Package ‘rgraph6’

March 9, 2022

Version 2.0-1

Title Representing Graphs as 'graph6', 'digraph6' or 'sparse6' Strings

Depends R (>= 2.10)

Imports methods, Rcpp (>= 1.0.3)

LinkingTo Rcpp

Suggests testthat, knitr, rmarkdown, igraph, network, Matrix

Description Encode network data as strings of printable ASCII characters. Implemented functions include encoding and decoding adjacency matrices, edgelists, igraph, and network objects to/from formats 'graph6', 'sparse6', and 'digraph6'. The formats and methods are described in McKay, B.D. and Piperno, A (2014) <doi:10.1016/j.jsc.2013.09.003>.

License GPL (>= 3)

RoxygenNote 7.1.2

Encoding UTF-8

LazyData true

URL https://mbojan.github.io/rgraph6/

BugReports https://github.com/mbojan/rgraph6/issues

VignetteBuilder knitr

NeedsCompilation yes

Author Michal Bojanowski [aut, cre] (<https://orcid.org/0000-0001-7503-852X>), Kozminski University), David Schoch [aut] (<https://orcid.org/0000-0003-2952-4812>)

Maintainer Michal Bojanowski <michal2992@gmail.com>

Repository CRAN

Date/Publication 2022-03-09 08:00:02 UTC
**Description**

Encode network data as strings of printable ASCII characters. Implemented functions include encoding and decoding adjacency matrices, edgelists, igraph, and network objects to/from formats 'graph6', 'sparse6', and 'digraph6'. The formats and methods are described in McKay, B.D. and Piperno, A (2014) <doi:10.1016/j.jsc.2013.09.003>.

**Details**

Formats 'graph6', 'sparse6' and 'digraph6' represent graphs as strings of printable ASCII characters. The formats are due to Brendan McKay who implemented them in his program Nauty (McKay 1978, 1980, 2003, McKay & Piperno 2014), and are described in detail here. Package *rgraph6* is a native R implementation of these formats.

The main functions are `as_graph6()`, `as_digraph6()`, `as_sparse6()` for encoding network data and `igraph_from_text()` and `network_from_text()` for decoding. There are also other low-level functions to decode directly from `digraph6`, `graph6`, and `sparse6`.

**Citation**

When using this package please cite it by referring to the following publications: McKay B, Piperno A (2014). "Practical graph isomorphism, II." *Journal of Symbolic Computation, 60*, 94-112.

Call citation(package="rgraph6") for more details and the BibTeX entry.

Author(s)

Maintainer: Michal Bojanowski <michal2992@gmail.com> (ORCID) (Kozminski University)
Authors:
• David Schoch <david@schochastics.net> (ORCID)

References

McKay, B. D., & Piperno, A. (2013). Nauty and Traces user's guide (Version 2.5). Computer Science Department, Australian National University, Canberra, Australia.

adjacency_from_text

Create adjacency matrices from 'graph6', 'sparse6', or 'digraph6' symbols

Description

Create adjacency matrices from 'graph6', 'sparse6', or 'digraph6' symbols

Usage

adjacency_from_text(object, ...)

Arguments

object character vector of 'graph6', 'sparse6', or 'digraph6' symbols

... other arguments, currently ignored

Details

If object contains 'sparse6' symbols, which are in fact encoded edgelists, the function will return corresponding adjacency matrices creating temporary igraph objects internally.

Value

A list of adjacency matrices.
Examples

# Graph6 symbols
sampleg6
adjacency_from_text(sampleg6)

# Sparse6 symbols
s6 <- c(":DgXI@G~", ":DgWCGb")
adjacency_from_text(s6)

# Digraph6 symbol
d6 <- "&N????C???D???G??C??????C????????C??Q@O?G?"
adjacency_from_text(d6)

Description

Generic function encoding directed networks as 'digraph6' symbol(s). See below for available methods.

Usage

as_digraph6(object)

## S3 method for class 'matrix'
as_digraph6(object)

## S3 method for class 'igraph'
as_digraph6(object)

## S3 method for class 'network'
as_digraph6(object)

## S3 method for class 'list'
as_digraph6(object)

## Default S3 method:
as_digraph6(object)

Arguments

object a matrix, an igraph object or a network object or a list thereof. See Methods section below.
as_graph6

Details

The 'digraph6' format is designed for directed graphs. Error is thrown in case it is given an undirected network.

