aheatmap: a Powerful Annotated Heatmap Engine
Package NMF - Version 0.22

Renaud Gaujoux
December 4, 2014

Abstract
This vignette showcases the main features of the annotated heatmap engine implemented by the function aheatmap. This engine is a highly enhanced modification of the function pheatmap from the pheatmap package\(^1\), and provides convenient and quick ways of producing high quality and customizable annotated heatmaps. Currently this function is part of the package NMF, but will most probably eventually constitute a separate package on its own.

Contents

1 Overview
2 Preliminaries
   2.1 Installation
   2.2 Sample data
3 Components
4 Annotation tracks
5 Column/row ordering
   5.1 Hierarchical clustering and dendrograms
   5.1.1 Display
5.2 Forced order
6 Borders
7 Colours
8 Labels
9 Legends
   9.1 Colour scale
   9.1.1 Colours and breaks
   9.1.2 Look and position
   9.2 Annotations
10 Session Info

1 Overview
The development of the function aheatmap started as modification of the function pheatmap from the pheatmap package\(^2\). The initial objective was to improve and increase its capabilities, as well as defining a simplified interface that was more consistent with the R core function heatmap. It is evolving into a general, flexible, powerful and easy to use engine for drawing annotated heatmaps.

The function aheatmap has many advantages compared to other heatmap functions such as heatmap, gplots::heatmap2, heatmap.plus::heatmap.plus, or pheatmap:

- Annotations: unlimited number of annotation tracks can be added to both columns and rows, with automated colouring for categorical and numeric variables.
- Compatibility with both base and grid graphics: the function can be directly called in drawing contexts such as grid, mfrow or layout. We believe that this is a feature many R users will enjoy, and that is strictly impossible with base heatmaps.
- Legends: default automatic legend and colouring;

\(^1\)http://cran.r-project.org/package=pheatmap
\(^2\)http://cran.r-project.org/package=pheatmap
• Customisation: clustering methods, annotations, colours and legend can all be customised, even separately for rows and columns;

• Convenient interface: many arguments provide multiple ways of specifying their value(s), which speeds up developing/writing and reduce the amount of code required to generate customised plots (e.g. see ??).

• Aesthetics: the heatmaps look globally cleaner, the image and text components are by default well proportioned relatively to each other, and all fit within the graphic device – if not set to an unreasonably small size.

2 Preliminaries

2.1 Installation

The aheatmap function is currently part of the NMF package3, which can be installed from any CRAN mirror or from the GitHub repository4, for the development version, with the following commands:

```r
# latest stable
install.packages('NMF')
# development version
devtools::install_github('NMF', 'renozao', 'devel')
```

2.2 Sample data

For the purpose of illustrating the capabilities of the function aheatmap, we first generate some random data that we will use throughout the vignette:

```r
# data matrix
x <- rmatrix(20, 10, .rng = 1234)
an_col <- list(Groups = gl(2, 5))
```

3 Components

Annotated heatmaps essentially use grid graphics5, composing the global picture by putting together the following components (or viewports in grid language):

dendrograms clusters and order columns/rows;

annotations are additional tracks that provide extra information about each column/row according to some associated auxiliary data;

data matrix , i.e. the heatmap itself, shown as coloured cells;

labels associates each column/row with some textual information;

legends such as value scales or color code used for the data matrix or annotations;

other information like main title, sub-title, extra information pane.

Figure 1 shows a diagram of two possible grid layout that combined the above listed components into a complete annotated heatmap.

---

3http://cran.r-project.org/package=NMF
4http://github.com/renozao/NMF
5Except for drawing dendrograms, which are plotted using the proven and well optimised base function plot.dendrogram.
Annotations tracks

Column/row ordering

The rows and/or columns of heatmaps are generally ordered in a way that highlight shared value patterns. This ordering can be automatically computed from the data itself, using hierarchical clustering algorithms, or forced to match known groups/order. Arguments Rowv, Colv, as well as distfun and hclustfun for automatic clustering, control how the ordering is performed and displayed. They accept the same values as the base function aheatmap, mimicking its behaviour, but also supports other convenient ways of specifying ordering and highlighting data patterns, some of which are illustrated in the rest of this section. We refer to the corresponding argument description on the man page ?aheatmap for a list of all supported values.

5.1 Hierarchical clustering and dendrograms

Dendrograms display result of applying a hierarchical clustering algorithm to the rows or columns, typically using the base function hclust.

By default aheatmap performs hierarchical clustering and show the associated dedrograms of both rows and columns, using the “euclidean” distance and the linkage method “complete”. However more custom clustering can also be specified:

```r
# default
aheatmap(x)
# use different clustering method for rows
aheatmap(x, Rowv = c('manhattan', 'average'), info = TRUE)
# use externally computed clustering
hc <- hclust(dist(x, method = 'minkowski'), method = 'centroid')
aheatmap(x, Rowv = hc, info = TRUE)
```
5.1.1 Display

# cluster rows but do not show dendrogram
a heatmap(x, Rowv = FALSE)
# cut column dendrogram into 3 clusters
a heatmap(x, Rowv = FALSE, Colv = 3L)
a heatmap(x, Rowv = FALSE, Colv = -3L)

a heatmap also provides some convenient shortcuts to use the dendextend package\(^6\) and produce enhanced cluster highlighting, with colors, text and boxes:

# cluster highlighting
a heatmap(x, Rowv = "#4", Colv = FALSE, main = "Colored clusters")
a heatmap(x, Rowv = "#4!", Colv = FALSE, main = "Boxed clusters")
# highlight cluster #2 only adding some red text
a heatmap(x, Colv = list("#302", text_col = 'red', text = 'Cluster B'), Rowv = FALSE
    , main = "Partial highlithing")

