Package 'fastbeta'

December 12, 2023

Version 0.2.0

Date 2023-12-11

Title Fast Estimation of Time-Varying Infectious Disease Transmission Rates

Description Methods for estimating time-varying infectious disease transmission rates from disease incidence time series, based on discretizations of an SIR 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 (>= 4.3.0)

Imports adaptivetau, deSolve, graphics, stats

Suggests grDevices, tools, utils

BuildResaveData no

NeedsCompilation yes

Author Mikael Jagan [aut, cre] (<https://orcid.org/0000-0002-3542-2938>)

Maintainer Mikael Jagan <jaganmn@mcmaster.ca>

Repository CRAN

Date/Publication 2023-12-11 23:00:02 UTC

R topics documented:

`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, 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` observation intervals of equal duration.
`prob`	a numeric vector of length `d+n` 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 length `d+1` 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. `delay` need not sum to 1 but must not sum to 0.
`start`	a numeric vector of length `d+n` 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` zeros, choosing `d0=which.max(delay)-1`.
`tol`	a tolerance indicating a stopping condition; see the reference.
`iter.max`	the maximum number of iterations.
`complete`	a logical flag indicating if the result should preserve successive updates to `start`.

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-`(iter+1)` matrix containing `start` and the `iter` successive updates, each divided by `prob`.
`chisq`	the chi-squared statistic(s) corresponding to `value`.
`iter`	the number of iterations performed.

References

Goldstein, E., Dushoff, J., Ma, J., Plotkin, J. B., Earn, D. J. D., & Lipsitch, M. (2020). 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

```
set.seed(2L)
n <- 200L
d <- 50L
p <- 0.1
prob <- plogis(rlogis(d + n, location = qlogis(p), scale = 0.1))
delay <- diff(pgamma(0L:(d + 1L), 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.int(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.int(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, 3L),
        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, 3L),
       bty = "n")

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

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

Description

Generates a discrete time estimate of a transmission rate $\beta(t)$ from an equally spaced incidence time series and other data.

Usage

```
fastbeta(series, constants, ...)
```

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.
`constants`	a numeric vector of the form `c(S0, I0, R0, gamma, delta)`, specifying an initial state and rates of removal and loss of immunity, in that order.
`...`	optional arguments passed to `deconvolve`, if the first column of `series` represents *reported* incidence or mortality rather than incidence.

Details

`fastbeta` works by discretizing the system of ordinary differential equations

$$\begin{aligned}\frac{\mathrm{d}S}{\mathrm{d}t} &= \nu(t) - \beta(t)SI + \delta R - \mu(t)S \\ \frac{\mathrm{d}I}{\mathrm{d}t} &= \beta(t)SI - \gamma I - \mu(t)I \\ \frac{\mathrm{d}R}{\mathrm{d}t} &= \gamma I - \delta R - \mu(t)R\end{aligned}$$

where t is understood to be a unitless measure of time relative to the duration of an observation interval, then computing the iteration

$$\begin{aligned}I_{t+1} &= \frac{(1 - \frac{1}{2}(\gamma + \mu_t))I_t + Z_{t+1}}{1 + \frac{1}{2}(\gamma + \mu_{t+1})} \\ R_{t+1} &= \frac{(1 - \frac{1}{2}(\delta + \mu_t))R_t + \frac{1}{2}\gamma(I_t + I_{t+1})}{1 + \frac{1}{2}(\delta + \mu_{t+1})} \\ S_{t+1} &= \frac{(1 - \frac{1}{2}\mu_t)S_t - Z_{t+1} + B_{t+1} + \frac{1}{2}\delta(R_t + R_{t+1})}{1 + \frac{1}{2}\mu_{t+1}} \\ \beta_t &= \frac{Z_t + Z_{t+1}}{2S_tI_t}\end{aligned}$$

where

$$\begin{aligned}X_t &\sim X(t) \quad [X = \beta, \mu, S, I, R] \\ Z_t &\sim \int_{t-1}^{t} \beta(s)S(s)I(s)\,\mathrm{d}s \\ B_t &\sim \int_{t-1}^{t} \nu(s)\,\mathrm{d}s\end{aligned}$$

and it is understood that indexing starts at $t = 0$. Z_t, B_t, and μ_t denote incidence and births between times $t - 1$ and t and the per capita natural mortality rate at time t; they are supplied together as argument `series`.

Value

A "multiple time series" object, inheriting from class `mts`, with four columns (named `S`, `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

```
data(sir.e02, package = "fastbeta")
a <- attributes(sir.e02)
str(sir.e02)
plot(sir.e02)

## We suppose that we have perfect knowledge of incidence,
## births, and the data-generating parameters
series <- cbind(sir.e02[, c("Z", "B")], mu = a[["mu"]](0))
colnames(series) <- c("Z", "B", "mu") # FIXME: stats:::cbind.ts mangles dimnames
constants <- c(S0 = sir.e02[[1L, "S"]],
               I0 = sir.e02[[1L, "I"]],
               R0 = sir.e02[[1L, "R"]],
               gamma = a[["gamma"]],
               delta = a[["delta"]])

X <- fastbeta(series, constants)
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 transmission rate $\beta(t)$ and r bootstrap estimates. Bootstrap estimates are computed for incidence time series simulated using sir, with transmission rate defined as the linear interpolant of the primary estimate.

