Package ‘rgexf’

March 25, 2015

Type Package

Encoding UTF-8

Title Build, Import and Export GEXF Graph Files

Version 0.15.3

Date 2015-03-24

Author George Vega Yon <gvegayon@caltech.edu> [aut], Jorge Fábrega La-coa <jorge.fabrega@uai.cl> [ctb], Joshua B. Kunst <jbkunst@gmail.com> [ctb]

Maintainer George Vega Yon <gvegayon@caltech.edu>

Description Create, read and write GEXF (Graph Exchange XML Format) graph files (used in Gephi and others). Using the XML package, it allows the user to easily build/read graph files including attributes, GEXF viz attributes (such as color, size, and position), network dynamics (for both edges and nodes) and edge weighting. Users can build/handle graphs element-by-element or massively through data-frames, visualize the graph on a web browser through "sig-majs" (a javascript library) and interact with the igraph package.


Depends XML, Rook, igraph

License GPL (>= 3)

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-03-24 18:27:03

R topics documented:

rgexf-package .................................................. 2
add.gexf.node .............................................. 3
check.dpl.edges .......................................... 5
checkTimes .................................................. 6
edge.list ................................................... 7
followers .................................................... 8
rgexf-package

Build, Import and Export GEXF Graph Files

Description

Create, read and write GEXF (Graph Exchange XML Format) graph files (used in Gephi and others).

Using the XML package, it allows the user to easily build/read graph files including attributes, GEXF viz attributes (such as color, size, and position), network dynamics (for both edges and nodes) and edge weighting.

Users can build/handle graphs element-by-element or massively through data-frames, visualize the graph on a web browser through "sigmajs" (a javascript library) and interact with the igraph package.

Finally, the functions igraph.to.gexf and gexf.to.igraph convert objects from igraph to gexf and vice versa keeping attributes and colors.

Please visit the project home for more information: https://bitbucket.org/gvegayon/rgexf.

Details

Package: rgexf
Type: Package
Version: 0.15.3
Date: 2015-03-24
License: GPL version 2 or later

Note

See the GEXF primer for details on the GEXF graph format: http://gexf.net/1.2draft/gexf-12draft-primer.pdf

Author(s)

George Vega Yon <gvegayon@caltech.edu> [aut]
add.gexf.node

Jorge Fabrega Lacoa <jorge.fabrega@uai.cl> [cnt],
Joshua B. Kunst <jbkunst@gmail.com> [cnt]

References

• rgexf project site: https://bitbucket.org/gvegayon/rgexf
• Gephi project site: https://gephi.org/
• GEXF project site: http://gexf.net/
• Nodos Chile project site (real life examples): http://nodoschile.org/
• Sigmasj project site: http://sigmajs.org/

Examples

## Not run:

demo(gexf) # Example of gexf command using fictional data.
demo(gexfattributes) # Working with attributes.
demo(gexfbasic) # Basic net.
demo(gexfdynamic) # Dynamic net.
demo(edge.list) # Working with edges lists.
demo(gexffull) # All the package.
demo(gexftwitter) # Example with real data of chilean twitter accounts.
demo(gexfdynamicandatt) # Dynamic net with static attributes.
demo(gexfbuildfromscratch) # Example building a net from scratch.
demo(gexfigraph) # Two-way gexf-igraph conversion
demo(gexfrandom) # A nice routine creating a good looking graph

## End(Not run)

---

add.gexf.node Adding and removing nodes/edges from gexf objects

Description

Manipulates gexf objects adding and removing nodes and edges from both, its dataframe representation and its XML representation.

Usage

```r
add.gexf.node(graph, id=NA, label=NA, start=NULL, end=NULL,
              vizAtt=list(color=NULL, position=NULL, size=NULL, shape=NULL, image=NULL),
              atts=NULL)
```

```r
add.gexf.edge(graph, source, target, id=NULL, type=NULL, label=NULL, start=NULL,
              end=NULL, weight=1, vizAtt = list(color=NULL, thickness=NULL, shape=NULL),
              atts=NULL, digits = getOption("digits"))
```

```r
rm.gexf.node(graph, id=NULL, number=NULL, rm.edges = TRUE)
```
add.gexf.node

rm.gexf.edge(graph, id=NULL, number=NULL)

add.node.spell(graph, id=NULL, number=NULL, start=NULL, end=NULL,
               digits = getOption("digits"))

add.edge.spell(graph, id=NULL, number=NULL, start=NULL, end=NULL,
               digits = getOption("digits"))

