nrow(incid),
  seed = NULL
)
```

Arguments

R	a matrix with dimensions containing values of the instantaneous reproduction number for each time step (row) and location (column), for the reference pathogen/strain/variant
incid	a multidimensional array containing values of the (local) incidence for each time step (1st dimension), location (2nd dimension) and pathogen/strain/variant (3rd dimension)
lambda	a multidimensional array containing values of the overall infectivity for each time step (1st dimension), location (2nd dimension) and pathogen/strain/variant (3rd dimension). The overall infectivity for a given location and pathogen/strain/variant represents the sum of the incidence for that location and that pathogen/strain/variant at all previous time steps, weighted by the current infectivity of those past incident cases. It can be calculated from the incidence 'incid' and the distribution of the serial interval using function 'compute_lambda')
priors	a list of prior parameters (shape and scale of a gamma distribution) for epsilon and R; can be obtained from the function 'default_priors'. The prior for R is assumed to be the same for all time steps and all locations
shape_epsilon	a value or vector of values of the shape of the posterior distribution of epsilon for each of the non reference variants, as returned by function 'get_shape_epsilon'
t_min	an integer >1 giving the minimum time step to consider in the estimation. Default value is 2 (as the estimation is conditional on observations at time step 1 and can therefore only start at time step 2).
t_max	an integer >'t_min' and <='nrow(incid)' giving the maximum time step to consider in the estimation. Default value is 'nrow(incid)'.
seed	a numeric value used to fix the random seed

Value

a value or vector of values for epsilon for each non reference pathogen/strain/variant, drawn from the marginal posterior distribution

Examples

```
n_loc <- 4 # 4 locations
n_v <- 3 # 3 strains
T <- 100 # 100 time steps
priors <- default_priors()
# constant incidence 10 per day everywhere
incid <- array(10, dim = c(T, n_loc, n_v))
incid <- process_I_multivariant(incid)
# arbitrary serial interval, same for both variants
w_v <- c(0, 0.2, 0.5, 0.3)
si_distr <- cbind(w_v, w_v, w_v)
lambda <- compute_lambda(incid, si_distr)
# Constant reproduction number of 1
R <- matrix(1, nrow = T, ncol = n_loc)
R[1, ] <- NA # no estimates of R on first time step
draw_epsilon(R, incid$local, lambda, priors, seed = 1)
```

draw_R *Draw R from marginal posterior distribution*

Description

Draw R from marginal posterior distribution

Usage

```
draw_R(
  epsilon,
  incid,
  lambda,
  priors,
  shape_R_flat = NULL,
  t_min = NULL,
  t_max = nrow(incid),
  seed = NULL
)
```

Arguments

epsilon	a value or vector of values for the relative transmissibility of the "new" pathogen/strain/variant(s) compared to the reference pathogen/strain/variant
incid	a multidimensional array containing values of the (local) incidence for each time step (1st dimension), location (2nd dimension) and pathogen/strain/variant (3rd dimension)
lambda	a multidimensional array containing values of the overall infectivity for each time step (1st dimension), location (2nd dimension) and pathogen/strain/variant (3rd dimension). The overall infectivity for a given location and pathogen/strain/variant represents the sum of the incidence for that location and that pathogen/strain/variant at all previous time steps, weighted by the current infectivity of those past incident cases. It can be calculated from the incidence ‘incid‘ and the distribution of the serial interval using function ‘compute_lambda‘)
priors	a list of prior parameters (shape and scale of a gamma distribution) for epsilon and R; can be obtained from the function ‘default_priors‘. The prior for R is assumed to be the same for all time steps and all locations
shape_R_flat	a vector of the shape of the posterior distribution of R for each time step t and each location l (stored in element (l-1)*(t_max - t_min + 1) + t of the vector), as obtained from function ‘get_shape_R_flat‘
t_min	an integer >1 giving the minimum time step to consider in the estimation. Default value is 2 (as the estimation is conditional on observations at time step 1 and can therefore only start at time step 2).
t_max	an integer >‘t_min‘ and <=‘nrow(incid)‘ giving the maximum time step to consider in the estimation. Default value is ‘nrow(incid)‘.
seed	a numeric value used to fix the random seed

Value

a matrix of the instantaneous reproduction number R for the reference pathogen/strain/variant for each time step (row) and each location (column) drawn from the marginal posterior distribution

Examples

```
n_v <- 2
n_loc <- 3 # 3 locations
T <- 100 # 100 time steps
priors <- default_priors()
# constant incidence 10 per day everywhere
incid <- array(10, dim = c(T, n_loc, n_v))
incid <- process_I_multivariant(incid)
# arbitrary serial interval, same for both variants
w_v <- c(0, 0.2, 0.5, 0.3)
si_distr <- cbind(w_v, w_v)
lambda <- compute_lambda(incid, si_distr)
# Epsilon = 1 i.e. no transmission advantage
epsilon <- 1
draw_R(epsilon, incid$local, lambda, priors, seed = 1, t_min = 2L)
```

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

Description

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

Usage

