Package 'fastbeta'

May 8, 2026

Version 0.5.1

Date 2025-10-25

Title Fast Approximation of Time-Varying Infectious Disease Transmission Rates

Description A fast method for approximating time-varying infectious disease transmission rates from disease incidence time series and other data, based on a discrete time approximation of an SEIR model, as analyzed in Jagan et al. (2020) <doi:10.1371/journal.pcbi.1008124>.

License GPL (>= 2)

URL https://github.com/davidearn/fastbeta

BugReports https://github.com/davidearn/fastbeta/issues

Depends R (>= 4.3)

Imports grDevices, graphics, stats

Suggests adaptivetau, deSolve, tools, utils

BuildResaveData no

NeedsCompilation yes

Author Mikael Jagan [aut, cre] (ORCID: <https://orcid.org/0000-0002-3542-2938>), David Earn [ctb] (ORCID: <https://orcid.org/0000-0003-3597-617X>)

Maintainer Mikael Jagan <jaganmn@mcmaster.ca>

Repository CRAN

Date/Publication 2025-10-26 19:10:02 UTC

Contents

`fastbeta-package` R *Package* **fastbeta**

Description

An R package for approximating time-varying infectious disease transmission rates from disease incidence time series and other data.

Details

The "main" function is `fastbeta`.

To render a list of available help topics, use `help(package = "fastbeta")`.

To report a bug or request a change, use `bug.report(package = "fastbeta")`.

Author(s)

Mikael Jagan <jaganmn@mcmaster.ca>

`cbind.ts` *Combine Time Series Objects*

Description

A replacement for the S3 method registered by package **stats** for generic function `cbind` and class `ts`. It sets column names following the rules employed by the internal default method for `cbind`. It exists to allow users to work around PR#18583, which shows that the method in package **stats** employs different and rather inconsistent rules. This function must be called directly, as it is not registered as a method for `cbind`.

Usage

```
## S3 method for class 'ts'
cbind(..., deparse.level = 1)
```

Arguments

`...`	vectors (including matrices), at least one inheriting from class `ts`.
`deparse.level`	an integer (0, 1, or 2) controlling how column names are determined for untagged arguments that are not matrices, following the internal default method for `cbind`.

Value

A "multiple time series" object, inheriting from class `mts`.

Examples

```
n <- 3L
x <- matrix(0, n, n, dimnames = list(NULL, LETTERS[seq_len(n)]))
y <- seq_len(n)
tsx <- ts(x)
tsy <- ts(y)
`~` <- identity
for (k in 0L:2L) {
cat(sprintf("k = %d:\n\n\n", k))
withAutoprint({
try(colnames(cbind   (   x,    y, deparse.level = k)))
try(colnames(cbind   ( tsx,  tsy, deparse.level = k)))
try(colnames(cbind.ts( tsx,  tsy, deparse.level = k)))
try(colnames(cbind   (~  x, ~  y, deparse.level = k)))
try(colnames(cbind   (~tsx, ~tsy, deparse.level = k)))
try(colnames(cbind.ts(~tsx, ~tsy, deparse.level = k)))
})
cat("\n\n")
}
rm(`~`)
```

---

deconvolve *Richardson-Lucy Deconvolution*

Description

Performs a modified Richardson-Lucy iteration for the purpose of estimating incidence from reported incidence or mortality, conditional on a reporting probability and on a distribution of the time to reporting.

Usage

```
deconvolve(x, prob = 1, delay = 1,
           start, x.pad = 0, tol = 1, iter.max = 32L, complete = FALSE)
```

Arguments

x	a numeric vector of length n giving the number of infections or deaths reported during n successive time intervals of equal duration.
prob	a numeric vector of length d+n, d=length(delay)-1, such that prob[d+i] is the probability that an infection during interval i is eventually reported. prob of length 1 is recycled.
delay	a numeric vector of positive length such that delay[j] is the probability that an infection during interval i is reported during interval i+j-1, given that it is eventually reported. Hence delay[j] is the probability of a delay by j-1 intervals and d=length(delay)-1 is the maximum delay. delay need not sum to 1 but must not sum to 0.
start	a numeric vector of length d+n, d=length(delay)-1, giving a starting value for the iteration. start[d+i] estimates the expected number of infections during interval i that are eventually reported. If missing, then a starting value is generated by padding x on the left and right with d-d0 and d0 elements equal to x.pad, choosing d0=which.max(delay)-1. Note that 0 is invariant under the iteration, hence it can be desirable to set x.pad to a small (relative to max(x)) positive number.
x.pad	a non-negative number, used only when start is unset; see above.
tol	a tolerance indicating a stopping condition; see the reference. Set to 0 if you want to perform no fewer than iter.max iterations.
iter.max	the maximum number of iterations.
complete	a logical flag indicating if the result should preserve successive updates to start.

Details

Do note that temporal alignment of x (length n) and y=deconvolve(x, ...)$value (length or number of rows d+n) requires padding x on the left, as in cbind(x=c(rep(NA, d), x), y); see the examples.

Value

A list with elements:

value	the result of updating start iter times then dividing by prob. If complete = TRUE, then value is a (d+n)-by-(1+iter) matrix containing start and the iter successive updates, each divided by prob.
chisq	the chi-squared statistics corresponding to value.
iter	the number of iterations performed.

subset(value, start == 0) is zero because zero is invariant under the iteration. If delay has l leading zeros and t trailing zeros, then head(value, t) and tail(value, l) are NaN due to divide-by-zero in the iteration. (Conceptually, x and delay provide no information about incidence during those intervals.)