Arguments

graph A gexf-class object.
id A node/edge id (normally numeric value).
label A node/edge label.
type Type of connection (edge).
number Index number(s) of a single or a group of nodes or edges.
weight Edge weight.
vizAtt A list of node/edge viz attributes (see write.gexf).
atts List of attributes, currently ignored.
source Source node’s id.
target Target node’s id.
start Starting time period
end Ending time period
rm.edges Whether to remove or not existing edges.
digits Integer. Number of decimals to keep for nodes/edges sizes. See print.default

Details

new.gexf.graph Creates a new gexf empty object (0 nodes 0 edges).
add.gexf.node and add.gexf.edge allow adding nodes and edges to a gexf object (graph) one at a time. rm.gexf.node and rm.gexf.edges remove nodes and edges respectively.
In the case of rm.gexf.node, by default every edge linked to the node that is been removed will also be removed (rm.edges = TRUE).
add.node.spell and add.edge.spell allow to include spells to specific nodes or edges in a gexf object.

Value

A gexf object (see write.gexf).

Author(s)

George Vega Yon <george.vega@nodoschile.org>,
Jorge Fabrega Lacoa <jorge.fabrega@nodoschile.org>
check.dpl.edges

References

The GEXF project website: http://gexf.net

Examples

```r
## Not run:
demo(gexfbuildfromscratch)

## End(Not run)
```

check.dpl.edges  Check (and count) duplicated edges

Description

Looks for duplicated edges and reports the number of instances of them.

Usage

```r
check.dpl.edges(edges, undirected=FALSE, order.edgelist=TRUE)
```

Arguments

- **edges**: A matrix or data frame structured as a list of edges
- **undirected**: Declares if the net is directed or not (does difference)
- **order.edgelist**: Weather to sort the resulting matrix or not

Details

check.dpl.edges looks for duplicated edges reporting duplicates and counting how many times each edge is duplicated.

For every group of duplicated edges only one will be accounted to report number of instances (which will be recognized with a value higher than 2 in the reps column), the other ones will be assigned a -1 at the reps value.

Function is mainly written in C, so speed gains are important.

Value

A three column data.frame with colnames “source”, “target” “reps”.

Author(s)

George Vega Yon <george.vega@nodoschile.org>
checkTimes

Description

Checks for correct time format

Usage

checkTimes(x, format = c("date", "dateTime", "float")

Arguments

x A string or vector of characters
format String, can be "date", "dateTime", "float"

Value

Logical.

Author(s)

George Vega Yon <george.vega@nodoschile.org>,
Jorge Fabrega Lacoa <jorge.fabrega@nodoschile.org>

Examples

test <- c("2012-01-17T03:46:41", "2012-01-17T03:46:410")
checkTimes(test, format="dateTime")
checkTimes("2012-02-01T00:00:00", "dateTime")
edge.list

Decompose an edge list

Description

Generates two data frames (nodes and edges) from a list of edges

Usage

dedge.list(x)

Arguments

x A matrix or data frame structured as a list of edges

Details

dedge.list transforms the input into a two-elements list containing a dataframe of nodes (with columns “id” and “label”) and a dataframe of edges. The last one is numeric (with columns “source” and “target”) and based on autogenerated nodes’ ids.

Value

A list containing two data frames.

Author(s)

George Vega Yon <george.vega@nodoschile.org>,
Jorge Fabrega Lacoa <jorge.fabrega@nodoschile.org>

Examples

edgelist <- matrix(
c("matthew","john",
"max","stephen",
"matthew","stephen"),
byrow=TRUE, ncol=2)

dedge.list(edgelist)
followers  

*Edge list with attributes*

### Description
Sample of accounts by december 2011.

### Usage
```r
data(followers)
```

### Format
A data frame containing 6065 observations.

### Source
Fabrega and Paredes (2012): “La politica en 140 caracteres” en Intermedios: medios de comunicacion y democracia en Chile. Ediciones UDP

---

### igraph.to.gexf

*Converting between gexf and igraph classes*

### Description
Converts objects between gexf and igraph objects keeping attributes, edge weights and colors.

### Usage
```r
igraph.to.gexf(igraph.obj, position=NULL)
gexf.to.igraph(gexf.obj)
```

### Arguments
- `igraph.obj`: An object of class igraph.
- `gexf.obj`: An object of class gexf.
- `position`: A three-column data-frame with XYZ coords.

