Package 'baytaAAR'

June 15, 2026

Type Package

Title Bayesian Transition Analysis with Markov Chain Monte Carlo

Version 1.0.3

Maintainer Nils Müller-Scheeßel <nils.mueller-scheessel@ufg.uni-kiel.de>

Description Provides Bayesian age estimation for bioarchaeological skeletal data using ordinal probit regression models implemented in 'JAGS' and 'NIMBLE'. The package is designed to handle multiple ordinal traits of adult individuals and incorporates a Gompertz prior on age to reflect population-level mortality. It accounts for estimation uncertainties and supports full customization of model parameters and Markov Chain Monte Carlo settings. For more details see Müller-Scheeßel et al. (2026) <doi:10.1002/ajpa.70289>.

License GPL-3

Encoding UTF-8

LazyData true

Imports tidyr, Rdpack, dplyr, scoringRules, ggpubr, flexsurv, coda, checkmate, stats

RoxygenNote 7.3.3

Suggests testthat (>= 3.0.0), withr, knitr, rmarkdown, ggplot2, gridExtra, bayesplot, tidybayes, rjags, runjags, ggridges

Depends R (>= 4.1.0), nimble

RdMacros Rdpack

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/ISAAKiel/baytaAAR, https://isaakiel.github.io/baytaAAR/

BugReports https://github.com/ISAAKiel/baytaAAR/issues

Config/Needs/website rmarkdown

NeedsCompilation no

Author Nils Müller-Scheeßel [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-7992-8722>),
Christoph Rinne [aut] (ORCID: <https://orcid.org/0000-0002-9829-6182>)

Repository CRAN

Date/Publication 2026-06-15 13:40:15 UTC

Contents

`age.comp.plot` *Plots of quality measures of age estimation*

Description

Visualisation of the difference between estimated and known age with the help of a combination of plots.

Usage

```
age.comp.plot(
  x,
  age_identifier = "age.s",
  known_age,
  mean_choice = "Mode",
  hdi_color = c("chartreuse4", "coral2")
)
```

Arguments

x	output from the function `diagnostic.summary()`.
age_identifier	a character string of either "age.s" or "age.s_c" to select the uncalibrated or calibrated age estimates. Default: "age.s".
known_age	a vector of known age-at-death. NAs are allowed and those entries will subsequently be ignored.
mean_choice	a character string of either "Mean", "Median" or "Mode". Default: "Mode".
hdi_color	a character vector of exactly two entries with color values to differentiate estimated ages within the HDI from those outside the HDI. Default: c("chartreuse4", "coral2")

Value

A ggplot object with 2 x 2 single plots, showing:

Top left Comparison of estimated *highest density intervals* with known ages, color1 = age within HDI, color2 = age outside HDI, individuals ordered according to known age-at-death.

Top right Comparison of the density of known ages with a Gompertz function derived from the arithmetic mean of the estimated population parameters α and β.

Bottom left Scatter plot of known and estimated ages with regression line in blue. The dotted line marks perfect equivalence.

Bottom right Slope of the regression line from the left bottom image (cf. goodness-of-fit measure `Residual_slope` from the function `age.comp.summary()`).

Examples

```
# select Spitalfields data with multiple traits
spitalfields_traits <- spitalfields[,c(2:6)]

# example with multinormal likelihood, please be patient
spitalfields_res <- bay.ta(algorithm = "mnorm",
method = spitalfields_traits)

# compute age summary statistics
age.comp.plot(spitalfields_res, known_age = spitalfields$Age)
```

---

age.comp.summary *Quality measures of age estimation*

Description

Comparison of estimated age and known age-at-death with the help of several Goodness-of-fit measures. For most of the measures smaller is better. The only exception is *corrPearson* where larger is better.

