Package ‘kairos’

June 18, 2022

Title Analysis of Chronological Patterns from Archaeological Count Data

Version 1.1.0

Description A toolkit for absolute dating and analysis of chronological patterns. This package includes functions for chronological modeling and dating of archaeological assemblages from count data. It provides methods for matrix seriation. It also allows to compute time point estimates and density estimates of the occupation and duration of an archaeological site.

License GPL (>= 3)

URL https://packages.tesselle.org/kairos/, https://github.com/tesselle/kairos

BugReports https://github.com/tesselle/kairos/issues

Depends R (>= 3.4)

Imports arkhe (>= 0.5.0), boot, dimensio (>= 0.2.2), extraDistr, ggplot2, Hmisc, methods, rlang, stats, utils

Suggests covr, folio, khroma, knitr, rmarkdown, testthat (>= 3.0.0), vdiffr (>= 1.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.0

Collate 'AllClasses.R' 'AllGenerics.R' 'aoristic.R' 'apportion.R' 'coerce.R' 'event_date.R' 'event_plot.R' 'event_resample.R' 'fit.R' 'kairos-package.R' 'mcd.R' 'mutators.R' 'plot.R' 'reexport.R' 'seriate_average.R' 'seriate_permute.R' 'seriate_rank.R' 'seriate_resample.R' 'show.R' 'subset.R' 'utilities.R' 'validate.R' 'zzz.R'

NeedsCompilation no

Author Nicolas Frerebeau [aut, cre] (<https://orcid.org/0000-0001-5759-4944>, Université Bordeaux Montaigne),
Brice Lebrun [ctb] (<https://orcid.org/0000-0001-7503-8685>, Université Bordeaux Montaigne),
Ben Marwick [ctb] (<https://orcid.org/0000-0001-7879-4531>, University of Washington),
Anne Philippe [ctb] (<https://orcid.org/0000-0002-5331-5087>, Université de Nantes)

Maintainer Nicolas Frerebeau <nicolas.frerebeau@u-bordeaux-montaigne.fr>

Repository CRAN

Date/Publication 2022-06-18 13:20:02 UTC

R topics documented:

`aoristic` *Aoristic Analysis*

Description

Computes the aoristic sum.

Usage

```
aoristic(x, y, ...)

## S4 method for signature 'numeric,numeric'
aoristic(
  x,
  y,
  step = 1,
  start = min(x, na.rm = TRUE),
  stop = max(y, na.rm = TRUE),
  weight = TRUE,
  groups = NULL
)

## S4 method for signature 'list,missing'
aoristic(x, step = 1, start = NULL, stop = NULL, weight = TRUE, groups = NULL)
```

Arguments

`x`	A `numeric` vector. If `y` is missing, must be a `list` (or a `data.frame`) with `numeric` components (columns) `from` and `to`.
`y`	A `numeric` vector. If missing, an attempt is made to interpret `x` in a suitable way.
`...`	Currently not used.
`step`	A length-one `integer` vector giving the step size, i.e. the width of each time step in the time series (in years CE; defaults to 1 - i.e. annual level).
`start`	A length-one `numeric` vector giving the beginning of the time window (in years CE).
`stop`	A length-one `numeric` vector giving the end of the time window (in years CE).
`weight`	A `logical` scalar: should the aoristic sum be weighted by the length of periods (default). If `FALSE` the aoristic sum is the number of elements within a time block.
`groups`	A `factor` vector in the sense that `as.factor(groups)` defines the grouping. If `x` is a `list` (or a `data.frame`), `groups` can be a length-one vector giving the index of the grouping component (column) of `x`.

Details

Aoristic analysis is used to determine the probability of contemporaneity of archaeological sites or assemblages. The aoristic analysis distributes the probability of an event uniformly over each temporal fraction of the period considered. The aoristic sum is then the distribution of the total number of events to be assumed within this period.

Muller and Hinz (2018) pointed out that the overlapping of temporal intervals related to period categorization and dating accuracy is likely to bias the analysis. They proposed a weighting method to overcome this problem. This method is not implemented here (for the moment), see the **aoristAAR** package.

Value

An AoristicSum object.

Author(s)

N. Frerebeau

References

Crema, E. R. (2012). Modelling Temporal Uncertainty in Archaeological Analysis. *Journal of Archaeological Method and Theory*, 19(3): 440-61. doi:10.1007/s1081601191223.

Johnson, I. (2004). Aoristic Analysis: Seeds of a New Approach to Mapping Archaeological Distributions through Time. *In* Ausserer, K. F., Börner, W., Goriany, M. & Karlhuber-Vöckl, L. (ed.), *Enter the Past - The E-Way into the Four Dimensions of Cultural Heritage*, Oxford: Archaeopress, p. 448-52. BAR International Series 1227. doi:10.15496/publikation2085

Müller-Scheeßel, N. & Hinz, M. (2018). *Aoristic Research in R: Correcting Temporal Categorizations in Archaeology*. Presented at the Human History and Digital Future (CAA 2018), Tubingen, March 21. `https://www.youtube.com/watch?v=bUBukex30QI`.

Palmisano, A., Bevan, A. & Shennan, S. (2017). Comparing Archaeological Proxies for Long-Term Population Patterns: An Example from Central Italy. *Journal of Archaeological Science*, 87: 59-72. doi:10.1016/j.jas.2017.10.001.

Ratcliffe, J. H. (2000). Aoristic Analysis: The Spatial Interpretation of Unspecific Temporal Events. *International Journal of Geographical Information Science*, 14(7): 669-79. doi:10.1080/136588100424963.

Ratcliffe, J. H. (2002). Aoristic Signatures and the Spatio-Temporal Analysis of High Volume Crime Patterns. *Journal of Quantitative Criminology*, 18(1): 23-43. doi:10.1023/A:1013240828824.

