Package 'baorista'

July 30, 2024

Title Bayesian Aoristic Analyses

Version 0.2.0

Description Provides an alternative approach to aoristic analyses for archaeological datasets by fitting Bayesian parametric growth models and non-parametric random-walk Intrinsic Conditional Autoregressive (ICAR) models on time frequency data (Crema (2024)<doi:10.1111/arcm.12984>). It handles event typochronology based timespans defined by start/end date as well as more complex user-provided vector of probabilities.

Depends R (>= 3.5.0), nimble (>= 0.12.0)

Imports stats,coda,graphics

Suggests knitr, rmarkdown

License GPL (>= 2)

Encoding UTF-8

LazyData true

VignetteBuilder knitr

RoxygenNote 7.2.3

Language en-GB

NeedsCompilation no

Author Enrico Crema [aut, cre] (<https://orcid.org/0000-0001-6727-5138>)

Maintainer Enrico Crema <enrico.crema@gmail.com>

Repository CRAN

Date/Publication 2024-07-30 14:20:06 UTC

Contents

`createProbMat` *Creates a probMat class object from user data*

Description

Converts either a data.frame with the start and end date of each event or matrix of probabilities values into a `probMat` class object.

Usage

```
createProbMat(x = NULL, pmat = NULL, timeRange = NULL, resolution = NULL)
```

Arguments

`x`	A data.frame containing the start and end date of the timespan of each event. Dates should be in BP, with the first column defining the start and the second column defining the end of the timespan.
`pmat`	A matrix of aoristic weights (probabilities), with row representing events and column representing timeblocks.
`timeRange`	A vector of two numerical values representing the start and end of the window of analysis in BP.
`resolution`	Resolution of the timeblock. Ignored if `pmat` is provided.

Value

An object of class `probMat`.

`dexpfit` *Estimate Exponential Growth rate from Aoristic data*

Description

Fits a double exponential growth model to `ProbMat` class objects.

Usage

```
dexpfit(
  x,
  niter = 1e+05,
  nburnin = 50000,
  thin = 10,
  nchains = 4,
  r1Prior = "dnorm(mean=0,sd=0.05)",
  r2Prior = "dnorm(mean=0,sd=0.05)",
  etaPrior = "dunif(min=1,max=z)",
  r1Sampler = NULL,
  r2Sampler = NULL,
  etaSampler = NULL,
  parallel = FALSE,
  seeds = 1:4
)
```

Arguments

`x`	A ProbMat class object
`niter`	Number of MCMC iterations. Default is 500,000.
`nburnin`	Number of iterations discarded for burn-in. Default is 250,000.
`thin`	Thinning interval
`nchains`	Number of MCMC chains
`r1Prior`	A string defining prior for the growth parameter r1. Default is 'dnorm(mean=0,sd=0.05)'.
`r2Prior`	A string defining prior for the growth parameter r2. Default is 'dnorm(mean=0,sd=0.05)'.
`etaPrior`	A string defining prior for the change point parameter eta. Default is 'dunif(1,z)', where 'z' is the number of time-blocks.
`r1Sampler`	A list containing settings for the MCMC sampler. Default is null and employs nimble's Default sampler (RW sampler).
`r2Sampler`	A list containing settings for the MCMC sampler. Default is null and employs nimble's Default sampler (RW sampler).
`etaSampler`	A list containing settings for the MCMC sampler. Default is null and employs nimble's Default sampler (RW sampler).
`parallel`	Logical specifying whether the chains should be run in parallel or not.
`seeds`	Random seed for each chain. Default is 1:4.

Details

The function fits a discrete bounded double exponential growth model on the observed data using MCMC as implemented by the nimble package. The Bayesian model consists of a two growth rate parameters (r1 and r2), with the change from r1 and r2 occurring at inferred point in time eta. Users can define suitable priors using character strings for the argument `rPrior1`,`rPrior2`, and `cPrior` (for details on how this should be specified please consult the nimble manual). Please note that the function returns posterior of the growth rate normalised by the resolution defined in the `ProbMat` class object. MCMC settings such as the choice the sampler, number of iterations, chains, etc can also be specified.

Value

A `fitteddoubleExp` class object containing the original ProbMat class object, posterior of the growth rate, along with its Gelman Rubin statistic and effective sample sizes.

`expfit` *Estimate Exponential Growth rate from Aoristic data*

Description

Fits an exponential growth model to `ProbMat` class objects.

