Package 'eratosthenes'

June 27, 2025

Title Archaeological Synchronism

Version 0.0.9

Description
Estimation of unknown historical or archaeological dates subject to relationships with other relative dates and absolute constraints, derived as marginal densities from the full joint conditional, using a two-stage Gibbs sampler with consistent batch means to assess convergence. Features reporting on Monte Carlo standard errors, as well as tools for rule-based estimation of dates of production and use of artifact types, aligning and checking relative sequences, and evaluating the impact of the omission of relative/absolute events upon one another.

License GPL (>= 3)

Imports stats, graphics, grDevices, Rcpp, Rdpack, paletteer

RdMacros Rdpack

Encoding UTF-8

RoxygenNote 7.3.2

LinkingTo Rcpp

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Stephen A. Collins-Elliott [aut, cre] (ORCID:
<https://orcid.org/0000-0002-5642-6903>)

Maintainer Stephen A. Collins-Elliott <sce@utk.edu>

Repository CRAN

Date/Publication 2025-06-27 19:40:02 UTC

Contents

`finds_d2l` *Convert Finds Data Frame (Context / Find-Type) to List Object*

Description

Performs the opposite of `finds_l2d`. Takes a `data.frame` object of two columns, containing the context in the first and the find-type in the second, and returns a `list` object for input in `gibbs_ad`. The value of the find `id` is automatically generated as an integer if not provided in a third column.

Usage

```
finds_d2l(input)

## S3 method for class 'data.frame'
finds_d2l(input)
```

Arguments

`input`	A two-column data frame of contexts (first column) and find-types (second column). An optional third column of an id number may be provided.

Value

A list of finds (each one a list) associated with contexts and their types.

Examples

```
f1 <- list(id = "find01", assoc = "D", type = c("type1", "form1"))
f2 <- list(id = "find02", assoc = "E", type = c("type1", "form2"))
f3 <- list(id = "find03", assoc = "G", type = c("type1", "form1"))
f4 <- list(id = "find04", assoc = "H", type = c("type2", "form1"))
f5 <- list(id = "find05", assoc = "I", type = "type2")
f6 <- list(id = "find06", assoc = "H", type = NULL)

artifacts <- list(f1, f2, f3, f4, f5, f6)
```

```
# convert list to data frame
artifacts_df <- finds_l2d(artifacts)

# convert data frame to list
artifacts_list <- finds_d2l(artifacts_df)
```

`finds_l2d` *Convert Finds List Object to Data Frame (Context / Find-Type)*

Description

Performs the opposite of `finds_d2l`. Takes a `list` object of finds and their types, used as input in `gibbs_ad`, and returns a `data.frame` of two columns, containing the context in the first and the find-type in the second, and the `id` of the object in the third.

Usage

```
finds_l2d(input)

## S3 method for class 'list'
finds_l2d(input)
```

Arguments

`input`	A list object of finds (each one a list) of associated contexts and types.

Value

A three-column data frame of contexts (first column) and find-types attested in that context (second column), along with the id number (third column).

Examples

```
f1 <- list(id = "find01", assoc = "D", type = c("type1", "form1"))
f2 <- list(id = "find02", assoc = "E", type = c("type1", "form2"))
f3 <- list(id = "find03", assoc = "G", type = c("type1", "form1"))
f4 <- list(id = "find04", assoc = "H", type = c("type2", "form1"))
f5 <- list(id = "find05", assoc = "I", type = "type2")
f6 <- list(id = "find06", assoc = "H", type = NULL)

artifacts <- list(f1, f2, f3, f4, f5, f6)

# convert list to data frame
artifacts_df <- finds_l2d(artifacts)
```

gibbs_ad *Gibbs Sampler for Archaeological Dates*

Description

A Gibbs sampler for dating archaeological events, to fit relative sequences to absolute, calendrical dates, along with rule-based production dates of artifact types. Relative events can be associated with *termini post quos* (*t.p.q.*) and *termini ante quos* (*t.a.q.*), which are entered as samples from a given probability density function $f(t)$. This function may take any form, a single date (i.e., with a probability of 1), a continuous uniform distribution (any time between two dates), or a bespoke density (as with calibrated radiocarbon dates). Relative events are modeled on a continuous uniform density between the latest antecedent event and earliest subsequent event.

Usage

```
gibbs_ad(
  sequences,
  finds = NULL,
  max_samples = 10^5,
  size = 10^3,
  mcse_crit = 0.5,
  tpq = NULL,
  taq = NULL,
  alpha_ = -5000,
  omega_ = 1950,
  trim = TRUE,
  rule = "naive"
)

## S3 method for class 'list'
gibbs_ad(
  sequences,
  finds = NULL,
  max_samples = 10^5,
  size = 10^3,
  mcse_crit = 0.5,
  tpq = NULL,
  taq = NULL,
  alpha_ = -5000,
  omega_ = 1950,
  trim = TRUE,
  rule = "naive"
)
```

Arguments

sequences	A `list` of relative sequences of elements (e.g., contexts).

