Package 'BaSTA2.0'

October 6, 2022

Type Package

Title Age-Specific Bayesian Survival Trajectory Analysis from Incomplete census or capture-recapture/recovery Data

Version 2.0.0

Date 2022-10-05

Author Fernando Colchero

Maintainer Fernando Colchero <colchero@imada.sdu.dk>

Depends R (>= 3.5.0)

Imports snowfall

Description Estimates survival and mortality with covariates from census or capture-recapture/recovery data in a Bayesian framework when many individuals are of unknown age. It includes tools for data checking, model diagnostics and outputs such as life-tables and plots.

License GPL

LazyLoad yes

LazyData yes

NeedsCompilation no

R topics documented:

`BaSTA2.0-package` *BaSTA: Parametric Bayesian estimation of age-specific survival for truncated and censored capture-recapture or census data.*

Description

This package estimates age-specific mortality and survival trajectories with covariates from capture-recapture/recovery or census data in a Bayesian framework when many individuals are of unknown age. It includes tools for data checking, model diagnostics, and outputs such as life-tables and plots.

Details

Package:	BaSTA2.0
Type:	Package
Version:	2.0.0
Date:	2022-10-05
License:	GNU General Public Licence
LazyLoad:	yes

Parametric estimation of age-specific survival and mortality from capture-mark-recapture or census data including individuals with unknown times of birth and death.

Author(s)

Fernando Colchero <colchero@imada.sdu.dk>

References

Colchero, F. and J.S. Clark (2012) Bayesian inference on age-specific survival from capture-recapture data for censored and truncated data. *Journal of Animal Ecology* 81, 139-149.

Colchero, F., O.R Jones and M. Rebke. (2012) BaSTA: an R package for Bayesian estimation of age-specific survival from incomplete mark-recapture/recovery data with covariates. *Methods in Ecology and Evolution* 3, 466-470.

Colchero, F., *et al.* (2021) The long lives of primates and the "invariant rate of aging" hypothesis. *Nature Communications* 12:3666

`basta` *Parametric Bayesian estimation of age-specific survival for left-truncated and right-censored capture-mark-recapture or census data.*

Description

This function performs multiple Markov Chain Monte Carlo (MCMC) simulations for the Bayesian estimation of age-specific mortality and survival when a large proportion of records have unknown times of birth and/or death. Survival parameters and unknown (i.e. latent) birth (and death, for CMR data) times are estimated, allowing the user to test a range of mortality patterns, and to test the effect of continuous and/or discrete covariates following Colchero and Clark's (2012) general approach.

Usage

```
basta(object, ...)

## Default S3 method:
 basta(object, dataType = "CMR", model = "GO",
  shape = "simple",  studyStart = NULL, studyEnd = NULL,
  minAge = 0, covarsStruct = "fused",  formulaMort = NULL,
  formulaRecap = NULL, recaptTrans = studyStart,  niter = 22000,
  burnin = 2001, thinning = 40, nsim = 1, parallel = FALSE,
  ncpus = 2, updateJumps = TRUE, negSenescence = FALSE,  ...)
```

Arguments

object	A data.frame to be used as an input data file for BaSTA. Note: BaSTA can take two types of datasets, namely capture-mark-recapture (CMR) or census data. See details for a description of the data structures.
dataType	A character string indicating if the data are for capture-mark-recapture (CMR) or census. Options are "CMR" (default) or "census".
model	The underlying mortality model to be used. "EX" = exponential,"GO" = Gompertz, "WE" = Weibull and "LO" = logistic (see details).
shape	The overall shape of the model. Values are: simple = no extra parameters added; Makeham = a constant parameter is added to the mortality; and bathtub = a Gompertz declining mortality for early ages and a constant parameter are added to the mortality model (see details).
studyStart	Only required for dataType = "CMR", an integer indicating the first year of the study.
studyEnd	Only required for dataType = "CMR", an integer indicating the last year of the study.
minAge	Age at which the analysis should start (see details).
covarsStruct	Character string that indicates how covariates should be evaluated. The options are: "fused", which defines all categorical variables as covariates for each mortality parameter and all continuous covariates under a proportional hazards structure; "prop.haz", which puts all covariates under a proportional hazards structure; and "all.in.mort" puts all covariates as a multilevel function of the mortality parameters (see details).
formulaMort	An object of class formula specifying the covariates to be included on the mortality part of the model. Note that the syntax should not include a dependent variable. If formulaMort = NULL, then no covariates are included in the analysis. See details.
formulaRecap	*Not yet implemented*, an object of class formula specifying the covariates to be included on the recapture part of the model. Note that the syntax should not include a dependent variable. If formulaMort = NULL, then no covariates are included in the analysis.
recaptTrans	A vector (of maximum length equal to the duration of the study) defining the recapture probability transition times (RPTP). These are points (years) where the recapture probability is thought to change. The default setting is for the recapture probability to be constant throughout the study, so the recaptTrans is simply defined as a single element vector of the first year of the observation period (e.g. c(1985)). If recapture probabilities were known to change at year say, 1990, the RPTP should be defined as c(1985, 1990).