See Also

`roc()`, plot()

Other chronological analysis: `apportion()`, `fit()`, `roc()`

Examples

```
## Aoristic Analysis
data("zuni", package = "folio")

## Set the start and end dates for each ceramic type
dates <- list(
  LINO = c(600, 875), KIAT = c(850, 950), RED = c(900, 1050),
  GALL = c(1025, 1125), ESC = c(1050, 1150), PUBW = c(1050, 1150),
  RES = c(1000, 1200), TULA = c(1175, 1300), PINE = c(1275, 1350),
  PUBR = c(1000, 1200), WING = c(1100, 1200), WIPO = c(1125, 1225),
  SJ = c(1200, 1300), LSJ = c(1250, 1300), SPR = c(1250, 1300),
  PINER = c(1275, 1325), HESH = c(1275, 1450), KWAK = c(1275, 1450)
)

## Keep only assemblages that have a sample size of at least 10
```

```
keep <- apply(X = zuni, MARGIN = 1, FUN = function(x) sum(x) >= 10)

## Calculate date ranges for each assemblage
span <- apply(
  X = zuni[keep, ],
  FUN = function(x, dates) {
    z <- range(unlist(dates[x > 0]))
    names(z) <- c("from", "to")
    z
  },
  MARGIN = 1,
  dates = dates
)

## Coerce to data.frame
span <- as.data.frame(t(span))

## Calculate aoristic sum (normal)
aorist_raw <- aoristic(span, step = 50, weight = FALSE)
plot(aorist_raw)

## Calculate aoristic sum (weights)
aorist_weighted <- aoristic(span, step = 50, weight = TRUE)
plot(aorist_weighted)

## Calculate aoristic sum (weights) by group
groups <- rep(c("A", "B", "C"), times = c(50, 90, 139))
aorist_groups <- aoristic(span, step = 50, weight = TRUE, groups = groups)
plot(aorist_groups)

## Rate of change
roc_weighted <- roc(aorist_weighted, n = 30)
plot(roc_weighted)

## Rate of change by group
roc_groups <- roc(aorist_groups, n = 30)
plot(roc_groups)
```

`AoristicSum-class` *Aoristic Sum*

Description

An S4 class to represent an aoristic analysis results.

Slots

`from` A `numeric` vector.

`to` A `numeric` vector.

`step` A length-one `numeric` vector giving the time-blocks width.

`weights` A `numeric` vector.

`breaks` A `numeric` vector giving the date break between time-blocks.

`blocks` A `character` vector giving the time-blocks.

`p` A `numeric` `array` giving the aorisitic probabilities.

`groups` A `character` vector.

Note

This class inherits from base `matrix`.

Author(s)

N. Frerebeau

See Also

Other classes: `CountApportion-class`, `EventDate-class`, `IncrementTest-class`, `MeanDate-class`, `PermutationOrder-class`, `RateOfChange-class`, `RefineCA-class`

`apportion` *Chronological Apportioning*

Description

Chronological Apportioning

Usage

```
apportion(object, ...)

## S4 method for signature 'data.frame'
apportion(
  object,
  s0,
  s1,
  t0,
  t1,
  from = min(s0),
  to = max(s1),
  step = 25,
  method = c("uniform", "truncated"),
  z = 2,
  progress = getOption("kairos.progress")
)

## S4 method for signature 'matrix'
```

```
apportion(
  object,
  s0,
  s1,
  t0,
  t1,
  from = min(s0),
  to = max(s1),
  step = 25,
  method = c("uniform", "truncated"),
  z = 2,
  progress = getOption("kairos.progress")
)
```

Arguments

`object`	An $m \times p$ numeric `matrix` or a `data.frame` of count data (absolute frequencies).
`...`	Currently not used.
`s0`	A length-m `numeric` vector giving the site beginning dates expressed in CE years (BCE years must be given as negative numbers).
`s1`	A length-m `numeric` vector giving the site end dates expressed in CE years (BCE years must be given as negative numbers).
`t0`	A length-p `numeric` vector giving the type beginning dates expressed in CE years (BCE years must be given as negative numbers).
`t1`	A length-p `numeric` vector giving the type end dates expressed in CE years (BCE years must be given as negative numbers).
`from`	A length-one `numeric` vector giving the beginning of the period of interest (in years CE).
`to`	A length-one `numeric` vector giving the end of the period of interest (in years CE).
`step`	A length-one `integer` vector giving the step size, i.e. the width of each time step for apportioning (in years CE; defaults to 25).
`method`	A `character` string specifying the distribution to be used (type popularity curve). It must be one of "`uniform`" (uniform distribution) or "`truncated`" (truncated standard normal distribution). Any unambiguous substring can be given.
`z`	An `integer` value giving the lower and upper truncation points (defaults to 2). Only used if `method` is "`truncated`".
`progress`	A `logical` scalar: should a progress bar be displayed?

Author(s)

N. Frerebeau

References

Roberts, J. M., Mills, B. J., Clark, J. J., Haas, W. R., Huntley, D. L. & Trowbridge, M. A. (2012). A Method for Chronological Apportioning of Ceramic Assemblages. *Journal of Archaeological Science*, 39(5): 1513-20. doi:10.1016/j.jas.2011.12.022.

See Also

Other chronological analysis: `aoristic()`, `fit()`, `roc()`

`CountApportion-class` *Count Apportioning*

Description

An S4 class to represent an artifact apportioning results. Gives the apportioning of artifact types (columns) per site (rows) and per period (dim. 3).

Slots

- `p` An `array` giving the probability of apportioning an artifact type to a given period.
- `method` A `character` string specifying the distribution used for apportioning (type popularity curve).
- `from` A length-one `numeric` vector giving the beginning of the period of interest (in years AD).
- `to` A length-one `numeric` vector giving the end of the period of interest (in years AD).
- `step` A length-one `integer` vector giving the step size, i.e. the width of each time step for apportioning (in years AD).

Note

This class inherits from base `array`.

Author(s)

N. Frerebeau

See Also

Other classes: `AoristicSum-class`, `EventDate-class`, `IncrementTest-class`, `MeanDate-class`, `PermutationOrder-class`, `RateOfChange-class`, `RefineCA-class` event *Event and Accumulation Dates*

Description

- `event()` fit a date event model.
- `predict_event()` and `predict_accumulation()` estimates the event and accumulation dates of an assemblage.