### Details
If the position argument is not NULL, the new gexf object will include the position viz-attribute.
Value

- For `igraph.to.gexf`: `gexf` class object
- For `gexf.to.igraph`: `igraph` class object

Author(s)

George Vega Yon <george.vega@nodoschile.org>

See Also

`layout`

Examples

```r
## Not run:

# Running demo
demo(gexfigraph)

# A simple graph without
gexf1 <- read.gexf("http://gephi.org/datasets/LesMiserables.gexf")
igraph1 <- gexf.to.igraph(gexf1)
gexf2 <- igraph.to.gexf(igraph1)

# A graph with attributes
gexf3 <- read.gexf("http://gexf.net/data/data.gexf")
igraph2 <- gexf.to.igraph(gexf3)
gexf4 <- igraph.to.gexf(igraph2)

## End(Not run)
```

---

**new.gexf.graph**  
*Build an empty gexf graph*

Description

Builds an empty `gexf` object containing all the class’s attributes.

Usage

```r
new.gexf.graph(defaultedgetype = "undirected",
meta = list(
    creator="NodosChile",
    description="A graph file writing in R using 'gexf'",
    keywords="gexf graph, NodosChile, R, rgexf"
))
```
Arguments

defaultedgetype
   "directed", "undirected", "mutual"

meta A List. Meta data describing the graph

Value

A gexf object.

Author(s)

George Vega Yon <george.vega@nodoschile.org>, Jorge Fabrega Lacoa <jorge.fabrega@nodoschile.org>

References

The GEXF project website: http://gexf.net

Examples

```r
## Not run:
demo(gexfbuildfromscratch)

## End(Not run)
```

---

read.gexf

*Reads gexf (.gexf) file*

Description

`read.gexf` reads gexf graph files and imports its elements as a gexf class object

Usage

`read.gexf(x)`

Arguments

- `x` String. Path to the gexf file.

Value

A gexf object.

Note

By the time attributes and viz-attributes aren’t supported.
**rgexf Methods**

**Author(s)**

George Vega Yon <george.vega@nodoschile.org>,
Jorge Fabrega Lacoa <jorge.fabrega@nodoschile.org>

**References**

The GEXF project website: [http://gexf.net](http://gexf.net)

**Examples**

```r
## Not run:
mygraph <- read.gexf("http://gephi.org/datasets/LesMiserables.gexf")

## End(Not run)
```

---

**rgexf Methods**

*S3 methods for gexf objects*

---

**Description**

Methods to print and summarize `gexf` class objects

**Usage**

```r
## S3 method for class 'gexf'
print(x, file=NA, replace=F, ...)
## S3 method for class 'gexf'
summary(object, ...)
## S3 method for class 'gexf'
plot(x, EdgeType = c("curve", "line"), output.dir = NULL, ...)
```

**Arguments**

- `x` An `gexf` class object.
- `object` An `gexf` class object.
- `file` String. Output path where to save the GEXF file.
- `replace` Logical. If `file` exists, TRUE would replace the file.
- `EdgeType` For the visualization
- `output.dir` String. The complete path where to export the sigmajs visualization
- `...` Ignored
Details

print.gexf displays the graph (XML) in the console. If file is not NA, a GEXF file will be exported to the indicated filepath.

summary.gexf prints summary statistics and information about the graph.

plot.gexf plots the graph object in the web browser using sigma-js javascript library. Generated files are stored at the OS’s “temp” folder. If output.dir is not NULL, then all files required to display the graph in the web browser will be saved in the output.dir.

Users must note that plot.gexf starts a server using the Rook package, otherwise it will not be possible to see the visualization (sigmajs requires this).

Value

print.gexf None (invisible NULL).
summary.gexf List containing some gexf object statistics.
plot.gexf None (invisible NULL).

Author(s)

George Vega Yon <george.vega@nodoschile.org>,
Joshua B. Kunst <jbkunst@nodoschile.org>

References

sigmajs project website http://sigmajs.org/.

