The Grammar of Graphics

Pedro Alcocer

January 11, 2010

ggplot2 is a plotting system for R, based on the grammar of graphics, which tries to take the good parts of the standard graphics utilities and none of the bad parts. It takes care of many of the fiddly 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.

This module is intended to be a superficial introduction to ggplot2.

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
```

2 Building plots

Suppose we are interested in discovering how the lexical decision reaction times (`RTlexdec`) 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 binned counts of 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.

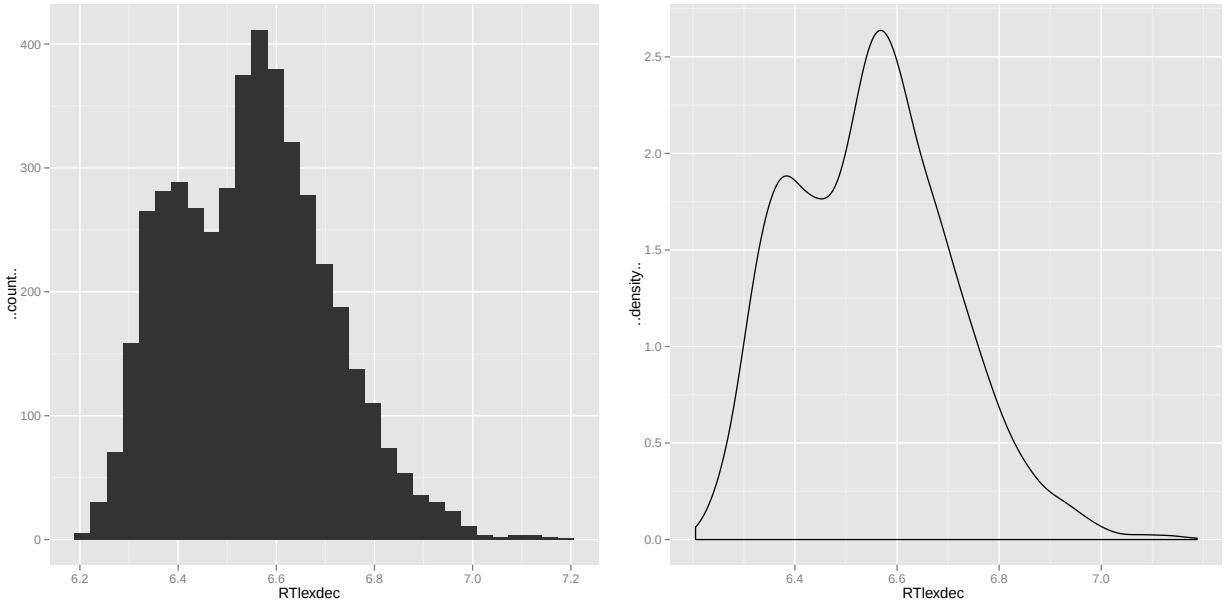


Figure 1: (Left) A histogram of the distribution of the `RTlexdec` response variable made with the `geom_histogram` geom. (Right) A smooth density estimate of the same, made with the `geom_density` geom.

```
> 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.

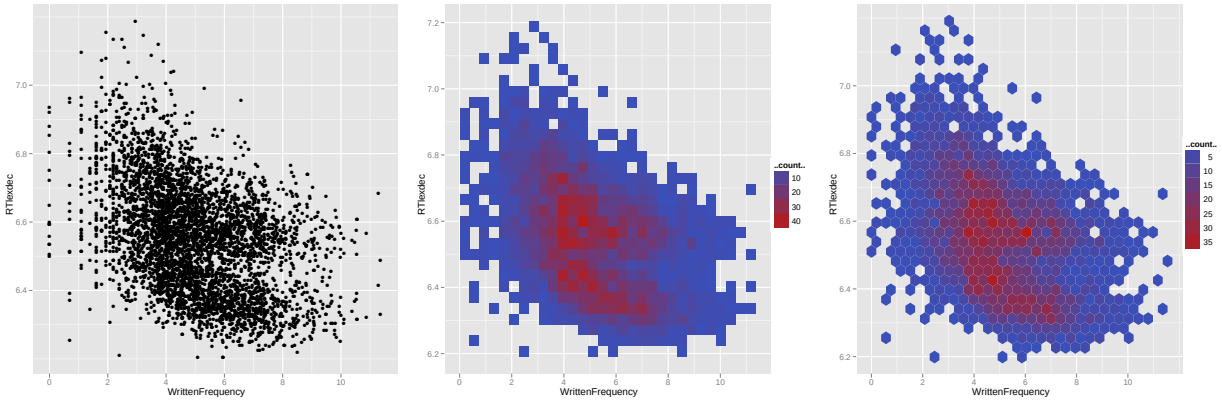


Figure 2: (Left) A scatter plot comparing written frequency (`WordCategory`) with log lexical decision reaction time (`RTlexdec`). (Center) A square-tessellated 2D histogram of the same relationship. (Right) A hexagon-tessellated 2D histogram.