Usage

```
event(object, dates, ...)

predict_event(object, data, ...)

predict_accumulation(object, data, ...)

## S4 method for signature 'data.frame,numeric'
event(object, dates, cutoff = 90, level = 0.95, ...)

## S4 method for signature 'matrix,numeric'
event(object, dates, cutoff = 90, level = 0.95, ...)

## S4 method for signature 'EventDate,missing'
predict_event(object, margin = 1, level = 0.95)

## S4 method for signature 'EventDate,matrix'
predict_event(object, data, margin = 1, level = 0.95)

## S4 method for signature 'EventDate,missing'
predict_accumulation(object)

## S4 method for signature 'EventDate,matrix'
predict_accumulation(object, data)

## S4 method for signature 'EventDate'
jackknife(object, level = 0.95, progress = getOption("kairos.progress"), ...)

## S4 method for signature 'EventDate'
bootstrap(
  object,
  level = 0.95,
  probs = c(0.05, 0.95),
  n = 1000,
  progress = getOption("kairos.progress"),
  ...
)
```

Arguments

object	An $m \times p$ numeric matrix or a data.frame of count data (absolute frequencies).
dates	A numeric vector of dates expressed in CE years (BCE years must be given as negative numbers). If named, the names must match the row names of object.
...	Further arguments to be passed to internal methods.
data	A numeric matrix or a data.frame of count data (absolute frequencies)for which to predict event and accumulation dates.
cutoff	An integer giving the cumulative percentage of variance used to select CA factorial components for linear model fitting (see details). All compounds with a cumulative percentage of variance of less than the cutoff value will be retained.
level	A length-one numeric vector giving the confidence level.
margin	A numeric vector giving the subscripts which the prediction will be applied over: 1 indicates rows, 2 indicates columns.
progress	A logical scalar: should a progress bar be displayed?
probs	A numeric vector of probabilities with values in $[0, 1]$ (see stats::quantile()). If NULL, quantiles are not computed.
n	A non-negative integer giving the number of bootstrap replications.

Details

This is an implementation of the chronological modeling method proposed by Bellanger and Husi (2012, 2013).

Event and accumulation dates are density estimates of the occupation and duration of an archaeological site (Bellanger and Husi 2012, 2013). The event date is an estimation of the *terminus post-quem* of an archaeological assemblage. The accumulation date represents the "chronological profile" of the assemblage. According to Bellanger and Husi (2012), accumulation date can be interpreted "at best [...] as a formation process reflecting the duration or succession of events on the scale of archaeological time, and at worst, as imprecise dating due to contamination of the context by residual or intrusive material." In other words, accumulation dates estimate occurrence of archaeological events and rhythms of the long term.

This method relies on strong archaeological and statistical assumptions (see vignette("kairos")).

Value

- event() returns an EventDate object.
- predict_event() returns a data.frame.
- predict_accumulation() returns a MeanDate object.
- bootstrap() and jackknife() return a data.frame.

Resampling

If `jackknife()` is used, one type/fabric is removed at a time and all statistics are recalculated. In this way, one can assess whether certain type/fabric has a substantial influence on the date estimate. A three columns `data.frame` is returned, giving the results of the resampling procedure (jackknifing fabrics) for each assemblage (in rows) with the following columns:

`mean` The jackknife mean (event date).

`bias` The jackknife estimate of bias.

`error` The standard error of predicted means.

If `bootstrap()` is used, a large number of new bootstrap assemblages is created, with the same sample size, by resampling each of the original assemblage with replacement. Then, examination of the bootstrap statistics makes it possible to pinpoint assemblages that require further investigation.

A five columns `data.frame` is returned, giving the bootstrap distribution statistics for each replicated assemblage (in rows) with the following columns:

`min` Minimum value.

`mean` Mean value (event date).

`max` Maximum value.

`Q5` Sample quantile to 0.05 probability.

`Q95` Sample quantile to 0.95 probability.

Note

All results are rounded to zero decimal places (sub-annual precision does not make sense in most situations). You can change this behavior with `options(kairos.precision = x)` (for `x` decimal places).

Bellanger *et al.* did not publish the data supporting their demonstration: no replication of their results is possible. This implementation must be considered **experimental** and subject to major changes in a future release.

Author(s)

N. Frerebeau

References

Bellanger, L. & Husi, P. (2013). Mesurer et modéliser le temps inscrit dans la matière à partir d'une source matérielle : la céramique médiévale. In *Mesure et Histoire Médiévale*. Histoire ancienne et médiévale. Paris: Publication de la Sorbonne, p. 119-134.

Bellanger, L. & Husi, P. (2012). Statistical Tool for Dating and Interpreting Archaeological Contexts Using Pottery. *Journal of Archaeological Science*, 39(4), 777-790. doi:10.1016/j.jas.2011.06.031.

Bellanger, L., Tomassone, R. & Husi, P. (2008). A Statistical Approach for Dating Archaeological Contexts. *Journal of Data Science*, 6, 135-154.

Bellanger, L., Husi, P. & Tomassone, R. (2006). Une approche statistique pour la datation de contextes archéologiques. *Revue de Statistique Appliquée*, 54(2), 65-81.

Bellanger, L., Husi, P. & Tomassone, R. (2006). Statistical Aspects of Pottery Quantification for the Dating of Some Archaeological Contexts. *Archaeometry*, 48(1), 169-183. doi:10.1111/j.1475-4754.2006.00249.x.

Poblome, J. & Groenen, P. J. F. (2003). Constrained Correspondence Analysis for Seriation of Sagalassos Tablewares. In Doerr, M. & Apostolis, S. (eds.), *The Digital Heritage of Archaeology*. Athens: Hellenic Ministry of Culture.

See Also

plot_event

Other dating methods: `mcd()`

Examples

```
## Not run:
utils::vignette("kairos")

## End(Not run)
```

`EventDate-class` *Date Model*

Description

S4 classes to store the event and accumulation times of archaeological assemblages.

Slots

`dates` A `numeric` vector of dates.

`model` A multiple linear model: the Gaussian multiple linear regression model fitted for event date estimation and prediction.

`cutoff` An length-one `integer` vector giving the cutoff value.

`keep` An `integer` vector.

Author(s)

N. Frerebeau

See Also

`dimensio::CA`

Other classes: `AoristicSum-class`, `CountApportion-class`, `IncrementTest-class`, `MeanDate-class`, `PermutationOrder-class`, `RateOfChange-class`, `RefineCA-class` `fit` *Frequency Increment Test*

Description

Frequency Increment Test

Usage

```
fit(object, dates, ...)

## S4 method for signature 'data.frame,numeric'
fit(object, dates)

## S4 method for signature 'matrix,numeric'
fit(object, dates)
```

Arguments

`object`	An $m \times p$ numeric `matrix` or a `data.frame` of count data (absolute frequencies).
`dates`	A `numeric` vector of dates expressed in CE years (BCE years must be given as negative numbers).
`...`	Currently not used.

