

Package: groupedChord (via r-universe)

October 19, 2026

Title Grouped Circular Flow Diagrams

Version 0.1.0

Description Create interactive bipartite circular flow diagrams with radial labels and an outer grouping ring. A common angular scale preserves link weights at both ends. Includes configurable colours, multiple widget instances, keyboard interaction, SVG export and Shiny bindings. Examples use entirely fictional workshop production data.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports htmlwidgets

Suggests shiny, testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

URL <https://aureliennicosiaulaval.github.io/groupedChord/>,
<https://github.com/AurelienNicosiaULaval/groupedChord>

BugReports <https://github.com/AurelienNicosiaULaval/groupedChord/issues>

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake make libuv1-dev

Repository <https://aureliennicosiaulaval.r-universe.dev>

Date/Publication 2026-10-07 12:23:15 UTC

RemoteUrl <https://github.com/AurelienNicosiaULaval/groupedChord>

RemoteRef HEAD

RemoteSha 9185a397033cc64b4a63a8a839e50b75b7520917

Contents

chord_palette	2
example_workshops	3
grouped_chord	3
groupedChordOutput	5
run_grouped_chord_demo	6
Index	7

chord_palette	<i>Stable named colours for circular diagrams</i>
---------------	---

Description

Generate a deterministic colour for each label. The result for a label does not depend on the other labels in the vector. Explicit palettes can also be passed to [grouped_chord()]. Names and the table supplement colour encoding.

Usage

```
chord_palette(labels, type = c("group", "source"))
```

Arguments

- labels Character labels to colour.
- type Palette role, either "group" or "source".

Value

A named character vector of hexadecimal colours.

Examples

```
chord_palette(c("Maison", "Jardin"))
```

example_workshops	<i>Entirely fictional production flows from workshops to objects</i>
-------------------	--

Description

Invented workshops, objects, families and quantities over three invented annual scenarios. Values are produced by a fixed arithmetic rule; no real project records, labels, proportions or dates were used. The unit is a fictional manufactured object. Values are illustrative, not observations.

Usage

```
example_workshops()
```

Value

A data frame of 108 rows with columns 'year', 'workshop', 'item', 'family' and 'quantity'. There are four workshops, eighteen objects and four object families. Each object has two workshop links per year.

Examples

```
d <- example_workshops()
head(d)
aggregate(quantity ~ year, d, sum)
```

grouped_chord	<i>Create a grouped circular flow diagram</i>
---------------	---

Description

Draw a bipartite flow diagram with straight radial labels and an optional outer grouping ring. Duplicate links are summed. Zero-weight rows are omitted. A target must belong to exactly one group. Missing, negative or non-finite weights are rejected rather than silently changed.

Usage

```
grouped_chord(
  data,
  source = "source",
  target = "target",
  value = "value",
  group = NULL,
  view = c("targets", "groups"),
  source_colours = NULL,
  group_colours = NULL,
  group_labels = NULL,
```

```

labels = list(source = "Sources", target = "Destinations", group = "Groupes"),
title = NULL,
unit = "poids",
footnote = character(),
locale = "fr-CA",
show_table = TRUE,
show_controls = TRUE,
source_label_width = 22,
context = list(),
width = NULL,
height = NULL,
elementId = NULL
)

```

Arguments

<code>data</code>	A data frame containing the links.
<code>source, target, value</code>	Names of the source, target and numeric weight columns.
<code>group</code>	Optional name of a column grouping the targets.
<code>view</code>	Either <code>"targets"</code> for individual targets or <code>"groups"</code> for totals by group. The latter requires a grouping column.
<code>source_colours, group_colours</code>	Optional named colour vectors. Names must cover all visible source or group labels. Extra names are allowed, so the same palette can be used for every filtered subset.
<code>group_labels</code>	Optional named character vector of short group labels, used on narrow screens. Full labels remain in the key and accessible names.
<code>labels</code>	Named list with <code>'source'</code> , <code>'target'</code> and <code>'group'</code> headings.
<code>title</code>	Optional title included in the widget and its SVG export.
<code>unit</code>	Label for the weight unit, for example <code>"fictional objects"</code> .
<code>footnote</code>	Character vector describing sources or interpretation limits.
<code>locale</code>	Locale used for numeric labels, default <code>"fr-CA"</code> .
<code>show_table</code>	Show a collapsible table of the underlying links.
<code>show_controls</code>	Show the widget's SVG download link. Shiny applications may instead use a download handler and <code>'input\$<outputId>_svg\$xml'</code> .
<code>source_label_width</code>	Approximate maximum characters per source label line. Targets keep straight, unwrapped names.
<code>context</code>	Named list of application metadata included with SVG exports, for example the selected year. Do not put credentials in this metadata.
<code>width, height</code>	Widget dimensions. Its content height adapts to the labels.
<code>elementId</code>	Optional HTML element identifier.

Details

The same angular unit is used at both ends of every ribbon. The scale is recalculated for each selection. Lengths in different widgets cannot be compared directly; use the numeric weights. The outer ring is a classification track: its length includes space for group labels and does not encode a weight.

Each widget has independent SVG identifiers, selections and exports. In Shiny, a click emits ‘input\$<outputId>_click’, a list with a ‘type’ (‘source’, ‘target’, ‘group’ or ‘link’) and the corresponding labels. The SVG snapshot is in ‘input\$<outputId>_svg’, with ‘xml’, ‘total’, ‘view’ and ‘context’. Non-Shiny pages receive a ‘groupedchord:select’ DOM event.

Value

An ‘htmlwidget’ of class ‘groupedChord’.

Examples

```
library(groupedChord)
d <- example_workshops()
d <- d[d$year == 2025, ]
grouped_chord(d, "workshop", "item", "quantity", group = "family",
  title = "Ateliers et objets inventes", unit = "objets fictifs")
```

groupedChordOutput	<i>Shiny bindings for grouped circular diagrams</i>
--------------------	---

Description

Shiny bindings for grouped circular diagrams

Usage

```
groupedChordOutput(outputId, width = "100%", height = "auto")

renderGroupedChord(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	Output identifier.
width, height	CSS dimensions; the default height lets content adapt.
expr	Expression creating a [grouped_chord()] widget.
env	Environment in which to evaluate ‘expr’.
quoted	Whether ‘expr’ has already been quoted.

Value

An output tag or a Shiny render function.

Examples

```
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {  
  run_grouped_chord_demo()  
}
```

run_grouped_chord_demo

Run the entirely fictional workshop demonstration

Description

Run the entirely fictional workshop demonstration

Usage

```
run_grouped_chord_demo(...)
```

Arguments

... Arguments forwarded to [shiny::runApp()].

Value

The result of 'shiny::runApp()', invisibly.

Examples

```
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {  
  run_grouped_chord_demo()  
}
```

Index

chord_palette, [2](#)

example_workshops, [3](#)

grouped_chord, [3](#)

groupedChordOutput, [5](#)

renderGroupedChord
 (groupedChordOutput), [5](#)

run_grouped_chord_demo, [6](#)