Package ‘datplot’

March 4, 2021

Type Package

Title Preparation of Object Dating Ranges for Density Plots (Aoristic Analysis)

Version 1.0.0

Maintainer Lisa Steinmann <lisa.steinmann@rub.de>

Description Converting date ranges into dating 'steps' eases the visualization of changes in e.g. pottery consumption, style and other variables over time. This package provides tools to process and prepare data for visualization and employs the concept of aoristic analysis.

License CC BY-SA 4.0

URL https://github.com/lsteinmann/datplot

BugReports https://github.com/lsteinmann/datplot/issues

Depends R (>= 3.3)

Suggests covr, devtools, dplyr, forcats, ggplot2, ggridges, knitr, readxl, reshape2, rmarkdown, stringr, testthat

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation yes

Author Lisa Steinmann [aut, cre] (<https://orcid.org/0000-0002-2215-1243>), Barbora Weissova [ctb] (<https://orcid.org/0000-0002-3297-6855>)

Repository CRAN

Date/Publication 2021-03-04 10:00:05 UTC

R topics documented:

`Beazley` *Beazley (sample of 1000)*

Description

A test dataset containing a data.frame how it should ideally be arranged to work with datplot.Data are gathered from the Beazley Archive Pottery Database (BAPD) – https://www.beazley.ox.ac.uk/pottery/default.htm and transformed to work with datplot

Usage

```
data(Beazley)
```

Format

A data frame with 1000 rows and 4 variables

Details

- Identifier (Vase.Number in BAPD)
- Technique: Sample contains only red- or blackfigured objects
- DAT_min. Integer: lower range of the dating, BCE in negative numbers
- DAT_max. Integer: upper range of the dating, BCE in negative numbers

Source

https://www.beazley.ox.ac.uk/pottery/default.htm

`calculate.outputrows` *Calculate output rows (internal)*

Description

an approximation(!) of the rows that will be needed to fit all the steps of the dating

Usage

```
calculate.outputrows(DAT_mat, stepsize)
```

Arguments

`DAT_mat`	a matrix as transformed by datsteps()
`stepsize`	the stepsize given to or by datsteps()

Value

the number of rows create.sub.objects should at least produce in order to fit all steps

`check.number` *Check for numbers (internal)*

Description

Checks if value is either numeric, integer or double and and returns TRUE.

Usage

```
check.number(value)
```

Arguments

`value`	A value to check

Value

TRUE if value is any kind of number, FALSE if value is not

`check.structure` *Check the Structure to be compatible with datsteps() (internal)*

Description

Checks if the object passed to datsteps() will work

Usage

```
check.structure(DAT_df)
```

Arguments

`DAT_df`	An object to check

Value

TRUE if object can be processed by datsteps(), error / FALSE if not

`create.sub.objects` *Create sub-objects for each object in a dataframe (internal)*

Description

Requires a matrix with 4 named columns as datsteps will hand to the function: * "index" (identifier so the values can later be reassigned to their ID and variable), * "datmin" (minimum dating as any kind of number), * "datmax" (maximum dating as any kind of number), * "weight" (as created by get.weights), * "step" (empty). It's expected that dates BCE are displayed as negative values while dates CE are positive values. Ignoring this will cause problems in any case, that would be fixed automatically by switch.dating().

Usage

```
create.sub.objects(DAT_mat, stepsize)
```

Arguments

`DAT_mat`	a matrix with 3 variables as prepared by datsteps()
`stepsize`	Number of years that should be used as an interval for creating dating steps.

Value

a longer matrix of the same structure to be further processed by datsteps() with a number of steps for each object `datsteps` *Create 'steps' of dates for each object in a dataframe*

Description

Requires a dataframe with 4 variables: ID (ideally factor), group (ideally factor), minimum date (int/numeric) and maximum date (int/numeric). It's expected that dates BCE are displayed as negative values while dates CE are positive values. Ignoring this will cause problems in any case.

Usage

```
datsteps(DAT_df, stepsize = 25)
```

Arguments

`DAT_df`	a dataframe with 4 variables: ID, group, minimum date (int/num) maximum date (int/num), _must_ be in this order, colnames are irrelevant; each object _must_ be one row.
`stepsize`	defaults to 5. Number of years that should be used as an interval for creating dating steps.

Value

a larger dataframe with a number of steps for each object as well as a 'weight' value, that is a quantification of how well the object is dated (lesser value means object is dated to larger timespans, i.e. with less confidence)

Examples

```
DAT_df_steps <- datsteps(DAT_df[1:100, ], stepsize = 25)
plot(density(DAT_df_steps$DAT_step))
```

`DAT_df` *datplot Testing data*

Description

A test dataset containing a data.frame how it should ideally be arranged to work with datplot. Data are not real and illustrate some common problems such as lower and upper dating in the wrong columns.

Usage

```
data(DAT_df)
```

Format

A data frame with 5000 rows and 4 variables

Details

- ID. Identifier of the Objects (has to be unique)
- var. Grouping variable, such as a Type or a Findspot
- DAT_min. Integer: lower range of the dating, BCE in negative numbers
- DAT_max. Integer: upper range of the dating, BCE in negative numbers

`generate.stepsize` *Determine stepsize (internal)*

Description

Determines stepsize by selecting the absolute minimum value between the upper and lower end of all dating ranges.

