## Package ‘FinNet’

August 10, 2023

**Type**  Package

**Title**  Quickly Build and Manipulate Financial Networks

**Version**  0.1.2

**Maintainer**  Fabio Ashtar Telarico <Fabio-Ashtar.Telarico@fdv.uni-lj.si>

**Description**  Providing classes, methods, and functions to deal with financial networks. Users can easily store information about both physical and legal persons by using pre-made classes that are studied for integration with scraping packages such as ‘rvest’ and ‘RSelectium’. Moreover, the package assists in creating various types of financial networks depending on the type of relation between its units depending on the relation under scrutiny (ownership, board interlocks, etc.), the desired tie type (valued or binary), and renders them in the most common formats (adjacency matrix, incidence matrix, edge list, ‘igraph’, ‘network’).

**License**  GPL (>= 3)

**URL**  https://fatelarico.github.io/FinNet.html

**BugReports**  https://github.com/FATelarico/FinNet/issues

**Encoding**  UTF-8

**RoxygenNote**  7.2.3

**Depends**  R (>= 2.10)

**Author**  Fabio Ashtar Telarico [aut, cre]

(<https://orcid.org/0000-0002-8740-7078>)

**Imports**  Matrix, grDevices, methods

**Suggests**  knitr, igraph, network, markdown, SPB, yahoofinance

**LazyData**  no

**VignetteBuilder**  knitr

**NeedsCompilation**  no

**Repository**  CRAN

**Date/Publication**  2023-08-10 06:50:20 UTC
## R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>as.firm</td>
<td>3</td>
</tr>
<tr>
<td>as.firm,financial_matrix-method</td>
<td>3</td>
</tr>
<tr>
<td>FF</td>
<td>4</td>
</tr>
<tr>
<td>FF-basic-methods</td>
<td>5</td>
</tr>
<tr>
<td>FF-comparison-methods</td>
<td>6</td>
</tr>
<tr>
<td>FF-math-methods</td>
<td>7</td>
</tr>
<tr>
<td>FF-nrow-ncol</td>
<td>8</td>
</tr>
<tr>
<td>FF-subset-method</td>
<td>8</td>
</tr>
<tr>
<td>FF.binary.both</td>
<td>9</td>
</tr>
<tr>
<td>FF.binary.management</td>
<td>10</td>
</tr>
<tr>
<td>FF.binary.ownership</td>
<td>11</td>
</tr>
<tr>
<td>FF.graph</td>
<td>12</td>
</tr>
<tr>
<td>FF.graph.custom</td>
<td>13</td>
</tr>
<tr>
<td>FF.naive.both</td>
<td>16</td>
</tr>
<tr>
<td>FF.naive.management</td>
<td>18</td>
</tr>
<tr>
<td>FF.naive.ownership</td>
<td>19</td>
</tr>
<tr>
<td>FF.net</td>
<td>20</td>
</tr>
<tr>
<td>FF.net.custom</td>
<td>21</td>
</tr>
<tr>
<td>FF.norm.both</td>
<td>24</td>
</tr>
<tr>
<td>FF.norm.management</td>
<td>26</td>
</tr>
<tr>
<td>FF.norm.ownership</td>
<td>27</td>
</tr>
<tr>
<td>find.firm</td>
<td>28</td>
</tr>
<tr>
<td>find.firms</td>
<td>30</td>
</tr>
<tr>
<td>find.people</td>
<td>32</td>
</tr>
<tr>
<td>firms_BKB</td>
<td>33</td>
</tr>
<tr>
<td>firms_US</td>
<td>33</td>
</tr>
<tr>
<td>FM</td>
<td>34</td>
</tr>
<tr>
<td>FO.binary</td>
<td>35</td>
</tr>
<tr>
<td>FO.naive</td>
<td>36</td>
</tr>
<tr>
<td>FO.norm</td>
<td>37</td>
</tr>
<tr>
<td>graph_methods</td>
<td>38</td>
</tr>
<tr>
<td>igraph_E_iterators</td>
<td>39</td>
</tr>
<tr>
<td>igraph_financial</td>
<td>40</td>
</tr>
<tr>
<td>igraph_v_iterators</td>
<td>40</td>
</tr>
<tr>
<td>network_financial</td>
<td>41</td>
</tr>
<tr>
<td>plot_igraph-methods</td>
<td>41</td>
</tr>
<tr>
<td>print.firm</td>
<td>42</td>
</tr>
<tr>
<td>query.firm</td>
<td>43</td>
</tr>
<tr>
<td>query.firms</td>
<td>44</td>
</tr>
<tr>
<td>query.firms.dataframe</td>
<td>46</td>
</tr>
<tr>
<td>register.firm</td>
<td>47</td>
</tr>
</tbody>
</table>

**Index**

50
**as.firm**  

*Coerce into (a list of) firm object(s)*

---

**Description**

Generic function to coerce other other classes into the S4 class firm representing a firm (legal person)

**Usage**

```r
as.firm(x, ...)
```

**Arguments**

- `x`  
  The object to coerce
- `...`  
  Arguments passed to class-specific methods

**Value**

An object of class firm or a (usually named) list of them, depending on the available method for the object being coerced.

**Author(s)**

Telarico, Fabio Ashtar

---

**as.firm,financial_matrix-method**  

*Coerce a financial_matrix object into a list of firm objects*

---

**Description**

`as.firm` method for an object of class financial_matrix

**Usage**

```r
## S4 method for signature 'financial_matrix'
as.firm(x, ...)
```

**Arguments**

- `x`  
  The financial_matrix object to coerce
- `...`  
  Optional arguments
Value

A (usually named) list of firm objects the length of which equals the number of rows and columns of the provided financial_matrix

Author(s)

Telarico, Fabio Ashtar

---

**FF**

*Create any firm-firm (FF) matrix*

**Description**

General function to create a firm-firm (FF) matrix

**Usage**

```r
FF(..., who, ties, id_as_firm_name = NULL, Matrix = NULL, self_ties = FALSE)
```

**Arguments**

- `...`
  - Either multiple objects of class firm or a list of such objects
- `who`
  - Whether to take into account: (ownership) co-ownership; (management) board interlocks, or both (recognises minimum unambiguous strings).
- `ties`
  - Type of ties to create. Possible values: binary; naive; share (see Details).
- `id_as_firm_name`
  - Whether to use the ticker as the firm’s name. Defaults to TRUE if all firms’ id is neither NULL nor NA.
- `Matrix`
  - Whether to use the `Matrix` package. Defaults to TRUE when any matrix in the pipeline contains more than 10,000 cells and the package is installed.
- `self_ties`
  - Whether to allow self-ties (a 'loop' in graph theory). Defaults to FALSE.

**Details**

See more specific functions for a detailed overview:

for board interlocks (who == 'management'):

- `FF.binary.management`, if ties = 'binary';
- `FF.binary.management`, if ties = 'naive';
- `FF.norm.management`, if ties = 'share'.

for co-ownership (who == 'ownership'):

- `FF.binary.ownership`, if ties = 'binary';
- `FF.naive.ownership`, if ties = 'naive';
• \texttt{FF.norm.ownership}, if ties = 'share'.

for both co-ownership and board interlocks (who == 'both'):

• \texttt{FF.binary.both}, if ties = 'binary';
• \texttt{FF.naive.both}, if ties = 'naive';
• \texttt{FF.norm.both}, if ties = 'share'.

Value

A matrix object of class \texttt{financial_matrix}(possibly using the \texttt{Matrix} package)

Author(s)

Telarico, Fabio Ashtar

See Also

\texttt{FF.binary.ownership} \texttt{FF.binary.management} \texttt{FF.naive.ownership} \texttt{FF.naive.management} \texttt{FF.norm.ownership} \texttt{FF.norm.management}

Examples

```r
# Create the normalised FF matrix of Berkshire Hathaway's holdings by boards interlocks
data('firms_BKB')
FF <- FF(firms_BKB, who = 'man', ties = 'share')
```

---

**FF-basic-methods**  
| Basic methods for objects of class \texttt{financial_matrix} |

Description

Basic methods for objects of class \texttt{financial_matrix}

Usage

```r
## S4 method for signature 'financial_matrix'
rownames(x, do.NULL = TRUE, prefix = "row")
```

```r
## S4 method for signature 'financial_matrix'
colnames(x, do.NULL = TRUE, prefix = "row")
```

Arguments

\begin{itemize}
  \item \texttt{x} \quad The \texttt{financial_matrix} object to operate on
  \item \texttt{do.NULL} \quad Whether to use NULL names. Defaults to FALSE
  \item \texttt{prefix} \quad Prefix for created names (if \texttt{do.NULL} is FALSE and names are NULL)
\end{itemize}
Details

Mind that usually the rows and columns are named after the firm’s tickers.

