Post01

Joe Kiefer

October 31, 2017

Intro

Topic: Producing interactive heat maps in R with 'heatmaply' package

Description: This post will discuss how to produce visually pleasing, comprehensive, and accurate heat maps for different types of data sets and data types. In-text citations will be of format ("Name of Article") since most of the info comes from articles/tutorials written by the creator of the package, Tal Galili.

Background

I am currently taking an ecological analysis class in order to fulfill my Biological Studies breadth requirement for Letters and Science. In the class, one of our lessons focused on different graphing and visalization methods. This included some examples of "heat maps," which in the context of the class were used for measuring weather patters, salinity of lakes and oceans, surface areas of a region that receive the most amount of rainfall, etc. In looking more at heat maps, I found that R has a package (among others) called 'heatmaply,' recently updated, that allows the user to produce these types of maps with interactive and customizable parameters.

The 'heatmaply' package is the only thing needed to get started, although similar maps can be created using 'gplots::heatmap.2' (see "Introduction" article in References section for more info).

```
library("heatmaply")
```

Heat map

A "heat map" is a representation of values from a data set or matrix using color to demonstrate certain levels or concentrations of values ("heatmaply: an r package"). Usually, the higher the value or concentration, the "hotter" the color on the map (yellow, red, orange).

heatmaply()

The heatmaply package allows the user to create interactive heat maps for data tables, frames, sets, matrices, etc.

Basic interactive operations

Moving the cursor over any part of the heat map will display the corresponding value for the appopriate column and row at that point ("Introduction to heatmaply"). For example, hovering over the center of the map will display the amount of gears (column) for the Plymouth Valiant (row) as 3.000 (the Valiant was a 3-speed automatic car this year).

When hovering, the cursor appears as a plus-sign (+). Clicking and dragging anywhere on the heat map will highlight the selected section, which will "zoom in" on the map and only show the rows and columns highlighted. Double-clicking on the map returns the view from zoomed-in to the entire map.

Hovering in the top right corner of the map will display a quick-use tool bar that includes zoom, pan, toggle lines, axes reset, and download commands (downloads as .png file).

The package also displays dendrograms (treemap-looking arms extending from the top and right of the graph), which are groupings/clusters of samples that are considered by the program to be similar ("heatmaply: an R package"). For some sets of data this will not be particularly applicable. However, if measuring specific group variables (the examples commonly used are 'plant species' or 'gene types') dendrograms can display these groupings as part of the heat map ("heatmaply: an R package").

What we get from default settings

Using the mtcars data table from R, we can observe the default parameters of the basic heatmaply() function. The independent variables (names cars, player names, months, etc.) are listed on the vertical axis, and the dependent variable aspects are displayed on the horizontal axis. The color legend is displayed to the right of the heat map, and is colored based on the length of the column parameter. By default larger column values show up as "hotter colors" (red, yellow, orange). See default map of mtcars and NBA stats below.

```
heatmaply(mtcars)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```

The names of the cars are on the vertical axis, 10 measured parameters on the horizontal axis, and color coded legend to the right demonstrates the color on the map (in this case, length of the column parameter for each car. Higher values = hotter colors).

```
load('nba2017-salary-points.RData')
datstats <- data.frame(read.csv('https://raw.githubusercontent.com/ucb-stat133/stat133-fall-2017/master/data/nba2017-stats.csv', stringsAsFactors = FALSE))
nbastats <- data.frame(datstats[, c(2,3,4,7,10,13,16,17,18,19,20,21,22)])
```

```
heatmaply(datstats)
```

Obviously, this is an enourmous amount of data for one map. However, the principles and technical/visual aspects of the data are the same as the mtcars graph above. For this map, we observe row numbers on the left vertical axis and the features on the horizontal axis. Since names are included in the data frame, they are displayed next to their corresponding rows on the right vertical axis, and are not color-coded according to any values (this is default for character/factor vectors) (“Introduction to heatmaply”).

Missing values

As can be seen from the NBA map, missing values (NA, NaN, etc.) are not given a color-coding, and instead appear as the color white (“Introduction to heatmaply”).

Useful customizable parameters

Labels and margins

As with graphs in base R and ggplot2, labelling the axes and creating a title are possible in heatmaply. The same commands from base R apply: xlab, ylab, main (“Introduction to heatmaply”). Margins are also adjustable using the “margins =” command (see both below) (“Introduction to heatmaply”).

Transforming data

“Scaling” helps to create a heat map more focused on deviations and distribution. Using the “scale =” command reproduces the graph, but changes the values (and corresponding colors) so that they reflect standard deviation from the mean (“Introduction to heatmaply”). This can be done according to either columns or rows (“Introduction to heatmaply”).

```
heatmaply(mtcars, xlab = "Features", ylab = "Cars",
          scale = "column",
        main = "Scaling",
        margins = c(60,100,40,20))
```

The colors of the graph now correspond to how many standard deviations the values in the column are from the mean value of that column. This gives a good visual indication of distribution, mean, and outliers without having to calculate summary stats.

“Normalizing” helps to ensure that data parameters in the matrix/frame that are binary (0 or 1), non-normal in distribution, or large compared to the values of other parameters do not skew the data/color-coding and distort the relationship of the variables (“Introduction to heatmaply”). It can also create a 0 to 1 scale for parameters with only two or three variables so that they are color coded according only those variables in the column (“Introduction to heatmaply”).