Value

A character vector of 'digraph6' symbols.

Methods (by class)

- matrix: Expects object to be a square matrix which is interpreted as an adjacency matrix of a directed graph.
- igraph: Igraph object needs to be a directed graph. Requires igraph package.
- network: Network object needs to be directed network. Requires network package.
- list: If object is a list the function is applied to each element. Consequently, it can be a list with a mixture of supported objects classes (adjacency matrices, igraph, or network objects).
- default: Throws an error about the unhandled class.

Examples

# From adjacency matrix ----------------------------------------------------
am <- matrix(c(
  0,1,0,
  0,0,1,
  1,0,0),
  byrow=TRUE, ncol=3, nrow=3)
as_digraph6(am)

# From igraph objects ------------------------------------------------------
if(requireNamespace("igraph", quietly=TRUE)) {
  g <- igraph::graph_from_adjacency_matrix(am)
as_digraph6(g)
}

# From network objects -----------------------------------------------------
if(requireNamespace("network", quietly=TRUE)) {
  net <- network::network(am)
as_digraph6(net)
}
Usage

as_graph6(object)

## S3 method for class 'matrix'
as_graph6(object)

## S3 method for class 'igraph'
as_graph6(object)

## S3 method for class 'network'
as_graph6(object)

## S3 method for class 'list'
as_graph6(object)

## Default S3 method:
as_graph6(object)

Arguments

object a matrix, an igraph object or a network object or a list thereof. See Methods section below.

Details

The 'graph6' format is designed for undirected graphs. Error is thrown in case it is given a directed graph.

Value

A character vector of 'graph6' symbols.

Methods (by class)

- matrix: Expects object to be a square matrix which is interpreted as an adjacency matrix of an undirected graph. The function reads only the upper triangle of the matrix and there is no test whether the matrix is symmetric.
- igraph: Igraph object needs to be an undirected graph. Requires igraph package.
- network: Network object needs to be a directed network. Requires network package.
- list: If object is a list the function is applied to each element. Consequently, it can be a list with a mixture of supported objects classes (adjacency matrices, igraph, or network objects).
- default: The default method throws an error about an unhandled class.

Examples

# From adjacency matrix
am <- matrix(c(0,1,0,1,1,0), nrow = 2, byrow = TRUE)

as_sparse6

1,0,0,
1,0,0
), byrow=TRUE, ncol=3)

as_graph6(am)

# From igraph objects -----------------------------------------------
if(requireNamespace("igraph", quietly=TRUE)) {
  g <- igraph::graph_from_adjacency_matrix(am, mode = "undirected")
  as_graph6(g)
}

# From network objects ------------------------------------------------
if(requireNamespace("network", quietly=TRUE)) {
  net <- network::network(am, directed=FALSE)
  as_graph6(net)
}

as_sparse6

Encode network data as 'sparse6' symbols

Description

Generic function encoding network data as 'sparse6' symbol(s). See below for available methods.

Usage

as_sparse6(object, ...)

## S3 method for class 'matrix'
as_sparse6(object, n = max(object, 0), ...)

## S3 method for class 'igraph'
as_sparse6(object, ...)

## S3 method for class 'network'
as_sparse6(object, ...)

## S3 method for class 'list'
as_sparse6(object, ...)

## Default S3 method:
as_sparse6(object, ...)

Arguments

object an edgelist, igraph, or network object or a list thereof. See Methods section below.
other arguments passed to/from other methods

n

number of vertices in the graph

Value

A character vector of 'sparse6' symbols.

Methods (by class)

• matrix: Expects object to be a two-column matrix of integers which is interpreted as an edgelist of an undirected graph. By default the network size is inferred to be the maximal element of object. This can be overridden by providing the network size via the n argument, the results will not be identical though (see the Examples).

• igraph: Igraph object needs to be an undirected graph. Requires igraph package.

• network: Network object needs to be a directed network. Requires network package.

