# Package: datplot (via r-universe)

August 25, 2024

**Type** Package

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

**Version** 1.1.1

**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** GPL (>= 3)

**URL** https://github.com/lsteinmann/datplot, https://lsteinmann.github.io/datplot/

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

**Depends** R (>= 3.3)

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

**VignetteBuilder** knitr

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.2.3

**Repository** https://lsteinmann.r-universe.dev

**RemoteUrl** https://github.com/lsteinmann/datplot

**RemoteRef** HEAD

**RemoteSha** 08d8ff088e17e1f3b4840c021ddf56e6e12450b1

# Contents



---

`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.carc.ox.ac.uk/carc/pottery 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.carc.ox.ac.uk/carc/pottery

`datsteps` *Create 'steps' of dates for each object in a data.frame*

## Description

This function transforms a data.frame of dated objects with associated data to a new data.frame which contains a row for each dating 'step' for each objects. Dating 'steps' can be single years (with 'stepsize = 1') or an arbitrary number that will be used as a guideline for the interval. It's expected that dates BCE are displayed as negative values while dates CE are positive values. Ignoring this will cause problems. If dates are provided in the wrong order in any number of rows they will automatically be switched.

The function along with a guide on how to use it and a case study is published in [Steinmann – Weissova 2021](https://doi.org/10.1017/aap.2021.8).

## Usage

```
datsteps(
  DAT_df,
  stepsize = 1,
  calc = "weight",
  cumulative = FALSE,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `DAT_df` | a data.frame with 4 variables: * 'ID' : An identifier for each row, e.g. an Inventory number (ideally character). * 'group' : A grouping variable, such as type or context (ideally factor). * 'DAT_min' : minimum dating (int/num), the minimum dating boundary for a single object, i.e. the earliest year the object may be dated to. * 'DAT_min' : maximum dating (int/num), the maximum dating boundary for a single object, i.e. the latest year the object may be dated to. The columns _must_ be in this order, column names are irrelevant; each row _must_ correspond to one datable entity / object. |
| `stepsize` | numeric, default is 1. Number of years that should be used as an interval for creating dating steps. |
| `calc` | method of calculation to use; can be either one of "weight" (default) or "probability": * "weight": use the [published original calculation](https://doi.org/10.1017/aap.2021.8) for weights, * "probability": calculate year-wise probability instead (only reasonable when 'stepsize = 1') |
| `cumulative` | FALSE (default), TRUE: add a column containing the cumulative probability for each object (only reasonable when 'stepsize = 1', and will automatically use probability calculation) |
| `verbose` | TRUE / FALSE: Should the function issue additional messages pointing to possible inconsistencies and notify of methods? |

## Value

an expanded data.frame in with each row represents a dating 'step'. Added columns contain the value of each step, the 'weight' or 'probability'- value for each step, and (if chosen) the cumulative probability.

## Examples

```
data("Inscr_Bithynia")
DAT_df <- Inscr_Bithynia[, c("ID", "Location", "DAT_min", "DAT_max")]
DAT_df_steps <- datsteps(DAT_df, 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

## get.histogramscale *Scaling Factor for Combined Histogram Plots*

### Description

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

### Usage

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

### Arguments

| | |
|---|---|
| DAT_df_steps | a data.frame as returned by [datsteps()]. (Will also work with a single number and a vector.) |
| binwidth | the bandwidth to use for the density function and histogram. Should equal the stepsize used to create the data.frame. If a data.frame 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

```
data("Inscr_Bithynia")
DAT_df <- Inscr_Bithynia[, c("ID", "Location", "DAT_min", "DAT_max")]
DAT_df_steps <- datsteps(DAT_df, stepsize = 25)
get.histogramscale(DAT_df_steps)

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

## get.probability *Calculate the probability for each year and each dated object*

### Description

Calculates the probability of each object being dated into each year / timeslot from two vectors of minimum and maximum dating. Returns a vector of probabilities.

### Usage

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

### Arguments

| | |
|---|---|
| DAT_min | a numeric vector containing the minimum date of each object |
| DAT_max | a numeric vector containing the maximum date of each object |

### Value

a vector of probabilities for each object being dated to any single year within the timespan (lesser value means object is dated to larger timespans, i.e. with less confidence).

### See Also

[datsteps()], [get.weights()]

---

## get.step.sequence *Calculate the sequence of dating steps*

---

### 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.

### Usage

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

### Arguments

| | |
|---|---|
| datmin | numeric value of the minimum dating of one object |
| datmax | numeric 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()]

### See Also

[datsteps()], [create.sub.objects()]

## Examples

```
min_year <- -494
max_year <- -334
sequence <- get.step.sequence(datmin = min_year, datmax = max_year, stepsize = 25)
sequence

min_year <- 1
max_year <- 100
sequence <- get.step.sequence(datmin = min_year, datmax = max_year, stepsize = 25)
sequence
```

---

# get.weights *Calculate the weights for each dated object*

---

## 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. See [publication](https://doi.org/10.1017/aap.2021.8) for information about how this is calculated.

## Usage

```
get.weights(DAT_min, DAT_max, verbose = FALSE)
```

## Arguments

| | |
|---|---|
| DAT_min | a numeric vector containing the minimum date of each object |
| DAT_max | a numeric vector containing the maximum date of each object |
| verbose | TRUE / FALSE: Should the function issue additional messages pointing to possible inconsistencies and notify of methods? |

## Value

a vector of 'weight'-values for the datsteps-data.frame, 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)

## See Also

[datsteps()], [get.probability()]

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*

## Description

Requires a data.frame with one variable and one value column.

## Usage

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

## Arguments

| | |
|---|---|
| DAT_df | a data.frame |
| var | index or name of the column that should be used as the group variable, OR "all" |
| val | index or name of the column that should be scaled (has to be numeric) |

## Value

the same data.frame, with the scaled values in the specified column

## Examples

```
data("Inscr_Bithynia")
DAT_df <- Inscr_Bithynia[, c("ID", "Location", "DAT_min", "DAT_max")]
DAT_df_steps <- datsteps(DAT_df, stepsize = 25)
DAT_df_scaled <- scaleweight(DAT_df_steps, var = 2, val = 5)
```

# Index