References

Goldstein, E., Dushoff, J., Ma, J., Plotkin, J. B., Earn, D. J. D., & Lipsitch, M. (2009). Reconstructing influenza incidence by deconvolution of daily mortality time series. *Proceedings of the National Academy of Sciences U. S. A.*, *106*(51), 21825-21829. doi:10.1073/pnas.0902958106

Examples

```
##
## Example 1: simulation
##

set.seed(2L)
n <- 200L
d <- 50L
p <- 0.1
prob <- plogis(rlogis(d + n, location = qlogis(p), scale = 0.1))
delay <- c(0, diff(pgamma(0L:d, 12, 0.4)))

h <- function (x, a = 1, b = 1, c = 0) a * exp(-b * (x - c)^2)
ans <- floor(h(seq(-60, 60, length.out = d + n), a = 1000, b = 0.001))

x0 <- rbinom(d + n, ans, prob)
x <- tabulate(rep(1L:(d + n), x0) +
              sample(0L:d, size = sum(x0), replace = TRUE, prob = delay),
              d + n)[-(1L:d)]

str(D0 <- deconvolve(x, prob, delay, complete = FALSE))
str(D1 <- deconvolve(x, prob, delay, complete =  TRUE))

matplot(-(d - 1L):n,
        cbind(x0, c(rep(NA, d), x), prob * D0[["value"]], p * ans),
        type = c("p", "p", "p", "l"),
        col = c(1L, 1L, 2L, 4L), pch = c(16L, 1L, 16L, NA),
        lty = c(0L, 0L, 0L, 1L), lwd = c(NA, NA, NA, 3),
        xlab = "Time", ylab = "Count")
legend("topleft", NULL,
       c("actual", "actual+delay", "actual+delay+deconvolution", "p*h"),
       col = c(1L, 1L, 2L, 4L), pch = c(16L, 1L, 16L, NA),
       lty = c(0L, 0L, 0L, 1L), lwd = c(NA, NA, NA, 3),
       bty = "n")

plot(0L:D1[["iter"]], D1[["chisq"]],
     xlab = "Iterations", ylab = quote(chi^2))
abline(h = 1, lty = 2L)


##
## Example 2: application to pneumonia and influenza
##

data(pneumonia, package = "fastbeta")
x <- pneumonia[["series"]][["deaths"]]
```

```
delay <- pneumonia[["delay"]][["gpg"]]

n <- length(x)
d <- length(delay) - 1L
r <- 30L

D2 <- deconvolve(x = x, delay = delay, tol = 0, iter.max = r,
                 complete = TRUE)
stopifnot(D2[["iter"]] == r,
          identical(dim(D2[["value"]]), c(d + n, 1L + r)),
          length(D2[["chisq"]]) == 1L + r,
          min(D2[["chisq"]]) < 1)

## Subscript for the first, critical, and last values:
j2 <- c(1L, which.max(D2[["chisq"]] < 1), 1L + r)

matplot(x = seq(from = pneumonia[["series"]][1L, "date"] - d,
                by = 1, length.out = d + n),
        y = cbind(c(rep(NA, d), x), D2[["value"]][, j2]),
        type = "o",
        col = c(1L, 4L, 2L, 3L), pch = 1L, lty = 1L, lwd = 1,
        xlab = "1918", ylab = "deaths")
legend("topleft", NULL,
       c("observed", sprintf("after %d iterations", j2 - 1L)),
       col = c(1L, 4L, 2L, 3L), pch = 1L, lty = 1L, lwd = 1,
       bty = "n")
```

`fastbeta` *Estimate a Time-Varying Infectious Disease Transmission Rate*

Description

Generates a discrete approximation of a time-varying infectious disease transmission rate from an equally spaced disease incidence time series and other data.

Usage

```
fastbeta(series, sigma = 1, gamma = 1, delta = 0,
         m = 1L, n = 1L, init, ...)
```

Arguments

`series`	a "multiple time series" object, inheriting from class `mts`, with three columns storing ("parallel", equally spaced) time series of incidence, births, and the per capita natural mortality rate, in that order.
`sigma`, `gamma`, `delta`	non-negative numbers. `m*sigma`, `n*gamma`, and `delta` are the rates of removal from each latent, infectious, and recovered compartment.
`m`	a non-negative integer indicating a number of latent stages.

`n`	a positive integer indicating a number of infectious stages.
`init`	a numeric vector of length `1+m+n+1` giving an initial state with compartments ordered as (S, E, I, R).
`...`	optional arguments passed to `deconvolve`, if the first column of `series` represents *observed* incidence rather than actual or estimated incidence.

Details

The algorithm implemented by `fastbeta` is based on an SEIR model with

- m latent stages (E^i, $i = 1, \ldots, m$);
- n infectious stages (I^j, $j = 1, \ldots, n$);
- time-varying rates β, ν, and μ of transmission, birth, and natural death; and
- constant rates $m\sigma$, $n\gamma$, and δ of removal from each latent, infectious, and recovered compartment, where removal from the recovered compartment implies return to the susceptible compartment (loss of immunity).

It is derived by linearizing of the system of ordinary differential equations

$$
\begin{aligned}
\mathrm{d}S \quad /\mathrm{d}t &= \quad \delta R \quad - (\lambda(t) + \mu(t))S \quad + \nu(t) \\
\mathrm{d}E^1 \quad /\mathrm{d}t &= \lambda(t)S \quad - (\ m\sigma + \mu(t))E^1 \\
\mathrm{d}E^{i+1}/\mathrm{d}t &= \ m\sigma E^i \quad - (\ m\sigma + \mu(t))E^{i+1} \\
\mathrm{d}I^1 \quad /\mathrm{d}t &= \ m\sigma E^m - (\ \ n\gamma + \mu(t))I^1 \\
\mathrm{d}I^{j+1}/\mathrm{d}t &= \ \ n\gamma I^j \quad - (\ \ n\gamma + \mu(t))I^{j+1} \\
\mathrm{d}R \quad /\mathrm{d}t &= \ \ n\gamma I^n \quad - (\quad \delta + \mu(t))R
\end{aligned}
\qquad \lambda(t) = \beta(t) \sum_j I^j
$$

and substituting actual or estimated incidence and births for definite integrals of λS and ν. This procedure yields a system of linear difference equations from which one recovers a discrete approximation of β:

$$
\begin{aligned}
E^1_{t+1} &= [(1 - \tfrac{1}{2}(m\sigma + \mu_t))E^1_t \qquad\qquad\qquad + Z_{t+1} \qquad\quad]/[1 + \tfrac{1}{2}(m\sigma + \mu_{t+1})] \\
E^{i+1}_{t+1} &= [(1 - \tfrac{1}{2}(m\sigma + \mu_t))E^{i+1}_t + \tfrac{1}{2}m\sigma(E^i_t \ + E^i_{t+1}) \qquad\quad]/[1 + \tfrac{1}{2}(m\sigma + \mu_{t+1})] \\
I^1_{t+1} &= [(1 - \tfrac{1}{2}(\ n\gamma + \mu_t))I^1_t \quad + \tfrac{1}{2}m\sigma(E^m_t + E^m_{t+1}) \qquad\quad]/[1 + \tfrac{1}{2}(\ n\gamma + \mu_{t+1})] \\
I^{j+1}_{t+1} &= [(1 - \tfrac{1}{2}(\ n\gamma + \mu_t))I^{j+1}_t + \tfrac{1}{2}\ n\gamma(I^j_t \ + I^j_{t+1}\ ) \qquad\quad]/[1 + \tfrac{1}{2}(\ n\gamma + \mu_{t+1})] \\
R_{t+1} &= [(1 - \tfrac{1}{2}(\quad \delta + \mu_t))R_t \quad + \tfrac{1}{2}\ n\gamma(I^n_t \ + I^n_{t+1}\ ) \qquad\quad]/[1 + \tfrac{1}{2}(\quad \delta + \mu_{t+1})] \\
S_{t+1} &= [(1 - \tfrac{1}{2}(\qquad \mu_t))S_t \quad + \tfrac{1}{2}\quad \delta(R_t \ + R_{t+1}) - Z_{t+1} + B_{t+1}]/[1 + \tfrac{1}{2}(\qquad \mu_{t+1})]
\end{aligned}
\qquad \beta_t = (Z_t + Z_{t+1})/(2S_t \sum_j I
$$

where we use the notation

$$
X_t \sim X(t) : X = S, E^i, I^j, R, Z, B, \mu, \beta
\qquad
\begin{aligned}
Z(t) &= \int_{t-1}^{t} \lambda(s)S(s)\,\mathrm{d}s \\
B(t) &= \int_{t-1}^{t} \nu(s)\,\mathrm{d}s
\end{aligned}
$$

and it is understood that the independent variable t is a unitless measure of time relative to the spacing of the substituted time series of incidence and births.

Value

A "multiple time series" object, inheriting from class mts, with 1+m+n+1+1 columns (named S, E, I, R, and beta) storing the result of the iteration described in 'Details'. It is completely parallel to argument series, having the same tsp attribute.

References

Jagan, M., deJonge, M. S., Krylova, O., & Earn, D. J. D. (2020). Fast estimation of time-varying infectious disease transmission rates. *PLOS Computational Biology*, *16*(9), Article e1008124, 1-39. doi:10.1371/journal.pcbi.1008124

Examples

```
if (requireNamespace("adaptivetau")) withAutoprint({

data(seir.ts02, package = "fastbeta")
a <- attributes(seir.ts02)
str(seir.ts02)
plot(seir.ts02)

## We suppose that we have perfect knowledge of incidence,
## births, and the data-generating parameters
series <- cbind.ts(seir.ts02[, c("Z", "B")], mu = a[["mu"]](0))

args <- c(list(series = series),
          a[c("sigma", "gamma", "delta", "m", "n", "init")])
str(args)

X <- do.call(fastbeta, args)
str(X)
plot(X)

plot(X[, "beta"], ylab = "Transmission rate")
lines(a[["beta"]](time(X)), col = "red") # the "truth"

})
```

---

fastbeta.bootstrap *Parametric Bootstrapping*

Description

A simple wrapper around fastbeta using it to generate a "primary" estimate of a time-varying transmission rate and r bootstrap estimates. Bootstrap estimates are computed for incidence time series simulated using seir, with transmission rate defined as the linear interpolant of the primary estimate.

Usage

```
fastbeta.bootstrap(r,
                   series, sigma = 1, gamma = 1, delta = 0,
                   m = 1L, n = 1L, init, ...)
```

Arguments

r	a non-negative integer indicating a number of replications.
series	a "multiple time series" object, inheriting from class mts, with three columns storing ("parallel", equally spaced) time series of incidence, births, and the per capita natural mortality rate, in that order.
sigma, gamma, delta	non-negative numbers. m*sigma, n*gamma, and delta are the rates of removal from each latent, infectious, and recovered compartment.
m	a non-negative integer indicating a number of latent stages.
n	a positive integer indicating a number of infectious stages.
init	a numeric vector of length 1+m+n+1 giving an initial state with compartments ordered as (S, E, I, R).
...	optional arguments passed to seir and/or deconvolve. Both take optional arguments prob and delay. When prob is supplied but not delay, seir and deconvolve receive prob as is. When both are supplied, seir receives prob as is, whereas deconvolve receives prob augmented with length(delay)-1 ones.

Value

A "multiple time series" object, inheriting from class mts, with 1+r columns storing the one primary and r bootstrap estimates. It is completely parallel to argument series, having the same tsp attribute.

Examples

```
if (requireNamespace("adaptivetau")) withAutoprint({

data(seir.ts02, package = "fastbeta")
a <- attributes(seir.ts02)
str(seir.ts02)
plot(seir.ts02)

## We suppose that we have perfect knowledge of incidence,
## births, and the data-generating parameters
series <- cbind.ts(seir.ts02[, c("Z", "B")], mu = a[["mu"]](0))

args <- c(list(r = 100L, series = series),
          a[c("sigma", "gamma", "delta", "m", "n", "init")])
str(args)

R <- do.call(fastbeta.bootstrap, args)
```

```
str(R)
plot(R)
plot(R, level = 0.95)

})
```

`fastbeta.matrix` *Calculate Coefficient Matrix for Iteration Step*

Description

Calculates the coefficient matrix corresponding to one step of the iteration carried out by `fastbeta`:

```
y <- c(1, E, I, R, S)
for (pos in seq_len(nrow(series) - 1L)) {
    L <- fastbeta.matrix(pos, series, ...)
    y <- L %*% y
}
```

Usage

```
fastbeta.matrix(pos,
                series, sigma = 1, gamma = 1, delta = 0,
                m = 1L, n = 1L)
```

Arguments

`pos`	an integer indexing a row (but not the last row) of `series`.
`series`	a "multiple time series" object, inheriting from class `mts`, with three columns storing ("parallel", equally spaced) time series of incidence, births, and the per capita natural mortality rate, in that order.
`sigma`, `gamma`, `delta`	non-negative numbers. `m*sigma`, `n*gamma`, and `delta` are the rates of removal from each latent, infectious, and recovered compartment.
`m`	a non-negative integer indicating a number of latent stages.
`n`	a positive integer indicating a number of infectious stages.