Details

The Frequency Increment Test (FIT) rejects neutrality if the distribution of normalized variant frequency increments exhibits a mean that deviates significantly from zero.

Value

An IncrementTest object.

Author(s)

N. Frerebeau

References

Feder, A. F., Kryazhimskiy, S. & Plotkin, J. B. (2014). Identifying Signatures of Selection in Genetic Time Series. *Genetics*, 196(2): 509-522. doi:10.1534/genetics.113.158220.

See Also

plot()

Other chronological analysis: `aoristic()`, `apportion()`, `roc()`

Examples

```
data("merzbach", package = "folio")

## Keep only decoration types that have a maximum frequency of at least 50
keep <- apply(X = merzbach, MARGIN = 2, FUN = function(x) max(x) >= 50)
counts <- merzbach[, keep]

## Group by phase
## We use the row names as time coordinates (roman numerals)
dates <- as.numeric(utils::as.roman(rownames(counts)))

## Frequency Increment Test
freq <- fit(counts, dates)

## Plot time vs abundance and highlight selection
plot(freq)
plot(freq, roll = TRUE, window = 5)
```

IncrementTest-class *Frequency Increment Test*

Description

An S4 class to represent a Frequency Increment Test results.

Usage

```
## S4 method for signature 'IncrementTest,ANY,missing'
x[[i]]
```

Arguments

x	An object from which to extract element(s) or in which to replace element(s).
i	A `character` string specifying elements to extract.

Functions

- `[[,IncrementTest,ANY,missing-method`: Extracts information from a slot selected by subscript i. i is a length-one character vector. Returns the corresponding slot values.

Slots

`counts` An $m \times p$ `numeric` matrix of count data.

`dates` A length-m `numeric` vector of dates.

`statistic` A `numeric` vector giving the values of the t-statistic.

`parameter` An `integer` giving the degrees of freedom for the t-statistic.

`p_value` A `numeric` vector giving the the p-value for the test.

Coerce

In the code snippets below, `x` is an `IncrementTest` object.

`as.data.frame(x)` Coerces to a `data.frame`.

Author(s)

N. Frerebeau

See Also

Other classes: `AoristicSum-class`, `CountApportion-class`, `EventDate-class`, `MeanDate-class`, `PermutationOrder-class`, `RateOfChange-class`, `RefineCA-class`

`mcd` *Mean Ceramic Date*

Description

Estimates the Mean Ceramic Date of an assemblage.

Usage

```
mcd(object, dates, ...)

## S4 method for signature 'numeric,numeric'
mcd(object, dates)

## S4 method for signature 'data.frame,numeric'
mcd(object, dates)

## S4 method for signature 'matrix,numeric'
mcd(object, dates)
```

Arguments

`object`	A length-p `numeric` vector, an $m \times p$ `numeric` `matrix` or `data.frame` of count data (absolute frequencies).
`dates`	A length-p `numeric` vector of dates expressed in CE years (BCE years must be given as negative numbers).
`...`	Currently not used.

Details

The Mean Ceramic Date (MCD) is a point estimate of the occupation of an archaeological site (South 1977). The MCD is estimated as the weighted mean of the date midpoints of the ceramic types (based on absolute dates or the known production interval) found in a given assemblage. The weights are the relative frequencies of the respective types in the assemblage.

A bootstrapping procedure is used to estimate the confidence interval of a given MCD. For each assemblage, a large number of new bootstrap replicates is created, with the same sample size, by resampling the original assemblage with replacement. MCDs are calculated for each replicates and upper and lower boundaries of the confidence interval associated with each MCD are then returned.

Value

A single `numeric` value or a MeanDate object.

Note

All results are rounded to zero decimal places (sub-annual precision does not make sense in most situations). You can change this behavior with `options(kairos.precision = x)` (for `x` decimal places).

Author(s)

N. Frerebeau

References

South, S. A. (1977). *Method and Theory in Historical Archaeology*. New York: Academic Press.

See Also

plot_mcd, `bootstrap()`, `jackknife()`, `simulate()`

Other dating methods: `event()`

Examples

```
## Mean Ceramic Date
## Coerce the zuni dataset to an abundance (count) matrix
data("zuni", package = "folio")

## Set the start and end dates for each ceramic type
dates <- list(
  LINO = c(600, 875), KIAT = c(850, 950), RED = c(900, 1050),
  GALL = c(1025, 1125), ESC = c(1050, 1150), PUBW = c(1050, 1150),
  RES = c(1000, 1200), TULA = c(1175, 1300), PINE = c(1275, 1350),
  PUBR = c(1000, 1200), WING = c(1100, 1200), WIPO = c(1125, 1225),
  SJ = c(1200, 1300), LSJ = c(1250, 1300), SPR = c(1250, 1300),
  PINER = c(1275, 1325), HESH = c(1275, 1450), KWAK = c(1275, 1450)
)

## Calculate date midpoints
```

```
mid <- vapply(X = dates, FUN = mean, FUN.VALUE = numeric(1))

## Calculate MCD
mc_dates <- mcd(zuni[100:125, ], dates = mid)
head(mc_dates)

## Plot
plot(mc_dates)


## Bootstrap resampling
boot <- bootstrap(mc_dates, n = 30)
head(boot)

## Jackknife resampling
jack <- jackknife(mc_dates)
head(jack)

## Simulation
sim <- simulate(mc_dates, n = 30, interval = "percentiles")
plot(sim)
```

`MeanDate-class` *Mean Date*

Description

An S4 class to store the weighted mean date (e.g. Mean Ceramic Date) of archaeological assemblages.

Arguments

`level` A length-one `numeric` vector giving the confidence level.

Slots

`types` A length-p `numeric` vector giving the dates of the (ceramic) types.

`weights` An $m \times p$ `integer` `matrix` giving the weights used.

`simulation` A three columns `numeric` matrix giving the summary statistics of the simulated dates (`mean` and `lower` and `upper` boundaries of the confidence interval).

`replications` An `integer` giving the number of bootstrap replications.

Coerce

In the code snippets below, `x` is a `MeanDate` object.

`as.data.frame(x)` Coerces to a `data.frame`.

Note

This class inherits from base numeric.