See Also

See also write.gexf

Examples

```r
## Not run:
# Data frame of nodes
people <- data.frame(id=1:4, label=c("juan", "pedro", "matthew", "carlos"),
                      stringsAsFactors=F)
# Data frame of edges
relations <- data.frame(source=c(1,1,1,2,3,4,2,4,4),
                         target=c(4,2,3,3,4,2,4,1,1))
# Building gexf graph
mygraph <- write.gexf(nodes=people, edges=relations)
# Summary and print
summary(mygraph)
print(mygraph, file="mygraph.gexf", replace=T)
# Plotting
```
switch.edges

plot(mygraph)

## End(Not run)

---

**switch.edges**

*Switches between source and target*

**Description**

Puts the lowest id node among every dyad as source (and the other as target)

**Usage**

`switch.edges(edges)`

**Arguments**

- `edges`: A matrix or data frame structured as a list of edges

**Details**

`edge.list` transforms the input into a two-elements list containing a dataframe of nodes (with columns “id” and “label”) and a dataframe of edges. The last one is numeric (with columns “source” and “target”) and based on autogenerated nodes’ ids.

**Value**

A list containing two data frames.

**Author(s)**

George Vega Yon &lt;george.vega@nodoschile.org&gt;

**Examples**

```r
relations <- cbind(c(1,1,3,4,2,5,6), c(2,3,1,2,4,1,1))
relations

switch.edges(relations)
```
twitteraccounts  

*Twitter accounts of Chilean Politicians and Journalists (sample)*

**Description**

Sample of accounts by December 2011.

**Usage**

`data(twitteraccounts)`

**Format**

A data frame containing 148 observations.

**Source**

Fabrega and Paredes (2012): “La politica en 140 caracteres” en Intermedios: medios de comuni-
cacion y democracia en Chile. Ediciones UDP

---

write.gexf  

*Builds a graph of gexf class*

**Description**

`write.gexf` takes a node matrix (or dataframe) and an edge matrix (or dataframe) and creates a
gexf object containing a data-frame representation and a gexf representation of a graph.

**Usage**

```r
write.gexf(
  nodes, edges, edgesLabel = NULL, edgesId = NULL,
  edgesAtt = NULL, edgesWeight = NULL,
  edgesVizAtt = list(
    color=NULL,
    size=NULL,
    shape= NULL
  ),
  nodesAtt = NULL,
  nodesVizAtt = list(
    color=NULL,
    position=NULL,
    size= NULL,
    shape=NULL,
    image= NULL
  ),
)```

```
write.gexf

nodeDynamic = NULL,
edgeDynamic = NULL,
digits =getOption("digits"),
output = NA, tFormat = "double",
defaultedgetype = "undirected",
meta = list(
    creator="NodosChile",
    description="A graph file writing in R using \"rgexf\"",
    keywords="gexf graph, NodosChile, R, rgexf"),
    keepFactors = FALSE,
    encoding = "UTF-8"
)

Arguments

nodes A two-column data-frame or matrix of “id”s and “label”s representing nodes.
edges A two-column data-frame or matrix containing “source” and “target” for each
each edge. Source and target values are based on the nodes ids.
edgesId A one-column data-frame, matrix or vector.
edgesLabel A one-column data-frame, matrix or vector.
edgesAtt A data-frame with one or more columns representing edges’ attributes.
edgesWeight A numeric vector containing edges’ weights.
edgesVizAtt List of three or less viz attributes such as color, size (thickness) and shape of the
edges (see details)
nodesAtt A data-frame with one or more columns representing nodes’ attributes
nodesVizAtt List of four or less viz attributes such as color, position, size and shape of the
nodes (see details)
nodeDynamic A two-column matrix or data-frame. The first column indicates the time at which
a given node starts; the second one shows when it ends. The matrix or data-frame
must have the same number of rows than the number of nodes in the graph.
edgeDynamic A two-column matrix or data-frame. The fist column indicates the time at which
a given edge starts; the second one shows when it ends. The matrix or data frame
must have the same number of rows than the number of edges in the graph.
digits Integer. Number of decimals to keep for nodes/edges sizes. See print.default
output String. The complete path (including filename) where to export the graph as a
GEXF file.
tFormat String. Time format for dynamic graphs (see details)
defaultedgetype “directed”, “undirected”, “mutual”
meta A List. Meta data describing the graph
keepFactors Logical, whether to handle factors as numeric values (TRUE) or as strings (FALSE)
by using as.character.
encoding Encoding of the graph.
Details

Just like `nodesVizAtt` and `edgesVizAtt`, `nodesAtt` and `edgesAtt` must have the same number of rows as nodes and edges, respectively. Using data frames is necessary as in this way data types are preserved.