Value

A character vector of length equal to the number of rows (or columns) in the \texttt{financial\_matrix} corresponding to the names of the rows (or columns)

Author(s)

Telarico, Fabio Ashtar

---

**Basic method to check to compare values in a financial\_matrix object**

Description

Basic method to check to compare values in a \texttt{financial\_matrix} object

Usage

```r
## S4 method for signature 'financial\_matrix,logical'
duplicated(x, incomparables = FALSE, ...)
```

```r
## S4 method for signature 'financial\_matrix,logical'
unique(x, incomparables = FALSE, ...)
```

Arguments

- \texttt{x}
  - The \texttt{financial\_matrix} object to operate on
- \texttt{incomparables}
  - Either:
    - a vector of values that cannot be compared
    - or \texttt{FALSE}, in which case all values can be compared
- \texttt{...}
  - Arguments passed to the relevant matrix method

Value

- \texttt{duplicated}: A logical array with the same dimensions and \texttt{dimnames} of the \texttt{financial\_matrix}'s matrix component.
- \texttt{unique}: The matrix component is coerced into a vector and then returned, but with only one copy of each duplicated element.

Author(s)

Telarico, Fabio Ashtar
Description

isSymmetric checks only the matrix-like part summary operates on all numeric attributes and the matrix-like part

Usage

```r
## S4 method for signature 'financial_matrix'
isSymmetric(object, ...)
## S4 method for signature 'financial_matrix'
summary(object, ...)
```

Arguments

- `object`: The `financial_matrix` object to operate on
- `...`: Arguments passed to the relevant matrix method

Details

Mathematical methods for `financial_matrix` objects

Value

- `isSymmetric`: a boolean, TRUE if the matrix is symmetric, FALSE otherwise;
- `summary`: a list of length equal to the number of numeric attributes possessed by the `financial_matrix` (maximum three, the matrix itself, revenues, and capitalisation) assumed as measured on the same scale and denominated in the same currency). Each element of the list of class `c('summaryDefault', 'table')` which has specialized `format` and `print` methods

Author(s)

Telarico, Fabio Ashtar
**FF-nrow-ncol**

*Number of rows/columns in a financial_matrix object*

**Description**

Unlike most other methods (i.e., duplicated, isSymmetric, summary, rownames, and colnames), these methods act on both the matrix-like and the other components of a financial_matrix object.

**Usage**

```r
## S4 method for signature 'financial_matrix'
ncol(x)
```

```r
## S4 method for signature 'financial_matrix'
nrow(x)
```

**Arguments**

- `x`:
  - The financial_matrix object to operate on.

**Details**

Checks if the length of the names matches that of the other attributes that are not NA or structurally of unitary length (i.e., the slots M and relation).

**Value**

A single numeric, the number of rows (columns) in the matrix. It also prints a message to the console if any of the object’s other attributes (e.g., capitalisation) is not conformed to the matrix’s dimensions.

**Author(s)**

Telarico, Fabio Ashtar

---

**FF-subset-method**

*Method to subset a financial_matrix*

**Description**

Subsets all components of a financial_matrix object.

**Usage**

```r
## S4 method for signature 'financial_matrix'
subset(x, ...)
```
FF.binary.both

Arguments

x  The financial_matrix object to operate on
...
Arguments passed to the relevant matrix method

Value

A financial_matrix object, subsetted to the desired firms

Author(s)

Telarico, Fabio Ashtar

FF.binary.both  Create a complete binary firm-firm (FF) matrix

Description

Function to create a binary firm-firm (FF) matrix based on both common ownership and board interlocks

Usage

FF.binary.both(
  ..., id_as_firm_name = NULL,
  Matrix = NULL,
  self_ties = FALSE,
  combining = "sum"
)

Arguments

...
Either multiple objects of class firm or a list of such objects
id_as_firm_name  Whether to use the ticker as the firm’s name. Defaults to TRUE if all firms’ id is neither NULL nor NA.
Matrix  Whether to use the Matrix package. Defaults to TRUE when any matrix in the pipeline contains more than 10,000 cells and the package is installed.
self_ties  Whether to allow self-ties (a ‘loop’ in graph theory). Defaults to FALSE.
combining  How to combine the FF matrix for managers and that for owners. Possible values:
  • sum;
  • mean or average;
  • min;
  • max;
Details

The ties' value will be: 1 if there is at least one common manager or owner, 0 otherwise.

Value

A matrix object of class financial_matrix (possibly using the Matrix package).

Author(s)

Telarico, Fabio Ashtar

See Also

FF FF.naive.both FF.norm.both

Examples

# Create the complete binary firm-firm matrix for the companies held by Berkshire Hathaway
data('firms_BKB')
FF <- FF.binary.both(firms_BKB)

Description

Function to create a binary firm-firm (FF) matrix based on board interlocks

Usage

FF.binary.management(
...
, id_as_firm_name = NULL,
Matrix = NULL,
self_ties = FALSE
)

Arguments

... Either multiple objects of class firm or a list of such objects
id_as_firm_name Whether to use the ticker as the firm’s name. Defaults to TRUE if all firms’ id is
neither NULL nor NA.
Matrix Whether to use the Matrix package. Defaults to TRUE when any matrix in the
pipeline contains more than 10,000 cells and the package is installed.
self_ties Whether to allow self-ties (a ‘loop’ in graph theory). Defaults to FALSE.
FF.binary.ownership

Value

A matrix object of class financial_matrix (possibly using the Matrix package)

Author(s)

Telarico, Fabio Ashtar

See Also

FF FF.binary.ownership FF.naive.ownership FF.naive.management FF.norm.ownership FF.norm.management

Examples

# Create the binary FF matrix of Berkshire Hathaway's holdings by boards interlock
data('firms_BKB')
FF <- FF.binary.management(firms_BKB)

Description

Function to create a binary firm-firm (FF) matrix based on common ownership

Usage

FF.binary.ownership(  ...
, id_as_firm_name = NULL,
Matrix = NULL,
self_ties = FALSE
)

Arguments

... Either multiple objects of class firm or a list of such objects
id_as_firm_name Whether to use the ticker as the firm's name. Defaults to TRUE if all firms' id is neither NULL nor NA.
Matrix Whether to use the Matrix package. Defaults to TRUE when any matrix in the pipeline contains more than 10,000 cells and the package is installed.
self_ties Whether to allow self-ties (a 'loop' in graph theory). Defaults to FALSE.

Value

A matrix object of class financial_matrix (possibly using the Matrix package)
Author(s)
Telarico, Fabio Ashtar

See Also
FF FF.binary.management FF.naive.ownership FF.naive.management FF.norm.ownership FF.norm.management

Examples

```r
# Create the binary FF matrix of Berkshire Hathaway's holdings by common ownership
data('firms_BKB')
FF <- FF.binary.ownership(firms_BKB)
```

Description
Create an object of class igraph from the package igraph using a FF matrix of class financial_matrix using all the default aesthetic options

Usage
```r
FF.graph(x, aesthetic = c("simple", "nice"))
```

Arguments
- `x`: A matrix-like object produced by FF and related functions.
- `aesthetic`: Choose a pre-set for the graph's look. Either 'simple' or 'nice' (see Details).

Details
This function does not allow for any of the additional arguments that can be passed to FF.graph.custom.

Value
A network in the desired format

Loops and values
Loops will be allowed if at least one of the matrix’s diagonal entries is not zero. The igraph will be valued if at least one entry of the matrix is neither zero nor one.