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. Each level will be plotted in a different color. [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))
```

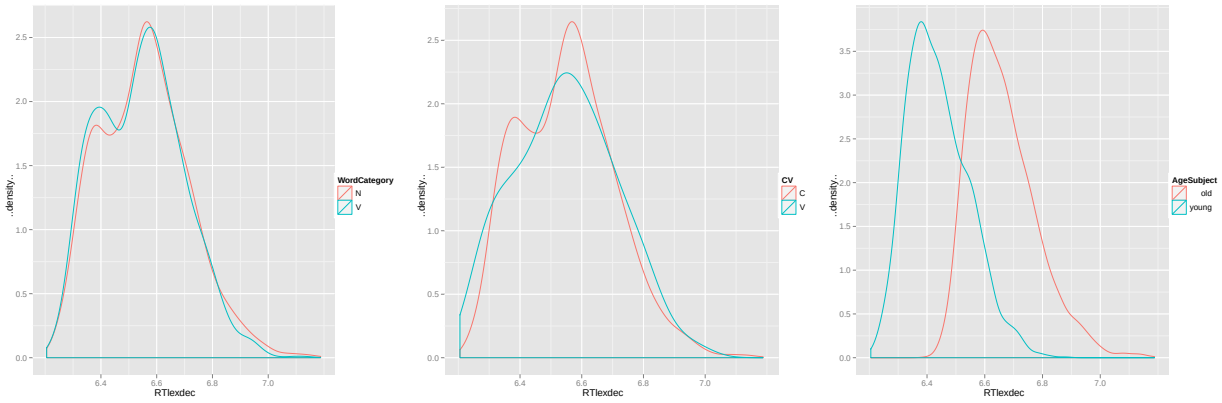


Figure 3: (Left) `WordCategory` (Center) `CV` (Right) `AgeSubject`

```
> p + geom_density(aes(color = LengthInLetters)) # Doesn't work.
> p + geom_density(aes(color = factor(LengthInLetters))) # Does what you expect.
```

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))
```

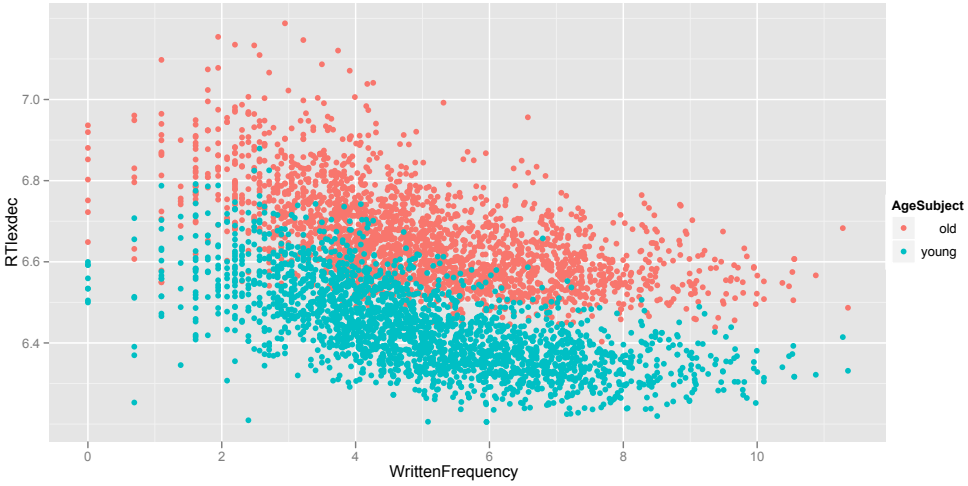


Figure 4: A scatter plot with `AgeSubject` split into different colors.

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 a good result, I'll also be adjusting some graphical 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 usefulness of the ggplot2 package.

