Package ‘tagcloud’

July 3, 2015

Type Package

Title Tag Clouds

Version 0.6

Date 2015-07-02

Author January Weiner

Maintainer January Weiner <january.weiner@gmail.com>

Description Generating Tag and Word Clouds.

License GPL (>= 2)

LazyLoad yes

Depends Rcpp (>= 0.9.4)

Imports RColorBrewer

Suggests extrafont,knitr

VignetteBuilder knitr

LinkingTo Rcpp

URL http://logfc.wordpress.com

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-07-03 11:17:02

R topics documented:

`editor.tagcloud` *Simple interactive editing of tag clouds*

Description

A minimalistic editor for object of the tagcloud class.

Usage

```
editor.tagcloud(boxes)
```

Arguments

`boxes`	An object of the tagcloud class, returned by the `tagcloud` function.

Details

`tagcloud` provides a minimalistic editor for tag clouds produced by `tagcloud` function. After `editor.tagcloud` is called, the tag cloud is plotted. First click selects the tag to be moved. The second click sends the tag such that its left lower corner is at the position indicated by the mouse. Right-clicking terminates the program.

Value

An object of the `tagcloud class` with the modified positions of the tags.

Author(s)

January Weiner <january.weiner@gmail.com>

See Also

`tagcloud`

Examples

```
## Not run:
data( gambia )
terms <- gambia$Term
tagcloud( terms )
boxes <- editor.tagcloud( boxes )


## End(Not run)
```

gambia *Results of GO enrichment analysis in TB*

Description

A data.frame object containing the results of a GO enrichment analysis from the GOstats package.

Format

A data frame with 318 observations on the following 9 variables.

GOBPID Pvalue OddsRatio ExpCount Count Size Term

GOBPID Gene Ontology (GO) biological process (BP) identifier

Pvalue P value from enrichment test

OddsRatio Measure of enrichment

ExpCount expected number of genes in the enriched partition which map to this GO term

Count number of genes in the enriched partition which map to this GO term

Size number of genes within this GO Term

Term Gene Ontology term description

Details

The data results from a microarray analysis of the whole blood transcriptome of tuberculosis (TB) patients compared to healthy individuals. Genes were sorted by their p-value and analysed using the GOstats package.

Significantly enriched GO terms are included in this data frame.

Source

Maertzdorf J., Ota M., Repsilber D., Mollenkopf H.J., Weiner J., et al. (2011) Functional Correlations of Pathogenesis-Driven Gene Expression Signatures in Tuberculosis. PLoS ONE 6(10): e26938. doi:10.1371/journal.pone.0026938

Examples

```
data(gambia)
tagcloud( gambia$Term, -log( gambia$Pvalue ) )
```

`plot.tagcloud` *Tag and Word Clouds*

Description

Functions to create and display plots called tag clouds, word clouds or weighted lists, in which a usually large number of words is displayed in size correlated with a numerical value (for example, frequency in a text or enrichment of a GO term). This makes it easier to visualize the prominence of certain tags or words. Also, it looks nice.

Usage

```
## S3 method for class 'tagcloud'
plot(x, family = NULL, add = FALSE, with.box = FALSE,
  col = NULL, sel = NULL, ...)

## S3 method for class 'tagcloud'
summary(object, ...)

tagcloud(tags, weights = 1, algorithm = "oval", scale = "auto",
  scale.multiplier = 1, order = "size", sel = NULL, wmin = NULL,
  wmax = NULL, floor = 1, ceiling = 3, family = NULL, col = NULL,
  fvert = 0, plot = TRUE, add = FALSE)
```

Arguments

`x,object`	An object of the type produced by tagcloud.
`family`	Font family to use, a vector containing font families to use for each tag. For the `tagcloud` function, the special keyword "random" can be used to assign random families (requires the extrafont package).
`add`	If TRUE, the tags will be added to the current plot instead of creating a new plot.
`with.box`	If TRUE, a rectangle will be plotted around each tag.
`col`	Color or a vector containing colors to use for drawing the tags.
`sel`	An integer or boolean vector indicating which terms from the provided list will be used to plot. The vectors `col` and `weights` will be filtered accordingly.
`tags`	A character vector containing words or tags to be shown on the plot.
`weights`	A numeric vector giving the relative proportions of text size corresponding to the given tag.
`algorithm`	Name of the algorithm to use. Can be "oval", "fill", "random", "snake", "list" or "clist". See Details.
`scale`	If "auto", text expansion will be calculated automatically to fit the available space. Otherwise, a numeric value used to modify the calculated text sizes; tune scale to achieve a better fit.

`scale.multiplier`	Multiplier for the final calculated text expansion parameter. Increase if there is too much empty space around the tag cloud; decrease if the tags go over the plot boundaries.
`order`	Determines in which order the tags will be drawn. Can be "size", "keep", "random", "height" or "width". See Details.
`wmin`	All items in the `weights` vector smaller than `wmin` will be changed to `wmin`
`wmax`	All items in the `weights` vector larger than `wmax` will be changed to `wmax`
`floor`	Minimal text size. See Details.
`ceiling`	Maximal text size. See Details.
`fvert`	Fraction of tags which will be rotated by 90 degrees counterclockwise.
`plot`	If FALSE, no plot will be produced.
`...`	Further arguments to be passed to downstream methods.

Details

The package `tagcloud` creates and plots tag clouds (word clouds). The algorithms in the package have been designed specifically with long tags (such as GO Term descriptions) in mind.

Term ordering: The available arguments are as follows:

- size – tags are ordered by size, that is, their effective width multiplied by their effective height. Default.
- keep – keep the order from the list of words provided
- random – randomize the tag list
- width – order by effective screen width
- height – order by effective screen height

By default, prior to plotting terms are ordered by size.