`finds` Optional. A `list` of finds related to (contained in) the elements of `sequences`.

`max_samples` Maximum number of samples to run. Default is `10^5`.

`size` The number of samples to take on each iteration of the main Gibbs sampler. Default is `10^3`.

`mcse_crit` Criterion for the Monte Carlo standard error to stop the Gibbs sampler, as based on depositional dates and absolute constraints. The number of Monte Carlo samples for production dates is identical to that depositional dates.

`tpq` A `list` containing *termini post quos*. Each object in the list consists of:

- `id` A `character` ID of the *t.p.q.*, such as a reference or number.
- `assoc` The element in `code` to which the *t.p.q.* is associated.
- `samples` A vector of samples drawn from the appertaining probability density function of that *t.p.q.*

`taq` A `list` containing *termini ante quos*. Each object in the list consists of:

- `id` A `character` ID of the *t.a.q.*, such as a reference or number.
- `assoc` The element in `code` to which the *t.p.q.* is associated.
- `samples` A vector of samples drawn from the appertaining probability density function of that *t.p.q.*

`alpha_` An initial *t.p.q.* to limit any elements which may occur before the first provided *t.p.q.* Default is `-5000`.

`omega_` A final *t.a.q.* to limit any elements which may occur after the after the last provided *t.a.q.* Default is `1950`.

`trim` A logical value to determine whether elements that occur before the first *t.p.q.* and after the last *t.a.q.* should be omitted from the results (i.e., to "trim" elements at the ends of the sequence, whose marginal densities depend on the selection of `alpha_` and `omega_`). Default is `TRUE`.

`rule` The rule for computing an estimated date of production of a find-type, either `"earliest"`, selecting a production date between the earliest deposition of that type and the next most earliest context, or `"naive"` (the default), which will select a production date any time between the distribution of that "earliest" date and the depositional date of that artifact.

Details

Gibbs sampling is a conventional method for calibrating and estimating radiocarbon dates in light of absolute constraints and relative sequences: see Buck et al. (1996); Buck et al. (1999); Bronk Ramsey (2009), the latter of which uses a mixture of Metropolis-Hastings and Gibbs.

In this implementation, two phases of Gibbs sampling are performed: an initial phase for selecting starting values and then the main sampler, with convergence evaluated using Monte Carlo standard errors (MCSE).

The initial Gibbs sampler results in a vector of starting values randomly sampled for each event up to $\sqrt{k}$ runs, where k is the total number of events. Starting values may therefore take some time to assign, but this initial sampling is necessary to avoid a catastrophic collapse due to floating point errors in the initial selection of random values and will also result in closer starting values with respect to marginal densities.

The main Gibbs sampler uses consistent batch means (CBM) determine convergence and hence when to end the main sampling run: there is no motivation to remove burn-in from the main sampling run nor to run multiple chains. CBM is assured to converge in distribution, see Jones et al. (2006); Flegal et al. (2008). A stopping point for the main sampler is therefore determined using the mean of the Monte Carlo standard errors (MCSE) across all random variates, which is the input of `mcse_crit` (the mean MCSE for all events). The input `max_samples` indicates the maximum number of simulations to run, but the sampler will stop if the specified criterion of `mcse_crit` is passed. The default mean MCSE is set at `mcse_crit = 0.5`, as the MCSE is measured in years (i.e. to allow for an error +/- 1 year), but, to be sure, individual events will have higher or lower MCSE than this mean criterion, whose primary purpose is as a stopping rule.

Note that the MCSE criterion is applied as a stopping rule for depositional dates and external constraints. The number of Monte Carlo samples for production dates of types is chosen to be identical to that need to pass `mcse_crit`, such that ultimately the final mean MCSE of all variates may differ from that of the criterion. Depending on the conditional structure of the relative sequences and the timescale of investigation, higher or lower MCSE may be more desirable or acceptable.

For the use dates of artifact type production, use, and deposition, see the `gibbs_ad_use` function.

Value

A `list` object of class `marginals` which contains the following:

- `deposition` A `list` of samples from the marginal density of each context's depositional date.
- `externals` A `list` of samples of the marginal density of each constraint (*t.p.q.* and *t.a.q.]*), as conditioned upon the occurrence of other depositional
- `production` If a `finds` object has been input, samples of the marginal density of the production date of finds types will be included in the output. If types are attested in trimmed contexts,
- `mcse` The Monte Carlo standard errors (MCSE) of the random variates (fixed t.p./a.q. will have a MCSE of 0.)

References

Bronk Ramsey C (2009). "Bayesian Analysis of Radiocarbon Dates." *Radiocarbon*, **51**, 337–360.

Buck CE, Cavanagh WG, Litton CD (1996). *Bayesian Approach to Interpreting Archaeological Data*. John Wiley and Sons, Chichester.