niter	The total number of MCMC steps.
burnin	The number of iterations for the burn in (see details).
thinning	The number of skipped MCMC steps to minimize serial autocorrelation (see details).
nsim	A numerical value for the number of simulations to be run.
parallel	A logical argument indicating whether the multiple simulations should be run in parallel or not. If TRUE, package **snowfall** is called and multiple simulations are run in parallel. If **snowfall** is not installed, the model is ran in series.
ncpus	a numerical value that indicates the number of cpus to be used if parallel is TRUE and package **snowfall** is installed. The default is 2 cpus. If package pkgsnowfall is not installed, the simulations are run in series.
updateJumps	A logical argument indicating wheter to update jump standard deviations (adaptive independent Metropolis) until an update rate of 0.25 is achieved (see details).
negSenescence	Logical indicating if negative senescence should be allowed, only applicable for model = "GO" and shape = "Makeham".
...	Additional arguments to be passed to function basta (see details)

Details

1) DATA TYPES:

The input object required by BaSTA needs to be constructed differently whether the data are of capture-mark-recapture or census.

a) capture-mark-recapture (CMR):

If dataType = "CMR", then the data frame requires the following structure. The first column is a vector of individual unique IDs, the second and third columns are birth and death years respectively. Columns $4, \ldots, T+3$ represent the observation window (i.e., recapture matrix) of T years. This is followed (optionally) by columns for categorical and continuous covariates (see bastaCMRdat for a CMR dataset example).

b) Census:

If dataType = "census", then the input data object requires at least five dates columns, namely "Birth.Date", "Min.Birth.Date", "Max.Birth.Date", "Entry.Date", and "Depart.Date". All dates need to be format as "%Y-%m-%d". In addition, a "Depart.Type" column is required with two types of departures "C" for Censored and "D" for dead (see bastaCensDat for a census dataset example).

2) INFERENCE ON AGE-SPECIFIC MORTALITY:

basta uses parametric mortality functions to estimate age-specific mortality (survival) from capture-recapture/recovery data. The mortality function describes how the risk of mortality changes with age, and is defined as $\mu(x|\theta)$, where x corresponds to age and θ is a vector of parameters to be estimated.

The model argument allows the user to choose between four basic mortality functions, namely (a) Exponential ("EX"; Cox and Oakes 1974), with constant mortality with age specified as

$$\mu_b(x|b) = b,$$

where $b > 0$, (b) the Gompertz mortality function ("GO"; Gompertz 1925, Pletcher 1999), calculated as

$$\mu_b(x|b) = exp(b_0 + b_1 x),$$

where $-\infty < b_0, b_1 < \infty$, (c) the Weibull mortality model ("WE"; Pinder III *et al.* 1978) calculated as

$$\mu_b(x|b) = b_0 b_1^{b_0} x^{b_0 - 1},$$

where $b_0, b_1 > 0$, and (d) the logistic mortality model ("LO"; Pletcher 1999), calculated as

$$\mu_b(x|b) = exp(b_0 + b_1 x)/(1 + b_2 exp(b_0)/b_1(exp(b_1 x) - 1)),$$

where $b_0, b_1, b_2 > 0$.

The shape argument allows the user to extend these models in order to explore more complex mortality shapes. The default value is "simple" which leaves the model as defined above. With value "Makeham", a constant is added to the mortality, making the model equal to $\mu_0(x|\theta) = \mu_b(x|b) + c$, where $\theta = [c, b]$. With value "bathtub", concave shapes in mortality can be explored. This is achieved by adding a declining Gompertz term and a constant parameter to the basic mortality model, namely

$$\mu_0(x|\theta) = exp(a_0 - a_1 x) + c + \mu_b(x|b)$$

,

where $-\infty < a_0 < \infty$, $a_1 \leq 0$ and $c \leq 0$.

3) COVARIATES:

Covariates are selected by means of the argument formulaMort, which requires an object of class formula, just as with other statistical inference functions such as lm or glm.

When covariates are included in the dataset, the basta function provides three different ways in which these can be evaluated by using argument covarsStruct:

1. "fused" will make the mortality parameters linear functions of all categorical covariates (analogous to a generalised linear model (GLM) structure) and will put all continuous covariates under a proportional hazards structure. Thus, for a simple exponential model with constant mortality of the form $\mu_0(x|b) = b$, the parameter is equal to $b = b_0 + b_1 z_1 + \ldots, b_k z_k$, where $[b_0, \ldots, b_k]$ are paramters that link the mortality parameter b with the categorical covariates $[z_1, \ldots, z_k]$.

2. "prop.haz" will put all covariates under a proportional hazards structure irrespective of the type of variable. In this case, the mortality model is be further extended by including a proportional hazards structure, of the form

$$\mu(x|\theta, \Gamma, Z_a, Z_c) = \mu_0(x|\theta, Z_a) exp(\Gamma Z_c)$$

,

where $\mu_0(x|\theta, Z_a)$ represents the mortality section as defined above, while the second term $exp(\Gamma Z_c)$ corresponds to the proportional hazards function. Z_a and Z_c are covariate (design) matrices for categorical and continuous covariates, respectively.

3. "all.in.mort" will put all covariates as linear functions of the survival parameters as explained above. Since most models require the lower bounds for the mortality parameters to be equal to 0, the only model that can be used for this test is Gompertz with shape set to "simple". In case these arguments are specified deferently, a warning message is printed noting that model will be forced to be "GO" and shape will be set to "simple".

4) MCMC SETTINGS:

The burnin argument represents the number of steps at the begining of the MCMC run that is be discarded. This sequence commonly corresponds to the non-converged section of the MCMC

sequence. Convergence and model selection measures are calculated from the remaining thinned parameter chains if multiple simulations are run, and all if all of them run to completion.

The thinning argument specifies the number of steps to be skipped in order to reduce serial autocorrelation. The thinned sequence, which only includes steps after burn in, is then used to calculate convergence statistics and model for selection.