Usage

```
expfit(
  x,
  niter = 1e+05,
  nburnin = 50000,
  thin = 10,
  nchains = 4,
  rPrior = "dnorm(mean=0,sd=0.05)",
  rSampler = NULL,
  parallel = FALSE,
  seeds = 1:4
)
```

Arguments

`x`	A ProbMat class object
`niter`	Number of MCMC iterations. Default is 500,000.
`nburnin`	Number of iterations discarded for burn-in. Default is 250,000.
`thin`	Thinning interval
`nchains`	Number of MCMC chains
`rPrior`	A string defining prior for the growth parameter r. Default is 'dnorm(mean=0,sd=0.05)'.

rSampler	A list containing settings for the MCMC sampler. Default is null and employs nimble's Default sampler (RW sampler).
parallel	Logical specifying whether the chains should be run in parallel or not.
seeds	Random seed for each chain. Default is 1:4.

Details

The function fits a discrete bounded exponential growth model on the observed data using MCMC as implemented by the nimble package. The Bayesian model consists of a single growth rate parameter (r), and users can define suitable priors using character strings for the argument `rPrior` (for details on how this should be specified please consult the nimble manual). The distribution parameters defined in `rPrior` is also used to generate initialisation values for the MCMC. Please note that the function returns posterior of the growth rate normalised by the resolution defined in the `ProbMat` class object. MCMC settings such as the choice the sampler, number of iterations, chains, etc can also be specified.

Value

A `fittedExp` class object containing the original ProbMat class object, posterior of the growth rate, along with its Gelman Rubin statistic and effective sample sizes.

---

icarfit *Fits a random walk ICAR model to Aoristic data*

Description

Estimates parameters of a multinomial probability distribution that describes the shape of the of the time-frequency distribution of an observed sets of events with chronological uncertainty. The function is wrapper for fitting a 1D random walk ICAR model via nimble.

Usage

```
icarfit(
  x,
  niter = 1e+05,
  nburnin = 50000,
  thin = 10,
  nchains = 4,
  sigmaPrior = "dexp(rate=1)",
  sigmaSampler = NULL,
  parallel = FALSE,
  seeds = 1:4
)
```

Arguments

x	A ProbMat class object
niter	Number of MCMC iterations. Default is 500,000.
nburnin	Number of iterations discarded for burn-in. Default is 250,000.
thin	Thinning interval
nchains	Number of MCMC chains
sigmaPrior	A string defining prior for the sigma parameter. Default is 'dexp(rate=1)'.
sigmaSampler	A list containing settings for the MCMC sampler. Default is null and employs nimble's Default sampler (RW sampler).
parallel	Logical specifying whether the chains should be run in parallel or not.
seeds	Random seed for each chain. Default is 1:4.

Details

The function estimates a vector temporally autocorrelated probability values on the observed data using MCMC as implemented by the nimble package. The model is effectively non-parametric, and at its core it is an implementation of a 1D random ICAR model. Users can specify the prior for the variance parameter through the argument `sigmaPrior`. Different settings for this parameter can greatly influence the estimates of the probability vectors. For example using `sigmaPrior=dexp(100)` instead of the default `sigmaPrior=dexp(1)` would lead to 'flatter' time-series with higher temporal autocorrelation. The distribution parameters defined in `sigmaPrior` is also used to generate initialisation values for the MCMC. Please consult the nimble package manual for the syntax required in specifying the prior. MCMC settings such as the choice the sampler, number of iterations, chains, etc can also be specified. Please not that the function is computationally intensive and might require a larger number of iterations to reach satisfactory MCMC convergence.

Value

A `fittedICAR` class object containing the original ProbMat class object, posteriors of the probabilities for each time-block and the variance parameter along with their MCMC diagnostics (Gelman Rubin statistic and effective sample size).

---

logisticfit *Fits a Logistic growth model on Aoristic data*

Description

Fits an exponential growth model to `ProbMat` class objects.