Usage

```
age.comp.summary(
  mcmc_list,
  known_age,
  mean_choice = "Mode",
  age_identifier = "age.s",
  ...
)
```

Arguments

mcmc_list	MCMC output from coda chains.
known_age	a vector of known age-at-death. NAs are allowed and those entries will subsequently be ignored.
mean_choice	a character string of either "Mean", "Median" or "Mode". Default: "Mode".
age_identifier	a character string of either "age.s" or "age.s_c" to select the uncalibrated or calibrated age estimates. Default: "age.s".
...	Arguments passed on to diagnostic.summary HDImass numeric. Value within 0 and 1. Default = 0.95. gelman_diag logical. If TRUE, the Gelman-Rubin diagnostics for computing the PSRF is invoked. Default: TRUE.

Value

A data.frame with one row and eight columns with age estimation quality parameters as follows:

- Bias Arithmetic mean of the difference between known and estimated age.
- corrPearson Correlation of known and estimated age.
- corr_p p-value of the correlation of known and estimated age.
- Residual_slope Slope of the regression line of the difference between known and estimated age.
- Inaccuracy Arithmetic mean of the absolute difference between known and estimated age.
- RMSE *Root mean square error* of known and estimated age.
- TMNLP *Test mean log posterior*, a local evaluation of the probability density at the point of known age.
- CRPS *Continuous ranked probability score*, a global evaluation of the probability density at the point of known age.

Examples

```
# select Spitalfields data with multiple traits
spitalfields_traits <- spitalfields[,c(2:6)]

# example with multinormal likelihood, please be patient
spitalfields_res <- bay.ta(framework = "NIMBLE", algorithm = "mnorm",
```

```
method = spitalfields_traits)

# compute age summary statistics
age.comp.summary(spitalfields_res, known_age = spitalfields$Age)
```

`age.estim.summary` *Summary of age estimates*

Description

Convenience function to quickly extract the age-related estimates from the result of the function `diagnostics.summary()`.

Usage

```
age.estim.summary(x, age_identifier = "age.s")
```

Arguments

`x`	output from function `diagnostics.summary()`
`age_identifier`	a character string of either "age.s" or "age.s_c" to select the uncalibrated or calibrated age estimates. Default: "age.s".

Value

A data.frame with mean, median and mode as well as the HDI ranges as specified in the output of `diagnostics.summary()` as columns and the following rows:

- `b` Gompertz parameter β.
- `a` Gompertz parameter α.
- `M` Modal age, derived from the Gompertz parameters α and β according to the equation (1 / β) * log(β / α) + minimum_age.
- `age_mean` Mean ages.
- `hdi_diff` *Highest density intervals*.

Examples

```
# select Sorsum data with auricular surface after Lovejoy et al. 1985
sorsum <- sorsum_as[,2]

# example with default settings, please be a little bit patient
sorsum_res <- bay.ta(method = sorsum)

sorsum_diag <- diagnostic.summary(sorsum_res)

# show summary of age-related estimates
age.estim.summary(sorsum_diag)
```

bay.ta *Bayesian Transition Analysis with JAGS or NIMBLE*

Description

bay.ta() implements latent trait analysis within a Bayesian Markov Chain Monte Carlo (MCMC) framework. It is intended to estimate the age-of-death of adult individuals for whom one or several ordinal traits have been assessed. It produces probability densities for the individual ages but also for the respective population as a whole. bay.ta() has been introduced and tested by Müller-Scheeßel et al. (2026).

