Package ‘rgexf’

August 12, 2021

Type Package

Encoding UTF-8

Title Build, Import and Export GEXF Graph Files

Version 0.16.2

Date 2021-08-10

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' visual 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 'gexf-js' (a 'javascript' library) and interact with the 'igraph' package.

URL https://gvegayon.github.io/rgexf/

BugReports https://github.com/gvegayon/rgexf/issues

Imports XML,
    igr aph,
    grDevices,
    utils,
    servr

License MIT + file LICENSE

LazyLoad yes

RoxygenNote 7.1.1

Suggests knitr,
    rmarkdown,
    tinytest,
    covr

VignetteBuilder knitr

Roxygen list(markdown = TRUE)
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).

Details

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 "gexf-js" (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://github.com/gvegayon/rgexf.

Note

See the GEXF primer for details on the GEXF graph format: https://gephi.org/gexf/1.2draft/gexf-12draft-primer.pdf
References

- rgexf project site: https://github.com/gvegayon/rgexf
- Gephi project site: https://gephi.org/
- GEXF project site: https://gephi.org/gexf/format/
- gexf-js project website: https://github.com/raphv/gexf-js
- Sigmasj project site: http://sigmajs.org/

Examples

```r
if (interactive()) {
  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
}
```

---

**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,
  id = NA,
  label = NA,
  start = NULL,
  end = NULL,
  vizAtt = list(color = NULL, position = NULL, size = NULL, shape = NULL, image = NULL),
  atts = NULL
)
```
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")
)

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

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.
start Starting time period
end Ending time period
vizAtt A list of node/edge viz attributes (see \texttt{write.gexf()}).
atts List of attributes, currently ignored.
source Source node's id.
target Target node's id.
**add.gexf.node**

- **type**: Type of connection (edge).
- **weight**: Edge weight.
- **digits**: Integer. Number of decimals to keep for nodes/edges sizes. See `print.default()`.
- **number**: Index number(s) of a single or a group of nodes or edges.
- **rm.edges**: Whether to remove or not existing edges.

**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`).

**Value**

A `gexf` object (see `write.gexf()`).

**Spells**

While the **start** and **end** attributes can be included in nodes and edges, spells provide a way to represent presence and absence of elements throughout time.

We can use spells to indicate windows during which the element is present or not. For example, a node that shows up from time 1 to time two and re-appears after time four can have two spells:

```
<spell start="1.0" end="2.0">
<spell start="4.0">
```

In the case of the functions **add.edge.spell** and **add.node.spell**, edges and nodes to which you want to add spells should already exist.

**Author(s)**

- George Vega Yon
- Jorge Fabrega Lacoa

**References**

The GEXF project website: https://gephi.org/gexf/format/

**Examples**

```r
if (interactive()) {
  demo(gexfbuildfromscratch)
}

# Creating spells -----------------------------------------------
g <- new.gexf.graph()
```
# Adding a few nodes + edges
```r
g <- add.gexf.node(g, id = 0, label = "A")
g <- add.gexf.node(g, id = 1, label = "B")
g <- add.gexf.node(g, id = 2, label = "C")
```
```r
g <- add.gexf.edge(g, source = 0, target = 1)
g <- add.gexf.edge(g, source = 0, target = 2)
```

# Now we add spells:
# - Node 0: 1.0 -> 2.0, 3.0 -> Inf
# - edge 1: 1.0 -> 2.0, 3.5 -> Inf
```r
g <- add.node.spell(g, 0, start = 1, end = 2)
g <- add.node.spell(g, 0, start = 3)
```
```r
g <- add.edge.spell(g, 1, start = 1, end = 2)
g <- add.edge.spell(g, 1, start = 3.5)
```

```r
g
```

---

**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 de difference)
- **order.edgelist**: Whether 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 an NA at the reps value.

**Value**

A three column data.frame with colnames “source”, “target” “reps”.
Author(s)

George Vega Yon

See Also

Other manipulation: `switch.edges()`

Examples

```r
# An edgelist with duplicated dyads
relations <- cbind(c(1,1,3,3,4,2,5,6), c(2,3,1,1,2,4,1,1))

# Checking duplicated edges (undirected graph)
check.dpl.edges(edges=relations, undirected=TRUE)
```

Description

Checks time

Usage

```r
checkTimes(x, format = "date")
```

Arguments

- `x` A string or vector char
- `format` String, can be “date”, “dateTime”, “float”

Value

Logical.

Author(s)

George Vega Yon

Jorge Fabrega Lacoa

Examples

```r
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
edge.list(x)

Arguments
x  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 auto-generated nodes’ ids.

