Package 'WeightedTreemaps'

December 12, 2024

Title Generate and Plot Voronoi or Sunburst Treemaps from Hierarchical Data

Version 0.1.4

Description Treemaps are a visually appealing graphical representation of numerical data using a space-filling approach. A plane or 'map' is subdivided into smaller areas called cells. The cells in the map are scaled according to an underlying metric which allows to grasp the hierarchical organization and relative importance of many objects at once. This package contains two different implementations of treemaps, Voronoi treemaps and Sunburst treemaps. The Voronoi treemap function subdivides the plot area in polygonal cells according to the highest hierarchical level, then continues to subdivide those parental cells on the next lower hierarchical level, and so on. The Sunburst treemap is a computationally less demanding treemap that does not require iterative refinement, but simply generates circle sectors that are sized according to predefined weights. The Voronoi tesselation is based on functions from Paul Murrell (2012) <https://www.stat.auckland.ac.nz/~paul/Reports/VoronoiTreemap/voronoiTreeMap.html>.

License GPL-3

URL https://github.com/m-jahn/WeightedTreemaps

BugReports https://github.com/m-jahn/WeightedTreemaps/issues

Depends R (>= 3.5.0)

Imports colorspace, dplyr, grid, lattice, methods, Rcpp, scales, sf, sp, tibble

Suggests knitr, parallel, rmarkdown

LinkingTo BH, Rcpp, RcppCGAL

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

SystemRequirements C++17

NeedsCompilation yes

Author Michael Jahn [aut, cre] (<https://orcid.org/0000-0002-3913-153X>),
David Leslie [aut],
Ahmadou Dicko [aut] (<https://orcid.org/0000-0002-9654-7582>),
Dunipace Eric [aut] (<https://orcid.org/0000-0001-8909-213X>),
Paul Murrell [aut, cph] (<https://orcid.org/0000-0002-3224-8858>)

Maintainer Michael Jahn <jahn@mpusp.mpg.de>

Repository CRAN

Date/Publication 2024-12-12 09:00:02 UTC

Contents

`as.data.frame.sunburstResult`
as.data.frame.sunburstResult

Description

Coerces a sunburstResult object to data frame.

Usage

```
## S3 method for class 'sunburstResult'
as.data.frame(x, ..., stringsAsFactors = FALSE)
```

Arguments

x	(sunburstResult) A sunburst treemap results object
...	(none) Not used
stringsAsFactors	(logical) Transform strings to factors, default FALSE

Value

Returns a data.frame

See Also

sunburstTreemap for generating the treemap that is the input for this function

---

as.data.frame.voronoiResult *as.data.frame.voronoiResult*

Description

Coerces a voronoiResult object to data frame, omitting polygon data.

Usage

```
## S3 method for class 'voronoiResult'
as.data.frame(x, ..., stringsAsFactors = FALSE)
```

Arguments

x	(voronoiResult) A voronoi treemap results object
...	(none) Not used
stringsAsFactors	(logical) Transform strings to factors, default FALSE

Value

Returns a data.frame

See Also

voronoiTreemap for generating the treemap that is the input for this function

cropped_voronoi *cropped_voronoi*

Description

Tesselates a plane using a set of XY coordinates

Usage

```
cropped_voronoi(sites)
```

Arguments

sites	(numeric matrix) The only input parameter for the function. A matrix with 3 columns: X and Y coordinates, as well as weights that are used for tesselation.

Details

The function is only intended for internal use. However, one can also use it directly for test purposes.

Value

A list of cell coordinates; one cell for each set of input coordinates.

drawTreemap *drawTreemap*

Description

Draws the treemap object that was obtained by running voronoiTreemap or sunburstTreemap. Many graphical parameters can be customized but some settings that determine the appearance of treemaps are already made during treemap generation. Such parameters are primarily cell size and initial shape of the treemap.