Buck CE, Christen JA, James GN (1999). "BCal: An On-line Bayesian Radiocarbon Calibration Tool." *Internet Archaeology*, **7**. https://intarch.ac.uk/journal/issue7/buck/.

Flegal JM, Haran M, Jones GL (2008). "Markov Chain Monte Carlo: Can We Trust the Third Significant Figure?" *Statistical Science*, **23**, 250–260. doi:10.1214/08STS257.

Jones GL, Haran M, Caffo BS, Neath R (2006). "Fixed-Width Output Analysis for Markov Chain Monte Carlo." *Journal of the American Statistical Association*, **101**, 1537–1547. doi:10.1198/016214506000000492.

Examples

```
x <- c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J")
y <- c("B", "D", "G", "H", "K")
z <- c("F", "K", "L", "M")
contexts <- list(x, y, z)

f1 <- list(id = "find01", assoc = "D", type = c("type1", "form1"))
f2 <- list(id = "find02", assoc = "E", type = c("type1", "form2"))
f3 <- list(id = "find03", assoc = "G", type = c("type1", "form1"))
f4 <- list(id = "find04", assoc = "H", type = c("type2", "form1"))
f5 <- list(id = "find05", assoc = "I", type = "type2")
f6 <- list(id = "find06", assoc = "H", type = NULL)

artifacts <- list(f1, f2, f3, f4, f5, f6)

# external constraints
coin1 <- list(id = "coin1", assoc = "B", type = NULL, samples = runif(100,-320,-300))
coin2 <- list(id = "coin2", assoc = "G", type = NULL, samples = seq(37, 41, length = 100))
  # seq(37, 41, length = 100) is equivalent in concept to runif(100, 37, 41))
destr <- list(id = "destr", assoc = "J", type = NULL, samples = 79)

tpq_info <- list(coin1, coin2)
taq_info <- list(destr)

result <- gibbs_ad(contexts, finds = artifacts, tpq = tpq_info, taq = taq_info)
```

---

gibbs_ad_use *Gibbs Sampler for Archaeological Dates: Artifact Use*

Description

Using the results of gibbs_ad, estimate a single density for the date of use of an artifact or artifact type. Multiple artifacts and types can be given, which will be pooled into a single type. For example, one can input several individual finds via their id number as comprising a type, or multiple (sub)types/classes as a single type, (e.g., "MGS V amphora", "MGS VI amphora", "MGS V/VI amphora" to construct one group).

Usage

```
gibbs_ad_use(
  marginalized,
  finds,
  id = NULL,
  type = NULL,
  type_name = NULL,
  max_samples = 10^5,
  size = 10^3,
```

```
  mcse_crit = 0.5
)

## S3 method for class 'marginals'
gibbs_ad_use(
  marginalized,
  finds,
  id = NULL,
  type = NULL,
  type_name = NULL,
  max_samples = 10^5,
  size = 10^3,
  mcse_crit = 0.5
)
```

Arguments

marginalized	A list object of class marginals, the output of gibbs_ad.
finds	Either the list object of finds used as input to produce marginals or a data.frame of two columns, the first listing the context and the second the incidence of the type in that context.
id	A vector of the id of one or more specific finds whose use date is to be estimated. The values of id must match those in the list of finds. If type is used, id is ignored.
type	A vector of one or more types to estimate a use density for. Must contain a value if id is NULL.
type_name	A customized label for the type (e.g., if one is selecting via id or has combined subtypes). If only type is used to select finds, the default will be that label Otherwise the default is simply "Type."
max_samples	Maximum number of samples to run. Default is 10^5.
size	The number of samples to take on each iteration of the main Gibbs sampler. Default is 10^3.
mcse_crit	Criterion for the Monte Carlo standard error to stop the Gibbs sampler. Only the MCSE of the use date is used as a stopping rule.

Details

Depending on whether one is using id numbers or type(s), the id or type argument is used, which takes a vector of the entries' names. The gibbs_ad_use function samples a use date between the production and depositional densities from the results of gibbs_ad, and in turn pools those densities for the production and deposition of the stipulated ids/type; the resulting list object does *not* express marginalized densities of production and deposition in light of the estimation of a use date.

See gibbs_ad for information on consistent batch means and Monte Carlo standard error, which are used to determined convergence for the use date.

Value

A `list` of class `use_marginals` of the density of a use date, conditional upon production and depositional dates.