Usage

```
bay.ta(
  framework = "NIMBLE",
  algorithm = "norm",
  multicore = FALSE,
  seed = as.integer(format(Sys.Date(), "%Y%m%d")),
  method,
  eta = 1,
  gomp_b = NA,
  error_sd = NA,
  minimum_age = 15,
  maximum_age = 100,
  parameters = c("b", "a", "beta0", "beta", "thresh", "age.s"),
  nChains = 3,
  adaptSteps = 2000,
  burnInSteps = 3000,
  thinSteps = 1,
  numSavedSteps = 10000,
  silent.jags = FALSE,
  silent.runjags = FALSE,
  verbose = TRUE
)
```

Arguments

framework	character string. Either JAGS or NIMBLE. Default: NIMBLE.
algorithm	character string. Either norm for 'simple' ordered regression or mnorm for multinormal ordered regression. Default: norm.
multicore	TRUE/FALSE. If TRUE each chain is assigned to a dedicated core. Default: FALSE.
seed	integer. Random number for reproducibility. In parallel processing, each cluster automatically gets different seeds. If no seed is specified, the value is set to today's date as integer.
method	matrix of integers, converted to matrix if not already matrix. Ordinal trait(s) for age estimation.

eta	numeric. Parameter for the LKJ distribution, must be > 0. Only used for multinormal ordered regression for the correlation matrix. 1 implies equal correlations, lower values assume stronger correlations. Default: 1.
gomp_b	numeric. Optional prior for parameter Gompertz beta. Default: NA.
error_sd	numeric. Optional error parameter for age estimates. Default: NA.
minimum_age	numeric. Minimum age for Gompertz distribution. Default: 15.
maximum_age	numeric. Maximum age for Gompertz distribution. Default: 100.
parameters	vector of character strings. Parameters to monitor.
nChains	integer. Number of chains. Default: 3.
adaptSteps	integer. Number of adaptation steps, ignored when framework is set to NIMBLE. Default: 2000.
burnInSteps	integer. Number of steps for burn-in. Default: 3000.
thinSteps	integer. Thinning, i. e. which *i*th step should be saved. Default: 1 (no thinning).
numSavedSteps	integer. Number of saved steps. Default: 10000. The total number of steps equals thinSteps × numSavedSteps.
silent.jags	TRUE/FALSE. Silent mode to run JAGS. Default: FALSE. Ignored when framework is set to NIMBLE.
silent.runjags	TRUE/FALSE Silent mode to run runjags. Default: FALSE. Ignored when framework is set to NIMBLE.
verbose	TRUE/FALSE. If TRUE, the current time stamp is displayed in the console, and after completion the elapsed time. Default: TRUE.

Details

bay.ta() is a wrapper for the functions bay.ta.jags() and bay.ta.nimble(). NIMBLE allows the user to run models with multinormal ordered regression, also with parallel clusters. In this respect, however, JAGS tends to be more stable. The latter presupposes, however, that you have installed JAGS outside of R.

Value

A list of MCMC chains of class coda::mcmc.list.

Data requirements

As input, bay.ta() assumes a matrix of trait expressions. In its simplest form, this may contain only one column with a single trait. NAs are allowed but neither must all entries in any of the rows be NA nor can this be the case for one or several of the columns. bay.ta will reject to run in such cases, and the offending rows or columns need to be removed from analysis. Please see the article on Chelsea 'Old church' for an example how this can be accomplished. The levels of all traits must start at 1. Binary traits are possible. Mixing of levels like 1.5 as short-cut for a trait-expression between 1 and 2, however, should be an absolute no-go as this would violate basic principles of ordinal scaling. Thus, for such cases a decision for one of the neighboring levels has to be made or they need to be set to NA. The nodes (= rows of the matrix) do not have to be fully observed for the multinormal model to run because with NIMBLE vers. 1.4.1., the NIMBLE team introduced a sampler for only partly observed multivariate normal random variables.

References

Müller-Scheeßel N, Rinne C, Fuchs K (2026). "A Fully Bayesian Approach to Adult Skeletal Age Estimation: Multivariate Latent Trait Modeling with Markov Chain Monte Carlo Sampling." *American Journal of Biological Anthropology*, **190**(2), e70289. doi:10.1002/ajpa.70289.