Usage

```
logisticfit(
  x,
  niter = 1e+05,
  nburnin = 50000,
  thin = 10,
  nchains = 4,
  rPrior = "dexp(rate=1/0.001)",
  mPrior = "dunif(min=1,max=z)",
  rSampler = NULL,
  mSampler = NULL,
  parallel = FALSE,
  seeds = 1:4
)
```

Arguments

x	A ProbMat class object
niter	Number of MCMC iterations. Default is 100,000.
nburnin	Number of iterations discarded for burn-in. Default is 50,000.
thin	Thinning interval
nchains	Number of MCMC chains
rPrior	A string defining prior for the growth parameter r. Default is 'dexp(1/0.01)'.
mPrior	A string defining prior for the point of maximum growth rate m. Default is 'dunif(1,z)', where 'z' is the number of time-blocks.
rSampler	A list containing settings for the MCMC sampler for the parameter 'r'. Default is null and employs nimble's Default sampler (RW sampler).
mSampler	A list containing settings for the MCMC sampler for the parameter 'm'. Default is null and employs nimble's Default sampler (RW sampler).
parallel	Logical specifying whether the chains should be run in parallel or not.
seeds	Random seed for each chain. Default is 1:4.

Details

The function fits a discrete bounded logistic growth model on the observed data using MCMC as implemented by the nimble package. The Bayesian model consists of two parameters, a growth rate (r) and a midpoint (m) defining the inflection point of the growth curve. Priors of the two parameters can be defined by the arguments `rPrior` and `mPrior`. In the latter case the object `z` is a placeholder for the number of blocks (e.g. the default 'dunif(1,z)' is a uniform across all blocks). Priors are defined by character strings following the syntax used by nimble. The distribution parameters defined in `rPrior` and `mPrior` are also used to generate initialisation values for the MCMC. Please note that the function returns posterior of the growth rate normalised by the resolution defined in the `ProbMat` class object. MCMC settings such as the choice the sampler, number of iterations, chains, etc can also be specified.

Value

A `fittedLogistic` class object containing the original ProbMat class object, posteriors of the growth rate and midpoint and their MCMC diagnostics (i.e. Gelman Rubin statistic and effective sample sizes).

`mcsim` *Monte-Carlo simulation on aoristic data*

Description

Samples multiple sets of random dates from aoristic weights

Usage

```
mcsim(x, nsim = 1000)
```

Arguments

`x`	A ProbMat class object
`nsim`	Number of Monte-Carlo simulations

Details

The function randomly assigns to each event a time-block based on its probability values (i.e. aoristic weight) and computes, for each time-block, the total number of simulated events. This process is repeated `nsim` time, allowing to estimate percentile-based intervals on the number of events per time-block (Crema 2012). It should be noted that while this approach accounts for chronological uncertainty, it provides only a description of the sample rather than the underlying population, and can be biased how the underlying archaeological periodisations define the time-spans of each event (see also Crema 2024 for discussion on limitations).

Value

An object of class `mcsimres` containing relevant metadata and a matrix with the number of events per time-block per Monte-Carlo simulation.

References

Crema, E. R. (2012). Modelling Temporal Uncertainty in Archaeological Analysis. Journal of Archaeological Method and Theory, 19(3), 440–461. doi:10.1007/s10816-011-9122-3 Crema, E.R. (2024). A Bayesian alternative to Aoristic analyses in archaeology. Archaeometry. doi:10.1111/arcm.12984

plot.fittedDExp *Plot double exponential model fitted to aoristic data*

Description

Plot posterior estimates of `fittedDExp` class objects.

Usage

```
## S3 method for class 'fittedDExp'
plot(
  x,
  hpd = c(0.5, 0.9),
  minortick = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = "Probability Mass",
  calendar = "BP",
  col = "black",
  lwd = 1,
  lty = 2,
  col1 = "steelblue",
  col2 = "lightblue",
  pch = 20,
  plot.legend = TRUE,
  legend.arg = NULL,
  ...
)
```

Arguments

`x`	An `fittedDExp` class object
`hpd`	A vector with two values defining the highest posterior density interval to display. Default is 0.5 and 0.9.
`minortick`	Interval for minor ticks in the x-axis label. Default is estimated based on timescale.
`ylim`	Limits of the y-axis. Default estimated from posterior ranges.
`xlab`	Label for the x-axis. Default based on `calendar`.
`ylab`	Label for the y-axis. Default is "Probability Mass".
`calendar`	Either `'BP'` or `'BCAD'`. Indicate whether the x-axis should be displayed in BP or BC/AD. Default is `'BP'`.
`col`	Color of posterior mean. Default is black.
`lwd`	Line width posterior mean. Default is 1.
`lty`	Line type posterior mean. Default is 2.
`col1`	Fill color for the first (inner) HPD interval. Default is 'steelblue'.

col2	Fill color for the second (outer) HPD interval. Default is 'lightblue'.
pch	Point symbol used to display mean posteriors. Default is 20.
plot.legend	Logical indicating whether to display a legend or not (default is TRUE).
legend.arg	List containing arguments to be directed to the legend() function.
...	Additional arguments affecting the plot.