Examples

```
x <- c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J")
y <- c("B", "D", "G", "H", "K")
z <- c("F", "K", "L", "M")
contexts <- list(x, y, z)

f1 <- list(id = "find01", assoc = "D", type = c("type1", "form1"))
f2 <- list(id = "find02", assoc = "E", type = c("type1", "form2"))
f3 <- list(id = "find03", assoc = "G", type = c("type1", "form1"))
f4 <- list(id = "find04", assoc = "H", type = c("type2", "form1"))
f5 <- list(id = "find05", assoc = "I", type = "type2")
f6 <- list(id = "find06", assoc = "H", type = NULL)

artifacts <- list(f1, f2, f3, f4, f5, f6)

# external constraints
coin1 <- list(id = "coin1", assoc = "B", type = NULL, samples = runif(100,-320,-300))
coin2 <- list(id = "coin2", assoc = "G", type = NULL, samples = seq(37, 41, length = 100))
destr <- list(id = "destr", assoc = "J", type = NULL, samples = 79)

tpq_info <- list(coin1, coin2)
taq_info <- list(destr)

result <- gibbs_ad(contexts, finds = artifacts, tpq = tpq_info, taq = taq_info)

# use dates by specifying ids
gibbs_ad_use(result, artifacts, id = c("find04", "find05"), max_samples = 2000, mcse_crit = 2)

# use dates by specifying types
gibbs_ad_use(result, artifacts, type = "type1", max_samples = 2000, mcse_crit = 2)
```

`histogram` *Histogram of Marginal Densities*

Description

Wrapper around `hist` to plot density histograms for select marginal densities (up to 12) in a single plot, from the results of `gibbs_ad`, or to plot density histograms of the production, deposition, and use of a type, from the results of `gibbs_ad_use`].

Usage

```
histogram(
  x,
  events = NULL,
  aspect = c("production", "use", "deposition"),
  display_name = "Type",
  breaks = "Freedman-Diaconis",
  xlim = NULL,
  ylim = NULL,
  xlab = "Year",
  palette = NULL,
  opacity = 1,
  legend_pos = "topright"
)

## S3 method for class 'marginals'
histogram(
  x,
  events = NULL,
  aspect = NULL,
  display_name = NULL,
  breaks = "Freedman-Diaconis",
  xlim = NULL,
  ylim = NULL,
  xlab = "Year",
  palette = NULL,
  opacity = 1,
  legend_pos = "topright"
)

## S3 method for class 'use_marginals'
histogram(
  x,
  events = NULL,
  aspect = c("production", "use", "deposition"),
  display_name = "Type",
  breaks = "Freedman-Diaconis",
  xlim = NULL,
  ylim = NULL,
  xlab = "Year",
  palette = NULL,
  opacity = 0.5,
  legend_pos = "topright"
)
```

Arguments

x	A list object of class marginals, the output of gibbs_ad, or of class use_marginals, to plot the output of gibbs_ad_use].
events	If plotting a marginals object, a vector or element of the event names to plot. Maximum number of events is 12.
aspect	If plotting a use_marginals object, that is, the output of gibbs_ad_use, a vector of one or more of c("production", "use", "deposition"). The default is all three.
display_name	If plotting a use_marginals object, the name of the artifact type to display in the histogram legend. Default is "Type".
breaks	The number or method of breaks in the histogram. Default is "Freedman-Diaconis". See hist for more.
xlim	The limits of the x-axis. Default is set to the min/max values of all samples.
ylim	The limits of the y-axis. This may need to be adjusted if densities have an extremely narrow interval.
xlab	Label for the x-axis. Default is "Year".
palette	A vector providing the color palette of the histogram. The default is "colorBlindness::paletteMartin (see palettes_d).
opacity	The opacity/transparency of the histograms for visualizing overlapping events, a value between 0 and 1 (default).
legend_pos	The position of the legend in the plot. Default is "topright".

Details

Also see also tidy_marginals for exporting the results of these functions into tidy data frame for custom plotting in e.g., ggplot2.

Value

A density histogram of the selected events/aspects.

A density histogram of the selected events/aspect.

Examples

```
x <- c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J")
y <- c("B", "D", "G", "H", "K")
z <- c("F", "K", "L", "M")
contexts <- list(x, y, z)

f1 <- list(id = "find01", assoc = "D", type = c("type1", "form1"))
f2 <- list(id = "find02", assoc = "E", type = c("type1", "form2"))
f3 <- list(id = "find03", assoc = "G", type = c("type1", "form1"))
f4 <- list(id = "find04", assoc = "H", type = c("type2", "form1"))
f5 <- list(id = "find05", assoc = "I", type = "type2")
f6 <- list(id = "find06", assoc = "H", type = NULL)
```

```
artifacts <- list(f1, f2, f3, f4, f5, f6)

# external constraints
coin1 <- list(id = "coin1", assoc = "B", type = NULL, samples = runif(100,-320,-300))
coin2 <- list(id = "coin2", assoc = "G", type = NULL, samples = seq(37, 41, length = 100))
destr <- list(id = "destr", assoc = "J", type = NULL, samples = 79)

tpq_info <- list(coin1, coin2)
taq_info <- list(destr)

result <- gibbs_ad(contexts, finds = artifacts, tpq = tpq_info, taq = taq_info)

# deposition of "B"
histogram(result, "B")

# deposition of "coin2" and deposition of "G"
histogram(result, c("coin2", "G"), opacity = 0.5)

# production of "type2" and deposition of "H"
histogram(result, c("H", "type2"), opacity = 0.5)

# production, use, and deposition of "type1"
type1_use <- gibbs_ad_use(result, artifacts, type = "type1",
                          max_samples = 1000, size = 500, mcse_crit = 2)
histogram(type1_use)
```

`ids_of_types` *Ids of Types*

Description

Given a `list` object of finds (with keys of `id`, `assoc`, `type` in each entry), return a vector of the `id` elements that belong to one or more specified type.