Author(s)

N. Frerebeau

See Also

Other classes: AoristicSum-class, CountApportion-class, EventDate-class, IncrementTest-class, PermutationOrder-class, RateOfChange-class, RefineCA-class

---

mutators *Get or Set Parts of an Object*

Description

Getters and setters to retrieve or set parts of an object.

Usage

```
get_dates(x)

get_groups(x)

get_model(x)

get_weights(x)

## S4 method for signature 'AoristicSum'
get_dates(x)

## S4 method for signature 'EventDate'
get_dates(x)

## S4 method for signature 'RateOfChange'
get_dates(x)

## S4 method for signature 'AoristicSum'
get_groups(x)

## S4 method for signature 'EventDate'
get_model(x)

## S4 method for signature 'AoristicSum'
get_weights(x)
```

```
## S4 method for signature 'CountApportion'
get_weights(x)

## S4 method for signature 'MeanDate'
get_weights(x)
```

Arguments

`x`	An object from which to get or set element(s).

Value

- `set_*()` returns an object of the same sort as `x` with the new values assigned.
- `get_*()` returns the part of `x`.

Author(s)

N. Frerebeau

`PermutationOrder-class` *Permutation Order*

Description

An S4 class to represent a permutation order.

Usage

```
## S4 method for signature 'PermutationOrder,ANY,missing'
x[[i]]
```

Arguments

`x`	An object from which to extract element(s) or in which to replace element(s).
`i`	A `character` string specifying elements to extract.

Functions

- `[[,PermutationOrder,ANY,missing-method`: Extracts information from a slot selected by subscript i. i is a length-one character vector. Returns the corresponding slot values.

Slots

`rows_order` An `integer` vector giving the rows permutation.

`columns_order` An `integer` vector giving the columns permutation.

`method` A `character` string indicating the seriation method used.

Author(s)

N. Frerebeau

See Also

dimensio::CA

Other classes: AoristicSum-class, CountApportion-class, EventDate-class, IncrementTest-class, MeanDate-class, RateOfChange-class, RefineCA-class

---

plot_aoristic *Plot Aoristic Analysis*

Description

Plot Aoristic Analysis

Usage

```
## S4 method for signature 'AoristicSum'
autoplot(object, ..., facet = TRUE)

## S4 method for signature 'AoristicSum,missing'
plot(x, facet = TRUE, ...)

## S4 method for signature 'RateOfChange'
autoplot(object, ..., level = 0.95, facet = TRUE)

## S4 method for signature 'RateOfChange,missing'
plot(x, level = 0.95, facet = TRUE, ...)
```

Arguments

`object, x`	An AoristicSum object.
`...`	Currently not used.
`facet`	A `logical` scalar: should a matrix of panels defined by groups be drawn?
`level`	A length-one `numeric` vector giving the confidence level.

Value

- `autoplot()` returns a `ggplot` object.
- `plot()` is called it for its side-effects: it results in a graphic being displayed (invisibly returns `x`).

Author(s)

N. Frerebeau

See Also

`aoristic()`, `roc()`

Other plotting methods: `plot_event`, `plot_fit`, `plot_mcd`, `plot_time()`

Examples

```
## Aoristic Analysis
data("zuni", package = "folio")

## Set the start and end dates for each ceramic type
dates <- list(
  LINO = c(600, 875), KIAT = c(850, 950), RED = c(900, 1050),
  GALL = c(1025, 1125), ESC = c(1050, 1150), PUBW = c(1050, 1150),
  RES = c(1000, 1200), TULA = c(1175, 1300), PINE = c(1275, 1350),
  PUBR = c(1000, 1200), WING = c(1100, 1200), WIPO = c(1125, 1225),
  SJ = c(1200, 1300), LSJ = c(1250, 1300), SPR = c(1250, 1300),
  PINER = c(1275, 1325), HESH = c(1275, 1450), KWAK = c(1275, 1450)
)

## Keep only assemblages that have a sample size of at least 10
keep <- apply(X = zuni, MARGIN = 1, FUN = function(x) sum(x) >= 10)

## Calculate date ranges for each assemblage
span <- apply(
  X = zuni[keep, ],
  FUN = function(x, dates) {
    z <- range(unlist(dates[x > 0]))
    names(z) <- c("from", "to")
    z
  },
  MARGIN = 1,
  dates = dates
)

## Coerce to data.frame
span <- as.data.frame(t(span))

## Calculate aoristic sum (normal)
aorist_raw <- aoristic(span, step = 50, weight = FALSE)
plot(aorist_raw)

## Calculate aoristic sum (weights)
aorist_weighted <- aoristic(span, step = 50, weight = TRUE)
plot(aorist_weighted)

## Calculate aoristic sum (weights) by group
groups <- rep(c("A", "B", "C"), times = c(50, 90, 139))
aorist_groups <- aoristic(span, step = 50, weight = TRUE, groups = groups)
plot(aorist_groups)

## Rate of change
roc_weighted <- roc(aorist_weighted, n = 30)
```

```
plot(roc_weighted)

## Rate of change by group
roc_groups <- roc(aorist_groups, n = 30)
plot(roc_groups)
```

---

plot_event *Event Plot*

Description

Produces an activity or a tempo plot.

Usage

```
## S4 method for signature 'EventDate'
autoplot(
  object,
  ...,
  type = c("activity", "tempo"),
  event = FALSE,
  select = 1,
  n = 500
)

## S4 method for signature 'EventDate,missing'
plot(x, type = c("activity", "tempo"), event = FALSE, select = 1, n = 500, ...)
```

Arguments

`object, x`	A EventDate object.
`...`	Currently not used.
`type`	A `character` string indicating the type of plot. It must be one of "`activity`" (default) or "`tempo`". Any unambiguous substring can be given.
`event`	A `logical` scalar: should the distribution of the event date be displayed? Only used if type is "`activity`".
`select`	A `numeric` or `character` vector giving the selection of the assemblage that are drawn.
`n`	A length-one non-negative `numeric` vector giving the desired length of the vector of quantiles for density computation.

Value

- `autoplot()` returns a `ggplot` object.
- `plot()` is called it for its side-effects: it results in a graphic being displayed (invisibly returns `x`).

Event and Acccumulation Dates

`plot()` displays the probability estimate density curves of archaeological assemblage dates (*event* and *accumulation* dates; Bellanger and Husi 2012). The *event* date is plotted as a line, while the *accumulation* date is shown as a grey filled area.

