Data visualization tutorial: exploring data and telling stories using ggplot2*

Peter Carbonetto *University of Chicago*

In this lesson, we will learn how to use ggplot2 to create simple yet effective data visualizations. The ggplot2 package is an incredibly powerful plotting interface that extends the base plotting functions in R. **This tutorial is also a Jupyter notebook in Google Colab.**

Some motivation

A good figure is an important part of an impactful research paper or presentation. A good figure is one that *tells an interesting story.*

Almost inevitably, creating a good figure takes iteration and refinement. You will rarely get it right on the first try.

For these reasons, taking the *programmatic approach* to creating plots is very powerful. It allows you to:

a. Create an endless variety of plots.
b. Reuse code to quickly create and revise plots.

In this tutorial we will explore the programmatic approach to plotting using **ggplot2**.

Setup

Do I have what I need? Download the tutorial materials from GitHub, and make sure you know where to find them.

If you have not already done so, install these packages:

```r
install.packages("ggplot2")
install.packages("cowplot")
install.packages("ggrepel")
install.packages("htmlwidgets")
install.packages("plotly")
```

I run this every time in Jupyter notebook or Google Colab to make the code outputs look like they do in RStudio:

*This document is included as part of the Basic Computing—Introduction to R tutorial packet for the BSD qBio Bootcamp, University of Chicago, 2023. **Current version**: August 24, 2023; **Corresponding author**: pcarbo@uchicago.edu. Thanks to Stefano Allesina, John Novembre, Stephanie Palmer and Matthew Stephens for their guidance.

```r
options(jupyter.rich_display = FALSE)
```

Do smaller dogs live longer?

The study of dogs is a surprisingly fruitful area of research! In this tutorial, we will make use of some data that was made available by the authors of a 2008 *Genetics* article, *Single-nucleotide-polymorphism-based association mapping of dog stereotypes.* These data are stored in a CSV file.

Our main analysis aim is to investigate the anecdotal claim that smaller breeds (such as Chihuahuas) live longer than larger breeds (such as Saint Bernards).

Our first ggplot plot

It will take us some time to understand *how* ggplot works, but let's start by quickly creating our first ggplot.

```r
library(ggplot2)
options(repr.plot.width = 4,repr.plot.height = 4,repr.plot.res = 180)
system("wget https://raw.githubusercontent.com/pcarbo/qBio9_stuff/main/dogs.csv")
dogs <- read.csv("dogs.csv",stringsAsFactors = FALSE)
ggplot(dogs,aes(x = height,y = weight)) +
  geom_point()
```

A first look at the data

When you load data into R for the first time, it is important to get a basic understanding of the data frame and its contents.

```r
# Add your code here.
```

What different types of data are in this table?

The often overlooked scatterplot

In this tutorial, we will learn about ggplot2 through one of the most basic data visualizations: *the scatterplot.*

The scatterplot is easily overlooked because it is so simple. But it can be one of the most effective ways to visualize relationships. And it has many uses.

With embellishments (adding labels, varying color, shape, size, *etc.*), scatterplots can produce stunning visualizations.

Our first ggplot2 (with "ugly" code)

Recall, our objective is to investigate the relationship between size and longevity in dogs. We'll use weight as a proxy for size.

```
# Add your code here.
```

Now for some more elegant code can accomplish the same thing. (This more elegant code comes with a "for experts only" warning.)

```
# Add your code here.
```

For the moment I want to focus on the "uglier" code because it highlights better the key elements of a ggplot2 plot:

1. The first input is the data (stored in a data frame).
2. The second input is an "aesthetic mapping", created using `aes` that defines how columns are mapped to features of the plot (axes, shapes, colors, *etc.*). *Note:* Many people use `aes` instead of `aes`.
3. A "geom", short for "geometric object", specifies the type of plot. ggplot2 has an excellent on-line reference at ggplot2.tidyverse.org explaining all the "geoms", from bar charts to contour plots, with code examples for each.
4. ggplot2 outputs a *ggplot object* `p`, which can be drawn to the screen with `print(p)` or just `p` (then hit "enter" or "return").

The distinguishing feature of ggplot2 is that plots are created by *adding layers*. This layering allows for infinite variety of plots to be created. The layering approach means that ggplot2 is easily extendible, and many R packages have been developed to enhance ggplot2. (We will use two of these packages, ggrepel and cowplot.)