• list: If object is a list the function is applied to each element. Consequently, it can be a list with a mixture of supported objects classes (edgelist matrices, igraph, or network objects).

• default: The default method fails gracefully.

See Also

The 'sparse6' format is designed for undirected graphs. Error is thrown in case it is given a directed graph.

Examples

# From edgelist matrix -----------------------------------------------
elm <- matrix(c(1, 2, 2, 3, 3, 4), ncol=2, byrow=TRUE)
as_sparse6(elm) # 1--2, 2--3, 3--4
as_sparse6(elm + 6) # 1, 2, 3, 4, 5, 6, 7--8, 8--9, 9--10
as_sparse6(elm, n = 10) # 1--2, 2--3, 3--4, 5, 6, 7, 8, 9, 10

# From igraph objects -----------------------------------------------
if(requireNamespace("igraph")) {
  g <- igraph::graph_from_edgelist(elm, directed=FALSE)
as_sparse6(g)
}

# From network objects -----------------------------------------------
if(requireNamespace("network")) {
  net <- network::network(elm, directed=FALSE)
as_graph6(net)
}
choose_format

Choose most efficient format heuristically

Description

Given a graph suggest the most efficient format out of 'graph6', 'sparse6' or 'digraph6'.

Usage

choose_format(object, ...)

## Default S3 method:
choose_format(object, ...)

## S3 method for class 'list'
choose_format(object, ...)

Arguments

object Igraph/network object or list thereof
...
other arguments, currently ignored

Details

If object is directed, the suggested format is 'digraph6'. If object is undirected the function suggests 'sparse6' if density is less than 0.15 and 'graph6' otherwise. This rule is approximate.

Value

Character value out of 'graph6', 'sparse6' or 'digraph6'. If object is a list, a vector of such values of the length equal to the length of object.

Examples

# From igraph ------------------------------------------------------
if(requireNamespace("igraph")) {
  g <- igraph::graph.famous("Zachary")
  choose_format(g)

  set.seed(123)
  glist <- list(
    igraph::sample_gnp(n = 15, p = 0.1),
    igraph::sample_gnp(n = 15, p = 0.2),
    igraph::sample_gnp(n = 15, p = 0.3),
    igraph::sample_gnp(n = 15, p = 0.15, directed = TRUE))

  choose_format(glist)
}

# From network -----------------------------------------------------
if(requireNamespace("network")) {
  m <- matrix(rbinom(25,1,.4),15,15)
  diag(m) <- 0
  g <- network::network(m, directed=FALSE)
  choose_format(g)
}

edgelist_from_text

Create edgelist matrices from 'graph6', 'sparse6', or 'digraph6' symbols

Description
Create edgelist matrices from 'graph6', 'sparse6', or 'digraph6' symbols

Usage
edgelist_from_text(object, ...)

Arguments
object character vector of 'graph6', 'sparse6', or 'digraph6' symbols
... other arguments, currently ignored

Details
If object contains 'graph6' or 'digraph6' symbols, which are in fact encoded adjacency matrices, the function will return corresponding edgelist matrices creating temporary igraph objects internally.

Value
A list of edgelist matrices.

Examples
# Graph6 symbols
sampleg6
edgelist_from_text(sampleg6)

# Sparse6 symbols
s6 <- c(":DgXI@G~", ":DgWCgCb")
edgelist_from_text(s6)

# Digraph6 symbol
d6 <- ":&N????C??D?_G??C?????_?C_??????C??Q@O?G?"
edgelist_from_text(d6)
Description

These functions take a vector of ‘digraph6’ symbols and return a list of other types of objects:

• `adjacency_from_digraph6()` creates adjacency matrices

• `igraph_from_digraph6()` creates ‘igraph’ objects. Requires package `igraph` to be installed.

• `network_from_digraph6()` creates ‘network’ objects. Requires package `network` to be installed.

Usage

adjacency_from_digraph6(d6)

igraph_from_digraph6(d6, ...)

network_from_digraph6(d6, ...)

Arguments

d6 character vector of ‘digraph6’ symbols

... other arguments, see Details.

Details

For `igraph_from_digraph6()` additional arguments are passed to `igraph::graph_from_adjacency_matrix()`.
For `network_from_digraph6()` additional arguments are passed to `network::as.network()`.