Instead, if aesthetic is set to 'simple':
- The width of the ties is 1;
- The colour of the ties is #b4b4b4 (Philippine Silver);
The size of the nodes is 5;
- The colour of the nodes is #081677 (Gentian blue).

Otherwise, if aesthetic is set to 'nice':
- The width of the ties is 1;
- The colour of the ties is a grey scale reflecting tie strength if the graph is valued, otherwise it is #b4b4b4 (Philippine Silver);
- The size of the nodes reflects their capitalisation if all firms have data on it and ranges between 1 and 5, otherwise it is 5 for all nodes;
- The colour of the nodes reflects their sector if all firms have data on it is taken from a built-in palette, otherwise it is #081677 (Gentian blue).

Author(s)
Telarico, Fabio Ashtar

See Also
FF.net FF.net.custom FF.graph.custom

Examples
# Create a nice graph representation of the binary FF of
# Berkshire Hataway's holdings based on common ownership
data("firms_BKB")
x <- FF.naive.ownership(firms_BKB)
FF.graph(x = x, aesthetic = 'nice')
Arguments

- **x**: A matrix-like object produced by `FF` and related functions.
- **vertex.size**: Which piece of information on the firms should be used to represent the nodes’ size (see Details).
- **vertex.colour**: Which piece of information on the firms should be used to represent the nodes’ colours (see Details).
- **edge.width**: Whether to use the edges’ width to represent tie strength. Defaults to `FALSE`.
- **edge.greyscale**: Whether to use the edges’ colour to represent tie strength through a grey scale. Defaults to `TRUE` if the matrix is valued.
- **directed**: Whether the network should be directed. Defaults to `TRUE`.
- **loops**: Whether the network should have loops. Defaults to `FALSE`.
- **weighted**: Whether the ties/edges should be weighted. Defaults to `TRUE` if any element of the matrix equals neither 0 nor 1.
- **...**: Aliases to the other parameters and additional settings (see Details).

Details

This function allows for a number of additional arguments.

Value

A network in the desired format.

What can be passed to **vertex.colour** and **vertex.size**

The pieces of information that is possible to pass to `vertex.size` and `vertex.colour` are:

- `capitalisation`, will be arranged into steps (see `capitalisation.bins` below)
- `revenue`, will be arranged into steps (see `revenues.bins` below)
- `legal_form`
- `sector`
- `currency`

What can be passed to **edge.width** and **edge.greyscale**

The pieces of information that is possible to pass to `edge.width` and `edge.greyscale` are:

- `capitalisation`
- `revenue`
**Additional parameters related to** vertex.size

The effect of the additional parameters that modify the behaviour of vertex.size are:

vertex.size.max (defaults to 5):

- if vertex.size or one of its aliases is specified, this is the size of the biggest vertex;
- if neither vertex.size nor any of its aliases is given, this is the size of ALL vertices.

vertex.size.min (defaults to 1):

- if vertex.size or one of its aliases is specified, this is the size of the smallest vertex;
- if neither vertex.size nor any of its aliases is given, it is ignored.

**Additional parameters related to** vertex.colour

The only additional parameter related to vertex.colour is vertex.colour.palette. It supports a vector of RGB or named colours (see colours for all named colours in R). It also accepts complete calls to functions that return a such a vector like RColorBrewer::brewer.pal(n, name) or viridisLite::viridis(n, option). If the palette is too short, it will be extended automatically using colorRampPalette. If the palette is not declared, but this argument is TRUE, it will default to the following vector of colours:

- #00204D, Oxford Blue
- #31446B, Police Blue
- #666970, Dim Grey
- #958F78, Artichoke
- #CBBA69, Dark Khaki
- #FFEA46, Gargoyle Gas

If the argument is FALSE, NULL or NA, the vertex will be coloured of #081677 (Gentian blue).

**Additional parameters related to** edge.width

edge.width.max (defaults to 5):

- if edge.width or one of its aliases is specified, this is the thickness of the thickest edge;
- if neither edge.width nor any of its aliases is given, this is the thickness of ALL edges

edge.width.min (defaults to 1):

- if edge.width or one of its aliases is specified, this is the thickness of the slimmest edge;
- if neither edge.width nor any of its aliases is given, it is ignored.
Additional parameters related to `edge.greyscale`

`edge.greyscale.darkest` (defaults to 5):
- if `edge.greyscale` or one of its aliases is specified, this is the thickness of the thickest edge;
- if neither `edge.greyscale` nor any of its aliases is given, this is the thickness of ALL edges

`edge.greyscale.fairest` (defaults to 1):
- if `edge.greyscale` or one of its aliases is specified, this is the thickness of the slimmest edge;
- if neither `edge.greyscale` nor any of its aliases is given, it is ignored.

Several aliases are accepted for all arguments, except `M`:
- for `vertex.size`: `node.size`
- for `vertex.colour`: `vertex.color`, `node.colour`, and `node.color`;
- for `edge.width`: `tie.width`
- for `edge.greyscale`: `tie.grayscale`, `tie.greyscale`, and `edge.grayscale`

Author(s)
Telarico, Fabio Ashtar

See Also
`FF.net` `FF.net.custom` `FF.graph`

Examples

```r
# Create the graph representation of the binary FF of
# Berkshire Hathaway's holdings based on common ownership
data("firms_BKB")
x <- FF.naive.ownership(firms_BKB)
FF.graph.custom(x = x, node.size = 3)
```

FF.naive.both | Create a complete naive-valued firm-firm (FF) matrix
---|---

Description
Function to create a naive-valued firm-firm (FF) matrix based on both common ownership and board interlocks
Usage

```
FF.naive.both(
  ..., 
  id_as_firm_name = NULL,
  Matrix = NULL,
  self_ties = FALSE,
  combining = "sum"
)
```

Arguments

- `...`: Either multiple objects of class `firm` or a list of such objects
- `id_as_firm_name`: Whether to use the ticker as the firm’s name. Defaults to `TRUE` if all firms' id is neither `NULL` nor `NA`.
- `Matrix`: Whether to use the `Matrix` package. Defaults to `TRUE` when any matrix in the pipeline contains more than 10,000 cells and the package is installed.
- `self_ties`: Whether to allow self-ties (a 'loop' in graph theory). Defaults to `FALSE`.
- `combining`: How to combine the FF matrix for managers and that for owners. Possible values:
  - `sum`;
  - `mean` or `average`;
  - `min`;
  - `max`;

Details

The ties' value will reflect the count of common owners and membership depending on `combining`:

- `sum`: sum of the counts;
- `mean` or `average`: average of the counts;
- `min`: minimum of the counts;
- `max`: maximum of the counts.

Value

A matrix object of class `financial_matrix` (possibly using the `Matrix` package)

Author(s)

Telarico, Fabio Ashtar

See Also

- `FF`
Examples

# Create the complete naive firm-firm matrix for the companies held by Berkshire Hathaway
data('firms_BKB')
FF <- FF.naive.both(firms_BKB)

FF.naive.management

Create a naive-valued firm-firm (FF) matrix for boards interlocks

Description

Function to create a naive-valued firm-firm (FF) matrix based on boards interlocks

Usage

FF.naive.management(
  ..., id_as_firm_name = NULL,
  Matrix = NULL,
  self_ties = FALSE
)

Arguments

... Either multiple objects of class firm or a list of such objects
id_as_firm_name Whether to use the ticker as the firm's name. Defaults to TRUE if all firms' id is
  neither NULL nor NA.
Matrix Whether to use the Matrix package. Defaults to TRUE when any matrix in the
  pipeline contains more than 10,000 cells and the package is installed.
self_ties Whether to allow self-ties (a 'loop' in graph theory). Defaults to FALSE.

Details

Naive-valued means simply counting the number of common managers.

Value

A matrix object of class financial_matrix(possibly using the Matrix package)

Author(s)

Telarico, Fabio Ashtar

See Also

FF FF.binary.ownership FF.binary.management FF.naive.ownership FF.norm.ownership FF.norm.management
Examples

```r
# Create the naive FF matrix of Berkshire Hathaway's holdings by boards interlocks
data('firms_BKB')
FF <- FF.naive.management(firms_BKB)
```

### Description

Function to create a naive-valued firm-firm (FF) matrix based on common ownership.

