Advanced plotting in R with ggplot2*

Pedro Alcocer

January 12, 2010
Winter Storm

ggplot2 [2] is a plotting system for R, based on the grammar of graphics [3], which tries to take care of many of the trivial details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics.

1 Preliminaries

Make certain you have installed the latest version of the packages that this document depends on with:

```
> install.packages("languageR")
> install.packages("ggplot2")
```

We will be using the `english` dataset from the `languageR` package. This data set gives mean visual lexical decision latencies and word naming latencies to 2284 monomorphemic English nouns and verbs, averaged for old and young subjects, with various predictor variables.

Learn more about the `english` dataset with:

```
> ?english
```

Actually inspect the dataset with:

```
> head(english)
     RTlexdec RTnaming Familiarity   Word AgeSubject WordCategory WrittenFrequency
1 6.543754 6.145044        2.37    doe      young            N         3.912023
2 6.397596 6.246882        4.43  whore      young            N         4.521789
3 6.304942 6.143756        5.60 stress      young            N         6.505784
4 6.424221 6.131878        3.87   pork      young            N         5.017280
5 6.450597 6.198479        3.93   plug      young            N         4.890349
6 6.531970 6.167726        3.27   prop      young            N         4.770685

> tail(english)
     RTlexdec RTnaming Familiarity  Word AgeSubject WordCategory WrittenFrequency
4563 6.608770 6.503839        3.70   spy        old            V         5.023881
4564 6.753998 6.446513        2.40   jag        old            V         2.079442
4565 6.711022 6.506979        3.17  hash        old            V         3.663562
4566 6.592332 6.386879        3.87  dash        old            V         5.043425
4567 6.565561 6.519884        4.97 flirt        old            V         3.135494
4568 6.667300 6.496624        3.03  hawk        old            V         4.276666
```

*This document is available online at http://pealco.net/downloads/ws2010_ggplot2.pdf

2 Building plots

Suppose we are interested in discovering how the lexical decision reaction times (`RTlexdec`) in this dataset are distributed. A histogram is a good way to represent a distribution. The x-axis should contain the reaction times and y-axis should contain counts of binned reaction times.

We begin a plot by describing what relationships we want to plot. The following command describes the relationship we're interested in (i.e., just the behavior of `RTlexdec`) and stores this description in the variable `p`. Calling `p` results in an empty plot window. This alone doesn't plot anything because we haven't specified the details of how to plot this abstract description.

```
> p <- ggplot(english, aes(x = RTlexdec))
> p
```

`ggplot2` thinks about plots as data relationships and ways to display those relationships. Typically, the data relationships are described with the `ggplot()` function. This function takes two arguments: (1) the data frame which contains your data, in this case `english` and (2) a description of the variables which you wish to plot within an `aes()` function call. In this case, we are only interested in plotting something along the x axis, so the call is `aes(x = RTlexdec)`.

How relationships are actually plotted is handled by the `geom` and `stat` family of commands. There are many geoms available to you.[1] For instance, to plot a histogram, we would add the geom `geom_histogram()` to the `p` object we created.

Figure 1: (Left) A histogram of the distribution of the `RTlexdec` response variable made with the `geom_histogram` geom. (Right) A smooth density estimate of the same, made with the `geom_density` geom.

```
> p + geom_histogram()
```

To plot a smooth density estimate, we use the `geom_density()` geom, instead.

```
> p + geom_density()
```

[1] For a complete list of geom and stat commands see the online `ggplot2` reference manual at `http://had.co.nz/ggplot2/`

2.1 Plotting two variables

Plotting two variables is similar to plotting one; it simply requires that you specify what should be plotted on the y axis. Below, we set up these relationships and store the abstract plot in `p2`. On the x axis, we'll have the written frequency of a word (`WrittenFrequency`) and on the y axis we'll have the lexical decision reaction time response variable `RTlexdec`.

```
> p2 <- ggplot(english, aes(x = WrittenFrequency, y = RTlexdec))
```

We can now plot this relationship on a scatterplot with the `geom_point()` geom.

```
> p2 + geom_point()
```

Likewise, without much trouble we can make two-dimensional histograms with square or hexagonal tessellations using the `geom_bin2d()` or `geom_hex()` geoms.

Figure 2: (Left) A scatter plot comparing written frequency (`WordCategory`) with log lexical decision reaction time (`RTlexdec`). (Center) A square-tessellated 2D histogram of the same relationship. (Right) A hexagon-tessellated 2D histogram.

So far, we haven't done much in `ggplot2` that we couldn't easily do in the base R graphics system (perhaps with the exception of the 2D histograms). Let's move into some territory where `ggplot2` excels.

3 Beyond base

Notice that the histogram and the density estimate in Figure 1 seem to have two peaks and that there seem to be two areas of concentration in the plots in Figure 2. This may indicate that we are looking at two overlapping distributions. `ggplot2` makes it very easy to split data by variable and display it in several ways.