Algorithms: There are four algorithms for placing tags on the plot implemented in tagcloud.

- oval – creates an oval cloud.
- fill – an attempt will be made to fill the available space
- random – randomly distribute tags over the available space. This algorithm is slow and not very effective
- snake – tags are placed clockwise around the first tag to plot
- list – create a list, one tag directly beneath another, justified left
- clist – create a list, one tag directly beneath another, centered

Algorithms `oval`, `fill` and `random` attempt to fill the available space by adjusting the scaling factor for the font sizes.

Calculation of tag sizes: Placing tags such that the empty space between the tags is minimized poses a non-trivial problem, because the effective bounding box of a displayed text is not linearly dependent on the `cex` parameter.

In tagcloud, first a `cex` parameter is calculated for each tag separately, based on the parameters `floor`, `ceiling` and the vector of weights. Note that all weights smaller than `wmin` are replaced by wmin and all weights larger than wmax are replaced by wmax. Then, effective heights and widths of the tags to be displayed are calculated using the strwidth and strheight functions.

Unless the argument scale is different from "auto", a scaling parameter for cex is automatically calculated based on the current area of the tags and the available plotting area. This usually results in a reasonable plot, but is neither guaranteed to occupy all of the available space without margins, nor that no tag will cross the view port.

Value

tagcloud returns an object of the tagcloud-class, which really is a data frame with the following columns:

- tags – the tags, words or phrases shown on the plot
- weights – a numeric vector that is used to calculate the size of the plotted tags
- family – name of the font family to be used in plotting
- vertical – whether the tag should be rotated by 90 degrees counter-clockwise
- x,y – coordinates of the left lower corner of the tags bounding box
- w,h – width and height of the bounding box
- cex – text expansion factor, see par
- s – surface of the tag (width x height)

The object of the tagcloud class can be manipulated using editor.tagcloud and displayed using plot, print and summary functions.

Note

Care should be taken when using extra fonts loaded by the extrafont package; not all fonts can be easily copied to a PDF file.

Some ideas in this package are based on the ‘wordcloud‘ package by Ian Fellows.

Author(s)

January Weiner <january.weiner@gmail.com>

See Also

editor.tagcloud – interactive editing of tagcloud objects.

strmultline – splitting multi-word sentences into lines for a better cloud display.

smoothPalette – mapping values onto a color gradient.

Examples

```
# a rather boring tag cloud
data( gambia )
terms <- gambia$Term
tagcloud( terms )
```

```
# tag cloud with weights relative to P value
# colors relative to odds ratio, from light
# grey to black
weights <- -log( gambia$Pvalue )
colors  <- smoothPalette( gambia$OddsRatio, max=4 )
tagcloud( terms, weights, col= colors, algorithm= "oval" )

# tag cloud filling the whole plot
tagcloud( terms, weights, col= colors, algorithm= "fill" )

# just a list of only the first ten terms
tagcloud( terms, weights, sel= 1:10,
          col= colors, algorithm= "list", order= "width" )

# oval, with line breaks in terms
terms <- strmultline( gambia$Term )
tagcloud( terms, weights, col= colors, algorithm= "oval" )

## Not run:
# shows available font families, scaled according to
# the total disk space occupied by the fonts
require( extrafont )
ft <- fonttable()
fi <- file.info( fonttable()$fontfile )
families <- unique( ft$FamilyName )
sizes    <- sapply( families,function( x ) sum( fi[ ft$FamilyName == x, "size" ] ) )
tagcloud( families, sizes, family= families )

## End(Not run)
```

`smoothPalette` *Replace a vector of numbers by a gradient of colors*

Description

Replace a vector of numbers by a vector of colors from a palette, such that values correspond to the colors on a smooth gradient.

Usage

```
smoothPalette(x, pal = NULL, max = NULL, min = NULL, n = 9,
  palfunc = NULL, na.color = "white")
```

Arguments

`x`	A numeric vector
`pal`	Character vector containing the color gradient onto which the numeric vector `x` will be mapped. By default, a gradient from white to black is generated. If it is a single character value, it will be treated as name of an RColorBrewer palette (see `brewer.pal`).

`max`	Values of `x` larger than `max` will be replaced by `max`
`min`	Values of `x` smaller than `min` will be replaced by `min`
`n`	Number of steps
`palfunc`	Palette function returned by colorRampPalette
`na.color`	NA values will be replaced by that color

Details

This function is used to map a continues numerical vector on an ordinal character vector, in especially a vector of colors. Color palette can be specified using an RColorBrewer palette name.

Value

A character vector of the same length as the numeric vector `x`, containing the matching colors.

Author(s)

January Weiner <january.weiner@gmail.com>

See Also

`tagcloud`

Examples

```
smoothPalette( 1:3 )
# will print:
# "#CCCCCC" "#666666" "#000000"

smoothPalette( 1:3, pal= "Blues" )
# will produce:
# "#F7FBFF" "#6BAED6" "#08306B"

x <- runif( 100 )
plot( 1:100, x, col= smoothPalette( x, pal= "BrBG" ), pch= 19 )
```

`strmultline` *Replace some spaces in multi-word sentences by newlines*

Description

Replace a space character by a newline in a multi-word sentence to get a better height / width ratio

Usage

```
strmultline(strings, ratio = 0.2)
```

Arguments

`strings`	a character vector containing the multi-word sentences to be split
`ratio`	the desired ratio height / width

Details

Very long tags, for example GO Term descriptions, make a bad tag cloud. `strmultline` tries to chop up such a long sentence into multiple (currently two) lines, to get a better height / width ratio.

Value

A character vector containing the modified sentences.

Author(s)

January Weiner <january.weiner@gmail.com>

See Also

`tagcloud`

Index