### Usage

```r
FF.naive.ownership(
  ..., id_as_firm_name = NULL, Matrix = NULL, self_ties = FALSE
)
```

### Arguments

- `...`: Either multiple objects of class `firm` or a list of such objects.
- `id_as_firm_name`: Whether to use the ticker as the firm’s name. Defaults to `TRUE` if all firms’ id is neither `NULL` nor NA.
- `Matrix`: Whether to use the `Matrix` package. Defaults to `TRUE` when any matrix in the pipeline contains more than 10,000 cells and the package is installed.
- `self_ties`: Whether to allow self-ties (a ‘loop’ in graph theory). Defaults to `FALSE`.

### Details

Naive-valued means simply counting the number of common owners.

### Value

A matrix object of class `financial_matrix` (possibly using the `Matrix` package).

### Author(s)

Telarico, Fabio Ashtari

### See Also

`FF.naive.ownership`, `FF.binary.ownership`, `FF.binary.management`, `FF.naive.management`, `FF.norm.ownership`, `FF.norm.management`
Examples

```r
# Create the naive FF matrix of Berkshire Hathaway's holdings by common ownership
data('firms_BKB')
FF <- FF.naive.ownership(firms_BKB)
```

---

**FF.net**

_Easily represent a firm-firm (FF) network using the package network_

---

**Description**

Create an object of class `network` from the package `network` using a FF matrix of class `financial_matrix` using all the default aesthetic options

**Usage**

```r
FF.net(x, aesthetic = c("simple", "nice"))
```

**Arguments**

- `x`: A matrix-like object produced by `FF` and related functions.
- `aesthetic`: Choose a pre-set for the network's look. Either 'simple' or 'nice' (see Details).

**Details**

This function does not allow for any of the additional arguments that can be passed to `FF.net.custom`.

**Value**

A network in the desired format

**Loops and values**

Loops will be allowed if at least one of the matrix’s diagonal entries is not zero. The network will be valued if at least one entry of the matrix is neither zero nor one.

Instead, if `aesthetic` is set to 'simple':

- The width of the ties is 1;
- The colour of the ties is `#b4b4b4` (Philippine Silver);
- The size of the nodes is 5;
- The colour of the nodes is `#081677` (Gentian blue).

Otherwise, if `aesthetic` is set to 'nice':

- The width of the ties is 1;
The colour of the ties is a grey scale reflecting tie strength if the network is valued, otherwise it is #b4b4b4 (Philippine Silver);

• The size of the nodes reflects their capitalisation if all firms have data on it and ranges between 1 and 5, otherwise it is 5 for all nodes;

• The colour of the nodes reflects their sector if all firms have data on it is taken from a built-in palette, otherwise it is #081677 (Gentian blue).

Author(s)
Telarico, Fabio Ashtar

See Also
FF.net.custom FF.graph FF.graph.custom

Examples

# Create a nice network representation of the binary FF of Berkshire Hataway’s holdings based on common ownership
data("firms_BKB")
x <- FF.naive.ownership(firms_BKB)
FF.net(x = x, aesthetic = 'nice')

---

**FF.net.custom** Represent a firm-firm (FF) network using the package network

**Description**
Create an object of class network from the package network using a FF matrix of class financial_matrix

**Usage**

```r
FF.net.custom(
x,  
vertex.size = NULL,  
vertex.colour = NULL,  
edge.width = NULL,  
edge.greyscale = NULL,  
directed = TRUE,  
loops = FALSE,  
weighted = any(x@M %in% c(0, 1)),  
...  
)
```
Arguments

- **x**: A matrix-like object produced by `FF` and related functions.
- **vertex.size**: Which piece of information on the firms should be used to represent the nodes’ size (see Details).
- **vertex.colour**: Which piece of information on the firms should be used to represent the nodes’ colours (see Details).
- **edge.width**: Whether to use the edges’ width to represent tie strength. Defaults to `FALSE`.
- **edge.greyscale**: Whether to use the edges’ colour to represent tie strength through a grey scale. Defaults to `TRUE` if the matrix is valued.
- **directed**: Whether the network should be directed. Defaults to `TRUE`.
- **loops**: Whether the network should have loops. Defaults to `FALSE`.
- **weighted**: Whether the ties/edges should be weighted. Defaults to `TRUE` if any element of the matrix equals neither 0 nor 1.
- **...**: Aliases to the other parameters and additional settings (see Details).

Details

This function allows for a number of additional arguments.

Value

A network in the desired format.

What can be passed to `vertex.colour` and `vertex.size`

The pieces of information that is possible to pass to `vertex.size` and `vertex.colour` are:

- capitalisation, will be arranged into steps (see `capitalisation.bins` below)
- revenue, will be arranged into steps (see `revenues.bins` below)
- legal_form
- sector
- currency

What can be passed to `edge.width` and `edge.greyscale`

The pieces of information that is possible to pass to `edge.width` and `edge.greyscale` are:

- capitalisation
- revenue
Additional parameters related to vertex.size

The effect of the additional parameters that modify the behaviour of vertex.size are:

vertex.size.max (defaults to 5):

- if vertex.size or one of its aliases is specified, this is the size of the biggest vertex;
- if neither vertex.size nor any of its aliases is given, this is the size of ALL vertices.

vertex.size.min (defaults to 1):

- if vertex.size or one of its aliases is specified, this is the size of the smallest vertex;
- if neither vertex.size nor any of its aliases is given, it is ignored.

Additional parameters related to vertex.colour

The only additional parameter related to vertex.colour is vertex.colour.palette. It supports a vector of RGB or named colours (see colours for all named colours in R). It also accepts complete calls to functions that return a such a vector like RColorBrewer::brewer.pal(n, name) or viridisLite::viridis(n, option). If the palette is too short, it will be extended automatically using colorRampPalette. If the palette is not declared, but this argument is TRUE, it will default to the following vector of colours:

- #00204D, Oxford Blue
- #31446B, Police Blue
- #666970, Dim Gray
- #958F78, Artichoke
- #CBBA69, Dark Khaki
- #FFEA46, Gargoyle Gas

If the argument is FALSE, NULL or NA, the vertex will be coloured of #081677 (Gentian blue).

Additional parameters related to edge.width

edge.width.max (defaults to 5):

- if edge.width or one of its aliases is specified, this is the thickness of the thickest edge;
- if neither edge.width nor any of its aliases is given, this is the thickness of ALL edges

edge.width.min (defaults to 1):

- if edge.width or one of its aliases is specified, this is the thickness of the slimmest edge;
- if neither edge.width nor any of its aliases is given, it is ignored.
Additional parameters related to `edge.greyscale`

- `edge.greyscale.darkest` (defaults to 5):
  - if `edge.greyscale` or one of its aliases is specified, this is the thickness of the thickest edge;
  - if neither `edge.greyscale` nor any of its aliases is given, this is the thickness of ALL edges

- `edge.greyscale.fairest` (defaults to 1):
  - if `edge.greyscale` or one of its aliases is specified, this is the thickness of the slimmest edge;
  - if neither `edge.greyscale` nor any of its aliases is given, it is ignored.