Value

No return value (plot function)

---

plot.fittedExp *Plot exponential model fitted to aoristic data*

Description

Plot posterior estimates of fittedExp class objects.

Usage

```
## S3 method for class 'fittedExp'
plot(
  x,
  hpd = c(0.5, 0.9),
  minortick = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = "Probability Mass",
  calendar = "BP",
  col = "black",
  lwd = 1,
  lty = 2,
  col1 = "steelblue",
  col2 = "lightblue",
  pch = 20,
  plot.legend = TRUE,
  legend.arg = NULL,
  ...
)
```

Arguments

x	An fittedExp class object
hpd	A vector with two values defining the highest posterior density interval to display. Default is 0.5 and 0.9.
minortick	Interval for minor ticks in the x-axis label. Default is estimated based on timescale.

ylim	Limits of the y-axis. Default estimated from posterior ranges.
xlab	Label for the x-axis. Default based on calendar.
ylab	Label for the y-axis. Default is "Probability Mass".
calendar	Either 'BP' or 'BCAD'. Indicate whether the x-axis should be displayed in BP or BC/AD. Default is 'BP'.
col	Color of posterior mean. Default is black.
lwd	Line width posterior mean. Default is 1.
lty	Line type posterior mean. Default is 2.
col1	Fill color for the first (inner) HPD interval. Default is 'steelblue'.
col2	Fill color for the second (outer) HPD interval. Default is 'lightblue'.
pch	Point symbol used to display mean posteriors. Default is 20.
plot.legend	Logical indicating whether to display a legend or not (default is TRUE).
legend.arg	List containing arguments to be directed to the legend() function.
...	Additional arguments affecting the plot.

Value

No return value (plot function)

---

plot.fittedICAR *Plot 1D ICAR model fitted to aoristic data*

Description

Plot posterior estimates of fittedICAR class objects.

Usage

```
## S3 method for class 'fittedICAR'
plot(
  x,
  hpd = c(0.5, 0.9),
  minortick = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = "Probability Mass",
  calendar = "BP",
  col1 = "steelblue",
  col2 = "lightblue",
  pch = 20,
  plot.legend = TRUE,
  legend.arg = NULL,
  ...
)
```

Arguments

x	An fittedICAR class object
hpd	A vector with two values defining the highest posterior density interval to display. Default is 0.5 and 0.9.
minortick	Interval for minor ticks in the x-axis label. Default is estimated based on timescale.
ylim	Limits of the y-axis. Default estimated from posterior ranges.
xlab	Label for the x-axis. Default based on calendar.
ylab	Label for the y-axis. Default is "Probability Mass".
calendar	Either 'BP' or 'BCAD'. Indicate whether the x-axis should be displayed in BP or BC/AD. Default is 'BP'.
col1	Fill color for the first (inner) HPD interval. Default is 'steelblue'.
col2	Fill color for the second (outer) HPD interval. Default is 'lightblue'.
pch	Point symbol used to display mean posteriors. Default is 20.
plot.legend	Logical indicating whether to display a legend or not (default is TRUE).
legend.arg	List containing arguments to be directed to the legend() function.
...	Additional arguments affecting the plot.

Value

No return value (plot function)

---

plot.fittedLogistic *Plot logistic model fitted to aoristic data*

Description

Plot posterior estimates of fittedLogistic class objects.

Usage

```
## S3 method for class 'fittedLogistic'
plot(
  x,
  hpd = c(0.5, 0.9),
  minortick = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = "Probability Mass",
  calendar = "BP",
  col = "black",
  lwd = 1,
  lty = 2,
  col1 = "steelblue",
```

```
  col2 = "lightblue",
  pch = 20,
  plot.legend = TRUE,
  legend.arg = NULL,
  ...
)
```

Arguments

x	An fittedExp class object
hpd	A vector with two values defining the highest posterior density interval to display. Default is 0.5 and 0.9.
minortick	Interval for minor ticks in the x-axis label. Default is estimated based on timescale.
ylim	Limits of the y-axis. Default estimated from posterior ranges.
xlab	Label for the x-axis. Default based on calendar.
ylab	Label for the y-axis. Default is "Probability Mass".
calendar	Either 'BP' or 'BCAD'. Indicate whether the x-axis should be displayed in BP or BC/AD. Default is 'BP'.
col	Color of posterior mean. Default is black.
lwd	Line width posterior mean. Default is 1.
lty	Line type posterior mean. Default is 2.
col1	Fill color for the first (inner) HPD interval. Default is 'steelblue'.
col2	Fill color for the second (outer) HPD interval. Default is 'lightblue'.
pch	Point symbol used to display mean posteriors. Default is 20.
plot.legend	Logical indicating whether to display a legend or not (default is TRUE).
legend.arg	List containing arguments to be directed to the legend() function.
...	Additional arguments affecting the plot.