Value

The returned object is:

• for `adjacency_from_digraph6()`, a list of the same length as its input of square symmetric adjacency matrices.

• for `igraph_from_digraph6()`, a list of ‘igraph’ objects

• for `network_from_digraph6()`, a list of ‘network’ objects

See Also

`as_digraph6()` for encoding objects as ‘digraph6’ symbols.
Examples

```r
am <- matrix(rbinom(16, 1, 0.3), 4, 4)
d6 <- as_digraph6(am)

# To adjacency matrix -----------------------------------------------
adjacency_from_digraph6(d6)

# To igraph objects -------------------------------------------------
if(requireNamespace("igraph", quietly=TRUE)) {
  igraph_from_digraph6(d6)
}

# To network objects ------------------------------------------------
if(requireNamespace("network", quietly=TRUE)) {
  network_from_digraph6(d6)
}
```

Description

These functions take a vector of 'graph6' symbols and return a list of other types of objects:

- `adjacency_from_graph6()` creates adjacency matrices
- `igraph_from_graph6()` creates 'igraph' objects. Requires package `igraph` to be installed.
- `network_from_graph6()` creates network objects. Requires package `network` to be installed.

Usage

```r
adjacency_from_graph6(g6)
igraph_from_graph6(g6, ...)
network_from_graph6(g6, ...)
```

Arguments

- `g6` character vector of 'graph6' symbols
- `...` other arguments, see Details.

Details

For `igraph_from_graph6()` additional arguments are passed to `igraph::graph_from_adjacency_matrix()`
For `network_from_graph6()` additional arguments are passed to `network::as.network()`
Value

The returned object is:

- for `adjacency_from_graph6()`, a list of the same length as its input of square symmetric adjacency matrices.

- for `igraph_from_graph6()`, a list of 'igraph' objects

- for `network_from_graph6()`, a list of network objects

See Also

`as_graph6()` for saving objects as 'graph6' symbols.

Examples

```r
A <- matrix(c(0,1,0,1,
1,0,1,0,
0,1,0,1,
1,0,1,0), 4, 4, byrow = TRUE)
g6 <- as_graph6(A)

# To adjacency matrix ------------------------------------------------------
adjacency_from_graph6(g6)

# To igraph objects --------------------------------------------------------
if(requireNamespace("igraph", quietly=TRUE)) {
  igraph_from_graph6(g6)
}

# To network objects -------------------------------------------------------
if(requireNamespace("network", quietly=TRUE)) {
  network_from_graph6(g6)
}
```

Description

These functions take a character vector of 'sparse6' symbols and return a list of other types of objects:

- `edgelist_from_sparse6()` creates edgelist matrices

- `igraph_from_sparse6()` creates 'igraph' objects. Requires package `igraph` to be installed.

- `network_from_sparse6()` creates 'network' objects. Requires package `network` to be installed.
from_sparse6

Usage

edgelist_from_sparse6(s6)

igraph_from_sparse6(s6)

network_from_sparse6(s6)

Arguments

s6 character vector of 'sparse6' symbols

Value

The returned object is:

- for `edgelist_from_sparse6()`, a list of the same length as its input of two-column edgelist matrices. The matrix has a gorder attribute storing the number of vertices in the graph.

- for `igraph_from_sparse6()`, a list of 'igraph' objects

- for `network_from_sparse6()`, a list of 'network' objects

See Also

`as_sparse6()` for encoding network data objects as 'sparse6' symbols.

Examples

```r
elm <- structure(c(1, 1, 2, 2, 4, 4, 5, 6, 9, 10, 7, 8, 4, 8, 6, 8,
                  8, 5, 4, 6), .Dim = c(10L, 2L))
s6 <- as_sparse6(elm, n = 10)

# To edgelist matrix -------------------------------------------------------
edgelist_from_sparse6(s6)

# To igraph object ---------------------------------------------------------
if(requireNamespace("igraph", quietly=TRUE)) {
  igraph_from_sparse6(s6)
}

# To network object --------------------------------------------------------
if(requireNamespace("network", quietly=TRUE)) {
  network_from_sparse6(s6)
}
```
**graph_as_text**

---

**Encode graph as text**

---

**Description**

Encode a graph as 'graph6', 'sparse6' or 'digraph6' choosing the format automatically.