Several aliases are accepted for all arguments, except `M`:

- for `vertex.size`: `node.size`
- for `vertex.colour`: `vertex.color`, `node.colour`, and `node.color`;
- for `edge.width`: `tie.width`
- for `edge.greyscale`: `tie.grayscale`, `tie.greyscale`, and `edge.grayscale`

**Author(s)**

Telarico, Fabio Ashtar

**See Also**

- `FF.net`  
- `FF.graph`  
- `FF.graph.custom`

**Examples**

```r
# Create the network representation of the binary FF of Berkshire Hathaway's holdings based on common ownership
data("firms_BKB")
x <- FF.naive.ownership(firms_BKB)
FF.net.custom(x = x, node.size = 3)
```

---

**FF.norm.both**

Create a complete normalised-valued firm-firm (FF) matrix

**Description**

Function to create a normalised-valued firm-firm (FF) matrix based on both common ownership and board interlocks
Usage

```
FF.norm.both(
    ..., 
    id_as_firm_name = NULL, 
    Matrix = NULL, 
    self_ties = FALSE, 
    combining = "sum"
)
```

Arguments

- `...`: Either multiple objects of class `firm` or a list of such objects
- `id_as_firm_name`: Whether to use the ticker as the firm’s name. Defaults to `TRUE` if all firms’ id is neither `NULL` nor `NA`.
- `Matrix`: Whether to use the `Matrix` package. Defaults to `TRUE` when any matrix in the pipeline contains more than 10,000 cells and the package is installed.
- `self_ties`: Whether to allow self-ties (a 'loop' in graph theory). Defaults to `FALSE`.
- `combining`: How to combine the FF matrix for managers and that for owners. Possible values:
  - `sum`
  - `mean` or `average`
  - `min`
  - `max`

Details

The ties’ value will reflect the count of common owners and membership depending on `combining`:
- `sum`: sum of the shares (normalised on 2);
- `mean` or `average`: average of the shares (normalised on 1);
- `min`: minimum of the shares (normalised on 1);
- `max`: maximum of the shares (normalised on 1).

Value

A matrix object of class `financial_matrix`(possibly using the `Matrix` package)

Author(s)

Telarico, Fabio Ashtar

See Also

`FF`, `FF.binary.both`, `FF.naive.both`
### Examples

```r
# Create the complete normalised firm-firm matrix for the companies held by Berkshire Hathaway
data('firms_BKB')
FF <- FF.norm.both(firms_BKB)
```

### Description

Function to create a normalised-valued firm-firm (FF) matrix based on boards interlocks

### Usage

```r
FF.norm.management(
  ..., 
  id_as_firm_name = NULL,
  Matrix = NULL,
  self_ties = FALSE
)
```

### Arguments

- `...`: Either multiple objects of class `firm` or a list of such objects
- `id_as_firm_name`: Whether to use the ticker as the firm’s name. Defaults to `TRUE` if all firms’ id is neither `NULL` nor `NA`.
- `Matrix`: Whether to use the `Matrix` package. Defaults to `TRUE` when any matrix in the pipeline contains more than 10,000 cells and the package is installed.
- `self_ties`: Whether to allow self-ties (a 'loop' in graph theory). Defaults to `FALSE`.

### Details

Normalised-valued means that weights represent the share of common managers.

### Value

A matrix object of class `financial_matrix` (possibly using the `Matrix` package)

### Author(s)

Telarico, Fabio Ashtar
**FF.norm.ownership**

See Also

FF FF.binary.ownership FF.binary.management FF.naive.ownership FF.naive.management FF.norm.ownership

Examples

```r
# Create the normalised FF matrix of Berkshire Hathaway's holdings by boards interlocks
data('firms_BKB')
FF <- FF.norm.management(firms_BKB)
```

---

**FF.norm.ownership**

Create a normalised-valued firm-firm (FF) matrix for common ownership

Description

Function to create a normalised-valued firm-firm (FF) matrix based on common ownership

Usage

```r
FF.norm.ownership(
  ..., id_as_firm_name = NULL,
  Matrix = NULL,
  self_ties = FALSE
)
```

Arguments

- `...`: Either multiple objects of class `firm` or a list of such objects
- `id_as_firm_name`: Whether to use the ticker as the firm’s name. Defaults to `TRUE` if all firms’ id is neither `NULL` nor `NA`.
- `Matrix`: Whether to use the `Matrix` package. Defaults to `TRUE` when any matrix in the pipeline contains more than 10,000 cells and the package is installed.
- `self_ties`: Whether to allow self-ties (a 'loop' in graph theory). Defaults to `FALSE`.

Details

Normalised-valued means that weights represent the share of common managers.

Value

A matrix object of class `financial_matrix`(possibly using the `Matrix` package)
find.firm  

Function to create a firm (legal person) using data from 'Yahoo! Finance'

Description

Tickers can be retrieved from [Yahoo! Finance](https://finance.yahoo.com/lookup/). This function requires the package `yahoofinancer` to be installed. It is available from the CRAN by running `install.packages('yahoofinancer')`.

Usage

```r
find.firm(
  ticker, 
  name = NULL, 
  ticker_is_id = TRUE, 
  legal_form = NULL, 
  sector_granularity = 1, 
  managers_remove_salutation_title = TRUE, 
  managers_only_surname = FALSE
)
```

Arguments

- **ticker**: Firm’s ticker.
- **name**: Provide the firm’s name. If not provided, NA, or NULL, will default to the string provided as ticker.
- **ticker_is_id**: Should the ticker be used as the firm’s id?
- **legal_form**: The firm’s legal form of the firm. Possible values: - a string (e.g., 'LLC', 'Private', 'GmbH', etc.); - NULL (default), in which case the function will set `legal_form` to 'JSC'; or - NA to specify no legal form.

Examples

```r
# Create the normalised FF matrix of Berkshire Hathaway's holdings by common ownership
data('firms_BKB')
FF <- FF.norm.ownership(firms_BKB)
```
sector_granularity

Sector in which the firm operates. Possible values: -0, NULL, or NA to omit the sector; -1 or 'generic' (default) for a generic description (e.g., 'Consumer Technology', 'Consumer Cyclical', 'Consumer Defensive'); -2 or 'specific' for a more granular description (e.g., 'Technology', 'Auto Manufacturers', 'Tobacco').

managers_remove_salutation_title

Yahoo! Finance provide salutation titles before the names of the managers. If this is TRUE (default), they will be omitted.

managers_only_surname

Yahoo! Finance provide first, middle, and last name of the managers. If this is TRUE (not recommended for large data sets), only the surname is returned.

Value

An object of the S4 class firm containing several fields, only the first one of which is mandatory:

name

Name of the firm (or ticker if no name was provided)

id

Firm’s ticker (if ticker_is_id was ‘TRUE’) or nothing (otherwise)

legal_form

Legal form of the firm (may be null)

sector

Sector in which the firm operates (may be null)

revenues

Yearly revenues

capitalisation

Capitalisation

management

Members of the board

ownership

Owner(s)

shares

Share owned by (each of) the owner(s)

currency

Currency

Author(s)

Telarico, Fabio Ashtar

See Also

register.firm find.firms

Examples

# Registering Apple automatically
#| Results are subject to the correct functioning of the package `yahoofinance`' 
#| and of the Yahoo! Finance API
find.firms

Function to create multiple firms (legal persons) using data from 'Yahoo! Finance'

Description

If legal_form is a vector containing: - one or more NULL elements, the corresponding firm's legal form will be JSC; - one or more NAs, the corresponding firm's legal form will be NA.

Usage

```r
find.firms(
  tickers,
  name = NULL,
  ticker_is_id = TRUE,
  legal_form = NULL,
  sector_granularity = 1,
  managers_remove_salutation_title = TRUE,
  managers_only_surname = FALSE
)
```

Arguments

tickers | The firms' ticker.
name | Provide the firms' names as a vector of the same length as tickers. If not provided, NA, or NULL, will default to the firm's ticker.
ticker_is_id | Should the ticker be used as the firm's id?
legal_form | The firm's legal form of the firm. Possible values: - a vector of strings (e.g., 'LLC', 'Private', 'GmbH', etc.) of the same length as tickers (see 'Details' for the interpretation of NAs and NULLs); - NULL (default), in which case the function will set legal_form to 'JSC' for all firms; or - NA to specify no legal form.
sector_granularity | Sector in which the firm operates. Possible values: - 0, NULL, or NA to omit the sector; - 1 or 'generic' (default) for a generic description (e.g., 'Consumer Technology', 'Consumer Cyclical', 'Consumer Defensive'); - 2 or 'specific' for a more granular description (e.g., 'Technology', 'Auto Manufacturers', 'Tobacco').
managers_remove_salutation_title | Yahoo! Finance provide salutation titles before the names of the managers. If this is TRUE (default), they will be omitted.
managers_only_surname | Yahoo! Finance provide first, middle, and last name of the managers. If this is TRUE (not recommended for large data sets), only the surname is returned.
Details

To ensure consistency, `ticker_is_id`, `sector_granularity`, `managers_remove_salutation_title`, and `managers_only_surname` cannot be vectors.