Value

No return value (plot function)

---

plot.mcsimres *Plot Monte-Carlo simulation results on aoristic data*

Description

Plot Monte-Carlo simulation based percentile intervals on frequency or rate of change of events.

Usage

```
## S3 method for class 'mcsimres'
plot(
  x,
  interval = 0.9,
  minortick = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  calendar = "BP",
  col = "black",
  lwd = 1,
  lty = 1,
  col.fill = "lightblue",
  pch = 20,
  type = "sum",
  plot.legend = TRUE,
  legend.arg = NULL,
  ...
)
```

Arguments

x	A mcsimres class object generated using the mcsim() function.
interval	A value between 0 and 1 defining the percentile interval. Default is 0.9.
minortick	Interval for minor ticks in the x-axis label. Default is estimated based on timescale.
ylim	Limits of the y-axis. Default estimated from posterior ranges.
xlab	Label for the x-axis. Default based on calendar.
ylab	Label for the y-axis. Default is "Probability Mass".
calendar	Either 'BP' or 'BCAD'. Indicate whether the x-axis should be displayed in BP or BC/AD. Default is 'BP'.
col	Color of Monte-Carlo simulation mean. Default is black.
lwd	Line width of Monte-Carlo mean. Default is 1.
lty	Line type Monte-Carlo mean. Default is 1.
col.fill	Fill color for the first (inner) percentile interval. Default is 'lightblue'.
pch	Point symbol used to display mean posteriors. Default is 20.
type	Determine whether to display total number of events (if set to 'sum') or the rate of change ('roc'), computed as (t0/t1)^(1/r)-1, where t0 is the number of events in given time-block t, t1 is the number of events of the next time-block t+1, and r is the size (in years) of the time-blocks. Defaults is 'sum'.
plot.legend	Logical indicating whether to display a legend or not (default is TRUE).
legend.arg	List containing arguments to be directed to the legend() function.
...	Additional arguments affecting the plot.

Value

No return value (plot function)

`plot.probMat` *Plot Aoristic Sums*

Description

Plot summed probabilities of aoristic weights.

Usage

```
## S3 method for class 'probMat'
plot(
  x,
  xlab = NULL,
  ylab = NULL,
  type = "asum",
  calendar = "BP",
  lwd = 1,
  col = 1,
  minortick = NULL,
  add = FALSE,
  ...
)
```

Arguments

`x`	`probMat` class object generated using the `generateProbMat()`.
`xlab`	Label for the x-axis. Default based on `calendar`.
`ylab`	Label for the y-axis. Default is 'Summed Probability' (if `type='asum'`) or 'Probability Mass' (when `type='dens'`).
`type`	Either 'asum' for Aoristic Sum, 'dens' for probability mass. Default is 'asum'.
`calendar`	Either `'BP'` or `'BCAD'`. Indicate whether the x-axis should be displayed in BP or BC/AD. Default is `'BP'`.
`lwd`	Line width. Default is 1.
`col`	Line col. Default is 'black'
`minortick`	Interval for minor ticks in the x-axis label. Default is estimated based on timescale
`add`	if set to `TRUE` adds the line and point graph on existing plot.
`...`	Additional arguments affecting the plot.

Value

No return value (plot function)

sampledf *Sample aoristic data (data.frame)*

Description

Sample datasets to illustrate data formats required for `createProbMat()`.

Usage

```
sampledf
```

Format

A data.frame class object with two columns defining the start and the end of each even (`sample.df`)

Examples

```
data(sampledf)
x  <- createProbMat(x=sampledf,timeRange = c(6500,4001),resolution= 100)
```

samplemat *Sample aoristic data (matrix)*

Description

Sample datasets to illustrate data formats required for `createProbMat()`.

Usage

```
samplemat
```

Format

A matrix class object storing the probability of each event (row) in each time-block (column)

Examples

```
data(samplemat)
x <- createProbMat(pmat=samplemat,timeRange = c(5000,3001),resolution=100)
plot(x)
```

Index