Usage

```
fastbeta.bootstrap(r, series, constants, ...)
```

Arguments

r	the number of replications.
series, constants	see fastbeta.

`...` optional arguments passed to `sir` and/or `deconvolve`. Both take optional arguments `prob` and `delay`. When `prob` is supplied but not `delay`, `sir` and `deconvolve` receive `prob` as is. When both are supplied, `sir` 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 `r+1` columns storing the primary and bootstrap estimates, in that order. It is completely parallel to argument `series`, having the same `tsp` attribute.

Examples

```
data(sir.e02, package = "fastbeta")
a <- attributes(sir.e02)
str(sir.e02)
plot(sir.e02)

## We suppose that we have perfect knowledge of incidence,
## births, and the data-generating parameters
series <- cbind(sir.e02[, c("Z", "B")], mu = a[["mu"]](0))
colnames(series) <- c("Z", "B", "mu") # FIXME: stats:::cbind.ts mangles dimnames
constants <- c(S0 = sir.e02[[1L, "S"]],
               I0 = sir.e02[[1L, "I"]],
               R0 = sir.e02[[1L, "R"]],
               gamma = a[["gamma"]],
               delta = a[["delta"]])

R <- fastbeta.bootstrap(100L, series, constants)
str(R)
plot(R)
plot(R, level = 0.95)
```

---

ptpi *Peak to Peak Iteration*

Description

Estimates the initial sizes of the susceptible, infected, and removed populations from a periodic, equally spaced incidence time series and other data. Interpret with care, notably when supplying time series that are only "roughly" periodic.

Usage

```
ptpi(series, constants, a = 0L, b = nrow(series) - 1L,
     tol = 1e-03, iter.max = 32L,
     complete = FALSE, backcalc = 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.
constants	a numeric vector of the form c(Sa, Ia, Ra, gamma, delta), specifying a starting value for the state at time a and rates of removal and loss of immunity, in that order.
a	the time of the first peak in the incidence time series. It is rounded internally to generate a 0-index of rows of series.
b	the time of the last peak in the incidence time series that is in phase with the first. It is rounded internally to generate a 0-index of rows of series.
tol	a tolerance indicating a stopping condition; see 'Details'.
iter.max	the maximum number of iterations.
complete	a logical indicating if the result should preserve the state at times a, ..., b in each iteration.
backcalc	a logical indicating if the state at time 0 should be back-calculated.
...	optional arguments passed to deconvolve, if the first column of series represents *reported* incidence or mortality rather than incidence.

Details

ptpi works by computing the iteration described in fastbeta from time $t = a$ to time $t = b$, iteratively, until the relative difference between the states at times a and b descends below a tolerance. The state at time a in the first iteration is specified by the user. In subsequent iterations, it is the value of the state at time b calculated in the previous iteration.

If backcalc = FALSE, then ptpi returns a list with component value equal to the state at time b in the last iteration. By periodicity, this value estimates the "true" state at time a.

If backcalc = TRUE, then value is back-calculated so that it estimates the "true" state at time 0. This works by inverting the transformation defining one step of the iteration, hence (components of) the back-calculated result can be nonsense if the inverse problem is ill-conditioned.

Value

A list with elements:

value	the estimated value of the initial state, which is the state at time a if backcalc = FALSE and the state at time 0 if backcalc = TRUE.
delta	the relative difference computed in the last iteration.
iter	the number of iterations performed.
X	if complete = TRUE, then a "multiple time series" object, inheriting from class mts, with dimensions c(b-a+1, 3, iter). X[, , i] gives the state at times a, ..., b in iteration i.

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

```
data(sir.e01, package = "fastbeta")
a <- attributes(sir.e01)
str(sir.e01)
plot(sir.e01)

## 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(sir.e01[, c("Z", "B")], mu = a[["mu"]](0))
colnames(series) <- c("Z", "B", "mu") # FIXME: stats:::cbind.ts mangles dimnames
constants <- c(Sa = sir.e01[[1L, "S"]],
               Ia = sir.e01[[1L, "I"]],
               Ra = sir.e01[[1L, "R"]],
               gamma = a[["gamma"]],
               delta = a[["delta"]])

plot(series[, "Z"])
a <- 8; b <- 216
abline(v = c(a, b), lty = 2)

L <- ptpi(series, constants, a = a, b = b, complete = TRUE, tol = 1e-06)
str(L)