Usage

```
drawTreemap(
  treemap,
  levels = 1:length(treemap@call$levels),
  color_type = "categorical",
  color_level = NULL,
  color_palette = NULL,
  color_steps = 10,
  border_level = levels,
  border_size = 6,
  border_color = grey(0.9),
```

```
  label_level = max(levels),
  label_size = 1,
  label_color = grey(0.9),
  label_autoscale = TRUE,
  title = NULL,
  title_size = 1,
  title_color = grey(0.5),
  legend = FALSE,
  legend_position = "left",
  legend_size = 0.1,
  custom_range = NULL,
  width = 0.9,
  height = 0.9,
  layout = c(1, 1),
  position = c(1, 1),
  add = FALSE
)
```

Arguments

treemap	(treemapResult) Either a voronoiResult or sunburstResult object that contains polygons and metadata as output from running voronoiTreemap or sunburstTreemap.
levels	(numeric) A numeric vector representing the hierarchical levels that are drawn. The default is to draw all levels.
color_type	(character) One of "categorical", "cell_size", "both", or "custom_color". For "categorical", each cell is colored based on the (parent) category it belongs. Colors may repeat if there are many cells. For "cell_size", cells are colored according to their relative area. For "both", cells are be colored the same way as for "categorical", but lightness is adjusted according to cell area. For "custom_color", a color index is used that was specified by custom_color during treemap generation. Use NULL to omit drawing colors.
color_level	(numeric) A numeric vector representing the hierarchical level that should be used for cell coloring. Must be one of levels. Default is to use the lowest level cells for Voronoi treemaps and all levels for sunburst treemaps.
color_palette	(character) A character vector of colors used to fill cells. The default is to use rainbow_hcl
color_steps	(numeric) Approximate number of steps for the color gradient to be used when drawing cells with color_type = "cell_size". Default step number is 10, and final step number can a vary a bit because pretty() is used to calculate a decent color range.
border_level	(numeric) A numeric vector representing the hierarchical level that should be used for drawing cell borders, or NULL to omit drawing borders, The default is that all borders are drawn.
border_size	(numeric) A single number indicating initial line width of the highest level cells. Is reduced each level, default is 6 pts. Alternatively a vector of length(border_level), then each border is drawn with the specified width.

`border_color` (character) A single character indicating color for cell borders, default is a light grey. Alternatively a vector of `length(border_level)`, then each border is drawn with the specified color.

`label_level` (numeric) A numeric vector representing the hierarchical level that should be used for drawing cell labels, or NULL to omit drawing labels. The default is the deepest level (every cell has a label).

`label_size` (numeric) A single number indicating relative size of each label in relation to its parent cell. Alternatively a numeric vector of `length(label_level)` that specifies relative size of labels for each level individually.

`label_color` (character) A single character indicating color for cell labels. Alternatively a vector of `length(label_level)`, then each label is drawn with the specified color.

`label_autoscale` (logical) Whether to automatically scale labels based on their estimated width. Default is TRUE.

`title` (character) An optional title, default to `NULL`.

`title_size` (numeric) The size (or 'character expansion') of the title.

`title_color` (character) Color for title.

`legend` (logical) Set to TRUE if a color key should be drawn. Default is FALSE.

`legend_position` (character) The position of the legend, one of "left" (default), "right", "top", or "bottom".

`legend_size` (numeric) The relative size of the legend (0 to 1), default is 0.1.

`custom_range` (numeric) A numeric vector of length 2 that can be used to rescale the values in `custom_color` to the range of choice. The default is `NULL` and it only has an effect if `custom_color` was specified when generating the treemap.

`width` (numeric) The width (0 to 0.9) of the viewport that the treemap will occupy.

`height` (numeric) The height (0 to 0.9) of the viewport that the treemap will occupy.

`layout` (numeric) Vector of length 2 indicating the number of rows and columns that the plotting area is supposed to be subdivided in. Useful only together with `position`, which indicates the position of the specific treemap. Use `add = TRUE` to omit starting a new page every time you call `drawTreemap()`.

`position` (numeric) Vector of length 2 indicating the position where the current treemap should be drawn. Useful only together with `layout`, which indicates the number of rows and columns the plotting area is subdivided into. Use `add = TRUE` to omit starting a new page every time you call `drawTreemap()`.

