Package 'rviewgraph'

May 7, 2020

Type Package

Title Animated Graph Layout Viewer

Version 1.3.1

Date 2020-05-06

Author Alun Thomas

Maintainer Alun Thomas <Alun.Thomas@utah.edu>

Description This is an 'R' interface to Alun Thomas's 'ViewGraph' 'Java' graph
viewing program.
It takes a graph specified as an incidence matrix, list of
edges, or in 'igraph' format and runs a graphical user interface that shows
an animation of a force directed algorithm positioning the vertices in two
dimensions. It works well for graphs of various structure of up to a few
thousand vertices. It's not fazed by graphs that comprise several
components. The coordinates can be read as an 'igraph' style layout
matrix at any time. The user can mess with the layout using a mouse,
preferably one with 3 buttons, and some keyed commands.

Suggests

Enhances igraph, Matrix

Depends rJava

SystemRequirements 'Java' version >= 8

License GPL-2

LazyLoad yes

Collate 'onLoad.R' 'rViewGraph.R'

Encoding UTF-8

RoxygenNote 7.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2020-05-07 10:30:02 UTC

R topics documented:

`rviewgraph-package` *Animated Graph Layout Viewer*

Description

rviewgraph provides an 'R' interface to Alun Thomas's 'ViewGraph' 'Java' graph viewing program.

Details

Package:	rviewgraph
Type:	Package
Version:	1.3.1
Date:	2020-05-06
License:	GPL-2
LazyLoad:	yes
SystemRequirements:	'Java' >= 8

This package provides an 'R' interface to Alun Thomas's 'ViewGraph' 'Java' graph viewing program. It takes a graph specified as an incidence matrix, array of edges, or in `igraph` format and runs a graphical user interface that shows an animation of a force directed algorithm positioning the vertices in two dimensions. It works well for graphs of various structure of up to a few thousand vertices. It's not fazed by graphs that comprise several components. The coordinates can be read as an `igraph` style layout matrix at any time. The user can mess with the layout using a mouse, preferably one with 3 buttons, and some keyed commands.

The only function that the user needs to know about in this package is `rViewGraph()`. It launches the GUI by delegating to specific functions depending on the class of the arguments.

`rViewGraph` *This is a function to create and start the 'Java' graph animation GUI.*

Description

Creates and starts an animated GUI for positioning the vertices of a graph in 2 dimensions.

Usage

```
rViewGraph(object, names, layout, directed, running, ...)

## Default S3 method:
rViewGraph(
  object,
  names = seq(max(object)),
  layout = NULL,
  directed = FALSE,
  running = TRUE,
  ...
)

## S3 method for class 'Matrix'
rViewGraph(
  object,
  names = 1:max(dim(object)),
  layout = NULL,
  directed = FALSE,
  running = TRUE,
  ...
)

## S3 method for class 'igraph'
rViewGraph(
  object,
  names = igraph::V(object)$name,
  layout = igraph::layout.random(object),
  directed = igraph::is.directed(object),
  running = TRUE,
  ...
)
```

Arguments

`object`	the object specifying the graph
`names`	the names of the nodes
`layout`	the starting positions of the vertices
`directed`	indicates whether or not the graph is directed
`running`	indicates whether or not to start with animation running
`...`	passed along extra arguments.

Details

Creates and starts a 'Java' GUI showing a real time animation of a Newton-Raphson optimization of a force function specified between the vertices of an arbitrary graph. There are attractive forces between adjacent vertices and repulsive forces between all vertices. The repulsions go smoothly to zero for a finite distance between vertices so that, unlike some other methods, different components don't send each other off to infinity.

This is a generic function that delegates its work to specific functions depending on the class of the first argument. It can handle an incidence matrix, a list of edges, and an `igraph` graph object.

The program is controlled by a slide bar, some buttons, the arrow, home and shift keys, but mostly by mouse operations. All three mouse buttons are used.

- The slide bar at the bottom controls the repulsive force in the energy equation used to set the coordinates.
- Mouse operations without shift key and without control key pressed.
  1. Left mouse: Drags a vertex. Vertex is free on release.
  2. Middle mouse: Drags a vertex. Vertex is fixed at release position.
  3. Right mouse: Translates the view by the amount dragged. A bit like putting your finger on a piece of paper and moving it.
  4. Double click with any mouse button in the background: Resets the vertices to new random positions.
- Mouse operations with shift key but without control key pressed.
  1. Left mouse: Drags a vertex and the component it is in. Vertex and component free on release.
  2. Middle mouse: Drags a vertex and the component it is in. Vertex and component are fixed at release positions.
  3. Right mouse: Translates the positions of the vertices relative to the position of the canvas by the amount dragged. This is useful to center the picture on the canvas ready for outputting.
- Mouse operations without shift key but with control key pressed.
  1. Left mouse: Click on a vertex to un-delete any deleted neighbours.
  2. Middle mouse: Click on a vertex to delete it from the graph.
  3. Double click left mouse: Replaces all deleted vertices in the graph.
  4. Double click middle mouse: Deletes all vertices from the graph.
- Mouse operations with shift key and with control key pressed.
  1. Left mouse: Click on a vertex to replace all vertices in the same component to the graph.
  2. Middle mouse: Click on a vertex to delete it and the component it is in from the graph.
- Key functions without shift key pressed. Mouse has to be in the picture canvas.
  1. Up arrow: Increases the scale of viewing by 10%.
  2. Down arrow: Decreases the scale of viewing by 10%.
  3. Left arrow: Rotates the view by 15 degrees clockwise.
  4. Right arrow: Rotates the view by 15 degrees anticlockwise.
  5. Home key: Undoes all scalings and rotations and places the origin at the center of the picture canvas.
- Key functions with shift key pressed. Mouse has to be in the picture canvas.
  1. Up arrow: Increases the vertex positions by 10% relative to the scale of the canvas.
  2. Down arrow: Decreases the vertex positions by 10% relative to the scale of the canvas.

3. Left arrow: Rotates the vertex positions by 15 degrees clockwise relative to the canvas orientation.
4. Right arrow: Rotates the vertex positions by 15 degrees anticlockwise relative to the canvas orientation.

Value

`rViewGraph` is intended only for interactive use. When used in a non-interactive envirionment it immediately exists returning the value `NULL`. Otherwise, all versions of `rViewGraph` return a list of functions that control the actions of the interactive viewer. None of the functions in the list take an argument.

`run()`	Starts the GUI running if it's not already doing so.
`stop()`	Stops the GUI running if it's running.
`hide()`	Stops the GUI and hides it.
`show()`	Shows the GUI. If it was running when `hide` was called, it starts running again.
`getLayout()`	Returns the coordinates of the vertices that are currently shown in the GUI. These are in the format that `igraph` expects for layouts.

Author(s)

Alun Thomas

Source

A full description of the force function and algorithm used is given by C Cannings and A Thomas, Inference, simulation and enumeration of genealogies. In D J Balding, M Bishop, and C Cannings, editors, The Handbook of Statistical Genetics. Third Edition, pages 781-805. John Wiley & Sons, Ltd, 2007.

Examples

```
# Viewing an Erdos Renyi random graph specified by random edges.
f = sample(100,size=200,replace=TRUE)
t = sample(100,size=200,replace=TRUE)
vft = rViewGraph(cbind(f,t))
#
# Edges can also be specified in \code{igraph} style.
e = c(t,f)
ve = rViewGraph(e)
```

Index