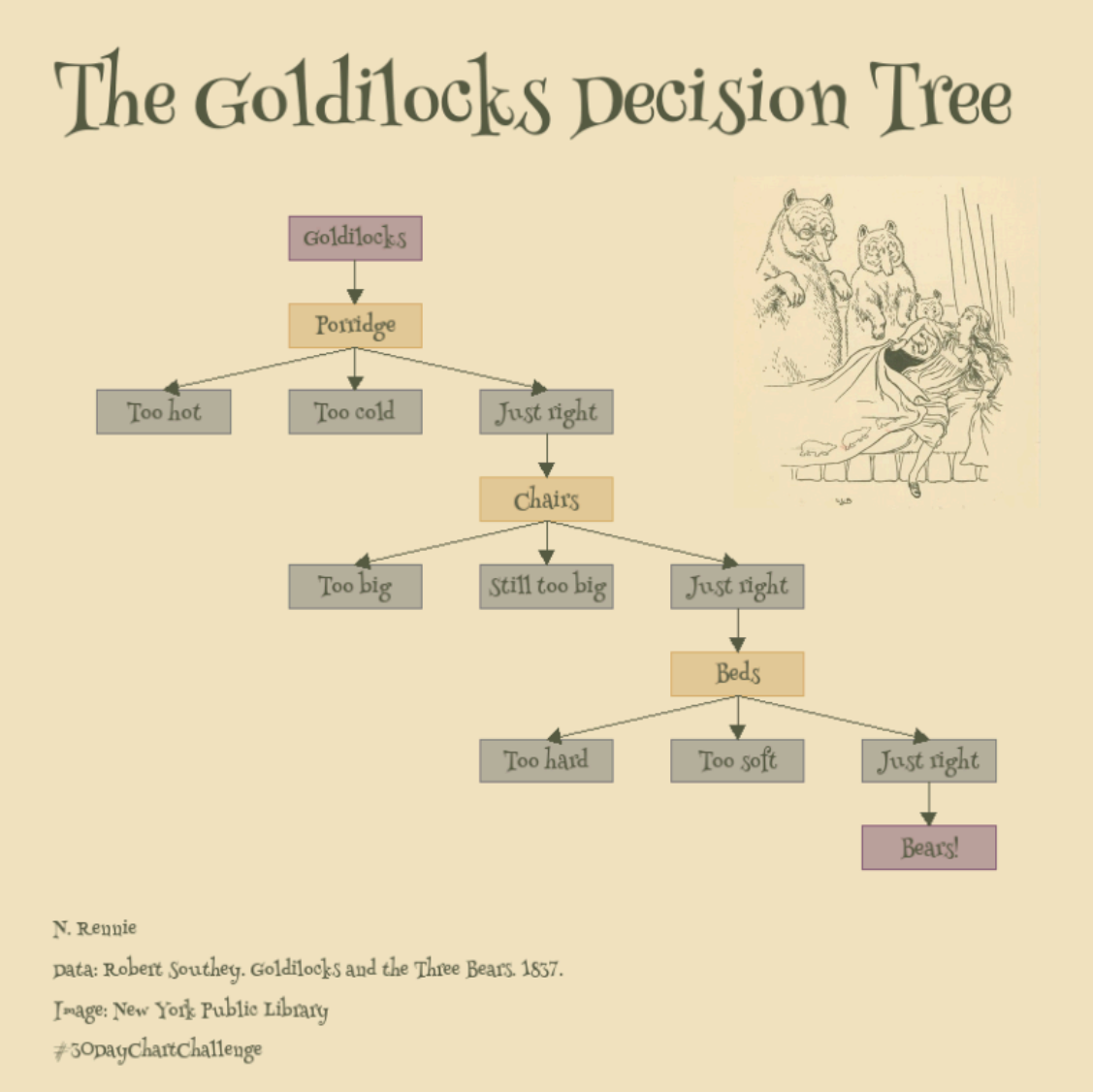
Introducing {ggflowchart}

Flowcharts can be a useful way to visualise complex processes, and the new R package {ggflowchart} makes them easy to create in R. This blog post shows you how.

May 12, 2023

Back in April 2022, I participated in the #30DayChartChallenge and for the *Storytelling* prompt on day 29 in the *Uncertainty* category, I created the **Goldilocks Decision Tree**. I also gave a talk to R-Ladies Nairobi on the challenge and used the flowchart as a live-coding example. A summary of the talk ended up as a blog post.