Examples

```
# select Sorsum data with auricular surface after Lovejoy et al. 1985
sorsum <- sorsum_as[,2]

# example with default settings
sorsum_res <- bay.ta(method = sorsum)

# example with framework JAGS
sorsum_res <- bay.ta(framework = "JAGS", method = sorsum)

# example with framework JAGS and multiple cores (parallel computing)
sorsum_res <- bay.ta(framework = "JAGS", multicore = TRUE, method = sorsum)

# example with 10,000 saved iterations and a thinning of 10 (= 100,000
# iterations)
sorsum_res <- bay.ta(method = sorsum, numSavedSteps = 10000, thin = 10)

# select Spitalfields data with multiple traits
spitalfields_traits <- spitalfields[,c(2:6)]

# example with multinormal likelihood, please be patient
spitalfields_res <- bay.ta(falgorithm = "mnorm",
method = spitalfields_traits)
```

---

corr.mat.mean *Extract correlation matrix from Cholesky factor*

Description

As the LKJ prior for the correlation matrix uses the Cholesky decomposition of the correlation matrix, getting the correlation indices from the coda chains is less straightforward than it seems. It involves taking the cross product from the resulting coda estimates.

Usage

```
corr.mat.mean(mcmc_list)
```

Arguments

mcmc_list	MCMC output from coda chains.

Value

A symmetric matrix with correlations between traits. The number of rows and columns corresponds to the number of traits.

Examples

```
# select Spitalfields data with multiple traits
spitalfields_traits <- spitalfields[,c(2:6)]

# example with multinormal likelihood, please be patient
spitalfields_res <- bay.ta(algorithm = "mnorm",
method = spitalfields_traits)

# compute correlation matrix
corr.mat.mean(spitalfields_res)
```

`diagnostic.summary` *Diagnostic summary of MCMC samples*

Description

Summarising diagnostics from a `coda::mcmc.list`, partly derived from *Kruschke 2015*.

Usage

```
diagnostic.summary(mcmc_list, HDImass = 0.95, gelman_diag = TRUE)
```

Arguments

`mcmc_list`	MCMC output from coda chains.
`HDImass`	numeric. Value within 0 and 1. Default = 0.95.
`gelman_diag`	logical. If TRUE, the Gelman-Rubin diagnostics for computing the PSRF is invoked. Default: TRUE.

Details

Because the first threshold is fixed, the Gelman-Rubin multivariate PSRF will always throw an error, so this is automatically set to `FALSE`. If the gelman diagnostics still produce an error, deactivate `gelman_diag` altogether by setting it to `FALSE`, too.

Value

A data.frame of class `diagnostic_summary` with the row names according to the parameters to be monitored and the following numeric columns:

- `PSRF Point est.` *Potential scale reduction factor* (= Gelman-Rubin statistic), a measure of the mixing of chains.
- `PSRF Upper C.I.` The upper limit of the 0.95-confidence interval of the PSRF.
- `Mean` Arithmetic mean of the estimates.
- `Median` Median of the estimates.
- `Mode` Mode of the estimates.
- `ESS` *Effective sample size*, a control of autocorrelation.
- `MCSE` *Monte Carlo standard error*.
- `HDImass` Credibility level of the *highest density interval*.
- `HDIlow` Start of the *highest density interval*.
- `HDIhigh` End of the *highest density interval*.

References

Kruschke JK (2015). *Doing Bayesian data analysis: a tutorial with R, JAGS, and Stan*. Academic Press, Amsterdam.

Examples

```
# select Sorsum data with auricular surface after Lovejoy et al. 1985
sorsum <- sorsum_as[,2]

# example with default settings, please be patient
sorsum_res <- bay.ta(method = sorsum)

# compute diagnostics of the MCMC samples
sorsum_diag <- diagnostic.summary(sorsum_res)

# show first rows
head(sorsum_diag)
```

`diagnostics.max.min` *Maximum and minimum diagnostic values*

Description

Convenience function to quickly extract maximum and mininum diagnostics values of the function `diagnostics.summary()` over all parameters. The maximum values of the PSRF should be below 1.1 while the minumum ESS should be above 10,000. If either of this is not the case, consider to increase the length of the chains, i. e. the number of iterations.