Value

A lower triangular matrix of size `1+m+n+1+1`.

Examples

```
if (requireNamespace("adaptivetau")) withAutoprint({

data(seir.ts02, package = "fastbeta")
a <- attributes(seir.ts02); p <- length(a[["init"]])
str(seir.ts02)
plot(seir.ts02)

## We suppose that we have perfect knowledge of incidence,
## births, and the data-generating parameters
series <- cbind.ts(seir.ts02[, c("Z", "B")], mu = a[["mu"]](0))

args <- c(list(series = series),
          a[c("sigma", "gamma", "delta", "init", "m", "n")])
str(args)

X <- unclass(do.call(fastbeta, args))[, seq_len(p)]
colnames(X)
Y <- Y. <- cbind(1, X[, c(2L:p, 1L)], deparse.level = 2L)
colnames(Y)

args <- c(list(pos = 1L, series = series),
          a[c("sigma", "gamma", "delta", "m", "n")])
str(args)

L <- do.call(fastbeta.matrix, args)
str(L)
symnum(L != 0)

for (pos in seq_len(nrow(series) - 1L)) {
    args[["pos"]] <- pos
    L. <- do.call(fastbeta.matrix, args)
    Y.[pos + 1L, ] <- L. %*% Y.[pos, ]
}
stopifnot(all.equal(Y, Y.))

})
```

---

pneumonia *Pneumonia and Influenza Mortality in Philadelphia, PA, 1918*

Description

Time series of deaths due to pneumonia and influenza in Philadelphia, PA from September 1, 1918 to December 31, 1918, as recorded in the "Special Tables of Mortality" of the U.S. Census Bureau.

Usage

```
data(pneumonia, package = "fastbeta")
```

Format

A named list with 2 components, `series` and `delay`. `series` is a data frame with 122 rows and 2 variables:

date date of the record.

deaths count of deaths due to influenza and pneumonia.

`delay` is a data frame with 64 rows and 3 variables:

nday number of days from infection to death.

goldstein, gpg probabilities, not summing to 1 due to rounding and truncation; see 'Source'.

Source

A script generating the `pneumonia` object is available as `system.file("scripts", "pneumonia.R", package = "fastbeta")`.

`series` is obtained from Table 2 in the first reference.

`delay` is obtained from the remaining references. Component `goldstein` is obtained from Figure 1 in the Supporting Information of Goldstein et al. (2009). Component gpg is obtained from the convolution of two gamma distributions, one for the time from infection to symptom onset fitted to Figure 1 in Moser et al. (1979) and another for the time from symptom onset to death fitted to Chart 2 in Keeton & Cushman (1918).

References

U.S. Census Bureau (1920). *Special Tables of Mortality from Influenza and Pneumonia: Indiana, Kansas, and Philadelphia, PA*. U.S. Department of Commerce. https://www.census.gov/library/publications/1920/demo/1918-mortality-special-tables.html

Goldstein, E., Dushoff, J., Ma, J., Plotkin, J. B., Earn, D. J. D., & Lipsitch, M. (2009). Reconstructing influenza incidence by deconvolution of daily mortality time series. *Proceedings of the National Academy of Sciences U. S. A.*, *106*(51), 21825-21829. doi:10.1073/pnas.0902958106

Moser, M. R., Bender, T. R., Margolis, H. S., Noble, G. R., Kendal, A. P., & Ritter, D. G. (1979). An outbreak of influenza aboard a commercial airliner. *Americal Journal of Epidemiology*, *110*(1), 1-6. doi:10.1093/oxfordjournals.aje.a112781

Keeton, R. W. & Cushman, A. B. (1918). The influenza epidemic in Chicago: the disease as a type of toxemic shock. *Journal of the Americal Medical Association*. *71*(24), 1962-1967.

Examples

```
data(pneumonia, package = "fastbeta")
str(pneumonia)

plot(deaths ~ date, pneumonia$series, xlab = "1918")

plot(goldstein/sum(goldstein) ~ nday, pneumonia$delay, type = "o",
     lty = 2, pch = 1, xlab = "days", ylab = "probability")
lines(gpg/sum(gpg) ~ nday, pneumonia$delay, type = "o",
      lty = 1, pch = 16)
```

ptpi *Peak to Peak Iteration*

Description

Approximates the state of an SEIR model at a reference time from an equally spaced, T-periodic incidence time series and other data. The algorithm relies on a strong assumption: that the incidence time series was generated by the asymptotic dynamics of an SEIR model admitting a locally stable, T-periodic attractor. Hence do interpret with care.

Usage

```
ptpi(series, sigma = 1, gamma = 1, delta = 0,
     m = 1L, n = 1L, init,
     start = tsp(series)[1L], end = tsp(series)[2L],
     tol = 1e-03, iter.max = 32L,
     backcalc = FALSE, complete = FALSE, ...)
```

Arguments

`series`	a "multiple time series" object, inheriting from class `mts`, with three columns storing ("parallel", equally spaced) time series of incidence, births, and the per capita natural mortality rate, in that order.
`sigma, gamma, delta`	non-negative numbers. `m*sigma`, `n*gamma`, and `delta` are the rates of removal from each latent, infectious, and recovered compartment.
`m`	a non-negative integer indicating a number of latent stages.
`n`	a positive integer indicating a number of infectious stages.
`init`	a numeric vector of length `1+m+n+1` giving an initial guess for the state at time `start`.
`start, end`	start and end times for the iteration, whose difference should be approximately equal to an integer number of periods. One often chooses the time of the first peak in the incidence time series and the time of the last peak in phase with the first.
`tol`	a tolerance indicating a stopping condition; see 'Details'.
`iter.max`	the maximum number of iterations.
`backcalc`	a logical indicating if the state at time `tsp(series)[1]` should be back-calculated from the state at time `start` if that is later.
`complete`	a logical indicating if intermediate states should be recorded in an array. Useful mainly for didactic or diagnostic purposes.
`...`	optional arguments passed to `deconvolve`, if the first column of `series` represents *observed* incidence rather than actual or estimated incidence.