The accumulation date can be displayed as a tempo plot (Dye 2016) or an activity plot (Philippe and Vibet 2017):

Tempo plot A tempo plot estimates the cumulative occurrence of archaeological events, such as the slope of the plot directly reflects the pace of change.

Activity plot An activity plot displays the first derivative of the tempo plot.

Author(s)

N. Frerebeau

References

Bellanger, L. & Husi, P. (2012). Statistical Tool for Dating and Interpreting Archaeological Contexts Using Pottery. *Journal of Archaeological Science*, 39(4), 777-790. doi:10.1016/j.jas.2011.06.031.

Dye, T. S. (2016). Long-Term Rhythms in the Development of Hawaiian Social Stratification. *Journal of Archaeological Science*, 71, 1-9. doi:10.1016/j.jas.2016.05.006.

Philippe, A. & Vibet, M.-A. (2017). Analysis of Archaeological Phases Using the R Package ArchaeoPhases. *Journal of Statistical Software, Code Snippets*, 93(1), 1-25. doi:10.18637/jss.v093.c01.

See Also

`event()`

Other plotting methods: `plot_aoristic`, `plot_fit`, `plot_mcd`, `plot_time()`

Examples

```
## Not run:
utils::vignette("kairos")

## End(Not run)
```

---

plot_fit *Detection of Selective Processes*

Description

Produces an abundance *vs* time diagram.

Usage

```
## S4 method for signature 'IncrementTest'
autoplot(object, ..., level = 0.95, roll = FALSE, window = 3)

## S4 method for signature 'IncrementTest,missing'
plot(x, level = 0.95, roll = FALSE, window = 3, ...)
```

Arguments

`object, x`	An object to be plotted.
`...`	Currently not used.
`level`	A length-one `numeric` vector giving the confidence level.
`roll`	A `logical` scalar: should each time series be subsetted to look for episodes of selection?
`window`	An odd `integer` giving the size of the rolling window. Only used if `roll` is `TRUE`.

Details

Results of the frequency increment test can be displayed on an abundance *vs* time diagram aid in the detection and quantification of selective processes in the archaeological record. If `roll` is `TRUE`, each time series is subsetted according to `window` to see if episodes of selection can be identified among decoration types that might not show overall selection. If so, shading highlights the data points where `fit()` identifies selection.

Value

- `autoplot()` returns a `ggplot` object.
- `plot()` is called it for its side-effects: it results in a graphic being displayed (invisibly returns `x`).

Note

Displaying FIT results on an abundance *vs* time diagram is adapted from Ben Marwick's original idea.

Author(s)

N. Frerebeau

See Also

`fit()`

Other plotting methods: `plot_aoristic`, `plot_event`, `plot_mcd`, `plot_time()`

Examples

```
data("merzbach", package = "folio")

## Keep only decoration types that have a maximum frequency of at least 50
keep <- apply(X = merzbach, MARGIN = 2, FUN = function(x) max(x) >= 50)
counts <- merzbach[, keep]

## Group by phase
## We use the row names as time coordinates (roman numerals)
dates <- as.numeric(utils::as.roman(rownames(counts)))

## Frequency Increment Test
freq <- fit(counts, dates)

## Plot time vs abundance and highlight selection
plot(freq)
plot(freq, roll = TRUE, window = 5)
```

`plot_mcd` *MCD Plot*

Description

MCD Plot

Usage

```
## S4 method for signature 'MeanDate'
autoplot(object, ..., select = NULL, decreasing = TRUE)

## S4 method for signature 'MeanDate,missing'
plot(x, select = NULL, decreasing = TRUE, ...)

## S4 method for signature 'SimulationMeanDate'
autoplot(object, ..., select = NULL, decreasing = TRUE)

## S4 method for signature 'SimulationMeanDate,missing'
plot(x, select = NULL, decreasing = TRUE, ...)
```

Arguments

`object, x`	A MeanDate object.
`...`	Currently not used.
`select`	A `numeric` or `character` vector giving the selection of the assemblage that are drawn.
`decreasing`	A `logical` scalar: should the sort be increasing or decreasing?

Value

- `autoplot()` returns a `ggplot` object.
- `plot()` is called it for its side-effects: it results in a graphic being displayed (invisibly returns `x`).

Author(s)

N. Frerebeau

See Also

`mcd()`

Other plotting methods: `plot_aoristic`, `plot_event`, `plot_fit`, `plot_time()`

Examples

```
## Mean Ceramic Date
## Coerce the zuni dataset to an abundance (count) matrix
data("zuni", package = "folio")

## Set the start and end dates for each ceramic type
dates <- list(
  LINO = c(600, 875), KIAT = c(850, 950), RED = c(900, 1050),
  GALL = c(1025, 1125), ESC = c(1050, 1150), PUBW = c(1050, 1150),
  RES = c(1000, 1200), TULA = c(1175, 1300), PINE = c(1275, 1350),
  PUBR = c(1000, 1200), WING = c(1100, 1200), WIPO = c(1125, 1225),
  SJ = c(1200, 1300), LSJ = c(1250, 1300), SPR = c(1250, 1300),
  PINER = c(1275, 1325), HESH = c(1275, 1450), KWAK = c(1275, 1450)
)

## Calculate date midpoints
mid <- vapply(X = dates, FUN = mean, FUN.VALUE = numeric(1))

## Calculate MCD
mc_dates <- mcd(zuni[100:125, ], dates = mid)
head(mc_dates)

## Plot
plot(mc_dates)


## Bootstrap resampling
boot <- bootstrap(mc_dates, n = 30)
head(boot)

## Jackknife resampling
jack <- jackknife(mc_dates)
head(jack)

## Simulation
sim <- simulate(mc_dates, n = 30, interval = "percentiles")
```

```
plot(sim)
```

`plot_time` *Abundance vs Time Plot*

Description

Produces an abundance *vs* time diagram.

Usage

```
plot_time(object, dates, ...)

## S4 method for signature 'data.frame,numeric'
plot_time(object, dates, facet = FALSE)

## S4 method for signature 'matrix,numeric'
plot_time(object, dates, facet = FALSE)
```

Arguments

object	An $m \times p$ numeric matrix or a data.frame of count data (absolute frequencies).
dates	A numeric vector of dates.
...	Currently not used.
facet	A logical scalar: should a matrix of panels defined by type/taxon be drawn?

