A Minimal Demo of glyphmaps

Garrett Grolemund

May 28, 2012

This document demonstrates the basic features of the `glyphmaps` package (name may change). `glyphmaps` was written to create plots like the ones below. Each was made with `glyphmaps`. The first two plots recreate plots from Wickham (2011). The last two recreate plots from Wickham et al. (Submitted).

To run the examples contained in this paper, you'll have to download the `glyphmaps` development package from `https://github.com/garrettgman/ggplyr` and install it with

```
library(devtools)
load_all("glyphmaps")

## Error: Can't find package glyphmaps
```

I haven't yet built the namespace or completed the documentation files, because the exact syntax / function membership hasn't been settled. As a result, the package will fail

```
Error in paste(repo, branch, sep = "/", collapse = ", ") :
  argument "repo" is missing, with no default
```

.

1 Glyphs

The above graphs are similar in that they are all built around glyphs. I define a glyph as a geom that visually summarises information about a subset of data. Glyphs can be quite simple, or very complex. Often glyphs contain their own internal coordinate systems. The star glyphs in Figure **??** illustrate how glyphs can contain an internal (minor) coordinate system and can still be plotted in an external (major) coordinate system.

2 glyphmaps geoms

`glyphmaps` introduces a number of new geoms that make it easy to create glyphs. These geoms can be used in conjunction with `ggplot2` functions to create layered plots. `glyphmaps` geoms behave just like `ggplot2` geoms, but take two additional arguments, described in Section 2.1.

2.1 new arguments

Each `glyphmaps` geom takes two new arguments,

```
Error in eval(expr, envir, enclos) : object 'glyphs' not found
```

and

```
Error in eval(expr, envir, enclos) : object 'reference' not found
```

.

```
Error in eval(expr, envir, enclos) : object 'glyphs' not found
```

controls how the data will be divided into subsets to be paired with individual glyphs.

```
Error in eval(expr, envir, enclos) : object 'reference' not found
```

lets the user draw reference features with the glyphs to facilitate comparisons.

```
Error in eval(expr, envir, enclos) : object 'glyphs' not found
```

takes the name of the variable(s) to be used to divide the data into the subsets that will correspond to different glyphs. The

```
Error in eval(expr, envir, enclos) : object 'glyphs' not found
```

argument works just like a

```
Error in eval(expr, envir, enclos) : object '.vars' not found
```

argument in a

```
Error in eval(expr, envir, enclos) : object 'ddply' not found
```

call. (In fact, the

```
Error in eval(expr, envir, enclos) : object 'glyphs' not found
```

argument in a `glyphmaps` geom will be passed into a

```
Error in eval(expr, envir, enclos) : object 'ddply' not found
```

call as the

```
Error in eval(expr, envir, enclos) : object '.vars' not found
```

argument.)

`glyphmaps` provides methods for drawing four types of reference objects behind glyphs: boxes, vertical lines, horizontal lines, and corner points. These objects help viewers make comparisons between data inside different glyphs.

3 Generalized glyphs

Users are not limited to the pre-packaged glyph geoms. Any single layer plot can be turned into a glyph using the

```
Error in eval(expr, envir, enclos) : object 'glyph' not found
```

and

```
Error in eval(expr, envir, enclos) : object 'plyr' not found
```

functions.

The plyr function also allows users to key aesthetics to subgroups of data.

4 Discussion - glyphs, embedded graphics, hierarchies

Glyphs are geometric objects (i.e, geoms) designed to display information within the geom. In other words, a glyph can display information even if it is drawn by itself, without references to an external coordinate system. In reality, all geoms are a type of glyph, but the term glyph is usually reserved for complicated geoms, such as those that contain their own internal coordinate systems.

Glyphs reveal a hierarchical structure of graphics: every plot is a collection of geoms, each of which can be thought of as its own self contained plot. Sometimes these subplots are not very interesting, as in the subplot created by a single point geom. At other times, thee subplots are quite complex, as in the star glyph of Figure **??**.

Graphs inherit this hierarchical structure from the data they describe. Data is produced through an iterative process of collecting observations, grouping observations and summarizing groups of observations to create more compact, information dense sets of data. Humans innately perform this process when collecting data; it is a cognitive pattern of the human brain which I will write about in the second cognitive chapter of my thesis.