S <- L[["X"]][, "S", ]
plot(S, plot.type = "single")
lines(sir.e01[, "S"], col = "red", lwd = 4) # the "truth"
abline(h = L[["value"]]["S"], v = a, col = "blue", lwd = 4, lty = 2)

## The relative error
L[["value"]]["S"] / sir.e01[1L + a, "S"] - 1
```

`sir` *Simulate Infectious Disease Time Series*

Description

Simulates time series of the susceptible, infected, and removed population sizes and corresponding time series of births, incidence, and *reported* incidence. Simulations are based on an SIR model with user-defined forcing and a simple model for observation error.

Usage

```
sir(n, beta, nu, mu, constants, stochastic = TRUE,
    prob = 1, delay = 1, useCompiled = TRUE, ...)
```

Arguments

`n`	a positive integer. The simulation uses `0:n` as time points, yielding `n` observation intervals of equal duration.
`beta, nu, mu`	functions of one or more argument returning transmission, birth, and natural death rates, respectively, at the time point indicated by the first argument. Arguments after the first must be strictly optional. The functions need not be vectorized.
`constants`	a numeric vector of the form `c(S0, I0, R0, gamma, delta)`, specifying an initial state and rates of removal and loss of immunity, in that order.
`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 reported. `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 reported during interval `j+i-1`, given that it is eventually reported. `delay` need not sum to 1 but must not sum to 0.
`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 `ode` (directly) or `ssa.adaptivetau` (via its list argument `tl.params`), depending on `stochastic`.

Details

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

$$\frac{\mathrm{d}S}{\mathrm{d}t} = \nu(t) - \beta(t)SI + \delta R - \mu(t)S$$
$$\frac{\mathrm{d}I}{\mathrm{d}t} = \beta(t)SI - \gamma I - \mu(t)I$$
$$\frac{\mathrm{d}R}{\mathrm{d}t} = \gamma I - \delta R - \mu(t)R$$

between times $t = 0, \ldots, n$, where t is understood to be a unitless measure of time relative to the duration of an observation interval. To generate time series of births and incidence

$$B(t) = \int_{t-1}^{t} \nu(s)\,\mathrm{d}s$$
$$Z(t) = \int_{t-1}^{t} \beta(s)S(s)I(s)\,\mathrm{d}s$$

the system is augmented with two additional equations describing *cumulative* incidence and *cumulative* births (with right hand sides given by the integrands above), and the *augmented* system with

five equations is integrated. Case reports are simulated by scaling incidence by `prob` and convolving the result with `delay`.

`sir(stochastic = TRUE)` works by simulating a Markov process corresponding to the augmented system, as described in the reference. Case reports are 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 an `(n+1)`-by-`(5+r)` numeric matrix `X`, where `r` is 0 if both `prob` and `delay` are missing (indicating no observation error) and 1 otherwise.

Rows correspond to time points `0:n`. Columns are named `S`, `I`, `R`, `B`, `Z`, and (if `r` is 1) `Z.obs`. `X[, 1:3]` give the state at each time, and `X[, 4:6]` give the number of births, infections, and infections *reported* during the observation interval ending at each time.

`X[1, 4:6]` is `NA`, and `X[2:length(delay), 6]` can contain incomplete information if `length(delay) >= 2`.

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

`sir.library`.

Examples

```
beta <- function (t, a = 1e-01, b = 1e-05)
b * (1 + a * cospi(t / 26))
nu <- function (t) 1e+03
mu <- function (t) 1e-03

S0 <- 5e+04
I0 <- 1e+03
R0 <- 1e+06 - S0 - I0
constants <- c(S0 = S0, I0 = I0, R0 = R0, gamma = 0.5, delta = 0)

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

set.seed(0L)
X <- sir(n, beta, nu, mu, constants, prob = prob, delay = delay)
str(X)
plot(X)
```

```
r <- 10L
Y <- do.call(cbind, replicate(r, simplify = FALSE,
sir(n, beta, nu, mu, constants, prob = prob, delay = delay)[, "Z.obs"]))
str(Y) # FIXME: stats:::cbind.ts mangles dimnames
plot(Y, plot.type = "single", col = seq_len(r), ylab = "Case reports")
```

`sir.library` *Often Used Simulations*

Description

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

Usage

```
data(sir.e01, package = "fastbeta") # "Example 01", and so on
data(sir.e02, package = "fastbeta")
```

Format

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

Source

Scripts sourced by `data` to reproduce the simulations are located in the 'data' subdirectory of the **fastbeta** installation; see, e.g. `system.file("data", "sir.e01.R", package = "fastbeta")`.

See Also

`sir`.

Examples

```
data(sir.e01, package = "fastbeta")
str(sir.e01)
plot(sir.e01)

data(sir.e02, package = "fastbeta")
str(sir.e02)
plot(sir.e02)
```

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

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

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