Value
A list containing two data frames.

Author(s)
George Vega Yon
Jorge Fabrega Lacoa

Examples
```r
edgelist <- matrix(
c("matthew","john",
  "max","stephen",
  "matthew","stephen"),
byrow=TRUE, ncol=2)
edge.list(edgelist)
```
followers

---

**Edge list with attributes**

**Description**
Sample of accounts by December 2011.

**Format**
A data frame containing 6065 observations.

**Source**
Fabrega and Paredes (2012): “La política en 140 caracteres” en Intermedios: medios de comunicación y democracia en Chile. Ediciones UDP

---

gexf-class

---

**Creates an object of class gexf**

**Description**
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**

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),
  nodeDynamic = NULL,
  edgeDynamic = NULL,
  digits =getOption("digits"),
  output = NA,
  tFormat = "double",
  defaultedgetype = "undirected",
  meta = list(creator = "NodosChile", description = "A GEXF file written in R with \"rgexf\"", keywords =

```r
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),  nodeDynamic = NULL,  edgeDynamic = NULL,  digits = getOption("digits"),  output = NA,  tFormat = "double",  defaultedgetype = "undirected",  meta = list(creator = "NodosChile", description = "A GEXF file written in R with \"rgexf\"", keywords =
```
"GEXF, NodosChile, R, rgexf, Gephi"),
keepFactors = FALSE,
encoding = "UTF-8",
vers = "1.3",
rescale.node.size = TRUE
)
write.gexf(nodes, ...)

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
dge. Source and target values are based on the nodes ids.
edgesLabel A one-column data-frame, matrix or vector.
edgesId 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
dges (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
odes (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.
vers Character scalar. Version of the GEXF format to generate. By default "1.3".
rescale.node.size Logical scalar. When TRUE it rescales the size of the vertices such that the largest
one is about \region.
...
Passed to gexf.
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.

nodeDynamic and edgeDynamic allow to draw dynamic graphs. It should contain two columns start and end, both allowing NaN value. It can be use jointly with tFormat which by default is set 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).

NA values in the first column are filled with the min of c(nodeDynamic,edgeDynamic), whereas if in the second column replaces with the max.

More complex time sequences like present/absent nodes and edges can be added with add.node.spell and add.edge.spell respectively.

Value

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

- meta : (list) Meta data describing the graph.
- mode : (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.
- edges : (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.
gexf-methods

Author(s)
George Vega Yon
Jorge Fabrega Lacoa

References
The GEXF project website: https://gephi.org/gexf/format/

See Also
new.gexf.graph()

Examples
if (interactive()) {
  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(gexfrandom)
}

gexf-methods
S3 methods for gexf objects

Description
Methods to print and summarize gexf class objects

Usage
## S3 method for class 'gexf'
print(x, file = NA, replace = F, ...)

## S3 method for class 'gexf'
summary(object, ...)
Arguments

- **x**: An gexf class object.
- **file**: String. Output path where to save the GEXF file.
- **replace**: Logical. If file exists, TRUE would replace the file.
- **...**: Ignored
- **object**: An gexf class object.

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.

Value

- `list("print.gexf")`
  - None (invisible NULL).
- `list("summary.gexf")`
  - List containing some gexf object statistics.

Author(s)