The updateJumps argument specifies wether to run a simulation to find appropriate jump standard deviations for theta and gamma parameters. If argument "nsim" is set to 1, then the simulation runs with the update jumps routine active. If "nsim" is larger than 1, then an initial simulation is ran to find apropriate jumps before the main analysis is ran.

5) ADDITIONAL ARGUMENTS:

Additional arguments for priors, jumps and start values can be passed on the ... section. For instance, argument thetaStart can be specified as a vector defining the initial values for each parameter in the survival model. If this argument is not specified, a set of random parameters is generated for each simulation. Similarly, argument gammaStart can be specified for all parameters in the proportional hazards section of the model. Jump standard deviations (i.e. the standard error in the Metropolis step) can be specified with arguments thetaJumps and gammaJumps. As with thetaStart, default values are assigned if these arguments are not specified.

To specify priors, arguments thetaPriorMean, thetaPriorSd, gammaPriorMean and gammaPriorSd can be used for prior means and standard errors for each survival and proportional hazards parameters. If not specified, default values are assigned.

The number of parameters in thetaStart, thetaJumps, thetaPriorMean and thetaPriorSd should be a vector or matrix for the parameters in the mortality function. The number of parameters will depend on the model chosen with model (see above). If the number of parameters specified does not match the number of parameters inherent to the model and shape selected, the function returns an error.

As described above, the number of parameters for gammaStart, gammaJumps, gammaPriorMean and gammaPriorSd arguments (i.e. section b), namely the proportional hazards section, will be a function of the number of continuous covariates if argument covarsStruct is "fused", or to the total number of covariates when covarsStruct is "prop.haz".

Value

params	If requested, a matrix with the thinned, converged parameter traces of all runs. This matrix is used to calculate quantiles for parameters, survival probability and mortality (see below).
theta	If requested, a matrix with only the parameters of the mortality function after convergence and thinning.
coefficients	A matrix with estimated coefficients (i.e. mean values per parameter on the thinned sequences after burnin), which includes standard errors, upper and lower 95% credible intervals, update rates per parameter, serial autocorrelation on the thinned sequences and the potential scale reduction factor for convergence (see Convergence value below).
names	Names of all parameters
DIC	Basic deviance information criterion (DIC) calculations to be used for model selection (Spiegelhalter *et al.* 2002, Celeux *et al.* 2006). Small differences between values should only be used a reference (see comments in Spiegelhalter *et al.* 2002). If all or some of the simulations failed, then the returned value is "Not calculated".

`KullbackLeibler`
: If called by `summary`, list with Kullback-Leibler discrepancy matrices between pair of parameters for categorical covariates (McCulloch 1989, Burnham and Anderson 2001) and McCulloch's (1989) calibration measure. If only one simulation was run or if no convergence is reached, then the returned value is "`Not calculated`".

`PS`
: If requested, a list with summary statistics of the PDF of ages at death, including the life expectancy, lifespan inequality, lifespan equality, and Gini index. These are separated by categorical covariate and, if continuous covariates are provided, they are evaluated at the average value of each continous covariate. The list object provides a table with the mean and lower and upper 95% credible intervals and vectors of the converged and thinned values for each variable.

`mort`
: If requested or called by functions `plot` or `plot.basta` median and 95% predictive intervals for the estimated mortality rates separated by categorical covariate and calculated at the mean for each continuous covariate, if provided.

`surv`
: If requested or called by functions `plot` or `plot.basta` median and 95% predictive intervals for the estimated survival probability, separated by categorical covariate and calculated at the mean for each continuous covariate, if provided.

`dens`
: If requested, median and 95% predictive intervals for the estimated probability density function of ages at death, separated by categorical covariate and calculated at the mean for each continuous covariate, if provided.

`x`
: If requested, a vector of the ages used to calculate `mort`, `surv`, and `dens`.

`cuts`
: An index vector per categorical covariate of the ages where the survival is larger than 0.05, used for display purposes when producing the plots with function `plot`.

`convergence`
: If requested, a matrix with convergence coefficients based on potential scale reduction as described by Gelman *et al.* (2004). If only one simulation was ran, then the returned value is "`Not calculated`".

`convmessage`
: Only used with functions `summary` and `print` to indicate whether the parameter traces converged appropriately.

`runs`
: A list object with the outputs of each individual MCMC run. Used with function `plot`.

`fullpar`
: A list object with the input parameter information for the model, including starting values, priors, initial jumps, lower bound, among other. Used with functions `summary` and `print`.

`simthe`
: A list object with information on the basic mortality model. Used with function `plot`.

`jumps`
: A list object with the final jump standard deviations for each parameter.

`covs`
: A list object with general information on the type of covariates, i.e., `cat` and `cont`, and the type of `covarStruct` used.

`settings`
: If called by `summary`, this is a vector indicating the number of iterations for each MCMC, the burn in sequence, the thinning interval, and the number of simulations that were run.

`modelSpecs`
: Model specifications inidicating the `model`, the `shape`, the covariate structure and the minimum age that were specified by the user.

`lifeTable`
: A period life table calculated from the estimated times of birth (and death for "`CMR`") accounting for truncation and censoring.

Author(s)

Fernando Colchero <colchero@imada.sdu.dk>

References

Burnham, K.P. and Anderson, D.R. (2001) Kullback-Leibler information as a basis for strong inference in ecological studies. *Widlife Research*, 28, 111-119.

Celeux, G., Forbes, F., Robert, C. P., and Titterington, D. M. (2006) Deviance information criteria for missing data models. *Bayesian Analysis*, 1(4), 651-673.