Usage

```
ids_of_types(input, type = NULL)

## S3 method for class 'list'
ids_of_types(input, type = NULL)
```

Arguments

`input`	A `list` object whose elements are a list containing the keys of `id`, `assoc`, `type`.
`type`	A vector or element

Value

A vector of ids within a `list` object of `finds` class,

Examples

```
f1 <- list(id = "find01", assoc = "D", type = c("type1", "form1"))
f2 <- list(id = "find02", assoc = "E", type = c("type1", "form2"))
f3 <- list(id = "find03", assoc = "G", type = c("type1", "form1"))
f4 <- list(id = "find04", assoc = "H", type = c("type2", "form1"))
f5 <- list(id = "find05", assoc = "I", type = "type2")
f6 <- list(id = "find06", assoc = "H", type = NULL)

artifacts <- list(f1, f2, f3, f4, f5, f6)

ids_of_types(artifacts, type = "type1")
ids_of_types(artifacts, type = c("type1", "type2"))
```

---

msd *Mean Squared Displacement of Events*

Description

Computes the mean squared displacement (MSD) of all events contained in the relative sequences and absolute constraints used in the execution of gibbs_ad.

Usage

```
msd(
  marginalized,
  sequences,
  finds = NULL,
  max_samples = 10^5,
  size = 10^3,
  mcse_crit = 0.5,
  tpq = NULL,
  taq = NULL,
  alpha_ = -5000,
  omega_ = 1950,
  rule = "naive"
)

## S3 method for class 'marginals'
msd(
  marginalized,
  sequences,
  finds = NULL,
  max_samples = 10^5,
  size = 10^3,
  mcse_crit = 0.5,
  tpq = NULL,
```

```
  taq = NULL,
  alpha_ = -5000,
  omega_ = 1950,
  rule = "naive"
)
```

Arguments

marginalized	The results of gibbs_ad.gibbs_ad.
sequences	A list of relative sequences of elements (e.g., contexts) used to compute marginalized.
finds	Optional. A list of finds related to (contained in) the elements of sequences.
max_samples	Maximum number of samples to run. Default is 10^5.
size	The number of samples to take on each iteration of the main Gibbs sampler. Default is 10^3.
mcse_crit	Criterion for the Monte Carlo standard error to stop the Gibbs sampler. A higher MCSE is recommended for situations with a higher number of events in order to reduce computational time.
tpq	A list containing *termini post quos* used to compute marginalized. See gibbs_ad for details.
taq	A list containing *termini ante quos* used to compute marginalized. See gibbs_ad for details.
alpha_	An initial *t.p.q.* to limit any elements which may occur before the first provided *t.p.q.* Default is -5000.
omega_	A final *t.a.q.* to limit any elements which may occur after the after the last provided *t.a.q.* Default is 1950.
rule	The rule for computing an estimated date of production. See gibbs_ad for details.

Details

The MSD entails the following jackknife/leave-one-out style routine:

- Each event is omitted from all relative and absolute sequences, and the function gibbs_ad is re-run to compute a "jackknifed" Monte Carlo mean for that event.
  - The squared difference of this jackknifed Monte Carlo mean and the original is then computed as its squared "displacement" in time.
  - The mean of the squared displacements of all events is then computed and attributed to the omitted event.

If an event has a low MSD, it bears a low impact on the rest of the events within the full joint conditional density. If it is has a high MSD, other events depend heavily upon its inclusion in the full joint density.

Trimming is not implemented in the computation of MSD, and so attention should be paid to the selection of alpha_ and omega_, and reported. This is owing to the way in which, if an absolute constraint (tpq or taq) is omitted that happens to be an earliest or latest bounding event, there still needs to be earliest and latest thresholds in place.

This function is fairly computationally intensive and thus a lower value of max_samples and a higher value of mcse_crit may be warranted

Value

Output is a list containing a data frame `MSD_stats` giving the mean MC date, the MCSE, the MSD, the variance of the squared displacements (not the standard error), and sample size, as well as a vector bounds of the values of `alpha_` and `omega_`.