```
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
method	see method in estimate_R (but EstimateR uses CamelCase where estimate_R uses snake_case for the method names)
n1	see n1 in estimate_R
n2	see n2 in estimate_R
Mean.SI	see config$mean_si in estimate_R
Std.SI	see config$std_si in estimate_R
Std.Mean.SI	see config$std_mean_si in estimate_R
Min.Mean.SI	see config$min_mean_si in estimate_R
Max.Mean.SI	see config$max_mean_si in estimate_R
Std.Std.SI	see config$std_std_si in estimate_R
Min.Std.SI	see config$min_std_si in estimate_R
Max.Std.SI	see config$max_std_si in estimate_R
SI.Distr	see config$si_distr in estimate_R
Mean.Prior	see config$mean_prior in estimate_R
Std.Prior	see config$std_prior in estimate_R
CV.Posterior	see config$cv_posterior in estimate_R
plot	Not used anymore, only there for compatibility
leg.pos	Not used anymore, only there for compatibility

---

estimate_advantage *Jointly estimate the instantaneous reproduction number for a reference pathogen/strain/variant and the relative transmissibility of a "new" pathogen/strain/variant*

Description

Jointly estimate the instantaneous reproduction number for a reference pathogen/strain/variant and the relative transmissibility of a "new" pathogen/strain/variant

Usage

```
estimate_advantage(
  incid,
  si_distr,
  priors = default_priors(),
  mcmc_control = default_mcmc_controls(),
  t_min = NULL,
  t_max = nrow(incid),
  seed = NULL,
  incid_imported = NULL,
  precompute = TRUE,
  reorder_incid = TRUE
)
```

Arguments

incid	a multidimensional array containing values of the incidence for each time step (1st dimension), location (2nd dimension) and pathogen/strain/variant (3rd dimension)
si_distr	a matrix with two columns, each containing the probability mass function for the discrete serial interval for each of the two pathogen/strain/variants, starting with the probability mass function for day 0 in the first row, which should be 0. each column in the matrix should sum to 1
priors	a list of prior parameters (shape and scale of a gamma distribution) for epsilon and R; can be obtained from the function 'default_priors'. The prior for R is assumed to be the same for all time steps and all locations
mcmc_control	a list of default MCMC control parameters, as obtained for example from function 'default_mcmc_controls'
t_min	an integer > 1 giving the minimum time step to consider in the estimation. The NULL, t_min is calculated using the function `compute_si_cutoff` which gets the maximum (across all variants) of the 95th percentile of the SI distribution.
t_max	an integer >'t_min' and <='nrow(incid)' giving the maximum time step to consider in the estimation. Default value is 'nrow(incid)'.
seed	a numeric value used to fix the random seed
incid_imported	an optional multidimensional array containing values of the incidence of imported cases for each time step (1st dimension), location (2nd dimension) and pathogen/strain/variant (3rd dimension). 'incid - incid_imported' is therefore the incidence of locally infected cases. If 'incid_imported' is NULL this means there are no known imported cases and all cases other than on those from the first time step will be considered locally infected.
precompute	a boolean (defaulting to TRUE) deciding whether to precompute quantities or not. Using TRUE will make the algorithm faster
reorder_incid	a boolean (defaulting to TRUE) deciding whether the incidence array can be internally reordered during the estimation of the transmission advantage. If TRUE, the most transmissible pathogen/strain/variant is temporarily assigned to [,,1] of the incidence array. We recommend the default value of TRUE as we find this to stabilise inference.

Value

A list with the following elements.

1. ‘epsilon‘ is a matrix containing the MCMC chain (thinned and after burnin) for the relative transmissibility of the "new" pathogen/strain/variant(s) compared to the reference pathogen/strain/variant. Each row in the matrix is a "new" pathogen/strain/variant and each column an iteration of the MCMC.
2. ‘R‘ is an array containing the MCMC chain (thinned and after burnin) for the reproduction number for the reference pathogen/strain/variant. The first dimension of the array is time, the second location, and the third iteration of the MCMC.
3. ‘convergence‘ is a logical vector based on the results of the Gelman-Rubin convergence diagnostic. Each element in ‘convergence‘ takes a value of TRUE when the MCMC for the corresponding epsilon has converged within the number of iterations specified and FALSE otherwise.
4. ‘diag‘ is a nested list of the point estimate and upper confidence limits of the Gelman-Rubin convergence diagnostics (as implemented in coda). The length of ‘diag‘ is equal to the number of rows in ‘epsilon‘. Each element of ‘diag‘ is a list of length 2 where the first element is called ‘psrf‘ and is a named list of the point estimate and upper confidence limits. The second elemnent is NULL and can be ignored.