Details

ptpi can be understood as an iterative application of fastbeta to a subset of series. The basic algorithm can be expressed in R code as:

```
w <- window(series, start, end); i <- nrow(s); j <- seq_along(init)
diff <- Inf; iter <- 0L
while (diff > tol && iter < iter.max) {
    init. <- init
    init <- fastbeta(w, sigma, gamma, delta, m, n, init)[i, j]
    diff <- sqrt(sum((init - init.)^2) / sum(init.^2))
    iter <- iter + 1L
}
value <- init
```

Back-calculation involves solving a linear system of equations; the back-calculated result can mislead if the system is ill-conditioned.

Value

A list with elements:

value	an approximation of the state at time start or at time tsp(series)[1L], depending on backcalc.
diff	the relative difference between the last two approximations.
iter	the number of iterations performed.
x	if complete = TRUE, then a "multiple time series" object, inheriting from class mts, with dimensions c(nrow(w), length(value), iter), where w = window(series, start, end). x[, , k] contains the state at each time(w) in iteration k.

References

Jagan, M., deJonge, M. S., Krylova, O., & Earn, D. J. D. (2020). Fast estimation of time-varying infectious disease transmission rates. *PLOS Computational Biology*, *16*(9), Article e1008124, 1-39. doi:10.1371/journal.pcbi.1008124

Examples

```
if (requireNamespace("deSolve")) withAutoprint({

data(seir.ts01, package = "fastbeta")
a <- attributes(seir.ts01); p <- length(a[["init"]])
str(seir.ts01)
plot(seir.ts01)

## We suppose that we have perfect knowledge of incidence,
## births, and the data-generating parameters, except for
## the initial state, which we "guess"
series <- cbind.ts(seir.ts01[, c("Z", "B")], mu = a[["mu"]](0))
```

```
plot(series[, "Z"])
start <- 23; end <- 231
abline(v = c(start, end), lty = 2)

set.seed(0L)
args <- c(list(series = series),
          a[c("sigma", "gamma", "delta", "m", "n", "init")],
          list(start = start, end = end, complete = TRUE))
init <- seir.ts01[which.min(abs(time(seir.ts01) - start)), seq_len(p)]
args[["init"]] <- init * rlnorm(p, 0, 0.1)
str(args)

L <- do.call(ptpi, args)
str(L)

S <- L[["x"]][, "S", ]
plot(S, plot.type = "single")
lines(seir.ts01[, "S"], col = "red", lwd = 4) # the "truth"
abline(h = L[["value"]]["S"], v = start, col = "blue", lwd = 4, lty = 2)

## Relative error
L[["value"]] / init - 1

})
```

`seir` *Simulate Infectious Disease Time Series*

Description

Simulates incidence time series based on an SEIR model with user-defined forcing and a simple model for observation error.

Note that simulation code depends on availability of suggested packages **adaptivetau** and **deSolve**. If the dependency cannot be loaded then an error is signaled.

Usage

```
seir(length.out = 1L,
     beta, nu = function (t) 0, mu = function (t) 0,
     sigma = 1, gamma = 1, delta = 0,
     m = 1L, n = 1L, init,
     stochastic = TRUE, prob = 1, delay = 1,
     aggregate = FALSE, useCompiled = TRUE, ...)

## A basic wrapper for the m=0L case:

 sir(length.out = 1L,
     beta, nu = function (t) 0, mu = function (t) 0,
     gamma = 1, delta = 0,
```

```
n = 1L, init,
stochastic = TRUE, prob = 1, delay = 1,
aggregate = FALSE, useCompiled = TRUE, ...)
```

Arguments

length.out	a non-negative integer indicating the time series length.
beta, nu, mu	functions of one or more arguments returning transmission, birth, and natural death rates at the time point indicated by the first argument. Arguments after the first must be strictly optional. The functions need not be vectorized.
sigma, gamma, delta	non-negative numbers. m*sigma, n*gamma, and delta are the rates of removal from each latent, infectious, and recovered compartment.
m	a non-negative integer indicating a number of latent stages.
n	a positive integer indicating a number of infectious stages.
init	a numeric vector of length 1+m+n+1 giving an initial state with compartments ordered as (S, E, I, R).
stochastic	a logical indicating if the simulation should be stochastic; see 'Details'.
prob	a numeric vector of length n such that prob[i] is the probability that an infection during interval i is eventually observed. prob of length 1 is recycled.
delay	a numeric vector of positive length such that delay[i] is the probability that an infection during interval j is observed during interval j+i-1, given that it is eventually observed. delay need not sum to 1 but must not sum to 0.
aggregate	a logical indicating if latent and infectious compartments should be aggregated.
useCompiled	a logical indicating if derivatives should be computed by compiled C functions rather than by R functions (which *may* be *byte*-compiled). Set to FALSE only if TRUE seems to cause problems, and in that case please report the problems with bug.report(package = "fastbeta").
...	optional arguments passed to lsoda (directly) or ssa.adaptivetau (via its list argument tl.params), depending on stochastic.

Details

Simulations are based on an SEIR model with

- m latent stages (E^i, $i = 1, \ldots, m$);
- n infectious stages (I^j, $j = 1, \ldots, n$);
- time-varying rates β, ν, and μ of transmission, birth, and natural death; and
- constant rates $m\sigma$, $n\gamma$, and δ of removal from each latent, infectious, and recovered compartment, where removal from the recovered compartment implies return to the susceptible compartment (loss of immunity).

`seir(stochastic = FALSE)` works by numerically integrating the system of ordinary differential equations

$$
\begin{aligned}
\mathrm{d}S &/\mathrm{d}t = \quad \delta R - (\lambda(t) + \mu(t))S + \nu(t) \\
\mathrm{d}E^1 &/\mathrm{d}t = \lambda(t)S - (\, m\sigma + \mu(t))E^1 \\
\mathrm{d}E^{i+1} &/\mathrm{d}t = m\sigma E^i - (\, m\sigma + \mu(t))E^{i+1} \\
\mathrm{d}I^1 &/\mathrm{d}t = m\sigma E^m - (\; n\gamma + \mu(t))I^1 \\
\mathrm{d}I^{j+1} &/\mathrm{d}t = n\gamma I^j - (\; n\gamma + \mu(t))I^{j+1} \\
\mathrm{d}R &/\mathrm{d}t = n\gamma I^n - (\quad \delta + \mu(t))R
\end{aligned}
\qquad \lambda(t) = \beta(t) \sum_j I^j
$$

where it is understood that the independent variable t is a unitless measure of time relative to an observation interval. To get time series of incidence and births, the system is augmented with two equations describing *cumulative* incidence and births

$$
\begin{aligned}
\mathrm{d}Z/\mathrm{d}t &= \lambda(t)S \\
\mathrm{d}B/\mathrm{d}t &= \nu(t)
\end{aligned}
$$

and the *augmented* system is numerically integrated. Observed incidence is simulated from incidence by scaling the latter by `prob` and convolving the result with `delay`.