Value

A ggplot object.

Author(s)

N. Frerebeau

See Also

Other plotting methods: plot_aoristic, plot_event, plot_fit, plot_mcd

Examples

```
data("merzbach", package = "folio")

## Coerce the merzbach dataset to a count matrix
## Keep only decoration types that have a maximum frequency of at least 50
keep <- apply(X = merzbach, MARGIN = 2, FUN = function(x) max(x) >= 50)
counts <- merzbach[, keep]

## Set dates
## We use the row names as time coordinates (roman numerals)
dates <- as.numeric(utils::as.roman(rownames(counts)))

## Plot abundance vs time
plot_time(counts, dates)
plot_time(counts, dates, facet = TRUE)
```

RateOfChange-class *Rate of Change*

Description

An S4 class to represent rates of change from an aoristic analysis.

Slots

`replicates` A non-negative `integer` giving the number of replications.

`breaks` A `numeric` vector giving the date break between time-blocks.

`groups` A `character` vector.

Note

This class inherits from base `array`.

Author(s)

N. Frerebeau

See Also

Other classes: `AoristicSum-class`, `CountApportion-class`, `EventDate-class`, `IncrementTest-class`, `MeanDate-class`, `PermutationOrder-class`, `RefineCA-class`

`RefineCA-class` *Partial Bootstrap CA*

Description

An S4 class to store partial bootstrap correspondence analysis results.

Usage

```
## S4 method for signature 'RefineCA,ANY,missing'
x[[i]]
```

Arguments

`x`	An object from which to extract element(s) or in which to replace element(s).
`i`	A `character` string specifying elements to extract.

Functions

- `[[,RefineCA,ANY,missing-method`: Extracts information from a slot selected by subscript i. i is a length-one character vector. Returns the corresponding slot values.

Slots

`hull` A three columns `numeric` matrix giving the vertices coordinates (`x`, `y`) of the convex hull and a identifier (`id`) to link each row to a variable.

`length` A `numeric` vector giving the convex hull maximum dimension length.

`keep` An `integer` vector giving the subscript of the variables to be kept.

`cutoff` A length-one `numeric` vector giving the cutoff value for samples selection.

Author(s)

N. Frerebeau

See Also

`dimensio::CA`

Other classes: `AoristicSum-class`, `CountApportion-class`, `EventDate-class`, `IncrementTest-class`, `MeanDate-class`, `PermutationOrder-class`, `RateOfChange-class` resample *Resampling Methods*

Description

- `bootstrap()` generate bootstrap estimations of a statistic.
- `jackknife()` generate jackknife estimations of a statistic.

Usage

```
bootstrap(object, ...)

jackknife(object, ...)

## S4 method for signature 'MeanDate'
bootstrap(object, n = 1000, f = NULL)

## S4 method for signature 'MeanDate'
jackknife(object)

## S4 method for signature 'MeanDate'
simulate(
  object,
  n = 1000,
  interval = c("student", "normal", "percentiles"),
  level = 0.8
)
```

Arguments

`object`	An object.
`...`	Currently not used
`n`	A non-negative `integer` specifying the number of bootstrap replications.
`f`	A `function` that takes a single numeric vector (the result of the resampling procedure) as argument.
`interval`	A `character` string giving the type of confidence interval to be returned. It must be one `"student"` (the default), `"normal"` or `"percentiles"`. Any unambiguous substring can be given.
`level`	A length-one `numeric` vector giving the confidence level.

Value

If `f` is `NULL`, `bootstrap()` and `jackknife()` return a `data.frame` with the following elements (else, returns the result of `f` applied to the `n` resampled values) :

original The observed value.

mean The bootstrap/jackknife estimate of mean.

bias The bootstrap/jackknife estimate of bias.

error The boostrap/jackknife estimate of standard erro.

Author(s)

N. Frerebeau

`roc` *Rate of Change*

Description

Computes the rate of change from an aoristic analysis.

Usage

```
roc(object, ...)

## S4 method for signature 'AoristicSum'
roc(object, n = 100)
```

Arguments

`object`	An `AoristicSum` object.
`...`	Currently not used.
`n`	A non-negative `integer` giving the number of replications (see details).

Value

A RateOfChange object.

Author(s)

N. Frerebeau

References

Baxter, M. J. & Cool, H. E. M. (2016). Reinventing the Wheel? Modelling Temporal Uncertainty with Applications to Brooch Distributions in Roman Britain. *Journal of Archaeological Science*, 66: 120-27. doi:10.1016/j.jas.2015.12.007.

Crema, E. R. (2012). Modelling Temporal Uncertainty in Archaeological Analysis. *Journal of Archaeological Method and Theory*, 19(3): 440-61. doi:10.1007/s1081601191223.

See Also

aoristic(), plot()

Other chronological analysis: aoristic(), apportion(), fit()

Examples

```
## Aoristic Analysis
data("zuni", package = "folio")

## Set the start and end dates for each ceramic type
dates <- list(
  LINO = c(600, 875), KIAT = c(850, 950), RED = c(900, 1050),
  GALL = c(1025, 1125), ESC = c(1050, 1150), PUBW = c(1050, 1150),
  RES = c(1000, 1200), TULA = c(1175, 1300), PINE = c(1275, 1350),
  PUBR = c(1000, 1200), WING = c(1100, 1200), WIPO = c(1125, 1225),
  SJ = c(1200, 1300), LSJ = c(1250, 1300), SPR = c(1250, 1300),
  PINER = c(1275, 1325), HESH = c(1275, 1450), KWAK = c(1275, 1450)
)

## Keep only assemblages that have a sample size of at least 10
keep <- apply(X = zuni, MARGIN = 1, FUN = function(x) sum(x) >= 10)

## Calculate date ranges for each assemblage
span <- apply(
  X = zuni[keep, ],
  FUN = function(x, dates) {
    z <- range(unlist(dates[x > 0]))
    names(z) <- c("from", "to")
    z
  },
  MARGIN = 1,
  dates = dates
)

## Coerce to data.frame
span <- as.data.frame(t(span))

## Calculate aoristic sum (normal)
aorist_raw <- aoristic(span, step = 50, weight = FALSE)
plot(aorist_raw)

## Calculate aoristic sum (weights)
aorist_weighted <- aoristic(span, step = 50, weight = TRUE)
plot(aorist_weighted)

## Calculate aoristic sum (weights) by group
groups <- rep(c("A", "B", "C"), times = c(50, 90, 139))
aorist_groups <- aoristic(span, step = 50, weight = TRUE, groups = groups)
plot(aorist_groups)

## Rate of change
roc_weighted <- roc(aorist_weighted, n = 30)
```

```
plot(roc_weighted)

## Rate of change by group
roc_groups <- roc(aorist_groups, n = 30)
plot(roc_groups)
```

seriation *Matrix Seriation*

Description

- `seriate_*()` computes a permutation order for rows and/or columns.
- `permute()` rearranges a data matrix according to a permutation order.
- `get_order()` returns the seriation order for rows and columns.