Examples

```
n_v <- 2
n_loc <- 3 # 3 locations
T <- 100 # 100 time steps
priors <- default_priors()
# constant incidence 10 per day everywhere
incid <- array(10, dim = c(T, n_loc, n_v))
# arbitrary serial interval, same for both variants
w_v <- c(0, 0.2, 0.5, 0.3)
si_distr <- cbind(w_v, w_v)

# Dummy initial values for the MCMC
R_init <- matrix(5, nrow = T, ncol = n_loc)
R_init[1, ] <- NA # no estimates of R on first time step
epsilon_init <- 5
x <- estimate_advantage(incid, si_distr, priors)
# Plotting to check outputs
par(mfrow = c(2, 2))
plot(x$epsilon, type = "l",
     xlab = "Iteration", ylab = "epsilon")
# Compare with what we expect with constant incidence in all locations
abline(h = 1, col = "red")
plot(x$R[10, 1, ], type = "l",
     xlab = "Iteration", ylab = "R time 10 location 1")
abline(h = 1, col = "red")
plot(x$R[20, 2, ], type = "l",
     xlab = "Iteration", ylab = "R time 20 location 2")
abline(h = 1, col = "red")
plot(x$R[30, 3, ], type = "l",
```

```
        xlab = "Iteration", ylab = "R time 30 location 3")
```

---

estimate_R *Estimated Instantaneous Reproduction Number*

Description

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

Usage

```
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),
  dt = 1L,
  dt_out = 7L,
  recon_opt = "naive",
  iter = 10L,
  tol = 1e-06,
  grid = list(precision = 0.001, min = -1, max = 1),
  backimputation_window = 0
)
```

Arguments

incid	One of the following • A vector (or a dataframe with a single column) of non-negative integers containing the incidence time series; these can be aggregated at any time unit as specified by argument dt • A dataframe of non-negative integers with either i) incid$I containing the total incidence, or ii) two columns, so that incid$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 Note that the cases from the first time step are always all assumed to be imported cases.
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 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. If the dates are known exactly use 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. If the dates are known exactly use 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

`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`.

`dt` length of temporal aggregations of the incidence data. This should be an integer or vector of integers. If a vector, this can either match the length of the incidence data supplied, or it will be recycled. For example, `dt = c(3L, 4L)` would correspond to alternating incidence aggregation windows of 3 and 4 days. The default value is 1 time unit (typically day).

`dt_out` length of the sliding windows used for R estimates (integer, 7 time units (typically days) by default). Only used if `dt > 1`; in this case this will superseed config$t_start and config$t_end, see `estimate_R_agg`.

`recon_opt` one of "naive" or "match", see `estimate_R_agg`.

`iter` number of iterations of the EM algorithm used to reconstruct incidence at 1-time-unit intervals(integer, 10 by default). Only used if `dt > 1`, see `estimate_R_agg`.

`tol` tolerance used in the convergence check (numeric, 1e-6 by default), see `estimate_R_agg`.

`grid` named list containing "precision", "min", and "max" which are used to define a grid of growth rate parameters that are used inside the EM algorithm used to reconstruct incidence at 1-time-unit intervals. Only used if `dt > 1`, see `estimate_R_agg`.

`backimputation_window` Length of the window used to impute incidence prior to the first reported cases. The default value is 0, meaning that no back-imputation is performed. If a positive integer is provided, the incidence is imputed for the first `backimputation_window` time units.

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.

R is estimated within a Bayesian framework, using a Gamma distributed prior, with mean and standard deviation which can be set using the ‘mean_prior‘ and ‘std_prior‘ arguments within the ‘make_config‘ function, which can then be used to specify ‘config‘ in the ‘estimate_R‘ function. Default values are a mean prior of 5 and standard deviation of 5. This was set to a high prior value with large uncertainty so that if one estimates R to be below 1, the result is strongly data-driven.

R is estimated on time windows specified through the ‘config‘ argument. These can be overlapping or not (see ‘make_config‘ function and vignette for examples).

