Review of the micromaps R package

Marcus W. Beck

June 25, 2015

To address:

- Do you find the syntax of the micromap package easy to understand?
- Could the main functions of the package, create_map_table, mmplot, and mmgroupedplot, be improved?
- Is there anything about the structure of micromap objects that could be improved for instance, in order to use micromap objects in R Shiny, or to make setting micomap plot options more expressive (such as adding plotting elements to a geom in ggplot2)?
- Is there some functionality of the package that you think is missing?

Ideas:

- Streamline use of the package functions for easier use, work only with `SpatialPolygonsDataFrame`. Retain options to include map and stat data separately if desired, but it should work with only a `SpatialPolygonsDataFrame` object.
  - It could be possible to create a very simple version of `mmplot` that works wtih only one input.
  - The `create_map_table` function could be used internally within `mmplot` to skip a step. I assume that this is used outside of `mmplot` because of the need to identify the 'id' column. I would suggest that the `create_map_table` function search for the 'id' column automatically based on a simple matching with the columns in the stats input. Again, this would be pretty simple if the input was a single `SpatialPolygonsDataFrame` object, i.e., the map and data attributes are already linked.
  - Data input that is linked to the map data could be provided in raw format, with a user-supplied summary function to improve flexibility and reduce burden on the user, e.g., additional arguments passed to mmplot or mmgroupedplot
  - Similarly, polygon simplification could be incorporated into the package, i.e., the raw spatial data in the `SpatialPolygonsDataFrame` could be used as input and simplified internally in the mmplot function using maptools or rgeos. Options to not do this could also be included so data that have already been simplified outside of R can still be used.

- I appreciate that the introduction guide provides instruction on creating new panel types. My perspective on package development is that it's useful to provide a means of customization, although users of the package will focus almost exclusively on the built-in features. I see two potential issues that limit the package from more effectively providing these tools. First, customization seems much too involved, which severely limits the number of users that can adopt the approach. Second, the built-in features rely too much on the format of the input data. Creating a better balance between these options could be the focus of further package development. I think a limitation of the current package, as noted above, is that the input data must already be summarized to work with each of the graph types. My suggestion is to modify the fundamental structure of `mmplot` to work with raw data. This could allow the user to specify a pre-existing plot type on the fly (e.g., dot, bar) without having the need to summarize the data beforehand. This could also allow the use of arbitrary functions to summarize the data. An example of this type of functionaly is provided by the `stat_summary` function of ggplot2. The user has the option to summarize data using any of a number of pre-existing functions (e.g., `mean_cl_boot`) or can provide their own in the call the the `fun.data` argument. I see the potential for micromaps to adopt these techniques if the raw data are included as input and a few additional functions are added to the package.

- I strongly encourage use of a formal object classification scheme for defining the functions (i.e., S3 or S4). This makes it much easier to determine what object classes can be used as input to each function. For example, the `create_map_table` function requires a `SpatialPolygonsDataFrame` object as input but this it not clear from the documentation - all that is specified is 'shapefile'. Defining an S3 method for this class should create a generic and `SpatialPolygonsDataFrame` definition in the help file and will also ensure that a sensible error is returned if an unsupported object class is passed to the function. This style of documentation is really easy using the roxygen and devtools packages. For example...

```
#' This is a function
#'
#' This is some more detail about the function
#'
#' @param input SpatialPolygonsDataFrame input
#' @param ... other arguments
#'
#' @export
mmplot <- function(input, ...) UseMethod(mmplot)

#' @rdname mmplot
#'
#' @export
#'
#' @S3method mmplot SpatialPolygonsDataFrame
```

```
mmplot.SpatialPolygonsDataFrame <- function(input, ...){

  # function stuff

}
```

- Related to the last point, the documentation could be improved in some places to clarify the exact object class required for the arguments to each function. For example, The help file for `mmplot` indicates that `stat.data` is a 'table' but this really should say 'data.frame'.

- This is more of a personal preference, but I'm not too crazy about a new graphics device being created each time a the function is executed. This can lead to rapid cluttering of the desktop.

- Maybe return a warning or error if colors does not equal length of groupings? Alternatively, the input colors could be passed to `colorRampPalette` to create a vector of colors equal in length to `grouping` based on an arbitrary number of color inputs. For example:

```
colors <- c("red", "blue")
grouping <- 5

# send this to each perceptual group
percep_cols <- colorRampPalette(colors)(grouping)
```

- Is the documentation in the Introduction Guide, which comes with the package, and the JSS article helpful?

  I had to do a bit of searching to find the Information Guide as it's buried in the R library. Maybe include the introduction guide as a vignette - will show up in searches and as a link on the CRAN web page. It can also be easily found with the following:

```
vignette('micromap')
```

  The introduction guide is helpful for showing the attributes that can be passed to `mmplot`, though it may be good to also include these in the help files for the package. My go-to for looking up function arguments are the help files yet I couldn't find any documentation outside of the introduction guide. Perhaps adding the attributes to the respective help files could be useful, e.g., all those under the 'dot' category on page 22 could go in the labels_att file.

- Seems like this package could be setup very nicely using an expressive style as in ggplot2 - this might be a lot of work though and probably requires substantial overhaul of package structure.

- Aspect ratio of maps changes with manual resizing of graphics device - this was caused by `scales = 'free_y'` in the `RankMaps` function. I suggested this is removed in a pull request to the EPA repo for micromaps. There is actually no reason to have free y scales on the maps since each facet has the same bounding region.
- From JSS manuscript - add histrograms, ts, symbol plots, etc. and incorporate interactivity (via shiny)
- Break up ideas into easy fixes and future features...