Package 'packcircles'

August 29, 2016

Type Package

Title Circle Packing

Version 0.1.1

Date 2015-07-22

Description Simple algorithm to arrange non-overlapping circles within a rectangle.

License MIT + file LICENSE

URL https://github.com/mbedward/packcircles

Imports Rcpp (>= 0.11.5)

Suggests ggplot2, gridExtra, knitr

LinkingTo Rcpp

VignetteBuilder knitr

NeedsCompilation yes

Author Michael Bedward [aut, cre],
David Eppstein [aut] (Original author of Python code for graph-based
circle packing ported to C++ for this package)

Maintainer Michael Bedward <michael.bedward@gmail.com>

Repository CRAN

Date/Publication 2015-07-26 09:11:33

R topics documented:

circleGraphLayout 2
circleLayout 3
circlePlotData 4
circleVertices 5
packcircles 6

Index **7**

`circleGraphLayout` *Find an arrangement of circles conforming to specified sizes and adjacencies.*

Description

Attempts to derive an arrangement of circles satisfying prior conditions for size and adjacency. Unlike the `circleLayout` function, this is a deterministic algorithm. Circles are classified as either internal or external. Viewing the pattern of adjacencies as a triangulated mesh, external circles are those on the boundary. Under the version of the algorithm implemented here, the radii of external circles are provided as inputs, while the radii of internal circles are derived as part of the output arrangement.

Usage

```
circleGraphLayout(internal, external)
```

Arguments

`internal`	A list of vectors of circle ID values where, in each vector, the first element is the ID of an internal circle and the remaining elements are the IDs of that circle's neighbours arranged as a cycle. The cycle may be clockwise or anti-clockwise but the same orering must be used for all vectors.
`external`	A data.frame or matrix of external circle radii, with circle IDs in the first column and radii in the second column.

Details

The `internal` argument specifies circle adjacencies (ie. tangencies). The format is an concise representation of graph edges, and consists of a list of vectors: one per internal circle. In each vector the first element is the ID value of the internal circle and the remaining values are IDs of neighbouring circles, which may be either internal or external.

The `external` argument is a data.frame which specifies the radii of external circles. Internal circle radii should not be specified as they are derived as part of the fitting algorithm. The function will issue an error if any internal circle IDs are present in the `external` data.

Value

A data.frame with columns for circle ID, centre X and Y ordinate, and radius.

The output arrangement as a data.frame with columns for circle ID, centre X and Y ordinates, and radius. For external circles the radius will equal input values.

Note

Please treat this function as experimental.

References

C.R. Collins & K. Stephenson (2003) An algorithm for circle packing. Computational Geometry Theory and Applications 25:233-256.

Examples

```
## Not run:
## Simple example with two internal circles surrounded by
## four external circles. Internal circle IDs are 1 and 2.
internal <- list( c(1, 3, 4, 5), c(2, 3, 4, 6) )

## Uniform radius for external circles
external <- data.frame(id=3:6, radius=1.0)

## Generate the circle packing
packing <- circleGraphLayout(internal, external)

## End(Not run)
```

`circleLayout` *Find an arrangement of circles given circle radii and initial positions.*

Description

Attempts to find a layout for a given set of circles within a rectangle such that is no overlap between circles.

Usage

```
circleLayout(xyr, xlim, ylim, maxiter = 1000, wrap = TRUE, weights = 1)
```

Arguments

`xyr`	3-column matrix or data.frame (centre X, centre Y, radius)
`xlim`	bounds in the X direction; either a vector for [xmin, xmax) or a single value interpreted as [0, xmax)
`ylim`	bounds in the Y direction; either a vector for [ymin, ymax) or a single value interpreted as [0, ymax)
`maxiter`	maximum number of iterations to attempt
`wrap`	whether to allow coordinate wrapping across bounds. If 'true', coordinate wrapping results in a toroidal space; if 'false', ordinates are simply restricted to bounds.
`weights`	an optional vector of numeric weights (0 to 1 inclusive) to apply to the distance each circle moves during pair-repulsion. A weight of 0 prevents any movement. A weight of 1 gives the default movement distance. A single value can be supplied for uniform weights. A vector with length less than the number of circles will be silently extended by repeating the final value. Any values outside the range [0, 1] will be clamped to 0 or 1.

Value

A list with components:

layout A 3-column matrix or data.frame (centre x, centre y, radius).

niter Number of iterations performed.

`circlePlotData` *Generate plotting data for circles.*

Description

Given a matrix or data.frame of circle layout data (circle centres, radii and optional IDs), generates a data.frame of plotting data to use with ggplot.

Usage

```
circlePlotData(layout, npoints = 25, xyr.cols = 1:3, id.col = NULL)
```

Arguments

`layout`	A matrix or data.frame of circle data (x, y, radius). May contain other columns, including an optional ID column.
`npoints`	The number of vertices to generate for each circle.
`xyr.cols`	Indices or names of columns for x, y, radius (in that order). Default is columns 1-3.
`id.col`	Optional index or name of column for circle IDs in output. If not provided, the output circle IDs will be the row numbers of the input circle data.

Value

A data.frame with columns: id, x, y; where id is the unique integer identifier for each circle.

See Also

`circleVertices`

Examples

```
## Not run:
## draw some random circles with ggplot
library(ggplot2)

xmax <- 100
ymax <- 100
rmin <- 10
rmax <- 20
N <- 20
```

```
## Random centre coordinates and radii
layout <- data.frame(id = 1:N,
                     x = runif(N, 0, xmax),
                     y = runif(N, 0, ymax),
                     r = runif(N, rmin, rmax))

## Get data for circle vertices
plotdat <- circlePlotData(layout, id.col=1, xyr.cols=2:4)

## Draw circles annotated with their IDs
ggplot() +
  geom_polygon(data=plotdat, aes(x, y, group=id), fill=NA, colour="black") +
  geom_text(data=layout, aes(x, y, label=id)) +
  coord_equal() +
  theme_bw()

## End(Not run)
```

`circleVertices` *Generate vertex coordinates for a circle.*

Description

Generates vertex coordinates for a circle with given centre and radius.

Usage

```
circleVertices(xc, yc, r, npoints = 25)
```

Arguments

`xc`	centre X
`yc`	centre Y
`r`	radius
`npoints`	number of vertices

Value

A 2-column matrix of X and Y values.

See Also

`circlePlotData`

packcircles *packcircles: arrange non-overlapping circles.*

Description

This package provides a simple algorithm to arrange circles of arbitrary radii within a rectangle such that there is no-overlap between circles.

Details

The algorithm is adapted from an example written in Processing by Sean McCullough (which no longer seems to be available online). It involves iterative pair-repulsion, in which overlapping circles move away from each other. The distance moved by each circle is proportional to the radius of the other to approximate inertia (very loosely), so that when a small circle is overlapped by a large circle, the small circle moves furthest. This process is repeated iteratively until no more movement takes place (acceptable layout) or a maximum number of iterations is reached (layout failure). To avoid edge effects, the bounding rectangle is treated as a toroid. Each circle's centre is constrained to lie within the rectangle but its edges are allowed to extend outside.

Index

circleGraphLayout, 2
circleLayout, 2, 3
circlePlotData, 4, *5*
circleVertices, *4*, 5

packcircles, 6
packcircles-package *(*packcircles*)*, 6