Usage

```
diagnostics.max.min(x)
```

Arguments

x	output from function `diagnostics.summary()`

Value

A data.frame with one row and the following numeric columns:

- `PSRF_max` Maximum value of the *potential scale reduction factor*.
- `PSRF_upper_max` Mximum value of the upper limit of the 0.95-confidence interval of the PSRF.
- `ESS_min` Minimum of the *effective sample size*.

Examples

```
# select Sorsum data with auricular surface after Lovejoy et al. 1985
sorsum <- sorsum_as[,2]

# example with default settings, please be atient
sorsum_res <- bay.ta(method = sorsum)

# compute diagnostics of the MCMC samples
sorsum_diag <- diagnostic.summary(sorsum_res)

# show maximum and minimum values
diagnostics.max.min(sorsum_diag)
```

---

prob.cat *Summed or mean probability densities per category*

Description

Summing or averaging probability densities per category. The resulting data.frames can be used, for example, to produce illustrative diagrams. See the vignettes for some examples.

Usage

```
prob.cat(
  mcmc_list,
  age_identifier = "age.s",
  group_vec,
  mode = c("mean", "summed")
)
```

Arguments

mcmc_list	MCMC output from coda chains.
age_identifier	a character string of either "age.s" or "age.s_c" to select the uncalibrated or calibrated age estimates. Default: "age.s".
group_vec	a vector specifying the grouping category.
mode	a string specifying the resulting data.frame of summed probabilities or mean probabilities per category. Either mean or summed.

Value

A data.frame with either probability summed by category or mean per category.

Examples

```
# select Spitalfields data with multiple traits
spitalfields_traits <- spitalfields[,c(2:6)]

# example with multinormal likelihood, please be patient
spitalfields_res <- bay.ta(framework = "NIMBLE", algorithm = "mnorm",
method = spitalfields_traits)

# compute averaging probabilities per category Sex
prob_cat_mean <- prob.cat(spitalfields_res, group_vec = spitalfields$Sex,
mode = "mean")

# compute summed probabilities per category Sex
prob_cat_summed <- prob.cat(spitalfields_res, group_vec = spitalfields$Sex,
mode = "summed")
```

---

sequential.binom.test *Sequential cumulative binomial test*

Description

The *cumulative binomial* test asserts if the expected coverage, i.e. the percentage of known ages within the *highest density intervals*, is within the confidence interval of the realized coverage. This wrapper function allows to run this test sequentially, i.e. with a sequence of expected coverage levels, at a confidence level of 0.95.

Usage

```
sequential.binom.test(
  mcmc_list,
  known_age,
  HDImass = 0.95,
  age_identifier = "age.s"
)
```

Arguments

mcmc_list	MCMC output from coda chains.
known_age	a vector of known age-at-death. NAs are allowed and those entries will subsequently be ignored.
HDImass	a numeric or a vector with the probability range.
age_identifier	a character string of either "age.s" or "age.s_c" to select the uncalibrated or calibrated age estimates. Default: "age.s".

Value

A dataframe with the number of rows equaling the length of the parameter HDImass and six columns as follows:

- coverage Expected coverage.
- n_in Absolute number of known ages within the *highest density intervals*.
- perc Realized coverage.
- CI_low Lower limit of the confidence interval for the realized coverage.
- CI_up Upper limit of the confidence interval for the realized coverage.
- p_value p-value of the binomial test. If significant, the expected coverage is outside of the confidence intervals of the realized coverage.