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`)
- I_imputed: the time series of incidence of imputed cases
- 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

Cori, A. et al. A new framework and software to estimate time-varying reproduction numbers during epidemics (AJE 2013). Wallinga, J. and P. Teunis. Different epidemic curves for severe acute respiratory syndrome reveal similar impacts of control measures (AJE 2004). Reich, N.G. et al. Estimating incubation period distributions with coarse data (Statis. Med. 2009)

See Also

- `make_config` for general settings of the estimation
- `discr_si` to build serial interval distributions
- `sample_posterior_R` to draw samples of R values from the posterior distribution from the output of `estimate_R()`

Examples

```
## 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(0,1,1,2,1,3,4,5,5,5,5,4,4,26,6,7,9)
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)
## the second plot produced shows, at each each day,
## the estimate of the reproduction number over the 7-day window
## finishing on that day.

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

## 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.

## Example with back-imputation:
## here we use the first 6 days of incidence to impute cases that preceded
## the first reported cases:

res_bi <- estimate_R(incid = Flu2009$incidence,
                 method = "parametric_si",
                 backimputation_window = 6,
                 config = make_config(list(
                      mean_si = 2.6,
```

```
                      std_si = 1,
                      t_start = t_start,
                      t_end = t_end)))
plot(res_bi, "R")

## We can see that early estimates of R are lower when back-imputation is
## used, even though the difference is marginal in this case.

## 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")

################
mcmc_control <- make_mcmc_control(
  burnin = 1000, # first 1000 iterations discarded as burn-in
  thin = 10, # every 10th iteration will be kept, the rest discarded
  seed = 1) # set the seed to make the process reproducible

R_si_from_data <- estimate_R(
  incid = MockRotavirus$incidence,
  method = "si_from_data",
  si_data = MockRotavirus$si_data, # symptom onset data
  config = make_config(si_parametric_distr = "G", # gamma dist. for SI
     mcmc_control = mcmc_control,
     n1 = 500, # number of posterior samples of SI dist.
     n2 = 50, # number of posterior samples of Rt dist.
     seed = 2)) # set seed for reproducibility

## 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_agg` *Estimated Instantaneous Reproduction Number from coarsely aggregated data*

Description

Estimated Instantaneous Reproduction Number from coarsely aggregated data

Usage

```
estimate_R_agg(
  incid,
  dt = 7L,
  dt_out = 7L,
  iter = 10L,
  tol = 1e-06,
  recon_opt = "naive",
  config = make_config(),
  method = c("non_parametric_si", "parametric_si"),
  grid = list(precision = 0.001, min = -1, max = 1)
)
```

Arguments

`incid`	aggregated incidence data, supplied as a vector
`dt`	length of temporal aggregations of the incidence data. This should be an integer or vector of integers. If a vector, this will be recycled. For example, `dt = c(3L, 4L)` would correspond to alternating incidence aggregation windows of 3 and 4 days. The default value is 7 time units (typically days) - see details.

dt_out	length of the sliding windows used for R estimates (numeric, 7 time units (typically days) by default). Only used if dt > 1; in this case this will supersede config$t_start and config$t_end, see estimate_R().
iter	number of iterations of the EM algorithm (integer, 10 by default)
tol	tolerance used in the convergence check (numeric, 1e-6 by default)
recon_opt	one of "naive" or "match" (see details).
config	an object of class estimate_R_config, as returned by function make_config.
method	one of "non_parametric_si" or "parametric_si" (see details).
grid	named list containing "precision", "min", and "max" which are used to define a grid of growth rate parameters that are used inside the EM algorithm (see details). We recommend using the default values.

Details

Estimation of the time-varying reproduction number from temporally-aggregated incidence data. For full details about how Rt is estimated within this bayesian framework, see details in estimate_R.

Here, an expectation maximisation (EM) algorithm is used to reconstruct daily incidence from data that is provided on another timescale. In addition to the usual parameters required in estimate_R, the estimate_R_agg() function requires that the user specify two additional parameters: dt and dt_out. There are two other parameters that the user may modify, iter and grid, however we recommend that the default values are used.

- dt is the length of the temporal aggregations of the incidence data in days. This can be a single integer if the aggregations do not change. E.g. for weekly data, the user would supply dt = 7L. If the aggregation windows vary in length, a vector of integers can be supplied. If the aggregations have a repeating pattern, for instance, if incidence is always reported three times per week on the same day of the week, you can supply a vector such as: dt = c(2L,2L,3L). If the aggregations change over time, or do not have a repeating pattern, the user can supply a full vector of aggregations matching the length of the incidence data supplied.
- dt_out is the length of the sliding window used to estimate Rt from the reconstructed daily data. By default, Rt estimation uses weekly sliding time windows, however, we recommend that the sliding window is at least equal to the length of the longest aggregation window (dt) in the data.
- recon_opt specifies how to handle the initial incidence data that cannot be reconstructed by the EM algorithm (e.g. the incidence data for the aggregation window that precedes the first aggregation window that R can be estimated for). If ‘"naive"‘ is chosen, the naive disaggregation of the incidence data will be kept. If ‘"match"‘ is chosen, the incidence in the preceding aggregation window will be reconstructed by assuming that the growth rate matches that of the first estimation window. This is ‘"naive"‘ by default.
- iter is the number of iterations of the EM algorithm used to reconstruct the daily incidence data. By default, iter = 10L, which has been demonstrated to exceed the number of iterations necessary to reach convergence in simulation studies and analysis of real-world data by the package authors (manuscript in prep).
- grid is a named list containing "precision", "min", and "max" which are used to define a grid of growth rate parameters used inside the EM algorithm. The grid is used to convert reproduction number estimates for each aggregation of incidence data into growth rates, which are then used

to reconstruct the daily incidence data assuming exponential growth. The grid will auto-adjust if it is not large enough, so we recommend using the default values.

There are three stages of the EM algorithm:

- Initialisation. The EM algorithm is initialised with a naive disaggregation of the incidence data. For example, if there were 70 cases over the course of a week, this would be naively split into 10 cases per day.
- Expectation. The reproduction number is estimated for each aggregation window, except for the first aggregation window (as there is no past incidence data). This means that the earliest the incidence reconstruction can start is at least the first day of the second aggregation window. Additionally, if the disaggregated incidence in subsequent aggregation windows is too low to estimate the reproduction number, this will mean that the reconstruction will not start until case numbers are sufficiently high.
- Maximisation. The reproduction number estimates are then translated into growth rates for each aggregation window (Wallinga & Lipsitch, 2007) and used to reconstruct daily incidence data assuming exponential growth. The daily incidence is adjusted by a constant to ensure that if the daily incidence were to be re-aggregated, it would still sum to the original aggregated totals. The expectation and maximisation steps repeat iteratively until convergence.

The daily incidence that is reconstructed after the final iteration of the EM algorithm is then used to estimate Rt using the same process as the original `estimate_R` function, with sliding weekly time windows used as the default.

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 daily incidence reconstructed by the EM algorithm. For the initial incidence that cannot be reconstructed (e.g. the first aggregation window and aggregation windows where incidence is too low to estimate Rt - see details) then the incidence returned will be the naive disaggregation of the incidence data used to initialise the EM algorithm.
- 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)

Rebecca Nash <r.nash@imperial.ac.uk> and Anne Cori <a.cori@imperial.ac.uk>

References

Nash RK, Cori A, Nouvellet P. Estimating the epidemic reproduction number from temporally aggregated incidence data: a statistical modelling approach and software tool. medRxiv pre-print. (medRxiv pre-print)

Wallinga & Lipsitch. How generation intervals shape the relationship between growth rates and reproductive numbers (Proc Biol Sci 2007).

See Also

- estimate_R for details of the core function

Examples