Glyph maps simultaneously expose data from multiple levels of the hierarchy. As a set of geoms, the glyphs reveal relationships between data points in the higher level, compact data set. As individual plots, each glyph retains information about the data points in the lower level group of data that it summarises. This dual display makes glyph maps particularly useful for certain data analysis tasks. It also provides two different approaches to constructing glyph maps.

Glyph maps can be built from the top down by treating each glyph as a geom within the plot of interest. ggplyr provides new geoms $\texttt{geom}_s tar, \texttt{geom}_r adar, \texttt{geom}_d art, and \texttt{geom}_p lyr which allow a user to quickly incorporate geoms into an exis$

```
ggplyr's top down methods help users fit glyphs into existing methods of visualization.  These
methods provide new geoms (often based on new grobs) that can be used alongside existing geoms
in ggplot2.
   Top down methods are difficult to implement in ggplot2 for two reasons

First, ggplot2 calculates aesthetics on the entire data set at once.  However, glyphs contain
aesthetics that must be keyed to subsets of the data.

Second, the final width and height of individual glyphs often depends on non-position aesthetics,
such as angle and length.  In the ggplot2 pipeline, these aesthetics are scaled right before
the plot ranges are trained.  As a result, the final widths of glyphs must be computed at draw
time and frequently place parts of the glyph outside the plot window.
```

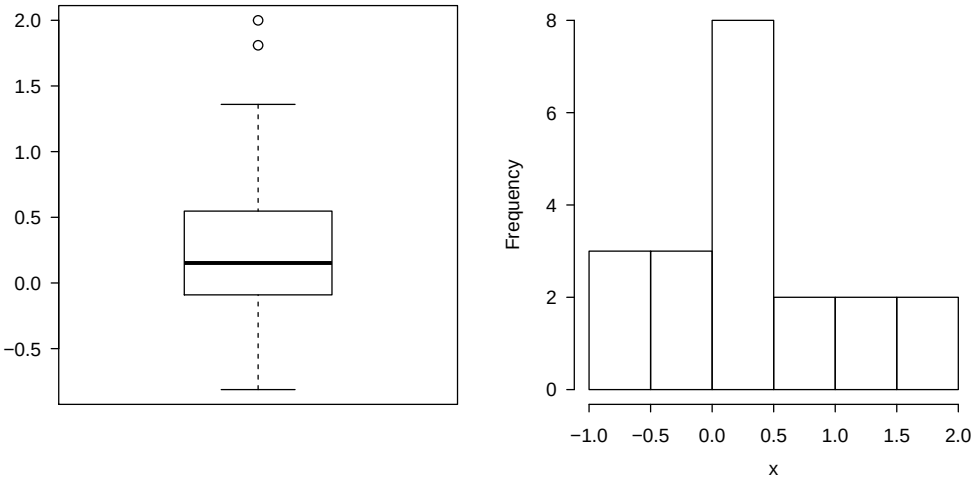
5.1 geom$_{p}lyr$

ggplyr provides a way for users to compute aesthetics by

```
set.seed(1121)
(x <- rnorm(20))

##  [1]  0.14496  0.43832  0.15319  1.08494  1.99954 -0.81188  0.16027  0.58589  0.36009
## [10] -0.02531  0.15088  0.11008  1.35968 -0.32699 -0.71638  1.80977  0.50840 -0.52746
## [19]  0.13272 -0.15594

mean(x)

## [1] 0.3217

var(x)

## [1] 0.5715
```

The first element of x is 0.145. Boring boxplots and histograms recorded by the PDF device:

```
## two plots side by side (option fig.show='hold')
par(mar = c(4, 4, 0.1, 0.1), cex.lab = 0.95, cex.axis = 0.9, mgp = c(2, 0.7, 0),
    tcl = -0.3, las = 1)
boxplot(x)
hist(x, main = "")
```



Do the above chunks work? You should be able to compile the TEX document and get a PDF file like this one: https://github.com/downloads/yihui/knitr/knitr-minimal.pdf. The Rnw source of this document is at https://github.com/yihui/knitr/blob/master/inst/examples/knitr-minimal.Rnw.

References

Hadley Wickham. The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, 40(1):1âĂŞ29, 2011. URL http://www.jstatsoft.org/v40/i01/.

Hadley Wickham, Heike Hofmann, Charlotte Wickham, and Diane Cook. Glyph-maps for visually exploring temporal patterns in climate data and models. *Environmetrics*, Submitted.