`add` (logical) Defaults to `FALSE`, creating a new page when drawing a treemap. When multiple treemaps should be plotted on the same page, this should be set to TRUE, and position of treemaps specified by `layout` and `position` arguments.

Value

The function does not return a value (except NULL). It creates a grid viewport and draws the treemap.

See Also

voronoiTreemap for generating the treemap that is the input for the drawing function

Examples

```
# load package
library(WeightedTreemaps)

# generate dummy data
df <- data.frame(
  A = rep(c("abcd", "efgh"), each = 4),
  B = letters[1:8],
  size = c(37, 52, 58, 27, 49, 44, 34, 45)
)

# compute treemap
tm <- voronoiTreemap(
  data = df,
  levels = c("B"),
  cell_size = "size",
  shape = "circle",
  positioning = "regular",
  seed = 123
)

# plot treemap with each cell colored by name (default)
drawTreemap(tm, label_size = 1, color_type = "categorical")

# plot treemap with each cell colored by name, but larger cells
# lighter and smaller cells darker
drawTreemap(tm, label_size = 1, color_type = "both")

# plot treemap with different color palette and style
drawTreemap(tm, label_size = 1, label_color = grey(0.3),
            border_color = grey(0.3), color_palette = heat.colors(6)
)

# ------------------------------------------

# load example data
data(mtcars)
mtcars$car_name = gsub(" ", "\n", row.names(mtcars))

# generate sunburst treemap
tm <- sunburstTreemap(
  data = mtcars,
  levels = c("gear", "cyl"),
  cell_size = "hp"
)

# draw treemap
drawTreemap(tm,
```

```
  title = "A sunburst treemap",
  legend = TRUE,
  border_size = 2,
  label_color = grey(0.6)
)
```

`get_polygons` *get_polygons*

Description

Extract the list of polygons from a voronoiResult object.

Usage

```
get_polygons(x, ...)
```

Arguments

x	(voronoiResult) A voronoi treemap results object
...	(none) Not used

Value

List of polygons, as class POLYGON from sf

See Also

`voronoiTreemap` for generating the treemap that is the input for this function

---

`Jahn_CellReports_2018` *Data from the publication of Jahn et al., CellReports, 2018*

Description

The dataset contains protein abundances of the Synechocystis sp. PCC6803 proteome. Protein abundance was determined using shotgun mass spectrometry. The data set also contains pathway information according to the cyanobase hierarchical annotation:

Usage

```
data(Jahn_CellReports_2018)
```

Format

A data frame with 19790 rows and 12 variables

Details

- protein - protein ID
- condition - combination from light and CO2
- light - light intensity in umol photons / m2 * s
- co2_concentration - CO2 concentration in
- mean_intensity - mean MS1 ion intensity
- mean_mass_fraction_norm - normalized mean mass fraction of protein
- sd_intensity - standard deviation from mean
- Process - functional annotation 1st level
- Pathway - functional annotation 2nd level
- Protein - functional annotation 3rd level
- Process.abbr - abbreviated Process
- Pathway.abbr - abbreviated Pathway
- Gene.names - trivial names of genes, if available

Source

https://pubmed.ncbi.nlm.nih.gov/30304686/

`poly_area` *poly_area*

Description

Mainly for internal use. Determines the area of a polygon based on its x and y coordinates. This function is a reimplementation of ‘soiltexture::TT.polygon.area()‘ and only re-implemented to avoid extra dependencies.

Usage

```
poly_area(poly_x, poly_y)
```

Arguments

`poly_x`	(numeric) X coordinates of each vertices of the polygon
`poly_y`	(numeric) Y coordinates of each vertices of the polygon

Value

A numeric, the area of the polygon.

See Also

soiltexture::TT.polygon.area()

poly_centroid *poly_centroid*

Description

Mainly for internal use. Determines the centroids of a polygon based on its x and y coordinates. This function is a reimplementation of ‘soiltexture::TT.polygon.centroids()‘ and only re-implemented to avoid extra dependencies.