```
## Example for constant aggregation windows e.g. weekly reporting

# Load data on SARS in 2003
data("SARS2003")

# this is daily data, but for this example we will aggregate it to weekly
# counts using the `aggregate_inc()` function
incid <- SARS2003$incidence
dt <- 7L
weekly_incid <- aggregate_inc(incid, dt)
si_distr <- SARS2003$si_distr

# estimate Rt using the default parameters (method "non_parametric_si")
method <- "non_parametric_si"
config <- make_config(list(si_distr = si_distr))
res_weekly <- estimate_R_agg(incid = weekly_incid,
                             dt = 7L, # aggregation window of the data
                             dt_out = 7L, # desired sliding window length
                             iter = 10L,
                             config = config,
                             method = method,
                             grid = list(precision = 0.001, min = -1, max = 1))

# Plot the result
plot(res_weekly)


## Example using repeating vector of aggregation windows e.g. for consistent
## reporting 3 times a week

# Using the SARS data again
data("SARS2003")

# For this example we will pretend data is being reported three times a week,
```

```
# in 2-day, 2-day, 3-day aggregations
incid <- SARS2003$incidence
dt <- c(2L,2L,3L)
agg_incid <- aggregate_inc(incid, dt)
si_distr <- SARS2003$si_distr

# estimate Rt using the default parameters (method "non_parametric_si")
method <- "non_parametric_si"
config <- make_config(list(si_distr = si_distr))
res_agg <- estimate_R_agg(incid = agg_incid,
                          dt = c(2L,2L,3L), # aggregation windows of the data
                          dt_out = 7L, # desired sliding window length
                          iter = 10L,
                          config = config,
                          method = method,
                          grid = list(precision = 0.001, min = -1, max = 1))

# Plot the result
plot(res_agg)

## Example using full vector of aggregation windows

dt <- rep(c(2L,2L,3L), length.out=length(agg_incid))
si_distr <- SARS2003$si_distr

# estimate Rt using the default parameters (method "non_parametric_si")
method <- "non_parametric_si"
config <- make_config(list(si_distr = si_distr))
res_agg <- estimate_R_agg(incid = agg_incid,
                          dt = dt, # aggregation windows of the data
                          dt_out = 7L, # desired sliding window length
                          iter = 10L,
                          config = config,
                          method = method,
                          grid = list(precision = 0.001, min = -1, max = 1))