Tickers can be retrieved from [Yahoo! Finance](https://finance.yahoo.com/lookup/). This function requires the package `yahoofinancer` to be installed. It is available from the CRAN by running `install.packages('yahoofinancer')`.

Value

An object of the S4 class `firm` containing several fields, only the first one of which is mandatory:

- `name`: Name of the firm (or ticker if no name was provided)
- `id`: Firm’ ticker (if `ticker_is_id` was ‘TRUE’) or nothing (otherwise)
- `legal_form`: Legal form of the firm (may be null)
- `sector`: Sector in which the firm operates (may be null)
- `revenues`: Yearly revenues
- `capitalisation`: Capitalisation
- `management`: Members of the board
- `ownership`: Owner(s)
- `shares`: Share owned by (each of) the owner(s)
- `currency`: Currency

Author(s)

Telarico, Fabio Ashtar

See Also

- `find.firm`

Examples

# Registering Apple, General Motors, and British American Tobacco automatically
#| Results are subject to the correct functioning of the package ‘yahoofinancer’
#| and of the Yahoo! Finance API
find.people

Extract all the unique people associated to at least one of the provided firm objects

Description

Extract all the unique people associated to at least one of the provided firm objects

Usage

find.people(..., who = c("managers", "owners", "both", "all"), sorting = TRUE)

Arguments

... Either multiple objects of class firm or a list of such objects
who Whether to extract the 'managers' or the 'owners' (minimum unambiguous string)
sorting Whether to sort the people by alphabetical order. Defaults to TRUE

Value

A vector containing the names of the individuals looked up. If

Author(s)

Telarico, Fabio Ashtar

Examples

# Find all the shareholders in companies that Berkshire Hathaway holds
data('firms_BKB')
shareholders <- find.people(firms_BKB, who = 'own')

# Find all those managing the companies that Berkshire Hathaway holds
data('firms_BKB')
managers <- find.people(firms_BKB, who = 'man')
firms_BKB  

*Complete Berkshire Hathaway Portfolio*

**Description**

Data on Apple (AAPL), General Motors (GM), and British American Tobacco (BTI) extracted from Yahoo! Finance (on May 20, 2023) and formatted as firm objects.

**Usage**

```r
data('firms_BKB')
```

**Format**

Three objects of class `firm`.

**Source**


---

firms_US  

*Three US firms*

**Description**

Data on Apple (AAPL), General Motors (GM), and British American Tobacco (BTI) extracted from Yahoo! Finance (on May 20, 2023) and formatted as firm objects.

**Usage**

```r
data('firms_US')
```

**Format**

Three objects of class `firm`.

**Source**

Function to create a (necessarily binary) firm-manager (FM) matrix

**Description**

Function to create a (necessarily binary) firm-manager (FM) matrix

**Usage**

```r
FM(..., id_as_firm_name = NULL, Matrix = NULL)
```

**Arguments**

- `...` Either multiple objects of class `firm` or a list of such objects
- `id_as_firm_name` Whether to use the ticker as the firm’s name. Defaults to `TRUE` if all firms’ id is neither `NULL` nor `NA`.
- `Matrix` Whether to use the `Matrix` package. Defaults to `TRUE` when there are more than 10,000 combinations and the package is installed.

**Value**

A matrix object of class `financial_matrix` (possibly using the `Matrix` package) in which:

- **the rows** Represent firms;
- **the columns** Represent managers (usually physical persons).

**Author(s)**

Telarico, Fabio Ashtar

**See Also**

`FO.binary` `FO.naive` `FO.norm`

**Examples**

```r
# Create the FM matrix of Berkshire Hathaway's holdings

data('firms_BKB')
FM <- FM(firms_BKB)
```
**FO.binary**

*Function to create a binary firm-owner (FO) matrix*

**Description**

Function to create a binary firm-owner (FO) matrix

**Usage**

`FO.binary(..., id_as_firm_name = NULL, Matrix = NULL)`

**Arguments**

- `...` Either multiple objects of class `firm` or a list of such objects
- `id_as_firm_name` Whether to use the ticker as the firm’s name. Defaults to `TRUE` if all firms’ id is neither `NULL` nor `NA`.
- `Matrix` Whether to use the `Matrix` package. Defaults to `TRUE` when there are more than 10,000 combinations and the package is installed.

**Value**

A matrix object of class `financial_matrix` (possibly using the `Matrix` package) in which:

- **the rows** Represent firms;
- **the columns** Represent owners (physical and legal persons).

**Author(s)**

Telarico, Fabio Ashtar

**See Also**

- `FM`
- `FO.naive`
- `FO.norm`

**Examples**

```r
# Create the binary FO matrix of Berkshire Hathaway's holdings

data('firms_BKB')
FO <- FO.binary(firms_BKB)
```
Description

The values are simply the value of the owner $j$'s stake in firm $i$.

Usage

```r
FO.naive(..., id_as_firm_name = NULL, Matrix = NULL)
```

Arguments

- `...`: Either multiple objects of class `firm` or a list of such objects
- `id_as_firm_name`: Whether to use the ticker as the firm's name. Defaults to `TRUE` if all firms' id is neither `NULL` nor `NA`.
- `Matrix`: Whether to use the `Matrix` package. Defaults to `TRUE` when there are more than 10,000 combinations and the package is installed.

Value

A matrix object of class `financial_matrix` (possibly using the `Matrix` package) in which:

- **the rows** Represent firms;
- **the columns** Represent owners (physical and legal persons).

Author(s)

Telarico, Fabio Ashtar

See Also

`FM FO.binary FO.norm`

Examples

```r
# Create the naive FO matrix of Berkshire Hathaway's holdings

data('firms_BKB')
FO <- FO.naive(firms_BKB)
```
FO.norm

Function to create a naive-valued firm-owner (FO) matrix

Description

The values represent the share of firm i’s capital owned by j.

Usage

FO.norm(..., id_as_firm_name = NULL, Matrix = NULL)

Arguments

... Either multiple objects of class firm or a list of such objects
id_as_firm_name Whether to use the ticker as the firm’s name. Defaults to TRUE if all firms’ id is neither NULL nor NA.
Matrix Whether to use the Matrix package. Defaults to TRUE when there are more than 10,000 combinations and the package is installed.

Value

A matrix object of class financial_matrix (possibly using the Matrix package) in which:

the rows Represent firms;
the columns Represent owners (physical and legal persons).

Author(s)

Telarico, Fabio Ashtar

See Also

FM FO.binary FO.naive

Examples

# Create the normalised FO matrix of Berkshire Hathaway's holdings
data('firms_BKB')
FO <- FO.norm(firms_BKB)
Description

The following functions are implemented:

- \( V_{\text{fin}} \) to retrieve the vertexes (igraph::V);
- \( \text{vcount}_{\text{fin}} \) to count the vertexes (igraph::vcount);
- \( \text{gorder}_{\text{fin}} \) as an alias to \( \text{vcount}_{\text{fin}} \) (igraph::gorder);
- \( \text{E}_{\text{fin}} \) to retrieve the edges (igraph::E);
- \( \text{gsize}_{\text{fin}} \) to count the edges (igraph::gsize);
- \( \text{ecount}_{\text{fin}} \) as an alias to \( \text{gsize}_{\text{fin}} \) (igraph::ecount);
- \( \text{plot}_\text{igraph}_{\text{fin}} \) to plot graphs (igraph::plot.igraph)

Usage

\begin{verbatim}
V(x)
vcount(x)
gorder(x)
E(x, ...)
ecount(x, ...)
gsize(x, ...)
plot_igraph(x, ...)
\end{verbatim}

Arguments

- \( x \) The igraph_financial object
- \( ... \) Other parameters passed to the corresponding igraph functions (see Details).