Usage

```
poly_centroid(poly_x, poly_y)
```

Arguments

poly_x	(numeric) X coordinates of each vertices of the polygon
poly_y	(numeric) Y coordinates of each vertices of the polygon

Value

A numeric, the area of the polygon.

See Also

soiltexture::TT.polygon.centroids()

poly_transform_shape *poly_transform_shape*

Description

Mainly for internal use. The function transforms ('projects') arbitrary input coordinates that are used as parent polygon or shape of the treemap. By default it scales and centers polygon coordinates on a square of 0 to 2000 units.

Usage

```
poly_transform_shape(poly, xy_max = 2000)
```

Arguments

poly	(list) named list of 'x' and 'y' coordinates of the polygon
xy_max	(numeric) maximum desired expansion of the polygon (default: 2000)

Value

A list with slots 'x' and 'y' containing numeric coordinates

print.sunburstResult *print.sunburstResult*

Description

Print method for sunburstResult

Usage

```
## S3 method for class 'sunburstResult'
print(x, ...)
```

Arguments

x	(sunburstResult) A sunburst treemap results object
...	(none) Not used

Value

Returns a data.frame

See Also

sunburstTreemap for generating the treemap that is the input for this function

print.voronoiResult *print.voronoiResult*

Description

Print method for voronoiResult

Usage

```
## S3 method for class 'voronoiResult'
print(x, ...)
```

Arguments

x	(voronoiResult) A voronoi treemap results object
...	(none) Not used

Value

Returns a data.frame

See Also

voronoiTreemap for generating the treemap that is the input for this function

---

`rounded_rect` *Coordinates to draw a rounded rectangle as parent cell for treemaps*

Description

Set of coordinates for a rounded rectangle as parent cell for treemaps

Usage

```
data(rounded_rect)
```

Format

A data frame with 50 rows and 2 variables

Details

- X - coordinate
- Y - coordinate

`summary.sunburstResult` *summary.sunburstResult*

Description

Summary method for sunburstResult

Usage

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

Arguments

`object`	(sunburstResult) A sunburst treemap results object
`...`	(none) Not used

Value

Returns a data.frame

See Also

sunburstTreemap for generating the treemap that is the input for this function

summary.voronoiResult *summary.voronoiResult*

Description

Summary method for voronoiResult.

Usage

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

Arguments

object	(voronoiResult) A voronoi treemap results object
...	(none) Not used

Value

Returns a data.frame

See Also

voronoiTreemap for generating the treemap that is the input for this function

sunburstTreemap *sunburstTreemap*

Description

Create sunburst treemaps where variables are encoded by size of circular sectors.

Usage

```
sunburstTreemap(
  data,
  levels,
  fun = sum,
  sort = TRUE,
  filter = 0,
  cell_size = NULL,
  custom_color = NULL,
  diameter_inner = 0.3,
  diameter_outer = 0.8,
  verbose = FALSE
)
```

Arguments

data	(data.frame) A data.frame with one column for each hierarchical level
levels	(character) Character vector indicating the column names to be used. The order of names must correspond to the hierarchical levels, going from broad to fine
fun	(function) Function to be used to aggregate cell sizes of parental cells
sort	(logical) Should the columns of the data.frame be sorted before?
filter	(logical) Filter the supplied data frame to remove very small sectors that may not be visible. The default is to not remove any sectors.
cell_size	(character) The name of the column used to control sector size. Can be one of `levels` or any other column with numerical data. NA or values equal or less than zero are not allowed. The values in this column are aggregated by the function specified by `fun`. If `sector_size = NULL`, sector size is simply computed by the number of members for the respective sector (corresponding to rows in the data.frame).
custom_color	(character) An optional column that can be specified to control cell color. Cell colors are determined when drawing the treemap using `drawTreemap`, but the default is to use one of `levels` or `cell size`. Any other data source that shall be used instead has to be included in the treemap generation and explicitly specified here. The default value is `NULL`.
diameter_inner	(numeric) The minimum inner diameter of the drawn map. Defaults to 0.3,
diameter_outer	(numeric) The maximum outer diameter of the drawn map. Defaults to 0.8
verbose	(logical) If verbose is TRUE (default is FALSE), basic information such as a success message is printed to the console.