# Plot the result
plot(res_agg)
```

`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 a objects 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

```
## 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)
```

`first_nonzero_incid` *Get the first day of non-zero incidence across all variants and locations.*

Description

Get the first day of non-zero incidence across all variants and locations.

Usage

```
first_nonzero_incid(incid)
```

Arguments

`incid`	a multidimensional array containing values of the incidence for each time step (1st dimension), location (2nd dimension) and pathogen/strain/variant (3rd dimension)

Details

For each variant, find the first day of non-zero incidence. The maximum of these is the smallest possible point at which estimation can begin.

Value

integer

Author(s)

Sangeeta Bhatia

`Flu1918` *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

Frost W. and E. Sydenstricker (1919) Influenza in Maryland: preliminary statistics of certain localities. Public Health Rep.(34): 491-504.

References

Cauchemez S. et al. (2011) Role of social networks in shaping disease transmission during a community outbreak of 2009 H1N1 pandemic influenza. Proc Natl Acad Sci U S A 108(7), 2825-2830.

Ferguson N.M. et al. (2005) Strategies for containing an emerging influenza pandemic in Southeast Asia. Nature 437(7056), 209-214.

Fraser C. et al. (2011) Influenza Transmission in Households During the 1918 Pandemic. Am J Epidemiol 174(5): 505-514.

Frost W. and E. Sydenstricker (1919) Influenza in Maryland: preliminary statistics of certain localities. Public Health Rep.(34): 491-504.

Vynnycky E. et al. (2007) Estimates of the reproduction numbers of Spanish influenza using morbidity data. Int J Epidemiol 36(4): 881-889.

White L.F. and M. Pagano (2008) Transmissibility of the influenza virus in the 1918 pandemic. PLoS One 3(1): e1498.

Examples

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

## estimate the reproduction number (method "non_parametric_si")
res <- estimate_R(Flu1918$incidence,
          method = "non_parametric_si",
          config = make_config(list(si_distr = Flu1918$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

Cauchemez S. et al. (2011) Role of social networks in shaping disease transmission during a community outbreak of 2009 H1N1 pandemic influenza. Proc Natl Acad Sci U S A 108(7), 2825-2830.

Morgan O.W. et al. (2010) Household transmission of pandemic (H1N1) 2009, San Antonio, Texas, USA, April-May 2009. Emerg Infect Dis 16: 631-637.

References

Cauchemez S. et al. (2011) Role of social networks in shaping disease transmission during a community outbreak of 2009 H1N1 pandemic influenza. Proc Natl Acad Sci U S A 108(7), 2825-2830.

Ferguson N.M. et al. (2005) Strategies for containing an emerging influenza pandemic in Southeast Asia. Nature 437(7056), 209-214.

Examples

```
## 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

Lessler J. et al. (2009) Outbreak of 2009 pandemic influenza A (H1N1) at a New York City school. New Eng J Med 361: 2628-2636.

References

Lessler J. et al. (2009) Outbreak of 2009 pandemic influenza A (H1N1) at a New York City school. New Eng J Med 361: 2628-2636.

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 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)
```

`get_shape_epsilon` *Precompute shape of posterior distribution for epsilon*

Description

Precompute shape of posterior distribution for epsilon

Usage

```
get_shape_epsilon(incid, lambda, priors, t_min = 2L, t_max = nrow(incid))
```

Arguments

`incid`	a multidimensional array containing values of the (local) incidence for each time step (1st dimension), location (2nd dimension) and pathogen/strain/variant (3rd dimension)
`lambda`	a multidimensional array containing values of the overall infectivity for each time step (1st dimension), location (2nd dimension) and pathogen/strain/variant (3rd dimension). The overall infectivity for a given location and pathogen/strain/variant represents the sum of the incidence for that location and that pathogen/strain/variant at all previous time steps, weighted by the current infectivity of those past incident cases. It can be calculated from the incidence 'incid' and the distribution of the serial interval using function 'compute_lambda')
`priors`	a list of prior parameters (shape and scale of a gamma distribution) for epsilon and R; can be obtained from the function 'default_priors'. The prior for R is assumed to be the same for all time steps and all locations
`t_min`	an integer >1 giving the minimum time step to consider in the estimation. Default value is 2 (as the estimation is conditional on observations at time step 1 and can therefore only start at time step 2).
`t_max`	an integer >'t_min' and <='nrow(incid)' giving the maximum time step to consider in the estimation. Default value is 'nrow(incid)'.

Value

a value or vector of values of the shape of the posterior distribution of epsilon for each of the non reference variants

Examples

```
n_loc <- 4 # 4 locations
n_v <- 3 # 3 strains
T <- 100 # 100 time steps
priors <- default_priors()
# constant incidence 10 per day everywhere
```