**Usage**

```r
graph_as_text(object, ...)
```

## Default S3 method:
```r
graph_as_text(object, ...)
```

## S3 method for class 'list'
```r
graph_as_text(object, ...)
```

**Arguments**

- `object`: igraph/network object or a list thereof
- `...`: other arguments, currently ignored

**Details**

If `object` is a list it may be a mixture of 'network' and 'igraph' objects.

**Value**

A character vector of encoded graphs.

**Methods (by class)**

- `default`: The default method chooses the encoding format automatically using `choose_format()`.
- `list`: The list method applies the default method to each element.

**See Also**

`choose_format()`

**Examples**

# From igraph -------------------------------------------
```r
if(requireNamespace("igraph")) {
  g <- igraph::graph.famous("Zachary")
  graph_as_text(g)

  glist <- list(
    igraph::sample_gnp(n = 15, p = 0.1),
```
igraph::sample_gnp(n = 15, p = 0.2),
igraph::sample_gnp(n = 15, p = 0.3))

graph_as_text(glist)
}

# From network -----------------------------------------------
if(requireNamespace("network")) {
  m <- matrix(rbinom(25, 1, .4), 5, 5)
  diag(m) <- 0
  g <- network::network(m, directed=FALSE)
  graph_as_text(g)
}

---

igraph_from_text

Create igraph objects from 'graph6', 'sparse6', or 'digraph6' symbols

Description
Create igraph objects from 'graph6', 'sparse6', or 'digraph6' symbols

Usage
igraph_from_text(object)

Arguments
object character vector of 'graph6', 'sparse6', or 'digraph6' symbols

Value
A list of 'igraph' objects.

Examples
if(requireNamespace("igraph", quietly=TRUE)) {
  # Graph6 symbols
  sampleg6
  igraph_from_text(sampleg6)

  # Sparse6 symbols
  s6 <- c("DgXI0G-", "DgWCgCb")
  igraph_from_text(s6)

  # Digraph6 symbol
  d6 <- "&N????_C??D??_G??C??????_C_??????C??Q@O?G?"
  igraph_from_text(d6)
}
is_graph6

Infer or test for graph6, sparse6, and digraph6 symbols

Description

Functions `is_graph6()`, `is_sparse6()`, and `is_digraph6()` test if elements of a character vector are valid symbols of particular type.

Function `guess_format()` tries to guess the type of the symbols used in `x`.

Usage

```r
is_graph6(x)

is_sparse6(x)

is_digraph6(x)

guess_format(x)
```

Arguments

- `x` character vector

Value

Logical vector of length equal to `length(x)` with `TRUE` if an element is a valid symbol and `FALSE` otherwise.

Function `guess_format()` returns a character vector of the same length as `x` with values "graph6", "sparse6", or "digraph6" depending on the type of symbol present, or `NA` if the symbol is unknown or matches more than one type.

Note

At this moment the test is performed using regular expressions. Theoretically it may result in false positives.

Examples

```r
all(is_graph6(g6))
all(is_sparse6(s6))
all(is_digraph6(d6))

# Vector mixing graphs in various formats
x <- g6
x[seq(2, 20, by = 3)] <- s6[seq(2, 20, by = 3)]
x[seq(3, 20, by = 3)] <- d6[seq(3, 20, by = 3)]
guess_format(x)
```
network_from_text  
Create network objects from 'graph6', 'sparse6', or 'digraph6' symbols

**Description**
Create network objects from 'graph6', 'sparse6', or 'digraph6' symbols

**Usage**
```r
network_from_text(object)
```

**Arguments**
- **object**: character vector of 'graph6', 'sparse6', or 'digraph6' symbols

**Value**
A list of 'network' objects.

**Examples**
```r
# complete graph in graph6 format
g6 <- "G----("

# random graph with 15 nodes
s6 <- ":NeF?bsl?aNC"

# random directed graph with 10 nodes
d6 <- "&I???GGGI?_gG??O???"

network_from_text(g6)
network_from_text(c(g6,s6,d6))
```

---

read_file6  
Read files of 'graph6', 'sparse6' or 'digraph6' symbols

**Description**
Read files of 'graph6', 'sparse6' or 'digraph6' symbols

**Usage**
```r
read_file6(path, type = "adjacency")
```
Arguments

path character; path to file name

type character; one of "adjacency", "edgelist", "igraph", or "network". Type of result returned.