Details

Implementing most basic iterators from the package igraph for objects of class igraph_financial

Value

The same result for both igraph and igraph_financial objects

- \( V \): A vertex sequence containing all vertices, in the order of their numeric vertex ids.
- \( \text{vcount} \) and \( \text{gorder} \): Number of vertices, numeric scalar.
igraph_E_iterators

• E: An edge sequence of the graph
• ecount and gsize: Number of edges, numeric scalar.
• plot_igraph: Returns NULL, invisibly. Called to print the graph to any R device. (see method and igraph::plot.igraph)

Author(s)
Telarico, Fabio Ashtar

igraph_E_iterators  igraph edge iterators for igraph_financial objects

Description
Methods to extend igraph edge iterators and functions to igraph_financial objects

Usage

## S4 method for signature 'igraph_financial'
E(x, ...)

## S4 method for signature 'igraph'
E(x, ...)

## S4 method for signature 'igraph_financial'
ecount(x, ...)

## S4 method for signature 'igraph'
ecount(x, ...)

## S4 method for signature 'igraph_financial'
gsize(x, ...)

## S4 method for signature 'igraph'
gsize(x, ...)

Arguments

x          The igraph_financial object
...
Other parameters passed to the corresponding method and/or igraph functions (see Details).

Value
The same result for both igraph and igraph_financial objects

• E: An edge sequence of the graph
• ecount and gsize: Number of edges, numeric scalar
**igraph_financial**

An S4 class for relational data extending the package

[R](https://igraph.org/igraph)

**Description**

An S4 class for the network objects produced by the **FF.graph** and **FF.graph.custom** to represent the relations between firms (legal person)

**Slots**

data  The representation of the network as a igraph object

**igraph_v_iterators**

**igraph vertex iterators for igraph_financial objects**

**Description**

Methods to extend igraph vertex iterators and functions to igraph_financial objects

**Usage**

```r
## S4 method for signature 'igraph_financial'
V(x)

## S4 method for signature 'igraph'
V(x)

## S4 method for signature 'igraph_financial'
vcount(x)

## S4 method for signature 'igraph'
vcount(x)

## S4 method for signature 'igraph_financial'
gorder(x)

## S4 method for signature 'igraph'
gorder(x)
```

**Arguments**

- **x**  The igraph_financial object
network_financial

Value

The same result for both igraph and igraph_financial objects

- V: A vertex sequence containing all vertices, in the order of their numeric vertex ids
- vcount and gorder: Number of vertices, numeric scalar

Author(s)

Telarico, Fabio Ashtar

network_financial  
An S4 class for relational data extending the package Rhrefhttps://statnet.org/network

Description

An S4 class for the network objects produced by the FF.net and FF.net.custom functions to represent the relations between firms (legal person)

Slots

data  The representation of the network as a network object

plot_igraph-methods  igraph plotting for igraph_financial objects

Description

Methods to extend codeigraph’s plotting functions to igraph_financial objects

Usage

```r
## S4 method for signature 'igraph_financial'
plot_igraph(x, ...)

## S4 method for signature 'igraph'
plot_igraph(x, ...)
```

Arguments

- `x`  The igraph_financial object
- `...`  Other parameters passed to the corresponding method and/or igraph functions (see Details).
Value

For both igraph and igraph_financial objects, returns NULL invisibly. It is called to print the graph to any R device. (see method and igraph::plot.igraph)

Author(s)

Telarico, Fabio Ashtar
query.firm

Function to extract information from a firm object (legal person)

Description

Function to extract information from a firm object (legal person)

Usage

query.firm(firm, which, naming = TRUE)

Arguments

firm Firm which to extract information from
which Information to extract, minimum unambiguous substring. Possible values (one or more): - name Name of the firm - id ID of the firm, usually the ticker (if provided or otherwise known) - legal_form Legal form of the firm - sector Sector in which the firm operates - revenues Yearly revenues - capitalisation Capitalisation - management Members of the board - ownership Owner(s) - shares Share owned by (each of) the owner(s) - currency Currency in which revenues and capitalisation are denominated
naming Whether to name the result after the queried information (defaults to TRUE)

Value

Depends on the information queried. One (or, if length(which) >= 2, a list of two or more) of the following:

name A string representing the name of the firm
id A string representing the ID of the firm (usually its ticker)
legal_form A string representing the firm’s legal form
sector A string indicating the sector in which the firm operates (possibly a NACE rev. 2 code)
revenues A numeric (double) quantifying yearly revenues
capitalisation A numeric (double) quantifying capitalisation
management A vector of strings representing the members of the board
ownership A vector of strings representing the owner(s)
shares A numeric (double) vector indicating the shares controlled by (each of) the owner(s)
currency A string indicating the currency in which revenues and capitalisation are denominated

Author(s)

Telarico, Fabio Ashtar
query.firms

Function to extract information from multiple firm object (legal person)

Description

This function can be fed either: - a (possibly named) list of objects of class firm (see examples 1 and 2); or - multiple objects of class firm (see example 3)

Usage

query.firms(..., which, naming = TRUE)

Arguments

... Object/s which to extract information from (see 'Details')

which Information to extract, minimum unambiguous sub-string. Possible values (one or more): - name Name of the firm - id ID of the firm, usually the ticker (if provided or otherwise known) - legal_form Legal form of the firm - sector Sector in which the firm operates - revenues Yearly revenues - capitalisation Capitalisation - management Members of the board - ownership Owner(s) - shares Shares Share owned by (each of) the owner(s) - currency Currency in which revenues and capitalisation are denominated

naming Whether to name the result after the query information (defaults to TRUE)
Value

Depends on the information queried. An object of class `list` (that, if `length(which)>=2`, contain multiple sub-lists) of the following:

- `name`: A string representing the name of the firm
- `id`: A string representing the ID of the firm (usually its ticker)
- `legal_form`: A string representing the firm’s legal form
- `sector`: A string indicating the sector in which the firm operates (possibly a NACE rev. 2 code)
- `revenues`: A numeric (double) quantifying yearly revenues
- `capitalisation`: A numeric (double) quantifying capitalisation
- `management`: A vector of strings representing the members of the board
- `ownership`: A vector of strings representing the owner(s)
- `shares`: A numeric (double) vector indicating the shares controlled by (each of) the owner(s)
- `currency`: A string indicating the currency in which revenues and capitalisation are denominated

Author(s)

Telarico, Fabio Ashtar

See Also

`query.firm` `query.firms.dataframe`

Examples

```r
# Query Apple's, GM's, and BTI's market cap and revenues
data('firms_US')
query.firms(firms_US, which = c('cap', 'rev'))

# Query GM's and BTI's management
data('firms_US')
query.firms(firms_US, which = 'man')

# Query Apple's and GM's revenues and currency
data('firms_US')
list2env(firms_US, envir = parent.frame())
query.firms(AAPL, GM, which = c('rev', 'curr'))
```
query.firms.dataframe  
*Function to extract information from multiple firm object (legal person) as a data frame*

**Description**

This function can be fed either: - a (possibly named) list of objects of class `firm` (see example 1); or

**Usage**

```r
query.firms.dataframe(..., which, naming = TRUE, transposing = TRUE)
```

**Arguments**

- `...`: Object/s which to extract information from (see 'Details')
- `which`: Information to extract, minimum unambiguous sub-string. Possible values (one or more):  - name Name of the firm - id ID of the firm, usually the ticker (if provided or otherwise known) - legal_form Legal form of the firm - sector Sector in which the firm operates - revenues Yearly revenues - capitalisation Capitalisation - management Members of the board - ownership Owner(s) - shares Share owned by (each of) the owner(s) - currency Currency in which revenues and capitalisation are denominated
- `naming`: Whether to name the result after the queried information (defaults to `TRUE`)
- `transposing`: If `TRUE` (default) each row will correspond to a firm and each column to a variable.

**Details**

It is not recommended to use this function with `management`, `ownership`, or `shares` unless `transposing == FALSE`.

**Value**

A data frame in structured as follows (or vice versa if `transposing == TRUE`):

- a row for each queried information; and
- a column for each number of firm.