Examples

```
x <- c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J")
y <- c("B", "D", "G", "H", "K")
z <- c("F", "K", "L", "M")
contexts <- list(x, y, z)

f1 <- list(id = "find01", assoc = "D", type = c("type1", "form1"))
f2 <- list(id = "find02", assoc = "E", type = c("type1", "form2"))
f3 <- list(id = "find03", assoc = "G", type = c("type1", "form1"))
f4 <- list(id = "find04", assoc = "H", type = c("type2", "form1"))
f5 <- list(id = "find05", assoc = "I", type = "type2")
f6 <- list(id = "find06", assoc = "H", type = NULL)

artifacts <- list(f1, f2, f3, f4, f5, f6)

# external constraints
coin1 <- list(id = "coin1", assoc = "B", type = NULL, samples = runif(100,-320,-300))
coin2 <- list(id = "coin2", assoc = "G", type = NULL, samples = seq(37, 41, length = 100))
destr <- list(id = "destr", assoc = "J", type = NULL, samples = 79)

tpq_info <- list(coin1, coin2)
taq_info <- list(destr)

result <- gibbs_ad(contexts, finds = artifacts, tpq = tpq_info, taq = taq_info)

result_msd <- msd(result, contexts, finds = artifacts, max_samples = 5000,
                  mcse_crit = 2, tpq = tpq_info, taq = taq_info)
```

`quae_antea` *Quae Antea*

Description

For a `list` of multiple partial sequences (of `vector` objects), generate another `list` which, for each element, gives the elements that occur before it ("*quae antea*"). This is analogous to a recursive trace through all partial sequences from right to left. An element `"alpha"` is added to all sets to avoid empty vectors. See also `quae_postea`.

Usage

```
quae_antea(obj)
```

```
## S3 method for class 'list'
quae_antea(obj)
```

Arguments

obj	A list of vector objects which represent ordered sequences.

Value

A list of vector objects, which contain the elements that occur before any one given element in the input sequences.

Examples

```
x <- c("A", "B", "C")
y <- c("B", "D", "E", "C", "F")
z <- c("C", "G")
a <- list(x, y, z)

quae_antea(a)
```

---

quae_postea *Quae Postea*

Description

For a list of multiple partial sequences (of vector objects), generate another list which, for each element, gives all elements that occur after it ("*quae postea*"). This is analogous to a recursive trace through all partial sequences from left to right. A final element "omega" is added to all sets to avoid empty vectors. See also quae_antea.

Usage

```
quae_postea(obj)

## S3 method for class 'list'
quae_postea(obj)
```

Arguments

obj	A list of vector objects which reperesent ordered sequences.

Value

A list of vector objects, which contain the elements that occur after any one given element in the input sequences.

Examples

```
x <- c("A", "B", "C")
y <- c("B", "D", "E", "C", "F")
z <- c("C", "G")
a <- list(x, y, z)

quae_postea(a)
```

`seq_adj` *Adjust Sequence to Target*

Description

Given an "input" sequence of elements and another "target" seqeunce that contains fewer elements in a different order, shift the order of the input sequence to match that of the target, keeping all other elements as proximate to one another as possible. This adjusted ranking is accomplished using piecewise linear interpolation between joint elements ranks. That is, joint rankings are plotted, with input rankings along the x axis and target rankings on the y axis. Remaining rankings in the input sequence are assigned a ranking of y based on the piecewise linear function between joint rankings. If the rank order of elements in the target are identical to those in the input, the result is identical to the input. A minimum number of three joint elements in both the input and target are required.

Usage

```
seq_adj(input, target)

## S3 method for class 'character'
seq_adj(input, target)
```

Arguments

`input`	A vector of elements in a sequence.
`target`	A vector of elements in a sequence, containing at least three of the same elements as `input`.

Value

A vector of the adjusted sequence.

Examples

```
x <- c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J") # the input sequence
y <- c("D", "A", "J") # the target sequence

seq_adj(x, y)
```

seq_check *Sequence Check*

Description

For a `list` of partial sequences (of `vector` objects), check to see that joint elements of each occur the same order. That is, for two sequences with elements A, B, C, D, E and B, D, F, E, all joint elements must occur in the same order to pass the check. Two sequences A, B, C, D, E and A, F, D, C, E would not pass this check as the elements C and D occur in different orders in either sequence.

Usage

```
seq_check(obj)

## S3 method for class 'list'
seq_check(obj)
```

Arguments

`obj`	A `list` of `vector` objects which represent a sequence.

Details

Event names `alpha` and `omega` are reserved for the ultimate boundaries of the chronological framework and should not be used in naming events in sequences.

Value

`TRUE` or `FALSE`

Examples

```
x <- c("A", "B", "C", "D", "E")
y <- c("B", "D", "F", "E")
a <- list(x, y)

seq_check(a)

z <- c("B", "F", "C")
b <- list(x, y, z)

seq_check(b)
```

sq_disp *Squared Displacement for a Target Event*

Description

Computes the squared displacement for a target event within the joint conditional density, estimating how much the omission of another event will change the date of that event. See also msd.