George G. Vega Yon
Joshua B. Kunst

See Also

See also `write.gexf`, `plot.gexf`

Examples

```r
if (interactive()) {
  # 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(4,2,3,4,2,4,1,1))

  # Building gexf graph
  mygraph <- gexf(nodes=people, edges=relations)

  # Summary and print
  summary(mygraph)

  # Summary and print
  summary(mygraph)

  write.gexf(mygraph, output="mygraph.gexf", replace=TRUE)

  # Plotting
}
gexf_js_config

plot(mygraph)
)

---

gexf_js_config  Visualizing GEXF graph files using gexf-js

Description

Using the gexf-js, a JavaScript GEXF viewer, this function allows you to visualize your GEXF on the browser. The function essentially copies a template website, the GEXF file, and sets up a configuration file. By default, the function then starts a webserver using the servr R package.

Usage

gexf_js_config(
  dir,
  graphFile = "network.gexf",
  showEdges = TRUE,
  useLens = FALSE,
  zoomLevel = 0,
  curvedEdges = TRUE,
  edgeWidthFactor = 1,
  minEdgeWidth = 1,
  maxEdgeWidth = 2,
  textDisplayThreshold = 9,
  nodeSizeFactor = 1,
  replaceUrls = TRUE,
  showEdgeWeight = TRUE,
  showEdgeLabel = TRUE,
  sortNodeAttributes = TRUE,
  showId = TRUE,
  showEdgeArrow = TRUE,
  language = FALSE)

  # S3 method for class 'gexf'
  plot(
    x,
    y = NULL,
    graphFile = "network.gexf",
    dir = tempdir(),
    overwrite = TRUE,
    htttd.args = list(),
    copy.only = FALSE,
    ...
  )
Arguments

dir           Directory where the files will be copied (tempdir() by default).
graphFile     Name of the gexf file.
showEdges     Logical scalar. Default state of the "show edges" button (nullable).
useLens       Logical scalar. Default state of the "use lens" button (nullable).
zoomLevel     Numeric scalar. Default zoom level. At zoom = 0, the graph should fill a
              800x700px zone
curvedEdges   Logical scalar. False for curved edges, true for straight edges this setting can’t
              be changed from the User Interface.

edgeWidthFactor
              Numeric scalar. Change this parameter for wider or narrower edges this setting can’t
              be changed from the User Interface.

minEdgeWidth   Numeric scalar.
maxEdgeWidth   Numeric scalar.
textDisplayThreshold
              Numeric scalar.

nodeSizeFactor Numeric scalar. Change this parameter for smaller or larger nodes this setting can’t
              be changed from the User Interface.
replaceUrls    Logical scalar. Enable the replacement of Urls by Hyperlinks this setting can’t
              be changed from the User Interface.
showEdgeWeight Logical scalar. Show the weight of edges in the list this setting can’t be changed
              from the User Interface.
showEdgeLabel  Logical scalar.

sortNodeAttributes
              Logical scalar. Alphabetically sort node attributes.
showId         Logical scalar. Show the id of the node in the list this setting can’t be changed
              from the User Interface.
showEdgeArrow  Logical scalar. Show the edge arrows when the edge is directed this setting can’t
              be changed from the User Interface.

language       Either FALSE, or a character scalar with any of the supported languages.
x              An object of class gexf.
y              Ignored.
overwrite      Logical scalar. When TRUE, the default, the function will overwrite all files
              copied from the template on the destination directory as specified by dir.
httd.args      Further arguments to be passed to servr::httd from the servr package.
copy.only      Logical scalar. When FALSE, the default, the function will make a call to
              servr::httd.
...            Further arguments passed to gexf.js_config
Details

Currently, the only languages supported are: German (de), English (en), French (fr), Spanish (es), Italian (it), Finnish (fi), Turkish (tr), Greek (el), Dutch (nl).

An important thing for the user to consider is the fact that the function only works if there are viz attributes, this is, color, size, and position. If the gexf object’s XML document does not have viz attributes, users can use the following hack:

```r
# Turn the object ot igraph and go back
x <- igraph.to.gexf(gexf.to.igraph(x))

# And you are ready to plot!
plot(x)
```

More details on this in the igraph.to.gexf function.

The files are copied directly from /tmp/Rtmp8GdiTL/Rinst5cce6c533540/rgexf/gexf-js. And the parameters are set up by modifying the following template file:

/tmp/Rtmp8GdiTL/Rinst5cce6c533540/rgexf/gexf-js/config.js.template

The server is lunched if and only if interactive() == TRUE.

References

gexf-js project website https://github.com/raphv/gexf-js.

Examples

```r
if (interactive()) {

  path <- system.file("gexf-graphs/lesmiserables.gexf", package="rgexf")
  graph <- read.gexf(path)
  plot(graph)
}
```

---

#### head.gexf

**head method for gexf objects**

**Description**

List the first n_nodes and n_edges of the gexf file.

**Usage**