Examples

```
# select Spitalfields data with multiple traits and convert to matrix
spitalfields_traits <- spitalfields[,c(2:6)]

# example with multinormal likelihood, please be patient
spitalfields_res <- bay.ta(algorithm = "mnorm",
method = spitalfields_traits)

# compute sequential binomial tests at expected probability levels 0.75
# and 0.95
sequential.binom.test(spitalfields_res, known_age = spitalfields$Age,
HDImass = c(0.75, 0.95))
```

---

sorsum_as *Sorsum: Example dataset*

Description

Neolithic population from a gallery grave in Hessen/Germany. From the published raw data, only the trait that was possible to assess most often (auricular surface after Lovejoy et al. 1985) is included. It is taken from Moser et al. 2025, table S69.

Format

A data frame with 38 rows and 2 columns:

- `id` individual id
- `auricular_surface` auricular surface after Lovejoy et al. 1985

References

Moser D, Pichler SL, Santos AL, Klimscha F, Fuchs K (2025). "The uncertain death. Estimating mortality structure by random sampling." *International Journal of Osteoarchaeology*, **35**(1), e3363. doi:10.1002/oa.3363.

Lovejoy CO, Meindl RS, Pryzbeck TR, Mensforth RP (1985). "Chronological metamorphosis of the auricular surface of the ilium: A new method for the determination of adult skeletal age at death." *American Journal of Physical Anthropology*, **68**(1), 15–28. doi:10.1002/ajpa.1330680103.

See Also

Other examples: `spitalfields`

`spitalfields` *Spitalfields: Example dataset*

Description

Known age-at-death of an Early Modern crypt population. The data is also available at the website of L. Konigsberg (s. link below).

Format

A data frame with 180 rows and 7 columns:

- `Age` known age-at-death
- `TO` trait 'Transverse organization'
- `ST` trait 'Surface texture'
- `MI` trait 'Microporosity'
- `MA` trait 'Macroporosity'
- `AP` trait 'Apical changes'
- `Sex` known sex

Source

<http://faculty.las.illinois.edu/lylek/Oxford/Buckberry.csv>

References

Buckberry JL, Chamberlain AT (2002). "Age estimation from the auricular surface of the ilium: a revised method." *American Journal of Physical Anthropology*, **119**(3), 231–239. doi:10.1002/ajpa.10130.

See Also

Other examples: `sorsum_as`

`threshold.chains` *Compute thresholds for chains*

Description

The computation of the thresholds on the log- and the age-scale is done outside of the MCMC simulation to spare the computation cost and the memory. The function returns a `coda::mcmc.list` which can be further processed.

Usage

```
threshold.chains(mcmc_list)
```

Arguments

`mcmc_list`	MCMC output from coda chains.

Value

A `coda::mcmc.list` for threshold values of traits on the age scale.

Examples

```
# select Sorsum data with auricular surface after Lovejoy et al. 1985
sorsum <- sorsum_as[,2]

# example with default settings, please be patient
sorsum_res <- bay.ta(method = sorsum)

# compute threshold chains
threshold_chains <- threshold.chains(sorsum_res)
```

`threshold.matrix` *Extract thresholds*

Description

A convenience function to extract mean thresholds values from the output of `diagnostic.summary()` which in turn was derived from a `coda::mcmc.list` computed with `threshold.chains()`

Usage

```
threshold.matrix(x, mean_choice = "Mode")
```

Arguments

`x`	output from function `diagnostic.summary()`
`mean_choice`	a character string of either "Mean", "Median" or "Mode". Default: "Mode".

Value

A matrix with threshold values of traits. The number of rows corresponds to the number of traits, and the number of columns to the maximum number of levels of one of the traits.

Examples

```
# select Sorsum data with auricular surface after Lovejoy et al. 1985
sorsum <- sorsum_as[,2]

# example with default settings, please be patient
sorsum_res <- bay.ta(method = sorsum)

# compute threshold chains
threshold_chains <- threshold.chains(sorsum_res)

# compute summary diagnostics
threshold_diag <- diagnostic.summary(threshold_chains)

# extract threshold matrix (for sorsum only 1 row)
threshold.matrix(threshold_diag)
```

Index