```
heatmaply(normalize(mtcars), xlab = "Features", ylab = "Cars",
        main = "Normalization",
        margins = c(60,100,40,20))
```

Observe that parameters such as ‘am’ only have 2 variables, “automatic” or “manual,” and that the map now reflects this (automatic = 0 (dark blue), manual = 1 (yellow)). This type of map provides more of a ordinal “ranking” within the columns based on the difference between min and max values.

“Percentize” is similar to “normalize,” but instead of ranking the values as shown above, it will divide them by the maximum rank of that column, which will return a percentile rank (“Introduction to heatmaply”). As stated in the “Introduction” document, this is essentially portraying the “percent of observations that got that value or below it.”

```
heatmaply(percentize(mtcars), xlab = "Features", ylab = "Cars",
        main = "Percentize",
        margins = c(60,100,40,20))
```

As an example of how this may scew data if not used correctly, for the 'am' column (binary, only 0 or 1), the max value of 1 stays at the 100 percentile as 1, but all of the 0s are reverted to their proportion in the column (how many times 0 appears divided by total rows) ("Introduction to heatmaply"). This only happens when there are "ties" (same value) in the data, in which case the 'rank' command can be used instead ("Introduction to heatmaply"). A correct "percentization" is shown in the column 'qsec,' where percentile ranks are correctly applied. As is evident, the "percentize" command should be used with caution.

Colors

Using the 'colors =' parameter can change the colors indicated on the graph (in case you want a "cold map" instead) ("Introduction to heatmaply").

```
heatmaply(mtcars, xlab = "Features", ylab = "Cars",
          scale = "column",
        main = "Scaling new colors",
        colors = c(3,5,14),
        margins = c(60,100,40,20))
```

Useful applications

As mentioned before, heat maps are another relatively comprehensive, simple, and effective method of visually presenting density in data, frequently occuring values, and values based on amount/size. This is not only true for displaying parameters, but also for concepts such as correlation, average, deviations, and rankings (see above and below), where hotter colors make identifying high values easier. The presence of color coding helps for quicker analysis and identification of trends in the data than what would be more time-consuming with simple summary statistics or other graphs. For example, as seen above, the "scaling" method can give a good, quick-reference indication of mean, distribution, etc. before having to compute the true summary stats of the data frame. The commands and codes are relatively simple, allow for a wide range of customization, and follow closely with similar graphing commands for base R and ggplot2 so that you do not have to spend a significant amount of time learning new commands. Isolation of data is quick and easy with the interactive, click-and-drag map interface of the heat map, and the graphs themselves can be included in .Rmd files, export easily to external files and folders, and can be observed in the Viewer frame. Overall, use of this package can serve to greatly supplement data projects with comprehensive, interactive graphics, and is also especially applicable for dealing with data regarding weather, geography, biology (genes), or any of the data styles we work with in class.

Example correlation graph for mtcars

```
heatmaply(cor(mtcars), main = 'Correlation of parameters', colors = c(14,11,1),
          margins = c(40, 40),
          k_col = 2, k_row = 2,
          limits = c(-1,1))
```

Example correlation graph for stats of NBA players

```
heatmaply(cor(nbastats), main = 'Correlation of NBA stats',
          limits = c(-1,1))
```

k_row/k_column here correspond to the width of each column. For a full list of all commands and parameters, see the package description document included in the References section. This map demonstrates each column variables' correlation with the others (darker = more correlated) ("Package heatmaply").

For the NBA data, we could also observe other trends, such as how "negative stats" (fouls, turnovers) relate to each other, salary, minutes, and field goals made:

```
nbabad <- data.frame(salary, datstats$field_goals_made, datstats$fouls, datstats$turnovers, datstats$minutes)

heatmaply(cor(nbabad), main = 'Correlation of NBA stats 2',
          limits = c(-1,1))
```

As is shown, there are nearly endless ways to reproduce different types of visual aids for data analysis with this package in a visually-striking and understandable way.

Limitations

While this package is very powerful, the fact that it is interactive and very robust in terms of producing color, lables, etc. means it does run slower (at least on my computer) when compared to other visualization methods. There is a good amount of info put inside the viewing field, so without changing the default margin parameters, the map can look messy due to overlap of row names with axis labels. It also seems to require more CPU when multiple maps are open. Also, in order to export a file containing these heat maps directly to github, a separate package must be downloaded and utilized (see above frame connected to the first mtcars heat map that starts with “We recommend”), unless the command ‘always-allow-html’ is used in the yaml header.

Extra stuff, more examples/info

This package is very recently updated (within the last 2 weeks), so it will likely be updated again in the coming months as bugs are worked out, more commands/parameters are added, and so on. The references below contain the main documentation and introduction for the package. Also, a new blog has been created on the r-statistics website dedicated to solving and discussing problems with the package (https://www.r-statistics.com/tag/heatmaply/). Stackoverflow also has a running forum dedicated to the package (https://stackoverflow.com/questions/tagged/heatmaply).

References

Galili, Tal. “Package heatmaply.” cran.r-project.org, 26 Oct. 2017. https://cran.r-project.org/web/packages/heatmaply/heatmaply.pdf. Accessed 30 Oct 2017.

Galili, Tal. “Introduction to heatmaply.” cran.r-project.org, 26 Oct. 2017. https://cran.r-project.org/web/packages/heatmaply/vignettes/heatmaply.html. Accesed 29 Oct 2017.

Galili, Tal. “heatmaply: an R package for creating interactive cluster heatmaps for online publishing.” r-statistics.com, 30 Oct. 2017. https://www.r-statistics.com/2017/10/heatmaply-an-r-package-for-creating-interacti ve-cluster-heatmaps-for-online-publishing/. Accessed 31 Oct. 2017.