The reactions were pretty positive, and the suggestions from Twitter that it should become its own R package have been floating around in my mind since then. So here it is! {ggflowchart} - the package for creating simple flowcharts using {ggplot2}.

What does {ggflowchart} do?

Flowcharts can be a useful way to visualise complex processes. However, I couldn't find an easy way to create a flowchart in R. There are a few packages for either drawing basic components of flowcharts (like {grid}), packages that are great for visualising complex network data where order doesn't really matter (like

{ggnetwork} and {igraph}), but none of them gave me the control over customisation I was used to with {ggplot2}.

{ggflowchart} tries to fill that gap. The aim of {ggflowchart} is to help R users make simple, good-looking flowcharts, with as little code as possible. It computes a layout, then uses existing {ggplot2} functions to stitch together rectangles, text, and arrows.

Installing {ggflowchart}

As of 11 May 2023, {ggflowchart} is officially available on CRAN. You can install {ggflowchart} using `install.packages("ggflowchart")`.

You can also install the development version from GitHub:

Copy

```
1 remotes::install_github("nrennie/ggflowchart")
```

At the moment, {ggflowchart} has reasonably few dependencies (most of them common R packages you're already likely to have installed if you've been working with {ggplot2}).

- dplyr,
- igraph,
- ggplot2,
- rlang,
- tibble,
- tidyr

A few examples

To show you how {ggflowchart} actually works, let's go through a couple of small examples. The examples explained here are also included in the vignettes for future reference.

Copy

```
1 library(ggflowchart)
```

The simplest flowchart, which may well be all you need, takes only a tibble (or data frame) containing two columns. One column detailing where an edge in the flowchart begins, and the second column detailing where it ends. Ideally, you'll name these columns `"from"` and `"to"` but if not, {ggflowchart} assumes the first two columns of your tibble relate to the start and end points of edges.

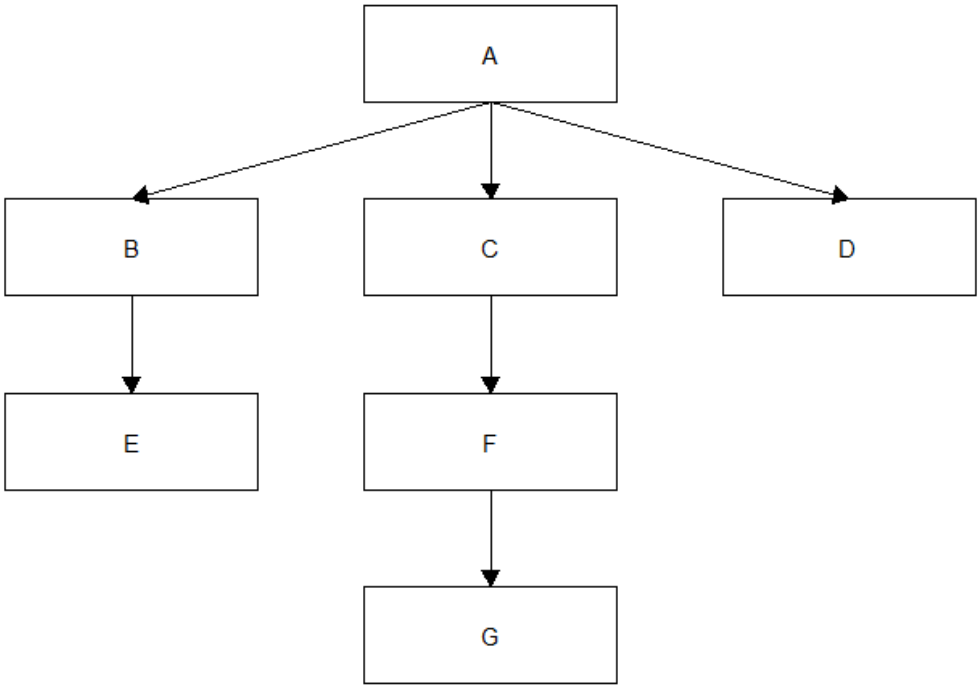
Copy

```
1 data <- tibble::tibble(from = c("A", "A", "A", "B", "C", "F"),
2                        to = c("B", "C", "D", "E", "F", "G"))
```

To construct the flowchart, you simply pass in the data frame of edges to the `ggflowchart()` function.