Usage

```
sq_disp(
  marginalized,
  target,
  sequences,
  finds = NULL,
  max_samples = 10^5,
  size = 10^3,
  mcse_crit = 0.5,
  tpq = NULL,
  taq = NULL,
  alpha_ = -5000,
  omega_ = 1950,
  rule = "naive"
)

## S3 method for class 'marginals'
sq_disp(
  marginalized,
  target,
  sequences,
  finds = NULL,
  max_samples = 10^5,
  size = 10^3,
  mcse_crit = 0.5,
  tpq = NULL,
  taq = NULL,
  alpha_ = -5000,
  omega_ = 1950,
  rule = "naive"
)
```

Arguments

marginalized	The results of gibbs_ad.gibbs_ad.
target	The target event (any event in marginalized) for which to estimate squared displacement.

sequences	A list of relative sequences of elements (e.g., contexts) used to compute marginalized.
finds	Optional. A list of finds related to (contained in) the elements of sequences.
max_samples	Maximum number of samples to run. Default is 10^5.
size	The number of samples to take on each iteration of the main Gibbs sampler. Default is 10^3.
mcse_crit	Criterion for the Monte Carlo standard error to stop the Gibbs sampler. A higher MCSE is recommended for situations with a higher number of events in order to reduce computational time.
tpq	A list containing *termini post quos* used to compute marginalized. See gibbs_ad for details.
taq	A list containing *termini ante quos* used to compute marginalized. See gibbs_ad for details.
alpha_	An initial *t.p.q.* to limit any elements which may occur before the first provided *t.p.q.* Default is -5000.
omega_	A final *t.a.q.* to limit any elements which may occur after the after the last provided *t.a.q.* Default is 1950.
rule	The rule for computing an estimated date of production. See gibbs_ad for details.

Details

Displacement is computed via the following jackknife/leave-one-out-style routine:

- Each event, excluding the target event itself, is omitted from all relative and absolute sequences, and the function gibbs_ad is re-run to compute a "jackknifed" Monte Carlo mean for the target event.
  - The squared difference of this jackknifed Monte Carlo mean and the original is then computed as its squared "displacement" in time.

If an event has a low squared displacement, it has a low impact on the dating of the target event. If it is has a high squared displacement, the target event's date depends heavily upon its inclusion in the full joint density.

Trimming is not implemented in the estimation of squared displacement, and so attention should be paid to the selection of alpha_ and omega_, and reported. This is owing to the way in which, if an absolute constraint (tpq or taq) is omitted that happens to be an earliest or latest bounding event, there still needs to be earliest and latest thresholds in place.

This function is fairly computationally intensive, and so a lower value of max_samples or higher value of mcse_crit may be warranted.

Value

Output is a list containing a data frame sq_disp giving the diplacement with respect to all other events and a vector bounds of the values of alpha_ and omega_.

Examples

```
x <- c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J")
y <- c("B", "D", "G", "H", "K")
z <- c("F", "K", "L", "M")
contexts <- list(x, y, z)

f1 <- list(id = "find01", assoc = "D", type = c("type1", "form1"))
f2 <- list(id = "find02", assoc = "E", type = c("type1", "form2"))
f3 <- list(id = "find03", assoc = "G", type = c("type1", "form1"))
f4 <- list(id = "find04", assoc = "H", type = c("type2", "form1"))
f5 <- list(id = "find05", assoc = "I", type = "type2")
f6 <- list(id = "find06", assoc = "H", type = NULL)

artifacts <- list(f1, f2, f3, f4, f5, f6)

# external constraints
coin1 <- list(id = "coin1", assoc = "B", type = NULL, samples = runif(100,-320,-300))
coin2 <- list(id = "coin2", assoc = "G", type = NULL, samples = seq(37, 41, length = 100))
destr <- list(id = "destr", assoc = "J", type = NULL, samples = 79)

tpq_info <- list(coin1, coin2)
taq_info <- list(destr)

result <- gibbs_ad(contexts, finds = artifacts, tpq = tpq_info, taq = taq_info)

# max_samples lowered and msce_crit raised for examples

# squared displacement for depositional context "E"
E_sqdisp <- sq_disp(result, target = "E", sequences = contexts,
                    max_samples = 3000, mcse_crit = 2, tpq = tpq_info, taq = taq_info)

# squared displacement for production of artifact type "type1"
type1_sqdisp <- sq_disp(result, target = "type1", sequences = contexts, finds = artifacts,
                      max_samples = 3000, mcse_crit = 2, tpq = tpq_info, taq = taq_info)
```

`synth_rank` *Synthetic Ranking*

Description

Using a `list` two or more partial sequences, all of which observe the same order of elements, create a single "synthetic" ranking. This is accomplished by counting the total number of elements after running a recursive trace through all partial sequences (via `quae_postea`). If partial sequences are inconsistent in their rankings, a `NULL` value is returned.

Usage

```
synth_rank(obj, ties = "average")
```

```
## S3 method for class 'list'
synth_rank(obj, ties = "average")
```

Arguments

obj	A list of vector objects which represent a sequence.
ties	The way in which ties are handled per the rank function. The default is "ties = average".

Value

A single vector containing the synthesized ranking.