```
incid <- array(10, dim = c(T, n_loc, n_v))
incid <- process_I_multivariant(incid)
# arbitrary serial interval, same for both variants
w_v <- c(0, 0.2, 0.5, 0.3)
si_distr <- cbind(w_v, w_v, w_v)
lambda <- compute_lambda(incid, si_distr)
get_shape_epsilon(incid$local, lambda, priors)
```

`get_shape_R_flat` *Precompute shape of posterior distribution for R*

Description

Precompute shape of posterior distribution for R

Usage

```
get_shape_R_flat(incid, priors, t_min = 2L, t_max = nrow(incid))
```

Arguments

`incid`	a multidimensional array containing values of the (local) incidence for each time step (1st dimension), location (2nd dimension) and pathogen/strain/variant (3rd dimension)
`priors`	a list of prior parameters (shape and scale of a gamma distribution) for epsilon and R; can be obtained from the function 'default_priors'. The prior for R is assumed to be the same for all time steps and all locations
`t_min`	an integer >1 giving the minimum time step to consider in the estimation. Default value is 2 (as the estimation is conditional on observations at time step 1 and can therefore only start at time step 2).
`t_max`	an integer >'t_min' and <='nrow(incid)' giving the maximum time step to consider in the estimation. Default value is 'nrow(incid)'.

Value

a vector of the shape of the posterior distribution of R for each time step t and each location l (stored in element (l-1)*(t_max - t_min + 1) + t of the vector)

Examples

```
n_v <- 2
n_loc <- 3 # 3 locations
T <- 100 # 100 time steps
priors <- default_priors()
# constant incidence 10 per day everywhere
incid <- array(10, dim = c(T, n_loc, n_v))
get_shape_R_flat(incid, priors)
```

`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. If the dates are known exactly use 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. If the dates are known exactly use 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

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)
converg_diag_clever
converg_diag_naive


## End(Not run)
```

---

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.

Usage

```
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.

——————— method "non_parametric_si" ———————-

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

——————— method "parametric_si" ———————

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" ———————

Method "uncertain_si" allows accounting for uncertainty on the serial interval distribution as described in Cori et al. AJE 2013. We allow the mean μ and standard deviation σ of the serial interval to vary according to truncated normal distributions. We sample n1 pairs of mean and standard deviations, $(\mu^{(1)}, \sigma^{(1)}), ..., (\mu^{(n_2)}, \sigma^{(n_2)})$, 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 n2 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 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" ———————

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. If the dates are known exactly use 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. If the dates are known exactly use 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 n1. For each element in the posterior sample of serial interval distribution, we then draw a sample of size n2 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 × n2 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_sample" ———————-

Method "si_from_sample" also allows accounting for uncertainty on the serial interval distribution. 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,
```

```
                            method = method,
                            si_data = MockRotavirus$si_data,
                            config = make_config(
                        list(si_parametric_distr = "G",
                        mcmc_control = make_mcmc_control(burnin = 1000,
                        thin = 10, seed = 1),
                        n1 = 500,
                        n2 = 50,
                        seed = 2)))
plot(R_si_from_data)

## End(Not run)
```

`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 fom 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

```
## 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

Groendyke C. et al. (2011) Bayesian Inference for Contact Networks Given Epidemic Data. Scandinavian Journal of Statistics 38(3): 600-616.

References

Groendyke C. et al. (2011) Bayesian Inference for Contact Networks Given Epidemic Data. Scandinavian Journal of Statistics 38(3): 600-616.

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.
```

mers_2014_15 *Data on Middle East Respiratory Syndrome (MERS) in Saudi Arabia.*

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

Cauchemez S, Nouvellet P, Cori A, Jombart T, Garske T, Clapham H, Moore S, Linden Mills H, Salje H, Collins C, et al. 2016. Unraveling the drivers of MERS-CoV transmission. Proc Natl Acad Sci 113: 9081-9086.

Examples

```
## 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

```
## 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)
```

Arguments

`incid`	One of the following • A vector (or a dataframe with a single column) of non-negative integers containing an incidence time series • A dataframe of non-negative integers with 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). Note that the cases from the first time step are always all assumed to be imported cases.
`si_distr`	Vector of probabilities giving the discrete distribution of the serial interval.

Details

The overall infectivity λ_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), weigthed 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 λ_t at each time step

Author(s)

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

References

Cori, A. et al. A new framework and software to estimate time-varying reproduction numbers during epidemics (AJE 2013).

See Also

discr_si, estimate_R

Examples