Copy

```
1 ggflowchart(data)
```



The `ggflowchart()` function does have a few additional arguments that you can use to change the appearance of the flowchart, shown below with their default values.

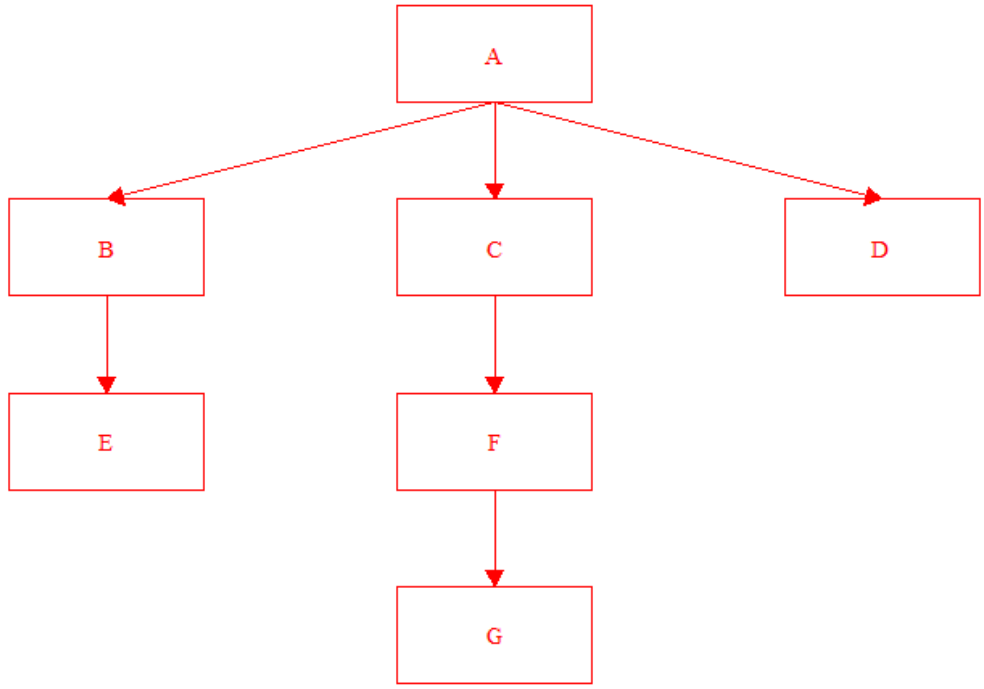
- `node_data = NULL`. An optional data frame specifying node attributes, including labels to appear in the boxes.
- `fill = "white"`. The fill colour of the node boxes.
- `colour = "black"`. The outline colour of the node boxes.
- `text_colour = "black"`. The colour of the text in the node boxes.
- `text_size = 3.88`. The size of the text in the node boxes.
- `arrow_colour = "black"`. The colour of the arrows between the node boxes.
- `arrow_size = 0.3`. The size of the arrows between the node boxes.
- `family = "sans"`. The font family of the text in the node boxes.
- `x_nudge = 0.35`. The width of the node boxes.
- `y_nudge = 0.25`. The height of the node boxes.
- `horizontal = FALSE`. The direction of the flowchart.

Additional arguments `color`, `text_color`, and `arrow_color` allow alternate spellings of colour.

For example, we may wish to switch to a red flowchart, with a sans serif font, and slightly more square boxes.