```r
## S3 method for class 'gexf'
head(x, n_nodes = 6L, n_edges = n_nodes, ...)
```
igraph.to.gexf

Arguments

x
An object of class gexf.
n_nodes, n_edges
Integers. Number of nodes and edges to print
... Ignored

Examples

fn <- system.file("gexf-graphs/lesmiserables.gexf", package = "rgexf")
g <- read.gexf(fn)
head(g, n_nodes = 5)

igraph.to.gexf

Converting between gexf and igraph classes

Description

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

Usage

igraph.to.gexf(igraph.obj, ...)
gexf.to.igraph(gexf.obj)

Arguments

igraph.obj
An object of class igraph.
... Further arguments passed to gexf().
gexf.obj An object of class gexf.

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 <g.vegayon@gmail.com>

See Also

layout()
Examples

```r
if (interactive()) {
  # Running demo
demo(gexfigraph)
}

fn <- system.file("gexf-graphs/lesmiserables.gexf", package = "rgexf")
gexf1 <- read.gexf(fn)
igraph1 <- gexf.to.igraph(gexf1)
gexf2 <- igraph.to.gexf(igraph1)

if (interactive()) {
  # Now, let's do it with a layout! (although we can just use
  # the one that comes with lesmiserables :))
  pos <- igraph::layout_nicely(igraph1)
  plot(
    igraph.to.gexf(igraph1, nodesVizAtt = list(position=cbind(pos, 0))),
    edgeWidthFactor = .01)
}
```

---

### 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 'rgexf'",
    keywords = "gexf graph, NodosChile, R, rgexf")
)
```

**Arguments**

- `defaultedgetype` "directed", "undirected", "mutual"
- `meta` A List. Meta data describing the graph

**Value**

A gexf object.
read.gexf

Author(s)
George Vega Yon
Jorge Fabrega Lacoa

References
The GEXF project website: https://gephi.org/gexf/format/

Examples
if (interactive()) {
  demo(gexfbuildfromscratch)
}

read.gexf

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.

Author(s)
George Vega Yon
Jorge Fabrega Lacoa

References
The GEXF project website: https://gephi.org/gexf/format/
Examples

```
fn <- system.file("gexf-graphs/lesmiserables.gexf", package = "rgexf")
mygraph <- read.gexf(fn)
```

---

**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**

e**dge.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 auto-generated nodes’ ids.

**Value**

A list containing two data frames.

**Author(s)**

George Vega Yon

**See Also**

Other manipulation: `check.dpl.edges()`

**Examples**

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

switch.edges(relations)
```
**twitteraccounts**

**Description**

Sample of accounts by December 2011.

**Format**

A data frame containing 148 observations.

**Source**

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

* **IO**
  - gexf-class, 9
  - read.gexf, 19
* **datasets**
  - followers, 9
  - twitteraccounts, 21
* **manipulation**
  - check.dpl.edges, 6
  - switch.edges, 20
* **manip**
  - add.gexf.node, 3
  - check.dpl.edges, 6
  - edge.list, 8
  - igraph.to.gexf, 17
  - new.gexf.graph, 18
  - switch.edges, 20
* **methods**
  - gexf-methods, 12
* **package**
  - rgexf-package, 2
* **utilities**
  - checkTimes, 7
  - add.edge.spell, 11
  - add.edge.spell (add.gexf.node), 3
  - add.gexf.edge (add.gexf.node), 3
  - add.gexf.node, 3
  - add.node.spell, 11
  - add.node.spell (add.gexf.node), 3
  - check.dpl.edges, 6, 20
  - checkTimes, 7
  - edge.list, 8
  - export-gexf (gexf-methods), 12
  - followers, 9
  - gephi (rgexf-package), 2
  - gexf, 16, 17
  - gexf (gexf-class), 9
  - gexf(), 17
  - gexf-class, 9
  - gexf-methods, 12
  - igraph.to.gexf (igraph.to.gexf), 17
  - gexf Js config, 14
  - head.gexf, 16
  - igraph.to.gexf, 16, 17
  - layout(), 17
  - new.gexf.graph, 18
  - new.gexf.graph(), 12
  - plot.gexf, 13
  - plot.gexf (gexf Js config), 14
  - print.default(), 5, 10
  - print.gexf (gexf-methods), 12
  - read.gexf, 19
  - rgexf (rgexf-package), 2
  - rgexf-package, 2
  - rm.gexf.edge (add.gexf.node), 3
  - rm.gexf.node (add.gexf.node), 3
  - servr::httd, 15
  - summary.gexf (gexf-methods), 12
  - switch.edges, 7, 20
  - twitteraccounts, 21
  - write.gexf, 13
  - write.gexf (gexf-class), 9
  - write.gexf(), 4, 5