3.1 Splitting variables into colors

Let's try to discover the predictor variable that is driving the two distributions. We'll consider three predictors: `WordCategory`, the category of the word that is being presented, noun or verb; `CV`, whether the word in question begins with a consonant or a vowel; and `AgeSubject`, whether the subject falls into the "young" age group or the "old" age group. We'll plot a smoothed density estimate and separate the two levels of the factor in question into different colors. [2]

[2] Notice that we are using the `aes()` command again. This command is really about *aes*thetic mappings between variables in the data and the visual properties of geoms. When we pass an `aes()` call to the `ggplot()` function, we are passing those aesthetic mappings

```
> p + geom_density(aes(color = WordCategory))
> p + geom_density(aes(color = CV))
> p + geom_density(aes(color = AgeSubject))
```

Figure 3: (Left) `WordCategory` (Center) `CV` (Right) `AgeSubject`

The rightmost plot in Figure 3 suggests that the two underlying distributions may be due to subject age. It seems that older subjects are slower at responding than younger subjects. We can see this again if we try a similar split on a scatterplot.

```
> p2 + geom_point(aes(color = AgeSubject))
```

Figure 4: A scatter plot with `AgeSubject` split into different colors.

Note that the predictor variable you are splitting by must be a factor in R. You will get an error if you try to split by a numerical predictor. You can, however, convert integer predictors into factors with `factor()`.

to any and all geoms that are added to that function. On the other hand, when we call `aes()` within a particular geom, the specified aesthetic mappings apply only to that geom.

3.2 Layering

Part of the flexibility that `ggplot2` offers is the ability to layer plots on top of each other. For instance, we can layer a 2D density contour on top of a scatter plot. In order to get an exceptionally good-looking result, we'll also be adjusting some cosmetic settings.

```
> # Set up a scatter plot with some transparency to prevent overplotting.
> # Note that color in this instance has nothing to do with variables, so doesn't
> # require aes().
> scatter <- geom_point(color = alpha("black", 1/3))
>
> # Set up a 2D density contour with line thickness 1, split by AgeSubject
> # into different colors. Note that size is outside aes().
> contour <- geom_density2d(size = 1, aes(color = AgeSubject))
>
> # Apply plots to data
> p2 + scatter
> p2 + contour
> p2 + scatter + contour
```

The rightmost plot in Figure 5 is a useful plot because it shows the raw data—no statistical opacity here—but also allows for a transparent and striking interpretation of that data: the two distributions are clearly overlapping. Furthermore these plots were generated with a minimal amount of syntax. Here we can really see the flexibility of the `ggplot2` package.

Figure 5: (Left) The first layer, a scatter plot. (Center) The second layer, a 2D density contour plot. (Right) The combined plot with raw data and contours.

3.3 Faceting

Faceting is another way to split data on a variable. Faceting separates two levels of a factor *in space* while keeping axes the same for easy comparison. It is an instantiation of Tufte's notion of "small multiples" [1]. For instance, let's facet a hexagonally-tiled 2D histogram by the `AgeSubject` variable.

```
> p2 + geom_hex() + facet_grid(. ~ AgeSubject)
```

Figure 6: A 2D hexagonally-tiled histogram, split by `AgeSubject`.

The `facet_grid()` layer takes something that looks like an R formula as an argument. A tilde (`~`) is required. A variable before the tilde will be tiled vertically; a variable after the tilde will be tiled horizontally. If you only want to facet one variable, place a period (`.`) in lieu of the second variable.

3.4 Stat layers

With `ggplot2` it is possible to plot the result of an impromptu (i.e., not pre-computed) statistical computation without much hassle. This is done with `stat` layers. The layer `stat_smooth()` computes a local regression (LOESS[3]) on the points specified by the `x` and `y` aesthetic mappings. It also computes a 95% confidence interval for this regression.

The simplest case involves adding a `stat_smooth()` layer to a scatter plot of points. The left plot in Figure 7 shows what this does. The LOESS regression assumes a non-linear regression in the data. As data becomes more sparse, the confidence intervals widen.

```
> p2 + geom_point() + stat_smooth()
```

We can get a bit fancier by throwing in some more aesthetic mappings. Now we'll color the points based on `AgeSubject` and we'll also have a regression line for each level in `AgeSubject`. The right plot in Figure 7 shows the result.

```
> p2 + geom_point(aes(color = AgeSubject)) + stat_smooth(aes(group = AgeSubject))
```

In fact, it is possible to compute any arbitrary statistic and plot it as a layer. If you can write a function for it, you can plot it. We'll see an example of this later.

