Package ‘PantaRhei’

October 12, 2022

**Title**  Plots Sankey Diagrams

**Version**  0.1.2

**Description**  Sankey diagrams are a powerful and visually attractive way to visualize the flow of conservative substances through a system. They typically consists of a network of nodes, and fluxes between them, where the total balance in each internal node is 0, i.e. input equals output. Sankey diagrams are typically used to display energy systems, material flow accounts etc. Unlike so-called alluvial plots, Sankey diagrams also allow for cyclic flows: flows originating from a single node can, either direct or indirect, contribute to the input of that same node. This package, named after the Greek aphorism Panta Rhei (everything flows), provides functions to create publication-quality diagrams, using data in tables (or spread sheets) and a simple syntax.

**Depends**  R (>= 3.5.0)

**License**  EUPL

**Encoding**  UTF-8

**LazyData**  true

**RoxygenNote**  7.1.1

**Imports**  grid, grDevices, utils

**Suggests**  knitr, rmarkdown, tibble

**VignetteBuilder**  knitr

**NeedsCompilation**  no

**Author**  Patrick Bogaart [aut, cre] (<https://orcid.org/0000-0002-8612-1289>)

**Maintainer**  Patrick Bogaart <pwbogaart@gmail.com>

**Repository**  CRAN

**Date/Publication**  2020-12-18 10:20:05 UTC

**R topics documented:**

- check_balance .......................................................... 2
- check_consistency ....................................................... 3
Description

For each substance involved, the balance per (internal) node is inspected. If outflow exceed inflow, or vice versa, a message is printed, and the function returns FALSE.

Usage

```
check_balance(nodes, flows, tolerance = 0.01)
```

Arguments

- `nodes` data.frame containing the nodes definition
- `flows` data.frame containing the flows definition
- `tolerance` numeric specifying a tolerance. Default is 0.01 (1%)

Value

TRUE if balanced, FALSE if not.

Examples

```
nodes <- data.frame(ID=c("A","B","C"), x=1:3, y=1:3, dir=c("right","right","stock"))
flows <- data.frame(from=c("A","B"), to=c("B","C"), quantity=c(10,10))
check_balance(nodes,flows)
```
**check_consistency**

*Check the consistence of the nodes, flows and palette data.frames*

**Description**

Check the consistence of the nodes, flows and palette data.frames

**Usage**

```r
check_consistency(nodes, flows, palette = NULL)
```

**Arguments**

- `nodes`: data.frame containing the nodes definition
- `flows`: data.frame containing the flows definition
- `palette`: data.frame containing the palette definition

**Value**

TRUE if all checks are passed; FALSE otherwise.

**Examples**

```r
nodes <- data.frame(ID=c("A","B"), x=1:2, y=0)
flows <- data.frame(from="A", to="B", quantity=10)
check_consistency(nodes, flows)
```

---

**MFA**

*Material Flow Account sample data*

**Description**

Dataset containing sample material flow account data, formatted for use within 'PantaRhei'

**Usage**

```r
MFA
```

**Format**

A list with three data frames:

- `nodes`
- `flows`
- `colors`
Description

Please read the user manual for more information.

parse_flows

Parse the information from a 'flows' definition table.

Description

This function checks the content of a flows definition, and appends some missing columns. It is mainly used internally, but can be invoked by the uses to see what it does.

Usage

parse_flows(flows, verbose = FALSE)

Arguments

flows data.frame containing the nodes definition
verbose logical: print some information?

Value

modified flows data.frame

Examples

Q0 <- data.frame(from="A", to="B", qty=10) # Note 'qty' as alias for quantity
str(Q0)
Q1 <- parse_flows(Q0)
str(Q1)
**parse_nodes**

*Parse the information from a 'nodes' definition table.*

**Description**

This function checks the content of a nodes definition, and appends some missing columns. It is mainly used internally, but can be invoked by the uses to see what it does.

**Usage**

```r
parse_nodes(nodes, verbose = FALSE)
```

**Arguments**

- `nodes` data.frame containing the nodes definition
- `verbose` logical: print some information?