`seir(stochastic = TRUE)` works by simulating a Markov process corresponding to the augmented system, as described in the reference. Observed incidence is simulated from incidence by binning binomial samples taken with probabilities `prob` over future observation intervals according to multinomial samples taken with probabilities `delay`.

Value

A "multiple time series" object, inheriting from class `mts`. Beneath the class, it is a `length.out`-by-`(1+m+n+1+2)` numeric matrix with columns `S`, `E`, `I`, `R`, `Z`, and `B`, where `Z` and `B` specify incidence and births as the number of infections and births since the previous time point.

If `prob` or `delay` is not missing, then there is an additional column `Z.obs` specifying *observed* incidence as the number of infections observed since the previous time point. The first `length(delay)` elements of this column contain partial counts.

References

Cao, Y., Gillespie, D. T., & Petzold, L. R. (2007). Adaptive explicit-implicit tau-leaping method with automatic tau selection. *Journal of Chemical Physics*, *126*(22), Article 224101, 1-9. doi:10.1063/1.2745299

See Also

`seir.auxiliary`, `seir.library`.

Examples

```
if (requireNamespace("adaptivetau")) withAutoprint({

beta <- function (t, a = 1e-01, b = 1e-05) b * (1 + a * sinpi(t / 26))
```

```
nu   <- function (t) 1e+03
mu   <- function (t) 1e-03

sigma <- 0.5
gamma <- 0.5
delta <- 0

init <- c(S = 50200, E = 1895, I = 1892, R = 946011)

length.out <- 250L
prob <- 0.1
delay <- diff(pgamma(0:8, 2.5))

set.seed(0L)
X <- seir(length.out, beta, nu, mu, sigma, gamma, delta, init = init,
          prob = prob, delay = delay, epsilon = 0.002)
##                                                ^^^^^
## default epsilon = 0.05 allows too big leaps => spurious noise
##
str(X)
plot(X)

r <- 10L
Y <- do.call(cbind.ts, replicate(r, simplify = FALSE,
seir(length.out, beta, nu, mu, sigma, gamma, delta, init = init,
     prob = prob, delay = delay, epsilon = 0.002)[, "Z.obs"]))
str(Y)
plot(window(Y, start = tsp(Y)[1L] + length(delay) / tsp(Y)[3L]),
     ##        ^^^^^
     ## discards points showing edge effects due to 'delay'
     ##
     plot.type = "single", col = seq_len(r), ylab = "Case reports")

})
```

`seir.auxiliary` *Auxiliary Functions for the SEIR Model without Forcing*

Description

Calculate the basic reproduction number, endemic equilibrium, and Jacobian matrix of the SEIR model without forcing.

Usage

```
seir.R0      (beta, nu = 0, mu = 0, sigma = 1, gamma = 1, delta = 0,
              m = 1L, n = 1L, N = 1)
seir.ee      (beta, nu = 0, mu = 0, sigma = 1, gamma = 1, delta = 0,
              m = 1L, n = 1L, N = 1)
seir.jacobian(beta, nu = 0, mu = 0, sigma = 1, gamma = 1, delta = 0,
              m = 1L, n = 1L)
```

Arguments

`beta`, `nu`, `mu`, `sigma`, `gamma`, `delta`	non-negative numbers. `beta`, `nu`, and `mu` are the rates of transmission, birth, and natural death. `m*sigma`, `n*gamma`, and `delta` are the rates of removal from each latent, infectious, and recovered compartment.
`m`	a non-negative integer indicating a number of latent stages.
`n`	a positive integer indicating a number of infectious stages.
`N`	a non-negative number indicating a population size for the `(nu == 0 && mu == 0)` case.

Details

If $\mu, \nu = 0$, then the basic reproduction number is computed as

$$\mathcal{R}_0 = N\beta/\gamma$$

and the endemic equilibrium is computed as

$$\begin{bmatrix} S \\ E^i \\ I^j \\ R \end{bmatrix} = \begin{bmatrix} \gamma/\beta \\ w\delta/(m\sigma) \\ w\delta/(n\gamma) \\ w \end{bmatrix}$$

where w is chosen so that the sum is N.

If $\mu, \nu > 0$, then the basic reproduction number is computed as

$$\mathcal{R}_0 = \nu\beta a^{-m}(1 - b^{-n})/\mu^2$$

and the endemic equilibrium is computed as

$$\begin{bmatrix} S \\ E^i \\ I^j \\ R \end{bmatrix} = \begin{bmatrix} \mu a^m/(\beta(1 - b^{-n})) \\ w a^{m-i} b^n(\delta + \mu)/(m\sigma) \\ w b^{n-j}(\delta + \mu)/(n\gamma) \\ w \end{bmatrix}$$

where w is chosen so that the sum is ν/μ, the population size at equilibrium, and $a = 1 + \mu/(m\sigma)$ and $b = 1 + \mu/(n\gamma)$.

Currently, none of the functions documented here are vectorized. Arguments must have length 1.

Value

`seir.R0` returns a numeric vector of length 1. `seir.ee` returns a numeric vector of length `1+m+n+1`. `seir.jacobian` returns a function of one argument `x` (which must be a numeric vector of length `1+m+n+1`) whose return value is a square numeric matrix of size `length(x)`.

See Also

`seir`, for the system of ordinary differential equations on which these computations are predicated.

seir.library *Often Used Simulations*

Description

Infectious disease time series simulated using seir, for use primarily in examples, tests, and vignettes. Users should not rely on simulation details, which may change between package versions.

Note that simulation code depends on availability of suggested packages **adaptivetau** and **deSolve**. If the dependency cannot be loaded then the value of the data set is NULL.