Copy

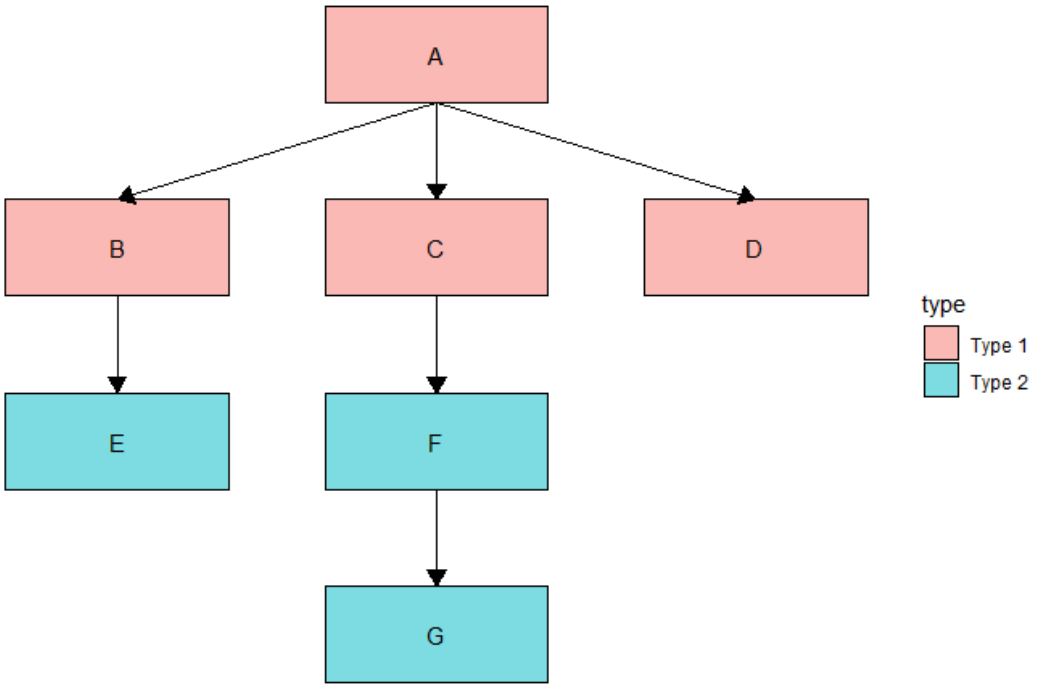
```
ggflowchart(data,
            colour = "red",
            text_colour = "red",
            arrow_colour = "red",
            family = "serif",
            x_nudge = 0.25)
```



The `fill` colour and `text_colour` can also be changed based on the name of a column in the `node_data` data frame. For example:

Copy

```
node_data <- tibble::tibble(
  name = c("A", "B", "C", "D", "E", "F", "G"),
  type = c("Type 1", "Type 1", "Type 1", "Type 1",
           "Type 2", "Type 2", "Type 2")
  )
ggflowchart(data, node_data, fill = type)
```



The column names to colour by can be either quoted or unquoted, e.g. `ggflowchart(data, node_data, fill = "type")` will produce the same result. Column names take priority over colour names. So if you have a column in `node_data` called `"blue"` - it will use the values in that column rather than colouring all nodes blue.

A more complex example showing how to change the layout using `scale_x_reverse()`, add titles, change the background colour, and edit the labels is included in another vignette.

What's coming next?

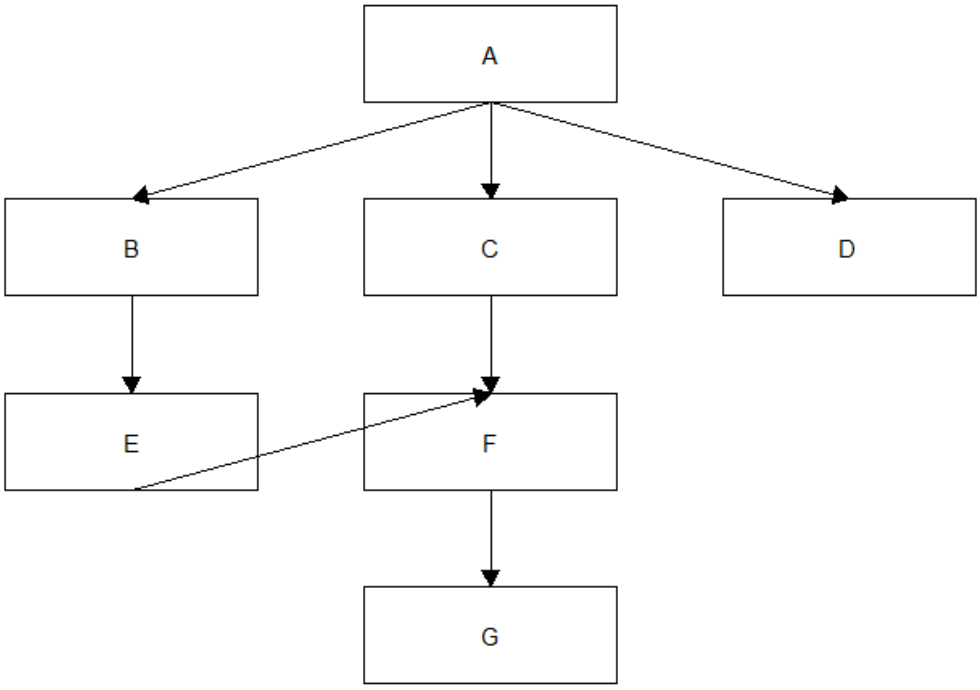
{ggflowchart} is currently a work in progress, and there are already a few things on my list to add into the next release! Upcoming features will include:

- Changing `linetype` and `arrow_colour` of arrows based on edge attributes.
- Changing `colour` of the node outline based on node attributes.
- Arrows between nodes on the same level.
- Ability to define custom layouts.