Usage

```
seriate_average(object, ...)

seriate_rank(object, ...)

refine(object, ...)

permute(object, order, ...)

get_order(x, ...)

## S4 method for signature 'data.frame'
seriate_average(object, margin = c(1, 2), axes = 1, ...)

## S4 method for signature 'matrix'
seriate_average(object, margin = c(1, 2), axes = 1, ...)

## S4 method for signature 'data.frame,PermutationOrder'
permute(object, order)

## S4 method for signature 'matrix,PermutationOrder'
permute(object, order)

## S4 method for signature 'PermutationOrder'
get_order(x, margin = c(1, 2))

## S4 method for signature 'data.frame'
seriate_rank(object, EPPM = FALSE, margin = c(1, 2), stop = 100)

## S4 method for signature 'matrix'
seriate_rank(object, EPPM = FALSE, margin = c(1, 2), stop = 100)
```

```
## S4 method for signature 'AveragePermutationOrder'
refine(object, cutoff, margin = c(1, 2), axes = c(1, 2), n = 30)

## S4 method for signature 'BootstrapCA'
refine(object, cutoff, margin = 1, axes = c(1, 2))
```

Arguments

object, x	An $m \times p$ numeric `matrix` or a `data.frame` of count data (absolute frequencies).
...	Further arguments to be passed to internal methods.
order	A PermutationOrder object giving the permutation order for rows and columns.
margin	A `numeric` vector giving the subscripts which the rearrangement will be applied over: `1` indicates rows, `2` indicates columns, `c(1, 2)` indicates rows then columns, `c(2, 1)` indicates columns then rows.
axes	An `integer` vector giving the subscripts of the CA axes to be used.
EPPM	A `logical` scalar: should the seriation be computed on EPPM instead of raw data?
stop	An `integer` giving the stopping rule (i.e. maximum number of iterations) to avoid infinite loop.
cutoff	A function that takes a numeric vector as argument and returns a single numeric value (see below).
n	A non-negative `integer` giving the number of bootstrap replications.

Value

- `seriate_*()` returns a PermutationOrder object.
- `permute()` returns either a permuted `matrix` or a permuted `data.frame` (the same as `object`).
- `bootstrap()` returns a RefineCA object.

Seriation

The matrix seriation problem in archaeology is based on three conditions and two assumptions, which Dunell (1970) summarizes as follows.

The homogeneity conditions state that all the groups included in a seriation must:

1. Be of comparable duration.
2. Belong to the same cultural tradition.
3. Come from the same local area.

The mathematical assumptions state that the distribution of any historical or temporal class:

1. Is continuous through time.
2. Exhibits the form of a unimodal curve.

Theses assumptions create a distributional model and ordering is accomplished by arranging the matrix so that the class distributions approximate the required pattern. The resulting order is inferred to be chronological.

The following seriation methods are available:

`seriate_average()` Correspondence analysis-based seriation (average ranking). Correspondence analysis (CA) is an effective method for the seriation of archaeological assemblages. The order of the rows and columns is given by the coordinates along one dimension of the CA space, assumed to account for temporal variation. The direction of temporal change within the correspondence analysis space is arbitrary: additional information is needed to determine the actual order in time.

`seriate_rank()` Reciprocal ranking seriation. These procedures iteratively rearrange rows and/or columns according to their weighted rank in the data matrix until convergence. Note that this procedure could enter into an infinite loop. If no convergence is reached before the maximum number of iterations, it stops with a warning.

Correspondence Analysis

`bootstrap()` allows to identify samples that are subject to sampling error or samples that have underlying structural relationships and might be influencing the ordering along the CA space.

This relies on a partial bootstrap approach to CA-based seriation where each sample is replicated `n` times. The maximum dimension length of the convex hull around the sample point cloud allows to remove samples for a given `cutoff` value.

According to Peebles and Schachner (2012), "[this] point removal procedure [results in] a reduced dataset where the position of individuals within the CA are highly stable and which produces an ordering consistent with the assumptions of frequency seriation."

Author(s)

N. Frerebeau

References

Desachy, B. (2004). Le sériographe EPPM: un outil informatisé de sériation graphique pour tableaux de comptages. *Revue archéologique de Picardie*, 3(1), 39-56. doi:10.3406/pica.2004.2396.

Dunnell, R. C. (1970). Seriation Method and Its Evaluation. *American Antiquity*, 35(03), 305-319. doi:10.2307/278341.

Ihm, P. (2005). A Contribution to the History of Seriation in Archaeology. In C. Weihs & W. Gaul (Eds.), *Classification: The Ubiquitous Challenge*. Berlin Heidelberg: Springer, p. 307-316. doi:10.1007/3540280847_34.

Peeples, M. A., & Schachner, G. (2012). Refining correspondence analysis-based ceramic seriation of regional data sets. *Journal of Archaeological Science*, 39(8), 2818-2827. doi:10.1016/j.jas.2012.04.040.

See Also

`dimensio::ca()`

Examples

```
## Replicates Desachy 2004 results
## Coerce dataset to abundance matrix
data("compiegne", package = "folio")

## Get seriation order for columns on EPPM using the reciprocal averaging method
## Expected column order: N, A, C, K, P, L, B, E, I, M, D, G, O, J, F, H
(indices <- seriate_rank(compiegne, EPPM = TRUE, margin = 2))

## Get permutation order
get_order(indices, 1) # rows
get_order(indices, 2) # columns

## Permute columns
(new <- permute(compiegne, indices))

## See the vignette
## Not run:
utils::vignette("seriation")

## End(Not run)
```

Index