**Author(s)**

Telarico, Fabio Ashtar

**See Also**

- `query.firm`
- `query.firms`
**register.firm**

Function to create a firm (legal person)

**Description**

Function to create a firm (legal person)

**Usage**

```r
register.firm(
  name,
  id = NA,
  legal_form = NA,
  sector = NA,
  sector_classif = NULL,
  revenues = NA,
  capitalisation = NA,
  management = NA,
  ownership = NA,
  shares = NA,
  currency = NA
)
```

**Arguments**

- **name**: Name of the firm
- **id**: Provide an ID code for the firm. Defaults to NA
- **legal_form**: Legal form of the firm (e.g., LLP, Inc, GmbH, Private, etc.)
- **sector**: Sector in which the firm operates (its values depend on the value of sector_classif)
- **sector_classif**: Which standard sector classification (if any) to be used. Possible values are - NACE for the Statistical Classification of Economic Activities in the European Community or 'Nomenclature statistique des Activités économiques dans la Communauté Européenne', revision 2; - NA for a custom classification (default if anything is provided); - NULL for no classification (default if nothing is provided).

**Examples**

```r
# Query Apple's, GM's, and BTI's market cap and revenues
data('firms_US')
query.firms.dataframe(firms_US, which = c('cap', 'rev'))

# Query GM's and BTI's market cap and revenues
data('firms_US')
list2env(firms_US, envir = parent.frame())
query.firms.dataframe(GM, BTI, which = c('cap', 'rev'))
```
revenues  Yearly revenues
capitalisation  Firm's capitalisation
management  Names of the members of the board
ownership  Names of the owner(s)
shares  Share owned by (each of) the owner(s)
currency  Currency in which the capitalisation and revenues are expressed (defaults to ‘USD’)

Value

An object of the S4 class `firm` containing several fields, only the first one of which is mandatory:

- name  Name of the firm
- id  ID of the firm, usually the ticker
- legal_form  Legal form of the firm
- sector  Sector in which the firm operates
- revenues  Yearly revenues
- capitalisation  Capitalisation
- management  Members of the board
- ownership  Owner(s)
- shares  Share owned by (each of) the owner(s)
- currency  Currency

Author(s)

Telarico, Fabio Ashtar

See Also

`find.firm`

Examples

```r
# Registering Apple manually
AAPL <- register.firm(name = 'Apple', id = 'AAPL', legal_form = 'GmbH',
  revenues = 81665400000, capitalisation = 2755039000000,
  management = my_vector <- c("Timothy D. Cook",
    "Luca Maestri",
    "Jeffrey E. Williams",
    "Katherine L. Adams",
    "Deirdre O'Brien",
    "Chris Kondo",
    "James Wilson",
    "Mary Demby",
    "Nancy Paxton",
    "Greg Joswiak"),
  ownership = c("Apple Corporation", "Tesoro de Mexico S.A. de C.V.", "Rothschild & Cie Banque",
    "Morgan Stanley & Co. Inc.", "Soros Fund Management Co.", "Vanguard Group Inc.",
    "BlackRock Fund Advisors", "Brown Brothers Harriman & Co."
  )
```

"Author(s)"

Telarico, Fabio Ashtar

"See Also"

`find.firm`

"Examples"
ownership = c('Vanguard Total Stock Market Index Fund',
        'Vanguard 500 Index Fund',
        'Fidelity 500 Index Fund',
        'SPDR S&P 500 ETF Trust',
        'iShares Core S&P 500 ETF',
        'Invesco ETF Tr-Invesco QQQ Tr, Series 1 ETF',
        'Vanguard Growth Index Fund',
        'Vanguard Institutional Index Fund-Institutional Index Fund',
        'Vanguard Information Technology Index Fund',
        'Select Sector SPDR Fund-Technology'),
shares = c(0.0290, 0.0218, 0.0104, 0.0102, 0.0084,
          0.0082, 0.0081, 0.0066, 0.0043, 0.0039),
currency = 'USD')

# Registering a coal-mining company indicating the sector using 'NACE' codes, without ID
set.seed(123456789)
firm_coalmining <- register.firm(
    name = 'A coal-mining firm',
    legal_form = 'Private',
    sector = 'B.05',
    sector_classif = 'NACE'
)

# Getting creative: Register a firm with coded owners and managers
set.seed(123456789)
firm_coded <- register.firm(
    name = 'Coded firm',
    revenues = sample(seq(1:100)/10, 1)*10^sample(1:5, 1),
    capitalisation = sample(seq(1:100)/10, 1)*10^sample(2:7, 1),
    management = c('Board Member', 'CEO', 'CTO', 'Activist investor'),
    ownership = c('State', 'Foreign investors'),
    shares = c(51, 49),
currency = 'EUR')
Index

* datasets
  - firms_BKB, 33
  - firms_US, 33
  - as.firm, 3
  - as.firm, financial_matrix-method, 3
  - colnames, financial_matrix-method (FF-basic-methods), 5
  - colorRampPalette, 15, 23
  - colours, 15, 23
  - duplicated, financial_matrix, logical-method (FF-comparison-methods), 6
  - E (graph_methods), 38
  - E,igraph-method (igraph_E_iterators), 39
  - E,igraph_financial-method (igraph_E_iterators), 39
  - ecount (graph_methods), 38
  - ecount,igraph-method (igraph_E_iterators), 39
  - ecount,igraph_financial-method (igraph_E_iterators), 39
  - FF, 4, 10–12, 14, 17–20, 22, 25, 27, 28
  - FF-basic-methods, 5
  - FF-comparison-methods, 6
  - FF-math-methods, 7
  - FF-nrow-ncol, 8
  - FF-subset-method, 8
  - FF.binary.both, 5, 9, 17, 25
  - FF.binary.management, 4, 5, 10, 12, 18, 19, 27, 28
  - FF.binary.ownership, 4, 5, 11, 11, 18, 19, 27, 28
  - FF.graph, 12, 16, 21, 24, 40
  - FF.graph.custom, 12, 13, 13, 21, 24, 40
  - FF.naive.both, 5, 10, 16, 25
  - FF.naive.management, 5, 11, 12, 18, 19, 27, 28

    FF.naive.ownership, 4, 5, 11, 12, 18, 19, 27, 28
    FF.net, 13, 16, 20, 24, 41
    FF.net.custom, 13, 16, 20, 21, 21, 41
    FF.norm.both, 5, 10, 17, 24
    FF.norm.management, 4, 5, 11, 12, 18, 19, 26, 28
    FF.norm.ownership, 5, 11, 12, 18, 19, 27, 27
    find.firm, 28, 31, 48
    find.firms, 29, 30
    find.people, 32
    firms_BKB, 33
    firms_US, 33
    FM, 34, 35–37
    FO.binary, 34, 35, 36, 37
    FO.naive, 34, 35, 36, 37
    FO.norm, 34–36, 37
    format, 7

    gorder (graph_methods), 38
    gorder,igraph-method (igraph_v_iterators), 40
    gorder,igraph_financial-method (igraph_v_iterators), 40
    graph_methods, 38
    gsize (graph_methods), 38
    gsize,igraph-method (igraph_E_iterators), 39
    gsize,igraph_financial-method (igraph_E_iterators), 39

    igraph_E_iterators, 39
    igraph_financial, 40
    igraph_financial-class (igraph_financial), 40
    igraph_v_iterators, 40
    isSymmetric,financial_matrix-method (FF-math-methods), 7

    list, 43–46

50
ncol, financial_matrix-method
  (FF-nrow-ncol), 8
network_financial, 41
network_financial-class
  (network_financial), 41
nrow, financial_matrix-method
  (FF-nrow-ncol), 8

plot_igraph (graph_methods), 38
plot_igraph, igraph-method
  (plot_igraph-methods), 41
plot_igraph, igraph_financial-method
  (plot_igraph-methods), 41
plot_igraph-methods, 41
print, 7
print, firm-method, 42

query.firm, 43, 45, 46
query.firms, 44, 44, 46
query.firms.dataframe, 44, 45, 46

register.firm, 29, 47
rownames, financial_matrix-method
  (FF-basic-methods), 5

subset, financial_matrix-method
  (FF-subset-method), 8
summary, financial_matrix-method
  (FF-math-methods), 7

unique, financial_matrix, logical-method
  (FF-comparison-methods), 6

V (graph_methods), 38
V, igraph-method (igraph_v_iterators), 40
V, igraph_financial-method
  (igraph_v_iterators), 40
vcount (graph_methods), 38
vcount, igraph-method
  (igraph_v_iterators), 40
vcount, igraph_financial-method
  (igraph_v_iterators), 40