---

\(^6\)http://cran.r-project.org/package=dendextend
A completely custom pre-formatted dendrogram can also be build and passed to `Rowv` (or `Colv`), controlling both ordering and display. For example, using the `dendextend` package\(^7\), one can format a dendrogram in a complex way and simply “plug” it into the row or column dendrogram panel:

```r
# use pre-formatted dedrogram
library(dendextend)
hc <- hclust(dist(t(x)))
hc <- as.dendrogram(hc) %>%
  set("nodes_pch", c(19,1,4)) %>%
  set("nodes_cex", c(2,1,2)) %>%
  set("nodes_col", c(3,4))
hc <- hc %>%
  set("branches_lwd", c(4,1)) %>%
  set("branches_lty", c(1,1,3)) %>%
  set("branches_col", c(1,2,3))
aheatmap(x, Colv = hc, Rowv = FALSE, main = "Custom dendrogram")
```

5.2 Forced order

Column and row order can also be forced to a given order by passing an integer indexing vector or `NA` for keeping the original order. It is important that the indexing vector is effectively an integer vector, as passing a numeric vector would only provides weights used when re-ordering the computed dendrogram. When weights are used, the resulting dendrogram is essentially the same, with branches (and leaves) ordered in different ways (see the two last heatmaps below).

```r
# original order
aheatmap(x, Rowv = NA, Colv = NA, main = 'Original')
# indexing vector
aheatmap(x, Rowv = NA, Colv = seq(ncol(x), 1), main = 'Forced (inverse)')
# not the same as numeric weight vector
aheatmap(x, Rowv = NA, Colv = as.numeric(seq(nrow(x), 1)), main = 'Values used as weights')
# compare with no weights
aheatmap(x, Rowv = NA, main = 'Default weighting')
```

6 Borders

Borders of the different elements can be controlled using argument `border`. Passing a single value draws a border on all relevant elements using the specified color, and a finer control is possible by providing a list of border graphical parameters for specific element separately (Figure 2):

- `cell` border around each cell in the data matrix;
- `matrix` border around the data matrix;

---
\(^7\)Code borrowed from [http://cran.r-project.org/web/packages/dendextend/vignettes/introduction.html](http://cran.r-project.org/web/packages/dendextend/vignettes/introduction.html)
**annCol** border around each cell in the column annotation;

**annRow** border around each cell in the row annotation;

**annLeg** border around each cell in the annotation legend(s);

**base** default specification to use for each element – if not otherwise defined.

---

**Figure 2:** Border control

---

### 7 Colours

### 8 Labels

### 9 Legends

Annotated heatmaps have two types of legends, one showing the colour-value scale used to visualise the data matrix and another one for the annotation tracks.

#### 9.1 Colour scale

The very principle of a heatmap is to bin data values into a certain number of intervals (or breaks), associating each of these with a given colour. The *colour scale* is the legend that provides details about how to read the resulting colour coded data matrix. As such, it serves multiple purposes:

- provide the mapping between colours and value intervals;
- show the actual range of displayed values;
- optionally show the overall distribution of values.
9.1.1 Colours and breaks

9.1.2 Look and position

As for other components in annotated heatmaps, the position of the colour scale is controlled by the argument `layout`, which can also be used to specify if the scale should expand over the full height/width or have a limited fixed size.

By default the scale is placed on the top-right corner of the data matrix, with a limited fixed size. Figure 3 illustrates how to easily obtain some other commonly used positions/look, through the use of special layout shortcuts. More options are available, as detailed in the manual page for `aheatmap`.

```r
# vertical on the right expanded over the full height
aheatmap(x, layout = '*')
# horizontal at the bottom-right corner
aheatmap(x, layout = '_')
# horizontal the bottom, expanded over the full width
aheatmap(x, layout = '_*')
# vertical on the left (when not using/showing row dendrogram)
# aheatmap(x, Rowv = NA, layout = '^')
```

Figure 3: Colour scale alternative layouts: the scale can be placed in different areas around the data matrix and expanded to full height/width.

9.2 Annotations

10 Session Info

- R version 3.1.2 (2014-10-31), x86_64-pc-linux-gnu
- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C, LC_TIME=en_ZA.UTF-8, LC_COLLATE=en_US.UTF-8, LC_MONETARY=en_ZA.UTF-8, LC_MESSAGES=en_US.UTF-8
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, utils
- Other packages: BH 1.54.0-5, bigmemory 4.4.6, bigmemory.sri 0.1.3, Biobase 2.26.0, BiocGenerics 0.12.1, cluster 1.15.3, colorspace 1.2-4, dendextend 0.17.5, knitr 1.8, NMF 0.22, pkgmaker 0.25.9, RColorBrewer 1.0-5, registry 0.2, rngtools 1.2.4, synchronicity 1.1.4
- Loaded via a namespace (and not attached): codetools 0.2-9, digest 0.6.4, doParallel 1.0.8, evaluate 0.5.5, foreach 1.4.2, formatR 1.0, ggplot2 1.0.0, grid 3.1.2, gridBase 0.4-7, gtable 0.1.2, highr 0.4, iterators 1.0.7, magrittr 1.5, MASS 7.3-35, munsell 0.4.2, plyr 1.8.1, proto 0.3-10, Rcpp 0.11.3, reshape2 1.4, scales 0.2.4, stringr 0.6.2, tools 3.1.2, whisker 0.3-2, xtable 1.7-4