[3] `http://en.wikipedia.org/wiki/Local_regression`

Figure 7: (Left) `stat_smooth()` on all the data. (Right) `stat_smooth()` split by `AgeSubject`

4 Real world examples

4.1 Self-paced reading

Self-paced reading results are usually presented as a line plot connecting the mean reading times at each sentence region, each condition getting its own line. Usually some measure of spread (e.g., standard error, confidence intervals) are presented at each point with error bars. This kind of complex plot is easily achievable with `ggplot2`.

```
> data.means <- read.csv("http://pealco.net/downloads/ws2010_spr.csv")
> head(data.means)
  region condition       RT        SE
1      a         a 556.5286  9.974514
2      a         b 544.1586 10.018679
3      a         c 540.4312 10.315718
4      b         a 538.8850 11.987531
5      b         b 548.2701 13.100119
6      b         c 540.8824 13.028911
> tail(data.means)
   region condition       RT       SE
31     i3         a 540.4339 12.38008
32     i3         b 551.4548 13.39948
33     i3         c 544.0227 13.85332
34      j         a 545.6344 10.41885
35      j         b 574.1764 11.76637
36      j         c 590.9825 12.31534
>
> # Describe the relationship that defines upper and lower limits for the error bars.
```

```
> limits <- aes(ymax = RT + SE, ymin = RT - SE)
> # Plot lines, points, and error bars. Always condition visual grouping by condition.
> ggplot(df, aes(x = region, y = RT, colour = condition, group = condition)) +
>     geom_line(size = 1, aes(linetype = condition)) +
>     geom_point(size = 3, aes(shape = condition)) +
>     geom_errorbar(limits, width = 0.1)
```

Figure 8: Self-paced reading data.

4.2 MEG

Here is some MEG data that was output by Cephalo.[4] It is from an MEG experiment that employed an oddball paradigm in order to elicit magnetic mismatch fields (MMFs). In these designs, one sound is played many times (the standard) and another sound is played occasionally (the deviant). Roughly speaking, if the perceptual system can detect a difference between the two sounds, then an MMF is elicited. This takes the form of a higher-amplitude waveform at about 200 ms after stimulus onset in deviant trials.

The data we're working with here has already been meaned into epochs by subject, channel, and condition. There is one subject, ten channels, and eight conditions. In total, there are 56080 observations.

```
> meg.data <- read.csv("http://pealco.net/downloads/ws2010_meg.csv")
> head(meg.data)
  subject condition sample channel hemisphere_x hemisphere_y factor0  factor1 amplitude
1   R1292        c1   -100    ch48         left     anterior    dlif standard -16.06218
2   R1292        c1   -100    ch76         left     anterior    dlif standard  13.48273
3   R1292        c1   -100    ch77         left     anterior    dlif standard  10.04500
4   R1292        c1   -100    ch79         left     anterior    dlif standard  34.84152
5   R1292        c1   -100    ch80         left     anterior    dlif standard -28.80825
6   R1292        c1   -100    ch85         left    posterior    dlif standard  12.98412
```

[4]MEG analysis software. http://pealco.net/cephalo

We are basically interested in seeing how the amplitude of the MEG signal changes over time. Time in this data set is encoded in the `sample` column and amplitude in the `amplitude` column. We are also interested (simultaneously) in the differences between conditions and the differences between the position of the stimulus (i.e., whether it occured as a standard or a deviant.) These are encoded in the columns `factor0` and `factor1`, respectively. Finally, we'd also like a measure of the magnitude of the brain's response to a stimulus. Since magnetic fields can be positive or negative and we purposefully choose both positive-going and negatively-going channels, we can't simply add amplitudes—they'll just cancel out. Instead we take the root mean square (RMS), which gets rid of polarity information while maintaining magnitude information. We'll add on a stat layer that computes this function for the plotted points.

```
> # Creat a root mean square function.
> rms <- function(x) { sqrt(mean(x**2)) }
>
> # Set up x and y aesthetic mappings
> meg.plot <- ggplot(meg.data, aes(x = sample, y = amplitude))
>
> # Apply a line geom, faceting, and a stat layer.
> meg.plot + geom_line(aes(group=channel), color=alpha("black", 1/4)) +
>            facet_grid(factor1 ~ factor0) +
>            stat_summary(fun.y = "rms", color = "red", geom="line")
```

Figure 9: MEG data.

Notice how this style of presentation makes it easy to compare standard epochs against deviant epochs. As has been the theme throughout this document, this advanced, complex, aesthetically-pleasing graphic was created with very little trouble to the data analyst. My hope is that I have been able to show how powerful a tool `ggplot2` is for creating data graphics.

References

[1] E.R. Tufte and G. Howard. *The visual display of quantitative information*. Graphics Press, Cheshire, CT, 1983.

[2] Hadley Wickham. *ggplot2: Elegant graphics for data analysis*. Springer-Verlag, New York, 2009.

[3] L. Wilkinson and G. Wills. *The grammar of graphics*. Springer Verlag, 2005.