Examples

```
x <- c("A", "B", "C", "D", "E")
y <- c("B", "D", "F", "E")
a <- list(x, y)

synth_rank(a)
```

---

tidy_marginals *Convert Marginals to Tidy (Molten) Data Frame*

Description

Takes the results of gibbs_ad or gibbs_ad_use and "melts" the list into a tidy data frame (Wickham 2014). Each row of the molten data frame will contain the index of the Monte Carlo sample, the sample itself, and then the event name.

Usage

```
tidy_marginals(input)

## S3 method for class 'marginals'
tidy_marginals(input)

## S3 method for class 'use_marginals'
tidy_marginals(input)
```

Arguments

input	An object of class marginals or use_marginals, the output of gibbs_ad or gibbs_ad_use.

Value

A data frame giving the MC sampling index (`idx`), the sample (`year`), and the event (`event`).

References

Wickham H (2014). "Tidy Data." *Journal of Statistical Software*, **59**, 1–23. doi:10.18637/jss.v059.i10.

Examples

```
x <- c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J")
y <- c("B", "D", "G", "H", "K")
z <- c("F", "K", "L", "M")

contexts <- list(x, y, z)

f1 <- list(id = "find01", assoc = "D", type = c("type1", "form1"))
f2 <- list(id = "find02", assoc = "E", type = c("type1", "form2"))
f3 <- list(id = "find03", assoc = "G", type = c("type1", "form1"))
f4 <- list(id = "find04", assoc = "H", type = c("type2", "form1"))
f5 <- list(id = "find05", assoc = "I", type = "type2")
f6 <- list(id = "find06", assoc = "H", type = NULL)

artifacts <- list(f1, f2, f3, f4, f5, f6)

# external constraints
coin1 <- list(id = "coin1", assoc = "B", type = NULL, samples = runif(100,-320,-300))
coin2 <- list(id = "coin2", assoc = "G", type = NULL, samples = seq(37, 41, length = 100))
destr <- list(id = "destr", assoc = "J", type = NULL, samples = 79)

tpq_info <- list(coin1, coin2)
taq_info <- list(destr)

result <- gibbs_ad(contexts, finds = artifacts, tpq = tpq_info, taq = taq_info)

tidy_marginals(result)
```

`traceplot` *Traceplot of Gibbs Samples*

Description

Wrapper around `plot` to make a traceplot of Gibbs samples from `gibbs_ad`. See `histogram` for plotting a density histogram of events.

Usage

```
traceplot(
  x,
  events = NULL,
  xlim = NULL,
  ylim = NULL,
  xlab = "Index",
  ylab = "Year",
  palette = NULL,
  opacity = 1,
  legend_pos = "topright",
  ...
)

## S3 method for class 'marginals'
traceplot(
  x,
  events = NULL,
  xlim = NULL,
  ylim = NULL,
  xlab = "Index",
  ylab = "Year",
  palette = NULL,
  opacity = 1,
  legend_pos = "topright",
  ...
)
```

Arguments

x	A list object of class marginals or use_marginals, the output of gibbs_ad or gibbs_ad_use respectively.
events	A vector or element of the event names to plot. Maximum number of events is 12.
xlim	The limits of the x-axis (optional).
ylim	The limits of the y-axis (optional).
xlab	Label for the y-axis. Default is "Index".
ylab	Label for the y-axis. Default is "Year".
palette	A vector providing the color palette of the histogram. The default is "colorBlindness::paletteMartin (see palettes_d).
opacity	The opacity/transparency of the traceplot, if visualizing overlapping events. A value between 0 and 1 (default).
legend_pos	The position of the legend in the plot. Default is "topright".
...	Additional graphical parameters passed to plot.

Details

Also see tidy_marginals for exporting the results of these functions into tidy data frame for custom plotting in e.g., ggplot2.

Value

A traceplot of the Gibbs samples of the selected events.

Examples

```
x <- c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J")
y <- c("B", "D", "G", "H", "K")
z <- c("F", "K", "L", "M")
contexts <- list(x, y, z)

f1 <- list(id = "find01", assoc = "D", type = c("type1", "form1"))
f2 <- list(id = "find02", assoc = "E", type = c("type1", "form2"))
f3 <- list(id = "find03", assoc = "G", type = c("type1", "form1"))
f4 <- list(id = "find04", assoc = "H", type = c("type2", "form1"))
f5 <- list(id = "find05", assoc = "I", type = "type2")
f6 <- list(id = "find06", assoc = "H", type = NULL)

artifacts <- list(f1, f2, f3, f4, f5, f6)

# external constraints
coin1 <- list(id = "coin1", assoc = "B", type = NULL, samples = runif(100,-320,-300))
coin2 <- list(id = "coin2", assoc = "G", type = NULL, samples = seq(37, 41, length = 100))
destr <- list(id = "destr", assoc = "J", type = NULL, samples = 79)

tpq_info <- list(coin1, coin2)
taq_info <- list(destr)

result <- gibbs_ad(contexts, finds = artifacts, tpq = tpq_info, taq = taq_info)

traceplot(result, "B")
traceplot(result, c("coin1", "B", "H"), opacity = 0.5)
```

Index