Colchero, F. and J.S. Clark (2012) Bayesian inference on age-specific survival from capture-recapture data for censored and truncated data. *Journal of Animal Ecology*. 81, 139-149.

Colchero, F., O.R. Jones and M. Rebke. (2012) BaSTA: an R package for Bayesian estimation of age-specific survival from incomplete mark-recapture/recovery data with covariates. *Method in Ecology and Evolution*. 3, 466-470.

Colchero, F., *et al.* (2021) The long lives of primates and the "invariant rate of aging" hypothesis. *Nature Communications* 12:3666

Cox, D. R., and Oakes D. (1984) *Analysis of Survival Data*. Chapman and Hall, London.

Gelman, A., Carlin, J.B., Stern, H.S. and Rubin, D.B. (2004) *Bayesian data analysis*. 2nd edn. Chapman & Hall/CRC, Boca Raton, Florida, USA.

Gompertz, B. (1825) On the nature of the function expressive of the law of human mortality, and on a new mode of determining the value of life contingencies. *Philosophical Transactions of the Royal Society of London*, 115, 513-583.

King, R. and Brooks, S.P. (2002) Bayesian model discrimination for multiple strata capture-recapture data. *Biometrika*, 89, 785-806.

McCulloch, R.E. (1989) Local model influence. *Journal of the American Statistical Association*, 84, 473-478.

Pinder III, J.E., Wiener, J.G. and Smith, M.H. (1978) The Weibull distribution: a new method of summarizing survivorship data. *Ecology*, 59, 175-179.

Spiegelhalter, D.J., Best, N.G., Carlin, B.P. and van der Linde, A. (2002) Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society: Series B*, 64, 583-639.

See Also

`summary.basta`, `print.basta`, `plot.basta` to visualise summary outputs for objects of class "basta".

`bastaCMRdat` and `bastaCensDat` for examples of input CMR and census datasets, respectively.

Examples

```
## ---------- #
## CMR data:
## ---------- #
## Load data:
data("bastaCMRdat", package = "BaSTA2.0")

## Check data consistency:
checkedData  <- DataCheck(bastaCMRdat, dataType = "CMR", studyStart = 51,
                          studyEnd = 70)

## Run short version of BaSTA on the data:
```

```
out <- basta(bastaCMRdat, studyStart = 51, studyEnd = 70, niter = 100,
             burnin = 11, thinning = 10, updateJumps = FALSE)

## ------------- #
## Census data:
## ------------- #
## Load data:
data("bastaCensDat", package = "BaSTA2.0")

## Check data consistency:
checkedData  <- DataCheck(bastaCensDat, dataType = "census")

## Run short version of BaSTA on the data:
out <- basta(bastaCensDat, dataType = "census", niter = 100, burnin = 11,
             thinning = 10, updateJumps = FALSE)

## --------------------- #
## Check BaSTA outputs:
## --------------------- #
## Print results:
summary(out, digits = 3)

## Plot traces for survival parameters:
plot(out)

## Plot posterior densities of survival parameters:
plot(out, densities = TRUE)

## Plot survival and mortality curves:
plot(out, plot.type = "demorates")
```

`bastaCensDat` *Example of census data for BaSTA analysis.*

Description

Simulated census data (i.e., continuous observation of individuals) for Bayesian Survival Trajectory Analysis (BaSTA).

Usage

```
data("bastaCensDat")
```

Format

A data frame with 500 observations on the following 8 variables.

`ID` ID for each individual.

`Birth.Date` Dates of birth, formated as "`YYYY-mm-dd`".

`Min.Birth.Date` Minimum estimated dates of birth, formated as "`YYYY-mm-dd`". If the date of birth is known, then `Min.Birth.Date` is equal to `Birth.Date`.

`Max.Birth.Date` Maximum estimated dates of birth, formated as "`YYYY-mm-dd`". If the date of birth is known, then `Max.Birth.Date` is equal to `Birth.Date`

`Entry.Date` Dates of entry to the study, formated as "`YYYY-mm-dd`".

`Depart.Date` Dates of departure from the study, formated as "`YYYY-mm-dd`".

`Depart.Type` a character vector indicating whether the `Depart.Date` is because of death (i.e., "`D`") or censored (i.e., "`C`").

`Sex` a character vector indicating the sex covariate.

Details

This dataset was created by stochastically simulating a hypothetical population with different mortality patterns between males and females and with proportional decreases in mortality as a function of a hypothetical continuous covariate (e.g. birth weight, average adult weight, etc.). The population was simulated for 40 years, with uniform times of birth within this period. Sex ratios were assumed to be 1:1. The time of death for each individual was inversed sampled from a Gompertz CDF of ages at death. The Gompertz parameters for females were: $b_0 = -3$ and $b_1 = 0.15$; and for males at $b_0 = -2$ and $b_1 = 0.2$.

The resulting dataset includes individuals where the data are left-truncated and/or right-censored. This is typical of capture mark recovery datasets.

Examples

```
## Load data:
data("bastaCensDat", package = "BaSTA2.0")

## Check data consistency:
checkedData  <- DataCheck(bastaCensDat, dataType = "census")
```

---

bastaCensOut *Output from a Bayesian Survival Trajectory Analysis (BaSTA) analysis on a simulated census dataset.*

Description

This dataset is the output of a BaSTA analysis on the simulated dataset `bastaCensDat`. The analysis consisted of four independent simulations run in parallel. The model chosen was Gompertz ("`GO`") with the `shape` argument set to "`simple`" and `covarStruct` set to "`fused`".