Usage

```
## if (requireNamespace("deSolve"))
data(seir.ts01, package = "fastbeta")
## else ...

## if (requireNamespace("adaptivetau"))
data(seir.ts02, package = "fastbeta")
## else ...
```

Format

A "multiple time series" object, inheriting from class mts, always a subset of the result of a call to seir, discarding transient behaviour. Simulation parameters may be preserved as attributes.

Source

Scripts sourced by data to reproduce the simulations are located in subdirectory 'data' of the **fastbeta** installation; see, e.g. system.file("data", "seir.ts01.R", package = "fastbeta").

See Also

seir.

Examples

```
if (requireNamespace("deSolve")) withAutoprint({

data(seir.ts01, package = "fastbeta")
str(seir.ts01)
plot(seir.ts01)

})

if (requireNamespace("adaptivetau")) withAutoprint({

data(seir.ts02, package = "fastbeta")
str(seir.ts02)
```

```
plot(seir.ts02)

})
```

`sir.aoi` *Solve the SIR Equations Structured by Age of Infection*

Description

Numerically integrates the SIR equations with rates of transmission and recovery structured by age of infection.

Usage

```
sir.aoi(from = 0, to = from + 1, by = 1,
        R0, ell = 1, eps = 0, n = max(length(R0), length(ell)),
        init = c(1 - init.infected, init.infected),
        init.infected = .Machine[["double.neg.eps"]],
        weights = rep(c(1, 0), c(1L, n - 1L)),
        F = function (x) 1, Fargs = list(),
        H = identity, Hargs = list(),
        root = NULL, root.max = 1L, root.break = TRUE,
        aggregate = FALSE, skip.Y = FALSE, ...)

## S3 method for class 'sir.aoi'
summary(object, name = "Y", tol = 1e-6, ...)
```

Arguments

`from`, `to`, `by`	passed to `seq.int` in order to generate an increasing, equally spaced vector of time points in units of the mean time spent infectious.
`R0`	a numeric vector of length `n` such that `sum(R0)` is the basic reproduction number and `R0[j]` is the contribution of infected compartment `j`. Otherwise, a numeric vector of length 1, handled as equivalent to `rep(R0/n, n)`.
`ell`	a numeric vector of length `n` such that `ell[j]` is the ratio of the mean time spent in infected compartment `j` and the mean time spent infectious; internally, `ell/sum(ell[R0 > 0])` is used, hence `ell` is determined only up to a positive factor. Otherwise (and by default), a numeric vector of length 1, handled as equivalent to `rep(1, n)`.
`eps`	a non-negative number giving the the ratio of the mean time spent infectious and the mean life expectancy; `eps = 0` implies that life expectancy is infinite (that there are no deaths).
`n`	a positive integer giving the number of infected compartments. Setting `n` and thus overriding the default expression is necessary only if the lengths of `R0` and `ell` are both 1.
`init`	a numeric vector of length 2 giving initial susceptible and infected proportions.

`init.infected`	a number in $(0, 1]$ used only to define the default expression for `init`; see 'Usage'.
`weights`	a numeric vector of length `n` containing non-negative weights, defining the initial distribution of infected individuals among the infected compartments. By default, all infected individuals occupy the first compartment.
`F`, `H`	functions returning a numeric vector of length 1 or of length equal that of the first formal argument. The body must be symbolically differentiable with respect to the first formal argument; see `D`.
`Fargs`, `Hargs`	lists of arguments passed to `F` or `H`. In typical usage, `F` and `H` define parametric families of functions of one variable, and `Fargs` and `Hargs` supply parameter values. For example: `H = function(x, h) x^h`, `Hargs = list(h = 0.996)`.
`root`	a function returning a numeric vector of length 1, with formal arguments `(tau, S, I, Y, dS, dI, dY, R0, ell)` (or a subset); otherwise, `NULL`. Roots of this function in the interval from `from` to `to` are sought alongside the numerical solution.
`root.max`	a positive integer giving a stopping condition for the root finder. Root finding continues until the count of roots found is `root.max`.
`root.break`	a logical indicating if the solver should stop once `root.max` roots are found. If `TRUE`, then the numerical solution ends at the last time point less than or equal to the last root.
`aggregate`	a logical indicating if infected compartments should be aggregated.
`skip.Y`	a logical indicating if solution of the equation for Y should not be attempted, e.g., because that equation seems stiffer than the rest.
`...`	optional arguments passed to the solver, function `lsoda` in package **deSolve**.
`object`	an R object inheriting from class `sir.aoi`, typically the value of a call to `sir.aoi`.
`name`	a character string in `colnames(object)`. Tail exponents of V are approximated, where, for example, $V = Y$ if `name = "Y"` and $V = \sum_j I_j$ (prevalence) if `name = "I"`.
`tol`	a positive number giving an upper bound on the relative change (from one time point to the next) in the slope of $\log(V)$, defining time windows in which growth or decay of V is considered to be exponential.

Details

The SIR equations with rates of transmission and recovery structured by age of infection are

$$\begin{aligned}
\mathrm{d}S &/\mathrm{d}t = \mu(1 - S) - (\textstyle\sum_j \beta_j F I_j) H(S) \\
\mathrm{d}I_1 &/\mathrm{d}t = (\textstyle\sum_j \beta_j F I_j) H(S) - (\gamma_1 + \mu) I_1 \\
\mathrm{d}I_{j+1} &/\mathrm{d}t = \gamma_j I_j - (\gamma_{j+1} + \mu) I_{j+1} \\
\mathrm{d}R &/\mathrm{d}t = \gamma_n I_n - \mu R
\end{aligned}$$

where $S, I_j, R \geq 0$, $S + \sum_j I_j + R = 1$, F is a forcing function, and H is a susceptible heterogeneity function. In general, F and H are nonlinear. In the standard SIR equations, F is 1 and H is the identity function.