**Value**

modified nodes data.frame

**Examples**

```r
n0 <- data.frame(ID=c("A","B"), x=1:2, y=0)
str(n0)
n1 <- parse_nodes(n0)
str(n1)
```

**parse_palette**

*Parse the information from a 'palette' definition table.*

**Description**

This function checks the content of a palette definition, and appends some missing columns. It is mainly used internally, but can be invoked by the uses to see what it does.

**Usage**

```r
parse_palette(palette, verbose = TRUE)
```

**Arguments**

- `palette` data.frame containing the palette definition
- `verbose` logical: print some information?
Value

modified palette data.frame

Examples

```r
p0 <- data.frame(substance="any", color="red")
str(p0)
p1 <- parse_palette(p0)
str(p1) # Should be the same!
```

sankey

Plots a Sankey diagram

Description

Plots a Sankey diagram

Usage

```r
sankey(
  nodes,
  flows,
  palette,
  node_style = list(),
  title = NULL,
  legend = FALSE,
  page_margin = 0.1,
  max_width = 0.2,
  rmin = 0.2,
  copyright = NULL,
  grill = NULL,
  verbose = FALSE
)
```

Arguments

- **nodes** data.frame, containing the nodes definition
- **flows** data.frame, containing the nodes definition
- **palette** data.frame, containing the nodes definition
- **node_style** list: containing node style specifiers:
  - **type** Character: Node type; possible values are "box", "bar" and "arrow".
  - **length** numeric: node length, as fraction plot size (default: 0.1).
  - **gp** an object of class gpar, typically the output from a call to the function gpar(). This is basically a list of graphical parameter settings, describing the colors etc of the node.
**label_pos** character: label position. values: auto, above, below, left, right, none.

**label_anchor** character: label position (overrides label_pos). Values are NW, N, NE, W, E, SW, S, SE.

**label_align** character: label alignment with respect to label_anchor. Values are NW, N, etc.

**label_gp** an object of class gpar, describing the font and color of the label text.

**mag_pos** similar to label_pos, but controls location of the node magnitude. Value inside plots the node magnitude inside the node. Value label plots the node magnitude beneath the node label.

**mag_anchor** similar to label_anchor.

**mag_align** similar to label_align.

**mag_gp** similar to label_gp.

**magFmt** character: format string for the node magnitude. default: "%.1f". see ?sprintf for more information.

**title** character: plot title. use strformat() to specify formatting.

**legend** logical or gpar: Specifies the plotting of a legend. valid values are NULL (default; no legend), TRUE (plot a legend using standard text size and color), or the output of a call to gpar(), to control legend text size and color.

**page_margin** numeric: Page margin. Either a scalar, an (x,y) vector or an (left,bot,rt,top) vector

**max_width** numeric: Maximum width of the flow bundles, in fraction of the plot size

**rmin** numeric: Minimum radius for flow path bends (as fraction of the diagram’s units)

**copyright** character: optional copyright statement?

**grill** logical: Plot a coordinate grill?

**verbose** logical: print some diagnostic messages?

---

**Value**

The modified nodes data.frame

**Examples**

```r
nodes <- data.frame(ID=c("A","B"), x=1:2, y=0)
flows <- data.frame(from="A", to="B", quantity=10, substance="stuff")
sankey(nodes, flows)

colors <- data.frame(substance="stuff", color="blue")
sankey(nodes, flows, colors)
sankey(nodes, flows, legend=TRUE)  # Plots default legend
sankey(nodes, flows, legend=grid::gpar(fontsize=18, ncols=2))  # Large fonts; 2 columns
```
strformat  

Format a string

Description

This function adds formatting information to a character string by storing this information as the character string’s attributes. Run the example to see how it works.

Usage

strformat(s, ...)

Arguments

s  character string to be formatted

... formatting specifiers to be forwarded to gpar()

Details

All formatting specifiers work as if gpar() would be called. (It is, behind the screen.)

Value

formatted string

Examples

s <- strformat("Hello, World", fontsize=18, col="red")
str(s)  # show object structure
Index

* datasets
  MFA, 3

check_balance, 2
check_consistency, 3

MFA, 3

PantaRhei, 4
parse_flows, 4
parse_nodes, 5
parse_palette, 5

sankey, 6
strformat, 8