Usage

```
data("bastaCensOut")
```

Format

A BaSTA output list (see details in `basta`).

Examples

```
## Load BaSTA output:
data("bastaCensOut", package = "BaSTA2.0")

## Plot traces for survival parameters:
plot(bastaCensOut)

## Plot posterior densities of survival parameters:
plot(bastaCensOut, densities = TRUE)

## Plot survival and mortality curves:
plot(bastaCensOut, plot.type = "demorates")
```

---

bastaCMRdat *Example of capture-mark-recapture data for BaSTA analysis.*

Description

Simulated capture-mark-recapture (CMR) data for Bayesian Survival Trajectory Analysis (BaSTA).

Usage

```
data("bastaCMRdat")
```

Format

A data frame with 500 observations on the following 25 variables (see `details`).

Details

This dataset was created by stochastically simulating a hypothetical population with different mortality patterns between males and females and with proportional decreases in mortality as a function of a hypothetical continuous covariate (e.g. birth weight). The population was simulated for 100 years, at each one of which 100 individuals were born. The number of females per generation was randomly drawn from a binomial distribution using function `rbinom` with probability of 0.5 (i.e. 1:1 sex ratio). The individual continuous covariate was randomly drawn from a random normal distribution (with function `rnorm`) with mean parameter equal to 0 (e.g. anomaly of weights) and standard deviation equal to 1. The time of death for each individual was inversed sampled from a Gompertz CDF of ages at death. The Gompertz parameters for females were: $b_0 = -4$ and $b_1 = 0.15$; and for males at $b_0 = -3$ and $b_1 = 0.15$. The gamma parameter for the continuous covariate was $\gamma = 0.2$.

The study was assumed to start at year 51 and to finish at year 70. Recapture probability was set to 0.6 and thus each observation per individual was randomly drawn from a Bernoulli trial with parameter $p = 0.6$. Captures at birth and recoveries were randomly drawn from a Bernoulli trial with parameters $p = 0.5$ and $p = 0.2$, respectively.

Therefore, the resulting dataset includes individuals where the data are left-truncated and/or right-censored. This is typical of capture mark recovery datasets.

Here is the description of each column:

`id`: ID for each individual.

`birth`: Integer times of birth or `NA` when unknown.

death: Integer times of death or NA when unknown.

Y.51 - Y.70: Columns of the recapture matrix

Sex: Character string indicating the sex for the individual record (i.e., "Female" and "Male")

Weight: Simumated body weights at birth.

Examples

```
## Load data:
data("bastaCMRdat", package = "BaSTA2.0")

## Check data consistency:
checkedData  <- DataCheck(bastaCMRdat, dataType = "CMR", studyStart = 51,
                          studyEnd = 70)
```

---

bastaCMRout *Output from a Bayesian Survival Trajectory Analysis (BaSTA) analysis on a simulated capture-mark-recapture (CMR) dataset.*

Description

This dataset is the output of a BaSTA analysis on the simulated dataset bastaCMRdat. The analysis consisted of four independent simulations run in parallel. Each simulation was run for 20,000 iterations. The model chosen was Gompertz ("GO") with the shape argument set to "simple" and covarStruct set to "fused".

Usage

```
data("bastaCMRout")
```

Format

A BaSTA output list (see details in basta).

Examples

```
## Load BaSTA output:
data("bastaCMRout", package = "BaSTA2.0")

## Plot traces for survival parameters:
plot(bastaCMRout)

## Plot posterior densities of survival parameters:
plot(bastaCMRout, densities = TRUE)

## Plot traces for proportional hazards parameter:
plot(bastaCMRout, trace.name = "gamma")

## Plot survival and mortality curves:
plot(bastaCMRout, plot.type = "demorates")
```

CensusToCaptHist *Constructs a capture-history matrix from repeated individual observations to be used in Bayesian Survival Trajectory Analysis (BaSTA).*

Description

This function takes a table of repeated observations per individual (e.g., discrete interval census), consisting of a pair of vectors with ID and observation date (e.g., year), and converts it into a capture-history matrix to be used as part of a BaSTA CMR data input.

Usage

```
CensusToCaptHist(ID, d, dformat = "%Y", timeInt = "Y")
```

Arguments

ID	A vector with individual IDs.
d	A vector of dates when each individual was observed (see details).
dformat	Defines the date format for d when d is of class character (see details).
timeInt	A one character string specifying which time interval should be used between capture occasions. Arguments are "Y" for years, "M" for months, "W" for weeks and "D" for days (see details)..

Details

The d argument can be specified as an object of class POSIXct or POSIXlt, as a vector of integer time intervals or as a character string indicating the day, month and year. (e.g. dd/mm/yyyy, mmddyyyy, mm-dd-yyyy etc.). When d is of class character then argument dformat needs to be specified using the same conventions as in function format.POSIXct for objects of class POSIXct or POSIXlt.

Author(s)

Owen R. Jones <jones@biology.sdu.dk>.

See Also

DataCheck for running a data check on the input data for function basta.FixCMRdata to fix potential issues for capture-mark-recapture data. bastaCMRdat for an example of input CMR datasets.

Examples

```
## Create a simulated vector of repeated IDs:
IDvec <- sort(sample(1:5, size = 15, replace = TRUE))

## Simulate dates (e.g., years) of observation per individual:
dVec <- rep(0, length(IDvec))
for(i in unique(IDvec)) {
  svec <- which(IDvec == i)
  dVec[svec] <- sort(sample(1990:1995, length(svec)))
}
```

```
## Construct the capture-recapture matrix:
Y <- CensusToCaptHist(ID = IDvec,  d = dVec)
```

`DataCheck` *Error checking for BaSTA input data.*

Description

A function to check the input data file for a Bayesian Survival Trajectory Analysis (BaSTA) for capture-mark-recapture (CMR) or census data.