Some of these features are easier than others, and some will require a bit of thought about design choices. Changing the arrow colours and node outline colours is a little bit tricky because we may end up with three separate `colour` scales here. Layouts are currently based on the tree output from {igraph} but users may wish to move some of the nodes around. Hopefully soon users could define their own `x` and `y` coordinates to position the boxes. Arrows between boxes on the same level, e.g. between `E` and `F` in the minimal example are currently a bit hit or miss…

Copy

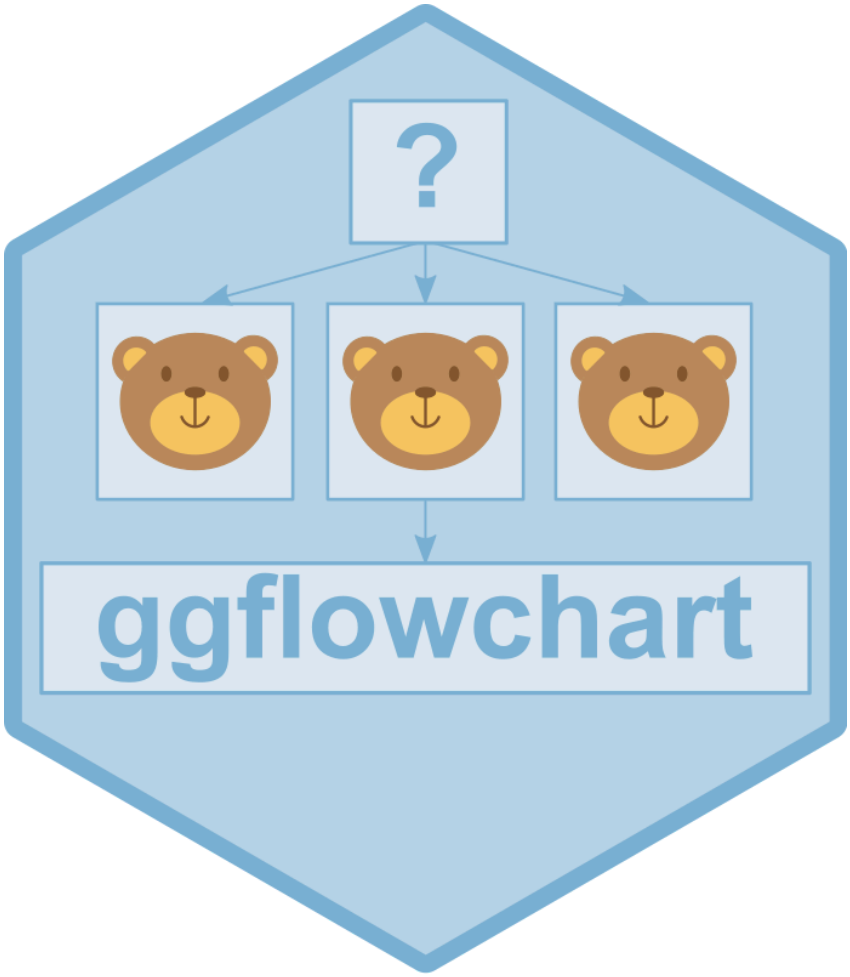
```
data <- tibble::tibble(from = c("A", "A", "A", "B", "C", "F", "E"),
                       to = c("B", "C", "D", "E", "F", "G", "F"))
ggflowchart(data)
```



These features are currently listed as issues on GitHub, and I will slowly work my way through them. If you have other suggestions for new features, or if you find a bug, please create an issue in the GitHub repository.

The most important part…

Of course, the most important part of any R package is the hex sticker! As a nod to the *Goldilocks Decision Tree* flowchart that inspired the package in the first place, the hex sticker for {ggflowchart} features three bears!



← Creating a cracked egg plot using {ggplot2} in R Learning Julia with #TidyTuesday and Tidier.jl →