`nodesVizAtt` and `edgesVizAtt` allow using visual attributes such as color, position (nodes only), size (nodes only), thickness (edges only) shape and image (nodes only).

- Color is defined by the RGBA color model, thus for every node/edge the color should be specified through a data-frame with columns r (red), g (green), b (blue) with integers between 0 and 256 and a last column with alpha values as a float between 0.0 and 1.0.
- Position, for every node, it is a three-column data-frame including x, y and z coordinates. The three components must be float.
- Size as a numeric colvector (float values).
- Thickness (see size).
- Node Shape (string), currently unsupported by Gephi, can take the values of disk, square, triangle, diamond and image.
- Edge Shape (string), currently unsupported by Gephi, can take the values of solid, dotted, dashed and double.
- Image (string), currently unsupported by Gephi, consists on a vector of strings representing URIs.

`nodesDynamic` and `edgesDynamic` allow to draw dynamic graphs. It should contain two columns `start` and `end`, both allowing NA value. It can be use jointly with `tFormat` which by default is setted as “double”. Currently accepted time formats are:

- Integer or double.
- International standard date yyyy-mm-dd.
- dateTime W3 XSD ([http://www.w3.org/TR/xmlschema-2/#dateTime](http://www.w3.org/TR/xmlschema-2/#dateTime)).

Value

A `gexf` class object (list). Contains the following:

- `meta` : (list) Meta data describing the graph.
- `node` : (list) Sets the default edge type and the graph mode.
- `atts definitions` : (list) Two data-frames describing nodes and edges attributes.
- `nodesVizAtt` : (data-frame) A multi-column data-frame with the nodes’ visual attributes.
- `edgesVizAtt` : (data-frame) A multi-column data-frame with the edges’ visual attributes.
- `nodes` : (data-frame) A two-column data-frame with nodes’ ids and labels.
- `edges` : (data-frame) A five-column data-frame with edges’ ids, labels, sources, targets and weights.
- `graph` : (String) GEXF (XML) representation of the graph.
Author(s)

George Vega Yon <george.vega@nodoschile.org>,
Jorge Fabrega Lacoa <jorge.fabrega@nodoschile.org>

References

The GEXF project website: http://gexf.net

See Also

new.gexf.graph

Examples

```bash
## Not run:
demo(gexf) # Example of gexf command using fictional data.
demo(gexfattributes) # Working with attributes.
demo(gexfbasic) # Basic net.
demo(gexfdynamic) # Dynamic net.
demo(edge.list) # Working with edges lists.
demo(gexffull) # All the package.
demo(gexftwitter) # Example with real data of chilean twitter accounts.
demo(gexfdynamicandatt) # Dynamic net with static attributes.
demo(gexfbuildfromscratch) # Example building a net from scratch.

## End(Not run)
```
Index

*Topic **IO**
  read.gexf, 10
  write.gexf, 14

*Topic **datasets**
  followers, 8
  twitteraccounts, 14

*Topic **manip**
  add.gexf.node, 3
  check.dpl.edges, 5
  edge.list, 7
  igraph.to.gexf, 8
  new.gexf.graph, 9
  switch.edges, 13

*Topic **methods**
  rgexf Methods, 11

*Topic **package**
  rgexf-package, 2

*Topic **utilities**
  checkTimes, 6

  add.edge.spell (add.gexf.node), 3
  add.gexf.edge (add.gexf.node), 3
  add.gexf.node, 3
  add.node.spell (add.gexf.node), 3

  check.dpl.edges, 5
  checkTimes, 6

  edge.list, 7
  export.gexf (rgexf Methods), 11

  followers, 8

  gephi (rgexf-package), 2
  gexf (rgexf-package), 2
  gexf.to.igraph (igraph.to.gexf), 8

  igraph.to.gexf, 8

  layout, 9

new.gexf.graph, 9, 17
plot.gexf (rgexf Methods), 11
print.default, 4, 15
print.gexf (rgexf Methods), 11
read.gexf, 10
rgexf (rgexf-package), 2
rgexf Methods, 11
rgexf-package, 2
rm.gexf.edge (add.gexf.node), 3
rm.gexf.node (add.gexf.node), 3
summary.gexf (rgexf Methods), 11
switch.edges, 13
twitteraccounts, 14
write.gexf, 4, 12, 14