Usage

```
DataCheck (object, dataType = "CMR", studyStart = NULL, studyEnd = NULL,  silent = TRUE)
```

Arguments

`object`	A `data.frame` to be used as an input data file for BaSTA. Note: BaSTA can take two types of datasets, namely capture-mark-recapture (`CMR`) or census data.
`dataType`	A `character string` indicating if the data are capture-mark-recapture (CMR) or census. Options are "`CMR`" (default) or "`census`".
`studyStart`	Only required for `dataType` = "`CMR`", an integer indicating the first year of the study.
`studyEnd`	Only required for `dataType` = "`CMR`", an integer indicating the last year of the study.
`silent`	Logical to indicate whether the results should be printed to the console.

Details

The function checks for inconsistencies in the dataset and reports them back. See `value` section for details on the types of errors detected by the function.

`DATA SPECIFICATIONS:`

`1) CMR data:` The input data `object` requires the following structure: the first column should be a vector of individual unique IDs, the second and third columns are birth and death years respectively. Columns $4, \ldots, T+3$ represent the observation window (i.e., recapture matrix) of T years. This is followed (optionally) by columns for categorical and continuous covariates.

`2) census data:` The input data `object` requires at least five dates columns, namely "Birth.Date", "Min.Birth.Date", "Max.Birth.Date", "Entry.Date", and "Depart.Date". All dates need to be format as "%Y-%m-%d". In addition, a "Depart.Type" column is required with two types of departures "C" for Censored and "D" for dead.

Value

`1) CMR data:`

`newData`	The original data frame (for consistency with previous versions of BaSTA).
`type1`	A vector of row numbers in the original data frame where there are deaths occurring before the study starts.

type2	A vector of row numbers in the original data frame where there are no birth/death AND no obervations.
type3	A vector of row numbers in the original data frame where there are births recorded after death.
type4	A vector of row numbers in the original data frame where there are observations (i.e. recaptures) after death.
type5	A vector of row numbers in the original data frame where there are observations (i.e. recaptures) before birth.
type6	A vector of row numbers in the original data frame where the year of birth is not a zero in the recapture matrix.
summary	List with summary information, e.g., sample size, number of records with known birth, number of records with known death, etc.
stopExec	Logical that indicates if the data are free of errors or not. i.e. TRUE = the data have no apparent errors, and FALSE = there is at leat one error.
probDescr	Character vector explaining the six types of problems the DataCheck functions looks for.
dataType	Type of dataset, i.e., "CMR".
studyStart	Integer indicating the study start time.
studyEnd	Integer indicating the study end time.

2) census data:

n	Integer for the number of rows (i.e., records) in the dataset.
stopExec	Logical that indicates if the data are free of errors or not. i.e. TRUE = the data have no apparent errors, and FALSE = there is at leat one error.
nas	List organised by column indicating whether NAs were detected in a given column.
DateRan	Matrix of dates ranges (as character strings) for each date column in the dataset.
probDescr	Character vector explaining the seven types of problems the DataCheck functions looks for.
MinBBirth	Vector of indices of rows where "Min.Birth.Date" was larger than "Birth.Date".
BirthMaxB	Vector of indices of rows where "Birth.Date" was larger than "Max.Birth.Date".
MinBMaxB	Vector of indices of rows where "Min.Birth.Date" was larger than "Max.Birth.Date".
BirthEntr	Vector of indices of rows where "Birth.Date" was larger than "Endtry.Date".
MinBEntr	Vector of indices of rows where "Min.Birth.Date" was larger than "Entry.Date".
MaxBEntr	Vector of indices of rows where "Max.Birth.Date" was larger than "Entry.Date".
EntrDep	Vector of indices of rows where "Entry.Date" was larger than "Depart.Date".
DepartType	Vector of indices of rows where "Depart.Type" does not fall within the "C" (i.e., censored) or "D" (i.e., uncensored or death) categories.
idUnCens	Vector of indices of rows for uncensored (i.e., death) records.
nUnCens	Integer indicating the number of uncensored records.
idCens	Vector of indices of rows for censored records.
nCens	Integer indicating the number of uncensored records.
idNoBirth	Vector of indices of rows for records with uncertain birth date.
nNoBirth	Integer indicating the number of records with uncertain birth date.

Author(s)

Fernando Colchero <colchero@imada.sdu.dk>

See Also

FixCMRdata to fix potential issues for capture-mark-recapture data.

Examples

```
## CMR data:
## --------- #
## Load data:
data("bastaCMRdat", package = "BaSTA2.0")

## Check data consistency:
checkedData  <- DataCheck(bastaCMRdat, dataType = "CMR", studyStart = 51,
                          studyEnd = 70)

## census data:
## ------------ #
## Load data:
data("bastaCensDat", package = "BaSTA2.0")

## Check data consistency:
checkedData  <- DataCheck(object = bastaCensDat, dataType = "census")

## Printed output:
## --------------- #
## Print DataCheck results:
print(checkedData)
```

FixCMRdata *Fix issues on CMR input data for BaSTA.*

Description

This function provides general fixes to common issues of capture-mark-recapture (CMR) data.

