Package 'InteractiveIGraph'

February 19, 2015

Type Package

Title interactive network analysis and visualization.

Version 1.0.6.1

Date 2013-03-02

Author Vygantas Butkus

Maintainer Vygantas Butkus <Vygantas.Butkus@gmail.com>

Depends igraph, grDevices

Description An extension of the package 'igraph'. This package create possibly to work with 'igraph' objects interactively.

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2013-05-07 22:05:15

R topics documented:

`InteractiveIGraph-package`
Interactive igraph

Description

This package is an extension of the package igraph. It allows to work with graph interactively similarly as with function tkplot, but it has advantage of much greater flexibility (with the cost of simplicity).

Please see doc/Illustrations.pdf for short graphical illustration of this package.

The main function in this package is InteractiveIGraph - in fact, all package is all about this function.

Details

Package:	InteractiveIGraph
Type:	Package
Version:	1.0.6.1
Date:	2013-03-02
Author:	Vygantas Butkus
Maintainer:	Vygantas Butkus <Vygantas.Butkus@gmail.com>
Institution:	the Bank of Lithuania
Depends:	igraph, grDevices
License:	GPL-2

This package is based on igraph, grDevices packages, therefore it has their advantages and disadvantages.

First of all, InteractiveIGraph works only on Windows and X11(type = "Xlib") screen devices, therefore if working with RStudio you should activate one of the devices. Secondly, with the large graphs, reploting becomes really annoying, so, it is really for small graphs. On the other hand customising large graphs in console might by even more annoying.

This package has several huge advantages comparing with tkplot. Firstly, it is much more flexible. You can change any attributes or vertexes or edges. Moreover, you may easily create any extra functionality in menu commands. On the other hand, it is not so much intuitive and stable. In the case of wrong entrees it may crash.

Warning

The system is tested only in Windows (sorry for that).

Note

This project is not fully completed, but unfortunately it is not in active development any more. Up till now it fulfills all my needs. In the case of new ideas I might join further development.

Author(s)

Author and maintainer: Vygantas Butkus <Vygantas.Butkus@gmail.com>.

See Also

The main function is InteractiveIGraph.

Examples

```
# this is a regular igraph object
gOrg <- erdos.renyi.game(15, 1/10,  directed = TRUE)
plot(gOrg)

# this is an 'InteractiveIGraph' object, up till now there is not much difference.
# Just some extra attributes.
g = InteractiveIGraph.Constructor(gOrg)
g = plot(g)

# now it is interactive. Please enjoy :)
if(interactive()){
  g = InteractiveIGraph(gOrg)
}
# p.s. if you want to save changes please press 'q' while ruining InteractiveIGraph().

# Select some vertices and apply commands, for example 'a', 'A', or 'g', 'b'.
# you may change any attributes by key 'ctrl-F' and then typing attributes command,
# for example 'color="green"'
# (it will work only on selected vertices).

# you may run any command by key 'ctrl-C' and typing command, for example 'print(V(g)$name)'

# with key 's' you can change selection mode - in this way you can select multiple vertices.
```

AdjustmentFunctions *Adjustment Functions*

Description

Special functions that control the behaviour of InteractiveIGraph.

Usage

```
PlotAdjustment.default(g)

ExtraInfo.default(type="V", ProgId=0, g=g)

BottomMenuAdjustment.default(g)
```

Arguments

g	an object of the class InteractiveIGraph.
type	the type of active object. Might be "V", "E", "G".
ProgId	The id of the active object.

Details

The user may create its own functions and replace the default funcions to change the behavor of the InteractiveIGraph. See examples.

Value

Adjustment functions must return an object of the class InteractiveIGraph.

See Also

InteractiveIGraph

Examples

```
gOrg <- erdos.renyi.game(15, 1/10,  directed = TRUE)
V(gOrg)$comment = paste("Some comment about vertex with number:", V(gOrg))


ExtraInfo.New <- function(type="V", ProgId=0, g=g){
msg = ""

  if(type=="V"){
    vid = which(V(g)$ProgId == ProgId)
    msg = paste(",", V(g)[vid]$comment)
  }
  if(type=="E"){
    msg = ", E..."
  }
  if(type=="G"){
    msg = ", G..."
  }

  Item = list(list(label=msg, RegionParams=list(XBufCof = 0, YBufCof=0.2),
RecParams=list(lwd = NA, border=NA), TextParams=list(cex=0.8)))
  g <- MenuAddItems(Items=Item,  MenuLine='Info', g=g)

  return(g)
}

if(interactive()){
  g = InteractiveIGraph(gOrg, ExtraInfo=ExtraInfo.New)
}
```

CommandList *CommandList*

Description

The list of commands that could be applied during InteractiveIGraph.

Usage

```
PrintCommandList()

CommandList
```

Details

Almost all commands (hot-keys) that could be applied in InteractiveIGraph are recorded in the CommandList. It is an object of the type list. The names of it's elements correspond to a hot-key and the elements contains a list with label, description and the function.

You can append this list, but be aware of the CommandList structure and be sure to return an object of the class InteractiveIGraph at the end of the function.

Please apply PrintCommandList function to find out possible commands.

There are some special keys that are not in the list.

- <Esc>Drop any selection and exit any mode.
- <ctrl-Z>Undo, it will rester your last action.
- <Enter>Enter to view mode if any object is active, or enters command if system is waiting for some input.
- <ctrl-V>Applies paste command if system is waiting for some input.
- <Backspace>Applies Backspace if system is waiting for some input.

Value

The list of commands.

See Also

InteractiveIGraph

Examples

```
PrintCommandList()
```

`InteractiveIGraph` *Interactive igraph*

Description

Interactive `igraph` - it creates opportunity to work with `igraph` interactively. It enables user to analyse graph properties and structure. It is possible to change graph it self or customise it appearance.