```
## 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

```
## S3 method for class 'estimate_R'
plot(
  x,
  what = c("all", "incid", "R", "SI"),
  plot_theme = "v2",
  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')).
plot_theme	A string specifying whether to use the original plot theme (plot_theme = "original") or an alternative plot theme (plot_theme = "v2"). The plot_theme is "v2" by default.
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

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; v2 theme by Rebecca Nash

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
## (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,
    n_sim = 10
  )
)

## 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)
```

`process_I_multivariant` *Process incidence input for multivariant analyses with estimate_advantage*

Description

Process incidence input for multivariant analyses with estimate_advantage

Usage

```
process_I_multivariant(incid, incid_imported = NULL)
```

Arguments

incid	a multidimensional array containing values of the incidence for each time step (1st dimension), location (2nd dimension) and pathogen/strain/variant (3rd dimension)
incid_imported	an optional multidimensional array containing values of the incidence of imported cases for each time step (1st dimension), location (2nd dimension) and pathogen/strain/variant (3rd dimension). 'incid - incid_imported' is therefore the incidence of locally infected cases. If 'incid_imported' is NULL this means there are no known imported cases and all cases other than on those from the first time step will be considered locally infected.

Value

a list with two elements. 1) 'local' a multidimensional array containing values of the incidence of locally infected cases for each time step (1st dimension), location (2nd dimension) and pathogen/strain/variant (3rd dimension) 2) 'imported' a multidimensional array containing values of the incidence of imported cases for each time step (1st dimension), location (2nd dimension) and pathogen/strain/variant (3rd dimension)

Examples

```
n_v <- 3 # 3 variants
n_loc <- 1 # 1 location
T <- 100 # 100 time steps
# constant incidence 10 per day everywhere
incid <- array(10, dim = c(T, n_loc, n_v))
process_I_multivariant(incid)
```

---

sample_posterior_R *sample from the posterior R distribution*

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

```
## 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$R$`Median(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
```

SARS2003 *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

Cori A. et al. (2009) Temporal variability and social heterogeneity in disease transmission: the case of SARS in Hong Kong. PLoS Comput Biol 5(8) : e1000471.

References

Cori A. et al. (2009) Temporal variability and social heterogeneity in disease transmission: the case of SARS in Hong Kong. PLoS Comput Biol 5(8): e1000471.

Lipsitch M. et al. (2003) Transmission dynamics and control of severe acute respiratory syndrome. Science 300(5627): 1966-1970.

Examples

```
## 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.
```

Smallpox1972 *Data on the 1972 smallpox epidemic in Kosovo*

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

Fenner F. et al. (1988) Smallpox and its Eradication. Geneva, World Health Organization.

References

Fenner F. et al. (1988) Smallpox and its Eradication. Geneva, World Health Organization.

Gani R. and S. Leach (2001) Transmission potential of smallpox in contemporary populations. Nature 414(6865): 748-751.

Riley S. and N. M. Ferguson (2006) Smallpox transmission and control: spatial dynamics in Great Britain. Proc Natl Acad Sci U S A 103(33): 12637-12642.

Examples

```
## 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.
```

`wallinga_teunis` *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

```
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 contains 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

Cori, A. et al. A new framework and software to estimate time-varying reproduction numbers during epidemics (AJE 2013). Wallinga, J. and P. Teunis. Different epidemic curves for severe acute respiratory syndrome reveal similar impacts of control measures (AJE 2004).

See Also

`discr_si`, `estimate_R`

Examples

```
## 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

```
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

aggregate_inc, 3

backimpute_I, 3

check_cdt_samples_convergence, 4
coarse2estim, 5
compute_lambda, 7
compute_si_cutoff, 8
compute_t_min, 9
covid_deaths_2020_uk, 9

default_mcmc_controls, 10
default_priors, 11
discr_si, 12, *23*, *43*, *51*, *60*
DiscrSI, 12
draw_epsilon, 13
draw_R, 15

estimate_advantage, 17
estimate_R, *5, 6*, *13*, 20, *27*, *29*, *31*, *40*, *51–53*, *60*
estimate_R(), *36*, *48*
estimate_R_agg, *21*, 26
estimate_R_plots, 30, *52, 53*
EstimateR, 16

first_nonzero_incid, 32
Flu1918, 33
Flu2009, 34
flu_2009_NYC_school, 35

get_shape_epsilon, 37
get_shape_R_flat, 38
grob, *31*, *53*

incidence, *20*, *59*
init_mcmc_params, 39

make_config, *23*, 40
make_mcmc_control, 45
Measles1861, 47
mers_2014_15, 48
MockRotavirus, 49

overall_infectivity, *13*, 50
OverallInfectivity, 50

plot.estimate_R, *31*, 52
process_I_multivariant, 54

sample_posterior_R, *23*, 55
SARS2003, 56
Smallpox1972, 58

wallinga_teunis, *31*, *52, 53*, 59
WT, 61