Introduction to ggplot2

N. Matloff

November 26, 2011

1 Introduction

Hadley Wickham’s ggplot2 package is a very popular alternative to R’s base graphics package. (Others include lattice, ggobi and so on.)

The ggplot2 package is an implementation of the ideas in the book, The Grammar of Graphics, by Leland Wilkinson, whose goal was to set out a set of general unifying principles for the visualization of data. For this reason, ggplot2 offers a more elegant and arguably more natural approach than does the base R graphics package.

The package has a relatively small number of primitive functions, making it relatively easy to master. But through combining these functions in various ways, a very large number of types of graphs may be produced.

The package is considered especially good in setting reasonable default values of parameters, and much is done without the user’s asking. Legends are automatically added to graphs, for instance.

The package is quite extensive (only a few functions, but lots of options), and thus this document is merely a brief introduction.

2 Installation and Use

Download and install ggplot2 with the usual install.packages() function, and then at each usage, load via library(). Here’s what I did on my netbook:

# did once:
> install.packages("ggplot2","/home/nm/R")
# do each time I use the package (or use .Rprofile)
> .libPaths("/home/nm/R")
> library(ggplot2)
3 Basic Structures

One operates in the following pattern:

- One begins with a call to `ggplot()`, which creates an R S3 object of class "ggplot",\(^1\) with a call

  \[ p \leftarrow \text{ggplot(your dataframe)} \]

  or

  \[ p \leftarrow \text{ggplot(your dataframe, aes(your args))} \]

  The resulting object \( p \) consists of a component named `data`, and other components containing information about the plot. Note that at this point, though, there is no plot, and nothing is displayed.

- One adds features to the plot via the + operator, which of course is an overloaded version of R’s built-in +, internally the function "+.ggplot". One can do this repeatedly, e.g. to superimpose several curves on the same graph.

- The function `aes()` (“aesthetics”) is used to map your data variables to graph attributes. This could be, for example, to specify which variable goes on the X-axis and which one in the Y-axis in a scatterplot, or for instance to indicate which variable will determine the color of any given point in the graph.

  One can set attributes in this way at various levels:

  - We can set attributes for the entire graph by calling `aes()` within our call to `ggplot()`, e.g. to specify which variables from our dataset we wish to plot.
  - We can set attributes specific to one of the + actions, such as specifying the color when we add a line to the picture.

  So for instance we could use `aes()` to specify our data variables either when we call `ggplot()`, so our data will apply to all operations, or when we call, say, `geom_point()`, to indicate data variables for this specific operation.

There are various types of objects that can be used as the second operand for the +. Examples are:

- **geoms** (“geometrics”): Geometric objects to be drawn, such as points, lines, bars, filled polygons and text.

- **position adjustments**: For instance, in a bar graph, this controls whether bars should be side by side, or stacked on top of each other.

- **facets**: Specifications to draw many graphs together, as panels in a large graph. You can have rows of panels, columns of panels, and rows and columns of panels!

- **themes**: Don’t like the gray background in a graph? Want nicer labeling, etc.? You can set each of these individually, but one of the built-in themes, or a user-contribute one, may prove to be to your liking, or you can write one that you anticipate using a lot.

\(^1\)The `ggplot2` package also offers a “quick and dirty” wrapper `qplot()` (“quick plot”), but this does not allow one to make full usage of the package’s capabilities.
4 Example: Census Data

The data set here consists of programmers (software engineers, etc.) and electrical engineers in Silicon Valley, in the 2000 Census. I’ve removed those with less than a Bachelor’s degree. The R object was a data frame named `pm`.

I first ran

```r
p <- ggplot(pm)
```

to set up the `ggplot` object. Next, I type

```r
p + geom_histogram(aes(HiDeg))
```

which produced a histogram of the highest-degree values of the workers:

![Histogram of highest-degree values](image)

Note that the + operation yields a new object of class "ggplot". Since the print function for that class actually plots the graph, the graph did appear on the screen. I could have saved the new object in a variable if needed.

I then decided to do a scatter plot of salary versus age:

```r
> p + geom_point(aes(x=Age, y=WgInc))
```

Note the role of `aes()` here; I used it to tell `geom_point()` which of my data variables would correspond to the X- and Y-axes.

This gave me this graph:
(As is often the case with large data sets, the points tend to “fill in” entire regions, one solution to which is to graph a random subset of the data, not done here.)

However, I wanted to separate the points according to highest degree level:

```r
> p + geom_point(aes(x=Age, y=WgInc, color=HiDeg))
```

Here I have three data variables informing `aes`: Age, wage income and highest degree. The argument `color` here means that I want the degree to be used for color coding the points:\footnote{Note that if I had wanted the same color to be set for all points, I'd set the `color` option \textbf{outside} the `aes()` call, as my second argument to `geom_point()`.

```r
2
```
Note the legend that was automatically included on the right.

Since some people might be viewing a black-and-white version of this document, I ran the command again, specifying point shape instead of point color:

```r
p + geom_point(aes(x=Age, y=WgInc, shape=HiDeg))
```

Here `ggplot2` decided to use a circle, a triangle and a square to represent Bachelor’s, Master’s and PhD workers:
Finally, since I’m interested in age discrimination in the industry, I decided to restrict my graph to those over age 40. The ggplot2 package cleverly exploits the R `subset()` function, allowing me to write

```r
p %>% subset(pm, Age > 40) + geom_point(aes(x=Age, y=WgInc, color=HiDeg))
```

The new operator `%>%` is mapped to ”+.ggplot”. The result was
Look at all those 0-income PhDs! (There was also a business income variable, which I did not pursue here, so maybe some had high incomes after all.)

5 For Further Information

Just plugging “ggplot2 tutorial,” “ggplot2 introduction,” “ggplot2 examples” and so on into your favorite search engine will give you tons of information.

Hadley’s book, ggplot2: Elegant Graphics for Data Analysis, is of course the definitive source, but also try his pictorial reference manual, at http://had.co.nz/ggplot2/.