A few improvements

No plot is ever right the first time. *What are ways we can improve our plot?* Add code for your improved plot here:

```
# Add your code here.
```

In ggplot, plots can be improved either by modifying the inputs to the functions you are already using (for example, `geom_point` has many settings that can be fiddled with), or by *adding layers*. The ggplot online reference has many helpful examples that illustrate the variety of ways plots can be improved.

Save your work

We've worked hard, and make considerable progress. This is a good point to save our work in an image file that can be shared with others. *What type of image file should we use?*

```
# Add your code here.
```

Plot the best-fit line

It has been estimated that an increase of 28 lbs in a dog's body weight corresponds to about a 1-year drop in expected lifespan (with a maximum lifespan of about 13 years). How well does this estimate agree with our data? Let's investigate this question by plotting the line that *best fits* these data (specifically, this is the "least-squares" fit).

```
# Add your code here.
```

Now let's add this "best fit" line to the plot. In case we make a mistake, let's call our new plot "p2" to avoid writing over our previous plot:

```
# Add your code here.
```

Now let's add another another "abline" layer to compare our estimate against the previous estimate (and name the new plot object "p3"):

```
# Add your code here.
```

Notice how easy it was to add layers to an existing plot!

Which breeds fit the trend, and which don't?

It would be helpful if we could tell which breeds are being plotted. Adding text labels to a ggplot is also done by adding a layer.

There's one catch here—there simply isn't enough real estate on the plot to accommodate all the breed names. This is a great opportunity to play around with a clever package, **ggrepel**, that adds the labels in a way that makes them more readable, and only adds them to the plot when possible. Let's appreciate how simply this sophisticated plot is created.

```
# Add your code here.
```

Notice that the labels are automatically redrawn as the plot is resized. Give it a try!

A QTL for weight

In the *Genetics* paper, the strongest *quantitative trait locus* (QTL) for weight was a QTL on chromosome 7. Using your ggplot coding skills, create a scatterplot to visualize the relationship between weight and the allele frequency at this QTL. *Is your code reproducible? i.e., could I run your code to get the same plot?*

```
# Add your code here.
```

Exploration: a surprising subtlety with color

So far, we have focussed on using the co-ordinate plane to visualize relationships. But other "aesthetics"—color, size, shape, *etc.*—can also be used to tell an evocative story. In the last chapter of this tutorial, we will explore the use of color to visualize relationships. In so doing, we will discover a complication.

We learned that the "shortcoat" column had values of 0 or 1 (with a few NAs).

Let's try varying the color of the points using the shortcoat column:

```
options(repr.plot.width = 5,repr.plot.height = 4,repr.plot.res = 180)
p <- ggplot(dogs,aes(x = weight,y = aod,color = shortcoat)) +
  geom_point() +
  theme_cowplot()
p
```

Are we happy with this plot? How could it be improved? Write your code for an improved plot here:

```
# Add your code here.
```

Optional: Try varying the shape of the points instead of color. If you have have written your ggplot code well, this should only involve a slight change to your code. You can use `scale_shape_manual` to select the shapes. Running `plot(0:23,pch = 0:23)` will give you the full list of shapes to choose from.

```
# Add your code here.
```

More on color

Carefully chosen colors can often be the difference between an effective visualization and an ineffective one. One the few complaints I have with ggplot2 is that the default colors choices are quite poor. So when you use ggplot2, you will often need to make adjustments to the colors. One rule-of-thumb is that "warmer" colors such as red and orange tend to draw the reader's attention.

There are several good resources on use of color in data visualization, and I will mention a couple here: Color Brewer (https://colorbrewer2.org); and a short article, "Color blindness", written by Bang Wong (*Nature Methods*, 2011). For more discussion, see the "Fundamentals of data visualization" book by Claus Wilke.

In our scatterplot, the best color choice is less clear, so I will let you experiment with different choices. To override the color defaults, add a `scale_color_manual` layer:

```
# Add your code here.
```

There are several different ways to specify colors. I prefer specifying colors by name; to get the full list of color names, run `colors()`.

The function that controls the color of a discrete variable has an odd name: `scale_color_manual`. This is because, in ggplot2, all methods that control the mapping of variables to colors, shapes, sizes, axes, *etc.*, start with `scale_`.

Obviously, there is much more to ggplot2. But once you are comfortable with these basic elements, you will find that almost everything else in ggplot2 is a variation of what we covered in this lesson.