Nondimensionalization using parameters $\mathcal{R}_{0,j} = \beta_j/(\gamma_j + \mu)$, $\ell_j = (1/(\gamma_j + \mu))/t_+$, and $\varepsilon = t_+/(1/\mu)$ and independent variable $\tau = t/t_+$, where $t_+ = \sum_{j:\mathcal{R}_{0,j}>0} 1/(\gamma_j + \mu)$ designates as a natural time unit the mean time spent infectious, gives

$$\begin{aligned}
\mathrm{d}S \quad /\mathrm{d}\tau &= \varepsilon(1 - S) - (\textstyle\sum_j (\mathcal{R}_{0,j}/\ell_j) F I_j) H(S) \\
\mathrm{d}I_1 \quad /\mathrm{d}\tau &= (\textstyle\sum_j (\mathcal{R}_{0,j}/\ell_j) F I_j) H(S) - (1/\ell_1 + \varepsilon) I_1 \\
\mathrm{d}I_{j+1}/\mathrm{d}\tau &= (1/\ell_j) I_j - (1/\ell_{j+1} + \varepsilon) I_{j+1} \\
\mathrm{d}R \quad /\mathrm{d}\tau &= (1/\ell_n) I_n - \varepsilon R
\end{aligned}$$

`sir.aoi` works with the nondimensional equations, dropping the last equation (which is redundant given $R = 1 - S - \sum_j I_j$) and augments the resulting system of $1 + n$ equations with a new equation

$$\mathrm{d}Y/\mathrm{d}\tau = (\textstyle\sum_j (\mathcal{R}_{0,j}/\ell_j) F I_j)(H(S) - 1/(\sum_j \mathcal{R}_{0,j} F))$$

due to the usefulness of the solution Y in applications.

Value

A "multiple time series" object, inheriting from class `sir.aoi` and transitively from class `mts`, storing the numerical solution. Beneath the class, it is a `T`-by-`(1+n+1)` numeric matrix of the form `cbind(S, I, Y)`, `T <= length(seq(from, to, by))`.

If `root` is a function, then an attribute `root.info` of the form `list(tau, state = cbind(S, I, Y))` stores the first `K` roots of that function and the state of the system at each root, `K <= root.max`.

If `aggregate = TRUE`, then infected compartments are aggregated so that the number of columns named `I` is 1 rather than `n`. This column stores prevalence, the proportion of the population that is infected. For convenience, there are 5 additional columns named `I.E`, `I.I`, `foi`, `inc`, and `crv`. These store the non-infectious and infectious components of prevalence (so that `I.E + I.I = I`), the force of infection, incidence (so that `foi * S = inc`), and the curvature of `Y`.

The method for `summary` returns a numeric vector of length 2 containing the left and right "tail exponents" of the variable V indicated by `name`, defined as the asymptotic values of the slope of $\log(V)$. `NaN` elements indicate that a tail exponent cannot be approximated because the time window represented by `object` does not cover enough of the tail, where the meaning of "enough" is set by `tol`.

Note

`sir.aoi` is not a special case of `sir` nor a generalization. The two functions were developed independently and for different purposes: `sir.aoi` to validate analytical results concerning the SIR equations as formulated here, `sir` to simulate incidence time series suitable for testing `fastbeta`.

Examples

```
if (requireNamespace("deSolve")) withAutoprint({

to <- 100; by <- 0.01; R0 <- c(0, 16); ell <- c(0.5, 1)

soln.peak <- sir.aoi(to = to, by = by, R0 = R0, ell = ell,
                     root = function (S, R0) sum(R0) * S - 1,
                     aggregate = TRUE)
```

```
str(soln.peak)

info.peak <- attr(soln.peak, "root.info")
to <- 4 * info.peak[["tau"]] # a more principled endpoint

soln <- sir.aoi(to = to, by = by, R0 = R0, ell = ell,
                aggregate = TRUE, atol = 0, rtol = 1e-12)
##                                ^^^^      ^^^^
## 'atol', 'rtol', ... are passed to deSolve::lsoda

head(soln)
tail(soln)

plot(soln) # dispatching stats:::plot.ts
plot(soln, log = "y")

(lamb <- summary(soln)) # left and right tail exponents

xoff <- function (x, k) x - x[k]
k <- c(16L, nrow(soln)) # c(1L, nrow(soln)) worse due to transient

plot(soln[, "Y"], log = "y", ylab = "Y")
abline(v = info.peak[["tau"]], h = info.peak[["state"]][, "Y"],
       lty = 2, lwd = 2, col = "red")
for (i in 1:2)
lines(soln[k[i], "Y"] * exp(lamb[i] * xoff(time(soln), k[i])),
      lty = 2, lwd = 2, col = "red")

wrap <-
function (root, ...)
attr(sir.aoi(to = to, by = by, R0 = R0, ell = ell,
             root = root, aggregate = TRUE, ...),
     "root.info")[["tau"]]
Ymax <- info.peak[["state"]][, "Y"]

## NB: want *simple* roots, not *multiple* roots
L <- list(function (Y) (Y - Ymax * 0.5)  ,
          function (Y) (Y - Ymax * 0.5)^2,
          function (Y) (Y - Ymax      )  ,
          function (Y) (Y - Ymax      )^2)
lapply(L, wrap)

## NB: critical values can be attained more than once
L <- list(function (Y, dY)             Y - Ymax * 0.5,
          function (Y, dY) if (dY > 0) Y - Ymax * 0.5 else 1,
          function (Y, dY) if (dY < 0) Y - Ymax * 0.5 else 1)
lapply(L, wrap, root.max = 2L)

})
```

---

smallpox *Smallpox Mortality in London, England, 1661-1930*

Description

Time series of deaths due to smallpox, deaths due to all causes, and births in London, England, from 1661 to 1930, as recorded in the London Bills of Mortality and the Registrar General's Weekly Returns.

Usage

```
data(smallpox, package = "fastbeta")
```

Format

A data frame with 13923 observations of 5 variables:

from start date of the record.

nday length of the record, which is the number of days (typically 7) over which deaths and births were counted.

smallpox count of deaths due to smallpox.

allcauses count of deaths due to all causes.

births count of births.

Source

A script generating the `smallpox` object from a CSV file accompanying the reference is available as `system.file("scripts", "smallpox.R", package = "fastbeta")`.

A precise description of the data set and its correspondence to the original source documents is provided in the reference.

References

Krylova, O. & Earn, D. J. D. (2020). Patterns of smallpox mortality in London, England, over three centuries. *PLOS Biology*, *18*(12), Article e3000506, 1-27. doi:10.1371/journal.pbio.3000506

Examples

```
data(smallpox, package = "fastbeta")
str(smallpox)
table(smallpox[["nday"]]) # not all 7 days, hence:
plot(7 * smallpox / as.double(nday) ~ from, smallpox, type = "l")
```

Index