Usage

```
FixCMRdata(object, studyStart, studyEnd, autofix = rep(0, 6),
                        silent = TRUE)
```

Arguments

object	A data.frame to be used as an input data file for BaSTA for dataType = "CMR". The first column is the individual's ID, the second and third columns are birth and death years respectively. Columns 4 to nt+3 represent the observation window of nt years. This is followed (optionally) by columns for covariate.
studyStart	An integer indicating the first year of the study.
studyEnd	An integer indicating the last year of the study.

autofix	A vector argument with a length of 6 indicating whether to automatically fix any errors (see details). This should be used with extreme caution. We recommend going back to the individual-based data and fixing each error "by hand".
silent	A logical argument indicating whether to print a detailed report to the screen or not.

Details

Argument autofix allows the user to fix the potential errors by specifying a code for each fix. Below are the descriptions of the actions that are taken depending on the error type and the fix code:

Type 1: 0 = do nothing; 1 = remove from dataframe.

Type 2: 0 = do nothing; 1 = remove from dataframe.

Type 3: 0 = do nothing; 1 = replace death records with 0; 2 = replace birth records with 0; 3 = replace both birth and death records with 0.

Type 4: 0 = do nothing; 1 = remove spurious post-death observations.

Type 5: 0 = do nothing; 1 = remove observations that pre-date year of birth.

Type 6: 0 = do nothing; 1 = replace birth year element of observation matrix with 0.

Value

For dataType = "CMR":

newData	A corrected data frame.
type1	A vector of row numbers in the original data frame where there are deaths occurring before the study starts.
type2	A vector of row numbers in the original data frame where there are no birth/death AND no obervations.
type3	A vector of row numbers in the original data frame where there are births recorded after death.
type4	A vector of row numbers in the original data frame where there are observations (i.e. recaptures) after death.
type5	A vector of row numbers in the original data frame where there are observations (i.e. recaptures) before birth.
type6	A vector of row numbers in the original data frame where the year of birth is not a zero in the recapture matrix.
stopExec	Logical that indicates if the data are free of errors or not. i.e. TRUE = the data have no apparent errors, and FALSE = there is at leat one error.

Author(s)

Fernando Colchero <colchero@imada.sdu.dk>

See Also

DataCheck for running a data check on the input data for function basta.

Examples

```
## Load data:
data("bastaCMRdat", package = "BaSTA2.0")

## Fix data:
fixedData  <- FixCMRdata(bastaCMRdat, studyStart = 51,
                         studyEnd = 70, autofix = rep(1, 6))
```

`summary.basta` *Summarizing and plotting Bayesian Survival Trajectory Analysis (BaSTA) model outputs.*

Description

These functions are all generic methods for class `basta`.

Usage

```
## S3 method for class 'basta'
summary(object, ...)
## S3 method for class 'basta'
print(x, ...)
## S3 method for class 'basta'
plot(x, plot.type = "traces", trace.name = "theta",
                       densities = FALSE, noCIs = FALSE, ...)
```

Arguments

`object`	An object of class `basta`.
`x`	An object of class `basta`.
`plot.type`	A character vector indicating the type of plot to be produced. Options are: "`traces`" for the MCMC traces; "`demorates`" for the resulting survival and mortality curves; and "`gof`" for a comparison between the estimated parametric survival function and the life table `lx` variable.
`trace.name`	Character string indicating the set of parameters or posteriors to be plotted. The options are: "`theta`" to plot the survival model parameters; "`gamma`" to plot the proportional hazards parameters (if it applies, else `plot.basta` returns an error); "`pi`" to plot the recapture probabilities.
`densities`	Logical indicating whether to plot the parameter posterior densities instead of the traces.
`noCIs`	Logical indicating whether the 95% credible intervals should be included in the plot.
`...`	Additional arguments passed to functions `print`, `summary` and `plot` (see details).

Details

For objects of class basta, the print function returns three summary elements describing the model and its results, namely: call, run, coefficients and, if convergence was reached, the DIC values for model fit. call describes the basic model used (i.e. exponential, Gompertz, Weibull or logistic), the shape chosen, "simple", "Makeham" or "bathtub", the covariate structure chosen, "fused", "prop.haz" or "all.in.mort" and which covariates (if any) were categorical and which continuous. Argument digits can be used for number formatting (see summary() or signif() for details).

The summary element coefficients prints out the estimated coefficients for all parameters in the model, as well as their standard errors and 95% upper and lower credible intervals. It also includes a measure of serial autocorrelation for each parameter calculated from the thinned parameter chains, an update rate per parameter, and the potential scale reduction factor for each parameter as a measure of convergence (Gelman *et al.* 2004).

Function summary includes all the previous elements, as well as a summary description of the priors and jump standard deviations for all survival parameters, a calibration of the Kullback-Leibler discrepancy as a measure of parameter similarities for those parameters associated to categorical covariates (McCulloch 1989), and a measure of model fit based on the deviance information criterion (DIC) (Spiegelhalter *et al.* 2002).

Function plot takes objects of class basta to create trace plots or, if the argument for plot.type is set to "demorates", it plots estimated survival probabilities and mortality rates with their 95% predictive intervals. If argument plot.trace is set to FALSE, argument xlim can be used to define a range of ages to visualize survival and mortality trends. Also, if logical argument noCI is set to TRUE, credible intervals around survival and mortality are not plotted, leaving only the mean trends. This can be handy when several categorical covariates have been evaluated and the plots get too crowded.

