tidyHeatmap: an R package for modular heatmap production based on tidy principles

Stefano Mangiola*1, 2 and Anthony T. Papenfuss1, 2, 3, 4

Background

The heatmap is a powerful tool for visualising multi-dimensional data, where individual values can be organised in a two-dimensional matrix and their values expressed as colours. Rows and columns can be ordered according to their reciprocal similarity using hierarchical clustering, and dendrograms can be added to the plot to facilitate the interpretation. Row- and column-wise visual annotations, such as coloured tiles, can also be included. Within the R environment, several packages have been developed to produce heatmaps. The simplest and most readily available tool, heatmap, is provided within the stats package (R Core Team, 2013) and offers basic heatmaps with simple tile annotations. The versatile package ggrepel2 can be also used to produce basic heatmaps (Wickham, 2016). More powerful software exists for producing fully annotated and/or multi-panel heatmaps, such as Pheatmap (Kolde, 2012), superheat (Barter & Yu, 2018) and ComplexHeatmap (Gu, Eils, & Schlesner, 2016). The versatility of these packages comes at the cost of adding complexity to the user interface, characterised by many parameters and annotation functions that introduce a steep learning curve to produce complex, clear, and good-looking graphics.

Statement of need

Considering (i) the utility and complexity of creating information-rich heatmaps, and (ii) the opportunity of increased coding efficiency and robustness offered by the tidy paradigm, a bridge between the two is very much needed. Recently, many tools for data science have been implemented according with tidy principles, this package aims to fill the gap for one of the most used data explorations tools.
Tidy paradigm for visualisation

tidyHeatmap is a R package that introduces tidy principles to the creation of information-rich heatmaps. It is available in the CRAN R repository. This package currently uses ComplexHeatmap as its graphical engine; however, due to its modular design it can be readily expanded to interface other engines. The command-line user interface is organised into (i) a main plotting utility; (ii) annotation layer utilities; and (iii) file-output utilities. The input is a tidy data frame with element (e.g., person), feature (e.g., physical characteristics) and value columns, with additional columns for independent variables for either elements (e.g., number of sport medals) or features (e.g., macroscopic or molecular characteristics). In this data structure, each observation is an element-feature pair.

<table>
<thead>
<tr>
<th>element</th>
<th>feature</th>
<th>value</th>
<th>annotation</th>
<th>group</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr</td>
<td>chr</td>
<td>fctr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>or</td>
<td>or</td>
<td>fctr</td>
<td>numeric</td>
<td></td>
</tr>
</tbody>
</table>

The input data frame streams along the utility path using the pipe operator from magrittr, allowing high modularity. The main utility allows the user to plot a base heatmap with dendrograms. The annotation utilities allow to serially add tile, point, bar and/or line annotation boxes to the side of the heatmap. The orientation of the annotations (row- or column-wise) is inferred based on the input data frame. The file-output utility allows the user to save vector or bitmap images directly from the R object, in the style of ggplot2. Row- or column-wise clusters can be defined effortlessly by applying the group_by function from dplyr (Wickham, François, Henry, & Müller, 2020) to the input data frame. Data transformation and row/column scaling is done internally. Together, this leads to a decrease of coding burden of 3 and 5 folds for lines and characters respectively compared to ComplexHeatmap (e.g., for Figure 1). Besides offering a modular and user-friendly interface, tidyHeatmap applies publication-ready aesthetics such as viridis (Garnier, 2018) and brewer (Neuwirth, 2014) colour palettes and automatic sizing of row and column labels to avoid overlapping (Figure 1).
Figure 1: Heatmap of the pasilla dataset including grouping and multiple annotations. Some annotation data was simulated for visualisation purposes.

The code interface consists of modular functions linked through the pipe operator. Custom colour palettes can be used by passing an array of colours or a colour function (e.g., circlize (Gu, Gu, Eils, Schlesner, & Brors, 2014)) to the palette argument of the annotation utilities.

```r
my_heatmap =

# Grouping
input_df %>%
group_by(pathway) %>%

# Plotting
heatmap(feature, element, value) %>%

# Annotation
add_tile(condition) %>%
add_tile(act) %>%
add_point(activation) %>%
add_bar(size) %>%
add_line(age)

# Saving
my_heatmap %>% save_pdf("my_file.pdf")
```

Conclusions

In order to perform complex tasks, the use of disjointed data structures demands time consuming and bug-prone information matching. Joint, tidy data frames decrease the cost/benefit ratio for the user, automating a large part on the data manipulation. tidyHeatmap introduces a modular paradigm for specifying information-rich heatmaps, just requiring column names as input. Due to its intuitive user interface and its advanced default aesthetic features, tidyHeatmap is ideal for the quick production of publication-ready heatmaps. This software is designed for modular expandability. Future directions include the incorporation of more static and interactive heatmap visualisation engines.

Acknowledgements

We acknowledge contributions all the Papenfuss Lab for feedback, and Maria Doyle for constant support.

References