Usage

```
generate.stepsize(DAT_mat)
```

Arguments

`DAT_mat`	a matrix as prepared by datsteps(), resp. a matrix witch columns "datmin" and "datmax" containing numeric/integer value of the dating ranges.

Value

stepsize

`get.histogramscale` *Scaling Factor for Combined Histogramm Plots*

Description

Requires a dataframe as produced by datsteps() or a number as DAT_df_steps. Calculated the value with which the y-axis of a density graph should be multiplied in order to be visible in the corresponding histogram.

Usage

```
get.histogramscale(DAT_df_steps, binwidth = "stepsize")
```

Arguments

`DAT_df_steps`	a dataframe as returned by datsteps (works also with a single number and a vector)
`binwidth`	the bandwidth to use for the density function and histogram. Should be stepsize used to create the dataframe. If a df as returned by datsteps() is given, stepsize can be automatically assigned using the corresponding attribute (binwidth = "stepsize")

Value

the value with which to scale the density curve to a histogram plot so that both will be visible

Examples

```
DAT_df_steps <- datsteps(DAT_df[1:100, ], stepsize = 25)
get.histogramscale(DAT_df_steps)

get.histogramscale(DAT_df_steps$DAT_step, binwidth = 20)
get.histogramscale(500, binwidth = 20)
```

`get.step.sequence` *Calculate the sequence of dating steps (internal)*

Description

Produces an appropriate sequence of years between the minimum and maximum dating. If they cannot be properly divided by the stepsize set beforehand, either three values are generated for objects that are dated to a range of more then 60 objects dated to a timespan of less or equal to 60 If they can be divided without residual, the normal sequence is returned. If there is a residual, the stepsize is modified depending on how large the residual is. (TODO: There is still a problem here that needs fixing.)

Usage

```
get.step.sequence(datmin = 0, datmax = 100, stepsize = 25)
```

Arguments

`datmin`	value of the minimum dating of one object
`datmax`	value of the maximum dating of one object
`stepsize`	the stepsize to be used

Value

sequence of steps to be created by create.sub.objects()

`get.weights` *Calculate the weights for each dated object (internal)*

Description

Calculates the weights from two vectors of minimum and maximum dating for each object. Returns a dataframe with the weight in the first column and FALSE in the second if two rows have the same value in both min and max dating.

Usage

```
get.weights(DAT_min, DAT_max)
```

Arguments

`DAT_min`	a vector containing the minimum date (a number) of each object
`DAT_max`	a vector containing the maximum date (a number) of each object

Value

the 'weight' value for the datsteps-dataframe, that is a quantification of how well the object is dated (lesser value means object is dated to larger timespans, i.e. with less confidence)

`Inscr_Bithynia` *Inscr_Bithynia*

Description

The data set was gathered by Barbora Weissova and published as part of her dissertation "Regional Economy, Settlement Patterns and the Road System in Bithynia (4th Century BC - 6th Century AD). Spatial and Quantitative Analysis.".

Usage

```
Inscr_Bithynia
```

Format

A data frame with 2878 rows and 9 variables:

- `ID` character COLUMN_DESCRIPTION
- `ikey` character ID at `https://inscriptions.packhum.org/` / `https://edh-www.adw.uni-heidelberg.de/home`, if available
- `Location` factor Findspot of the Inscription (City)
- `Source` character Corpus/Citation of the Inscription

`Dating` character Original Chronological Assessment, may contain inconsistencies

`Language` factor Language of the Inscription, can either be Latin, Greek, or both

`uncertain_dating` logical TRUE if Dating is not certain, FALSE if dating is certain

`DAT_min` integer lower border of the dating timespan, negative values for BCE, positive values for CE

`DAT_max` integer upper border of the dating timespan, negative values for BCE, positive values for CE

`URL` Link to the inscription (if available) at https://inscriptions.packhum.org/ or https://edh-www.adw.uni-heidelberg.de/home

Source

Weissova, Barbora. 2019. "Regional Economy, Settlement Patterns and the Road System in Bithynia (4th Century BC - 6th Century AD). Spatial and Quantitative Analysis." Dissertation, Berlin: Freie Universität Berlin. https://refubium.fu-berlin.de/handle/fub188/23730, partially after https://inscriptions.packhum.org/

`scaleweight` *Scales the content of a column according to group membership*

Description

Requires a dataframe with one variable and one value column.

Usage

```
scaleweight(DAT_df, var = c("all", 2), val = 5)
```

Arguments

`DAT_df`	a dataframe
`var`	the index of the column that should be used as the group variable, OR "all" (note: all non-numeric values will result in the weight being scaled across all objects)
`val`	the column that should be scaled (value / sum(values))

Value

the same dataframe, with scaled values in the specifies column

switch.dating *Switch values where dating is in wrong order (internal)*

Description

Requires a dataframe with 4 variables: ID (ideally factor), group (ideally factor), minimum date (int/numeric) and maximum date (int/numeric) and DAT_err as a vector of indices where dating is in wrong order.

Usage

```
switch.dating(DAT_df, DAT_err)
```

Arguments

`DAT_df`	a dataframe with 4 variable: ID, group, minimum date (int/num) maximum date (int/num)
`DAT_err`	a vector containing the indices of the dates which are in wrong order

Value

corrected DAT_mat

Index