Other arguments for plot include names.legend to indicate alternative names for the legend of vital rates plots. Also, when plot.trace is FALSE, argument xlim can be changed to display only a subest of the support. When argument noCI is TRUE, then the credible intervals around the vital rates are not displayed.

Value

Function summary() outputs the folowing values:

coefficients	A matrix with estimated coefficients (i.e. mean values per parameter on the thinned sequences after burnin), which includes standard errors, upper and lower 95% credible intervals, update rates per parameter (commonly the same for all survival and proportional hazards parameters), serial autocorrelation on the thinned sequences and the potential scale reduction factor for convergence (see Convergence value below).
DIC	Basic deviance information criterion (DIC) calculations to be used for model selection (Spiegelhalter *et al.* 2002).
KullbackLeibler	List with Kullback-Leibler discrepancy matrices between pair of parameters for categorical covariates (McCulloch 1989, Burnham and Anderson 2001) and McCulloch's (1989) calibration measure. If only one simulation was ran or if no convergence was reached, then the returned value is "Not calculated".
convergence	A matrix with convergence coefficients based on potential scale reduction as described by Gelman *et al.* (2004). If only one simulation was ran, then the returned value is "Not calculated".

modelSpecs	Model specifications inidicating the model, the shape and the covariate structure that were specified by the user.
settings	A vector indicating the number of iterations for each MCMC, the burn in sequence, the thinning interval, and the number of simulations that were run.

Author(s)

Fernando Colchero <colchero@imada.sdu.dk>, Owen R. Jones <jones@biology.sdu.dk> and Maren Rebke <maren.rebke@avitec-research.de>

References

Gelman, A., Carlin, J.B., Stern, H.S. and Rubin, D.B. (2004) *Bayesian data analysis*. 2nd edn. Chapman & Hall/CRC, Boca Raton, Florida, USA.

McCulloch, R.E. (1989) Local model influence. *Journal of the American Statistical Association*, 84, 473-478.

Spiegelhalter, D.J., Best, N.G., Carlin, B.P. and Van Der Linde, A. (2002) Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society: Series B* 64, 583-639.

See also:

Colchero, F. and J.S. Clark (2012) Bayesian inference on age-specific survival from capture-recapture data for censored and truncated data. *Journal of Animal Ecology*. 81(1):139-149.

Colchero, F., O.R. Jones and M. Rebke. (2012) BaSTA: an R package for Bayesian estimation of age-specific survival from incomplete mark-recapture/recovery data with covariates. *Method in Ecology and Evolution*. DOI: 10.1111/j.2041-210X.2012.00186.x

See Also

basta

Examples

```
## Load BaSTA output:
data("bastaCMRout", package = "BaSTA2.0")

## Print summary output:
summary(bastaCMRout)

## Plot traces for mortality parameters (theta):
plot(bastaCMRout)

## Plot traces for proportional hazards parameters (gamma):
plot(bastaCMRout, trace.name = "gamma")

## Plot traces for recapture probability(ies) (pi):
plot(bastaCMRout, trace.name = "pi")

## Plot predicted mortality and survival:
plot(bastaCMRout, plot.type = "demorates")

## Change the color for each covariate on
## the predicted vital rates:
plot(bastaCMRout, plot.type = "demorates",
     col = c("dark green", "dark blue"))
```

```
## Change the color and the legend text:
plot(bastaCMRout, plot.type = "demorates",
     col = c("dark green", "dark blue"),
     names.legend = c("Females", "Males"))

## Plot predicted mortality and survival
## between 2 and 8 years of age:
plot(bastaCMRout, plot.trace = FALSE, xlim = c(2, 8))

## Plot predicted mortality and survival
## between 2 and 8 years of age without
## credible intervals:
plot(bastaCMRout, plot.trace = FALSE, xlim = c(2, 8),
     noCI = TRUE)

## Plot parameter densities and predicted vital
## rates in the same plot (i.e. fancy):
plot(bastaCMRout, fancy = TRUE)

## Change colors and legend names for the
## "fancy" plot:
plot(bastaCMRout, fancy = TRUE, col = c("dark green", "dark blue"),
     names.legend = c("Females", "Males"))
```

`summary.bastaCheckCMR` *Summary of outputs from the data checking function in BaSTA.*

Description

These function provide general information about the input data set to be used for BaSTA, as well as detailes on potential issues that could have been found on the data.

Usage

```
## S3 method for class 'bastaCheckCMR'
summary(object, ...)

## S3 method for class 'bastaCheckCens'
summary(object, ...)

## S3 method for class 'bastaCheckCMR'
print(x, ...)

## S3 method for class 'bastaCheckCens'
print(x, ...)
```

Arguments

`object`	Output from function `DataCheck`
`x`	Output from function `DataCheck`
`...`	Additional arguments passed to functions `print` and `summary` (see details).

Details

Both functions print to the screen the results of the DataCheck function.

Author(s)

Fernando Colchero <colchero@imada.sdu.dk>

See Also

DataCheck for running a data check on the input data for function basta, FixCMRdata to fix potential issues for capture-mark-recapture data.

Examples

```
## CMR data:
## --------- #
## Load data:
data("bastaCMRdat", package = "BaSTA2.0")

## Check data consistency:
checkedData  <- DataCheck(bastaCMRdat, dataType = "CMR", studyStart = 51,
                          studyEnd = 70)

## census data:
## ------------ #
## Load data:
data("bastaCensDat", package = "BaSTA2.0")

## Check data consistency:
checkedData  <- DataCheck(object = bastaCensDat, dataType = "census")

## Printed output:
## --------------- #
## Print DataCheck results:
print(checkedData)
```

Index