Details

This function returns a treemap object instead of a plot. In order to actually draw the treemap, use `drawTreemap`.

Value

'sunburstTreemap' returns an object of the formal class 'sunburstResult'. It is essentially a list of objects related to the graphical representation of the treemap (polygons, labels, cell data) as well as data from the call of the function. It contains the following slots:

cells	'list' of polygons for drawing a treemap
data	'data.frame', the original data that was supplied to calling 'voronoiTreemap'
call	'list' of arguments used to call 'voronoiTreemap'

See Also

`drawTreemap` for drawing the treemap.

Examples

```
# load example data
data(mtcars)
mtcars$car_name = gsub(" ", "\n", row.names(mtcars))

# generate treemap;
# by default cell (sector) size is encoded by number of members per group
tm <- sunburstTreemap(
  data = mtcars,
  levels = c("gear", "cyl"),
  cell_size = "hp"
)

# draw treemap with default options
drawTreemap(tm,
  title = "A sunburst treemap",
  legend = TRUE,
  border_size = 2,
  layout = c(1, 3),
  position = c(1, 1)
)

# use custom color palette
drawTreemap(tm,
  title = "Use custom palette",
  legend = TRUE,
  color_palette = rep(c("#81E06E", "#E68CFF", "#76BBF7"), c(3, 4, 5)),
  border_size = 2,
  label_level = 2,
  label_size = 0.7,
  label_color = grey(0.5),
  layout = c(1, 3),
  position = c(1, 2),
  add = TRUE
)

# color cells (sectors) based on cell size
drawTreemap(tm,
  title = "Coloring encoded by cell size",
  color_type = "cell_size",
  legend = TRUE,
  color_palette = rev(heat.colors(10)),
  border_size = 3,
  border_color = grey(0.3),
  label_level = 1,
  label_size = 2,
  label_color = grey(0.5),
  layout = c(1, 3),
  position = c(1, 3),
  add = TRUE
)
```

voronoiTreemap *voronoiTreemap*

Description

Create nested additively weighted Voronoi treemaps.

Usage

```
voronoiTreemap(
  data,
  levels,
  fun = sum,
  sort = TRUE,
  filter = 0,
  cell_size = NULL,
  custom_color = NULL,
  shape = "rectangle",
  maxIteration = 100,
  error_tol = 0.01,
  convergence = "intermediate",
  seed = NULL,
  positioning = "regular",
  verbose = FALSE,
  debug = FALSE
)
```

Arguments

data	(data.frame) A data.frame with one column for each hierarchical level
levels	(character) Character vector indicating the column names to be used. The order of names must correspond to the hierarchical levels, going from broad to fine
fun	(function) Function to be used to aggregate cell sizes of parental cells
sort	(logical) Should the columns of the data.frame be sorted before treemap generation?
filter	(numeric) Filter the supplied data frame to remove very small cells that may not be visible. The default is to remove cells with a relative target area below a threshold of zero (no negative values allowed). Computation time can increase when many small cells are present. For example, a threshold of 0.01 filters out all observations/cells below 1 % of the total area.
cell_size	(character) The name of the column used to control cell size. Can be one of `levels` or any other column with numerical data. NA or values equal or less than zero are not allowed as the cell area needs to be positive. The values in this column are aggregated by the function specified by `fun`. If `cell_size = NULL`, cell area is simply computed by the number of members for the respective cell (corresponding to rows in the data.frame).

`custom_color` (character) An optional column that can be specified to control cell color. Cell colors are determined when drawing the treemap using `drawTreemap`, but the default is to use one of `levels` or `cell size`. Any other data source that shall be used instead has to be included in the treemap generation and explicitly specified here. The default value is `NULL`.