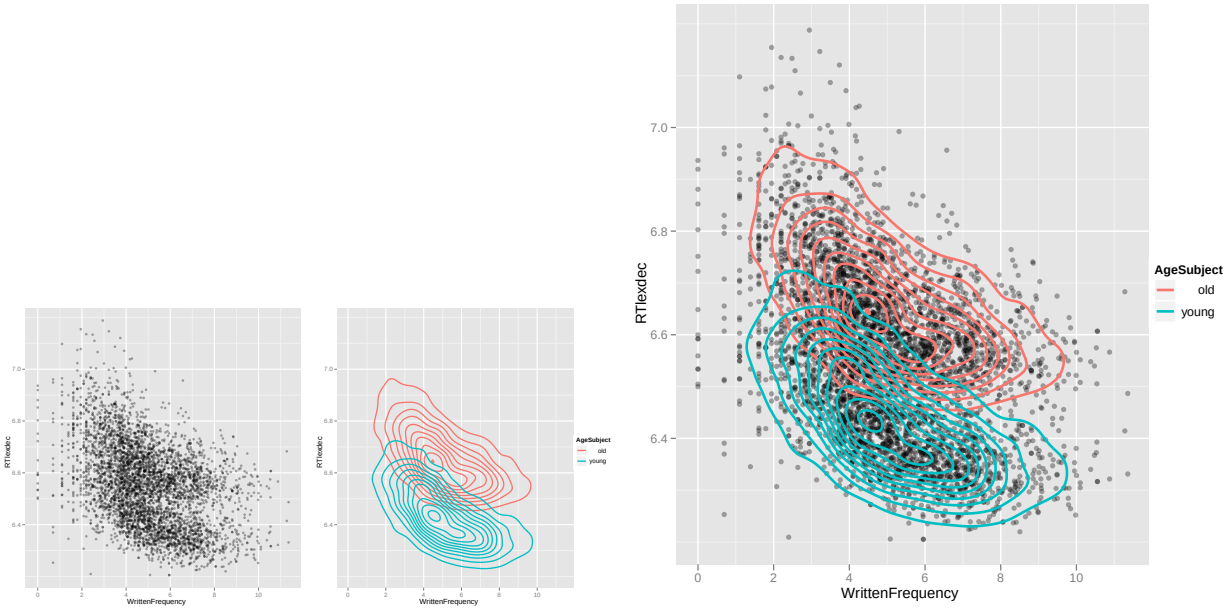
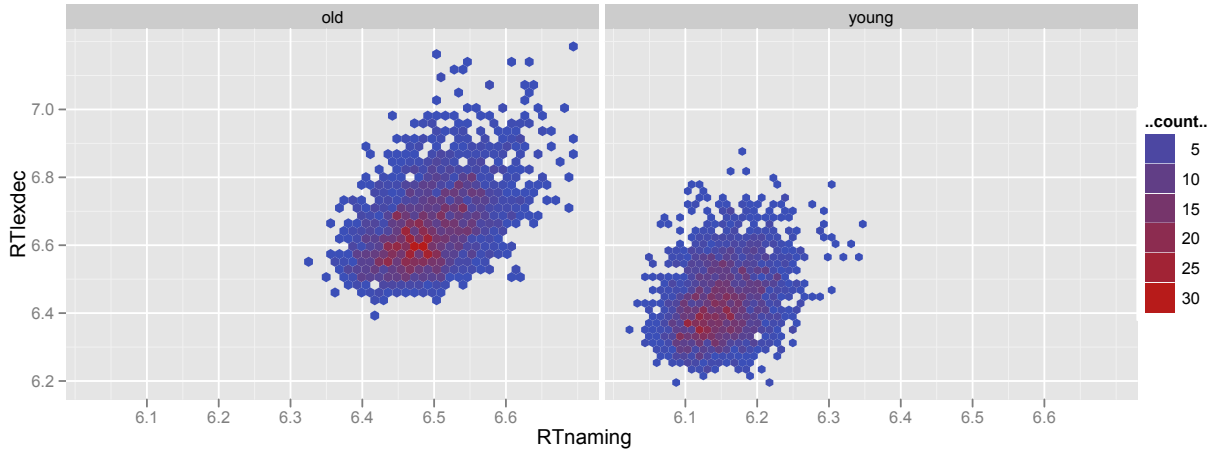


Figure 5: (Left) The first layer, a scatter plot. (Center) The second layer, a 2D density contour plot. (Right) The combined plot with raw data and contours.

3.3 Faceting

Faceting is another way to split data on a variable. Faceting separates two levels of a factor *in space*. For instance, let's facet a hexagonally-tiled 2D histogram by the `AgeSubject` variable.

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



4 Stat layers

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

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