Please see `doc/Illustrations.pdf` for short graphical illustration of this function.

Usage

```
InteractiveIGraph(g, ...)

InteractiveIGraph.Constructor(g, ...)

## S3 method for class 'InteractiveIGraph'
plot(x, ...)

## S3 method for class 'InteractiveIGraph'
as.igraph(x, KeepAttr = NULL, ...)
```

Arguments

`g, x`	an object of the class `InteractiveIGraph`.
`KeepAttr`	by default all special `InteractiveIGraph` attributes are deleted during transformation. User might keep them if required.
`...`	further arguments. See details and examples.

Details

Before running `InteractiveIGraph` firstly you should create simple `igraph` object. Later on, you must be sure you are running correct screen device. `InteractiveIGraph` works only on `window` and `X11(type = "Xlib")` screen devices, therefore if working with RStudio (or smth.) you should activate one of the correct devices.

After running the function, you can play around by using computer mouse directly on screen device. You can grab vertex (or group if them), move it, change any attributes. You may even create, or delete vertices and edges. If you want to save the results, press key <s> - it will `dump` object to text file. More over you can save the picture by presing <Ctrl-B> or press <q>, that stands for 'quite' - it will end the interaction program and assign the last value of the object to predefined variable. See examples.

Keyboard is active if the graphical device is active. To apply the command you just need to know a hot-key. While ruining `InteractiveIGraph` you may press <m> that stands for 'menu' and all the list will be printed in the console. The list of commands are kept in `CommandList`. This list might

by modified by user, therefore, any functionality might my easily added by creating new hot-key in the CommandList. You can find out almost all the commands by exploring CommandList (run function PrintCommandList). For example if you want to see minimum.spanning.tree of your graph you should pres button <t> while running InteractiveIGraph. You may see how this is realized with the command CommandList[["t"]]

There are some special features, that user might found interesting:

- active vertices. Vertices may by active / not active. hot-key: a, A, ctrl-A.
- grouping. Vertices might be grouped interactively. hot-key: g, G, ctrl-G.
- blocking. Groups might be 'closed' in blocks, Isolating huge amount of vertices that not actual in the moment. hot-key: b (while any groups is active).

It should be notice that there are several modes, that influence the output of commands. All modes might by seen in console by pressing <M>. Main modes are:

- select. The select mode enables select multiple vertices(edges) by multiple clicking. It means, vertices are not unselected after new click.
- ActiveObject. If any object is labeled as active (in bottom menu), then special commands might be applied to it. For example delete, block.
- ViewObject. If it is in object's view mode it is labeled in the 'main' of the graph. Special treatment of the commands might by applied. For example, if it is view mode of group, then selecting vertices and pressing "g" will join or disjoin vertices to group (instead of creating new group.)
- Input. Specifies case, then system is waiting for some kind of input, for example command.
- AllEdges. By default, if there are more then one vertices selected no edge indexes are shown. This mode forces to show active vertex edges.

If you are in the mode that you don't want to be, press <Esc> - it will restore default.

In ... you may specify many parameters form igraph, namely: 'layout' 'xlim', 'ylim', 'main', 'sub', 'axes', 'xlab', 'ylab' , 'add'. User may specify the margins with parameter 'mar'. Any vertex or edge parameters should be add via V and E.

User may also specify its own Adjustment Functions, namely 'PlotAdjustment', 'BottomMenuAdjustment', 'ExtraInfo'.

And there is special attribute 'MainMenu' - this allows to create any menu that show by pressing right mouse button.

By default a very simple Menu is created. Menu is a list with special structure you can explore it with graph$Menu$MenuList$MainMenu. Unfortunately, this part is in developing stage - it is now OK to make it work, but not to be user-friendly.

Value

An object of the class InteractiveIGraph. It is the extension of the igraph with special attributes.

Warning

The function is tested only in windows (sorry for that).

Sometimes the program crashes. Most often it crashes while assigning wrong attributes values. For example if trying to assign color attribute with command `'color="gree"'` instead of `'color="green"'` it will crash, because there are no color "gree". It is very hard to make system flexible and safe - wherefore be sure assigning correct values. It should me mentioned that in very rare moments the reasons of crashing are not identified, thence it is recommended from time to time to press 'ctrl-S' - it will `dump` current graph to text file - thus you work wont be lost.

Note

This package is based on `igraph`, `grDevices` packages, therefore it has their advantages and disadvantages.

First of all, `InteractiveIGraph` works only on `Windows` and `X11(type = "Xlib")` screen devices, therefore if working with RStudio yous should activate one of the devices. Secondly, with the large graphs, reploting becomes really annoying, so, it is really for small graphs. On the other hand customising large graphs in console might my even more annoying.

Author(s)

Vygantas Butkus

See Also

`CommandList`, Adjustment Functions.

Examples

```
# this is a regular igraph object
gOrg <- erdos.renyi.game(15, 1/10,  directed = TRUE)
plot(gOrg)

# this is an 'InteractiveIGraph' object, up till now there is not much difference. Just some extra attributes.
g = InteractiveIGraph.Constructor(gOrg)
g = plot(g)

# now it is interactive. Please enjoy :)
if(interactive()){
  g = InteractiveIGraph(gOrg)
}
# p.s. if you want to save changes please press 'q' while ruining InteractiveIGraph().

# Select some vertices and apply commands, for example 'a', 'A', or 'g', 'b'.
# you may change any attributes by key 'ctrl-F' and then typing attributes command,
# for example 'color="green"'
# (it will work only on selected vertices).

# you may run any command by key 'ctrl-C' and typing command, for example 'print(V(g)$name)'
```

```
# with key 's' you can change selection mode - in this way you can select multiple vertices.
```

Index