`shape` (list or character) Set the initial shape of the treemap. Currently supported are the keywords "rectangle", "rounded_rect", "circle" or "hexagon". Alternatively the user can supply a named list with coordinates for a custom polygon. The slots of the list must be labeled 'x' and 'y'. The coordinates are not tested for validity, use on your own risk.

`maxIteration` (numeric) Force algorithm to stop at this number of iterations for each parent cell. The algorithm usually converges to an acceptable solution fairly quickly, so it seems reasonable to restrict this number in order to save computation time. However, more iterations give higher accuracy.

`error_tol` (numeric) The allowed maximum error tolerance of a cell. The algorithm will stop when all cells have lower error than this value. It is calculated as the absolute difference of a cell's area to its target area. The default is 0.01 (or 1 %) of the total parental area. Note: this is is different from a relative per-cell error, where 1 % would be more strict.

`convergence` (character) One of "slow", "intermediate", or "fast". Intermediate (default) and fast try to adjust cell weights stronger such that the algorithm converges faster towards the final size of the cell. However this comes at the price of stability, with a larger number of polygons possibly being misformed, e.g. by having self-intersections. Set convergence to "slow" if you experience problems to calculate treemaps with very unequal cell sizes or very large treemaps.

`seed` (integer) The default seed is NULL, which will lead to a new random sampling of cell coordinates for each tesselation. If you want a reproducible arrangement of cells, set seed to an arbitrary number.

`positioning` (character) Algorithm for positioning of starting coordinates of child cells in the parental cell using `spsample()`; "random" for completely random positions, "regular" for cells aligned to a grid sorted from bottom to top by name, "clustered" with regular positions of cells but sorted by name from inside out. Two variants "regular_by_area" and "clustered_by_area" will work as their counterparts but will sort by cell target area instead of cell name. `positioning` can be a single character or a vector of `length(levels)` to allow different positioning algorithms for each level.

`verbose` (logical) If verbose is TRUE (default is FALSE), messages with statistics for each iteration of a treemap as well as a success message are printed to the console.

`debug` (logical) If debug is TRUE (default is FALSE), the solution for each iteration is drawn to the viewport to allow some visual inspection. The weights, target area, and difference are printed to the console. It is not recommended to set this option to TRUE unless you know what you are doing, as it makes treemap generation much slower.

Details

This is a recursive wrapper function, making use of the original implementation of the voronoi tesselation from Paul Murrell, University of Auckland. The original functions were obtained and slightly modified from https://www.stat.auckland.ac.nz/~paul/Reports/VoronoiTreemap/voronoiTreeMap.html This function returns a treemap object instead of a plot. In order to actually draw the treemap, use drawTreemap.

Value

'voronoiTreemap' returns an object of the formal class 'voronoiResult'. It is essentially a list of objects related to the graphical representation of the treemap (polygons, labels, cell data) as well as data from the call of the function. It contains the following slots:

cells	'list' of polygons for drawing a treemap
data	'data.frame', the original data that was supplied to calling 'voronoiTreemap'
call	'list' of arguments used to call 'voronoiTreemap'

See Also

drawTreemap for drawing the treemap.

Examples

```
# load package
library(WeightedTreemaps)

# generate dummy data
df <- data.frame(
  A = rep(c("abcd", "efgh"), each = 4),
  B = letters[1:8],
  size = c(37, 52, 58, 27, 49, 44, 34, 45)
)

# compute treemap
tm <- voronoiTreemap(
  data = df,
  levels = c("B"),
  cell_size = "size",
  shape = "circle",
  positioning = "regular",
  seed = 123
)

# plot treemap with each cell colored by name (default)
drawTreemap(tm, label_size = 1, color_type = "categorical")

# plot treemap with each cell colored by name, but larger cells
# lighter and smaller cells darker
drawTreemap(tm, label_size = 1, color_type = "both")

# plot treemap with different color palette and style
```

```
drawTreemap(tm, label_size = 1, label_color = grey(0.3),
            border_color = grey(0.3), color_palette = heat.colors(6)
)
```

Index