Details

File pointed to by path is a text file with one graph symbol per line. Optional headers of the form \texttt{>>graph6<<} or \texttt{>>sparse6<<} in the first line (and without the newline after the header) are ignored and removed.

Value

A list of decoded graphs in the form of objects determined by type.

Examples

```r
  g6_file <- tempfile()
  write(sampleg6,g6_file)
  read_file6(g6_file, type = "adjacency")
  unlink(g6_file)
```

Description

Objects \texttt{g6}, \texttt{s6}, and \texttt{d6} are vectors of codes in 'graph6', 'sparse6', and 'digraph6' representations respectively. Object \texttt{sampleg6} is a vector of 'graph6' codes.

Usage

\texttt{g6}

\texttt{s6}

\texttt{d6}

\texttt{sampleg6}

Format

The three objects \texttt{g6}, \texttt{s6}, and \texttt{d6} are character vectors of length 20 corresponding to undirected (in case of \texttt{g6} and \texttt{s6}) and directed (in case of \texttt{d6}) graphs of varying sizes and densities.

Object \texttt{sampleg6} is a character vector of length 9 of undirected graphs in 'graph6' format.
Details

Graphs in g6, s6, and d6 objects were generated using the common algorithm which consists of the following steps:

1. For each value from the vector of sizes of the node set (15, 30, 60, 120)...
2. ... generate a vector of edge counts (size of the edge set) of length 5 ranging from a single edge up to an edge count corresponding to the density of 0.2.
3. Given the node set sizes (item 1) and edge set sizes (item 2) sample undirected graphs from GNM model.
4. These undirected graphs are encoded in g6 and s6
5. Directed graphs were created by turning undirected edges to directed arcs in an arbitrary manner. These are encoded in the d6 object.
Index

* datasets
  sampleg6, 19
* package
  rgraph6-package, 2
  _PACKAGE (rgraph6-package), 2
adjacency_from_digraph6
  (from_digraph6), 11
adjacency_from_digraph6(), 11
adjacency_from_graph6 (from_graph6), 12
adjacency_from_graph6(), 12, 13
adjacency_from_text, 3
as_digraph6, 4
as_digraph6(), 2, 11
as_graph6, 5
as_graph6(), 2, 13
as_sparse6, 7
as_sparse6(), 2, 14
choose_format, 9
choose_format(), 15
d6 (sampleg6), 19
digraph6, 2
edgelist_from_sparse6 (from_sparse6), 13
edgelist_from_sparse6(), 13, 14
edgelist_from_text, 10
from_digraph6, 11
from_graph6, 12
from_sparse6, 13
g6 (sampleg6), 19
graph6, 2
graph_as_text, 15
guess_format (is_graph6), 17
guess_format(), 17
igraph::graph_from_adjacency_matrix(),
  11, 12
igraph_from_digraph6 (from_digraph6), 11
igraph_from_digraph6(), 11
igraph_from_graph6 (from_graph6), 12
igraph_from_graph6(), 12, 13
igraph_from_sparse6 (from_sparse6), 13
igraph_from_sparse6(), 13, 14
igraph_from_text, 16
igraph_from_text(), 2
is_digraph6 (is_graph6), 17
is_digraph6(), 17
is_graph6, 17
is_graph6(), 17
is_sparse6 (is_graph6), 17
is_sparse6(), 17
network::as.network(), 11, 12
network_from_digraph6 (from_digraph6),
  11
network_from_digraph6(), 11
network_from_graph6 (from_graph6), 12
network_from_graph6(), 12, 13
network_from_sparse6 (from_sparse6), 13
network_from_sparse6(), 13, 14
network_from_text, 18
network_from_text(), 2
read_file6, 18
rgraph6 (rgraph6-package), 2
rgraph6-package, 2
s6 (sampleg6), 19
sampleg6, 19
sparse6, 2