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

Peter Carbonetto *University of Chicago*

In this lesson, you will use ggplot2 to create effective data visualizations. The ggplot2 package is a powerful plotting interface that extends the base plotting functions in R. The main difference with Advanced Computing is that we take a more in-depth look at ggplot2 and plotting strategies.

Motivation

Why plot? One reason is to gain insights from your data, what we sometimes call "exploratory data visualization." Another reason is to tell a story (say, in a research paper). Both will involve iteration and refinement.

For these reasons, the *programmatic approach* is a powerful aapproach data visualization. This will allow you to:

1. Create an endless variety of plots.
2. Reuse code to quickly create and revise plots.

In this tutorial we will explore the programmatic approach to plotting using **ggplot2**, an increasingly popular package for creating plots in R.

Setup

Download the tutorial materials to your computer, and make sure you know where to find them.

Before starting the tutorial, I suggest quiting applications that are not needed and other "clutter" to reduce distractions.

Launch RStudio. It is best if you start with a fresh workspace; you can refresh your environment by selecting **Session > Clear Workspace** from the RStudio menu. Also, make sure your R working directory is the same directory containing the tutorial materials; you can run `getwd()` and `list.files()` to check this.

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

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

*This document is included as part of the Data Visualization tutorial packet for the BSD qBio Bootcamp, University of Chicago, 2021. **Current version**: August 25, 2021; **Corresponding author**: pcarbo@uchicago.edu. Thanks to John Novembre, Stephanie Palmer, Stefano Allesina and Matthew Stephens for their guidance.

Hands-on exericse: Do smaller dogs live longer?

In this tutorial, we will make use 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 that was included with the rest of the tutorial materials.

Our aim is to investigate the anecdotal claim that dogs representing breeds of small size (e.g., Chihuahuas) live longer than those from larger-sized breeds (e.g., Saint Bernards).

S. cerevisiae

Import the data into R

By now, you should know how to import data from a CSV file. This code reads the data from a CSV file and creates a data frame, `dogs`. (Make sure your working directory is set to the location of these data, otherwise R will not be able to find the file.)

```
dogs <- read.csv("dogs.csv", stringsAsFactors = FALSE)
```

A first look at the data

After loading data into R for the first time, I recommend getting a basic understanding of the data frame and its contents. Here are some simple commands I often use:

```
nrow(dogs)
ncol(dogs)
names(dogs)
head(dogs)
tail(dogs)
summary(dogs)
```

What different types of data are in this table? Note that "aod" stands for "age of death".

Let's take a closer look at the `shortcoat` column:

```
unique(dogs$shortcoat)
table(addNA(dogs$shortcoat))
```

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 scientific data. Embellishments (e.g., varying colors, shapes, sizes, labels) can produce stunning visualizations.

Scatterplots are typically used to investigate whether there is an interesting—or surprising—relationship between two (continuous) variables. But the scatterplot has many other uses. For example, the scatterplot can highlight problems with the data (e.g., "outliers").

Although simple, we can explore the key features of ggplot2 via the scatterplot. Before doing this, though, let's first create a scatterplot of weight vs. longevity using the base R function, `plot`. (No special packages are used here.)

```
plot(dogs$weight, dogs$aod)
```

What story does this plot tell? Write code here to add this story to the plot:

Our first ggplot2 (with ugly code)

Now let's recreate this same scatterplot using ggplot2. The benefits of ggplot2 over `plot` will not be immediately clear.

```
library(ggplot2)
p1 <- ggplot(dogs, aes_string(x = "weight", y = "aod"))
p1 <- ggplot_add(geom_point(), p1)
```

Where is the plot? We need to call `print`, which tells R to draw the plot to the screen:

```
print(p1)
```

Some slightly simpler code can accomplish the same thing:

```
p1 <- ggplot(dogs, aes_string(x = "weight", y = "aod")) +
  geom_point()
```

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" that defines how columns are mapped to features of the plot (axes, shapes, colors, *etc*).
3. A "geom", short for "geometric object", specifies the type of plot. ggplot2 has an excellent on-line reference (ggplot2.tidyverse.org) explaining all the "geoms", from bar charts to contour plots, with code examples for each.
4. ggplot2 outputs a *ggplot object*, which can be drawn to the screen with `print` (or you can do other things with it).

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.)

Some improvements

Our plot can be improved. For example we can use the `labs` function to add axis labels, and change how the points are plotted:

```
p1 <- ggplot(dogs, aes_string(x = "weight", y = "aod")) +
  geom_point(shape = 1) +
  labs(x = "body weight (lbs)", y = "longevity (years)")
```

In what other ways can the plot can be improved? Add code for your improved plot here:

```

```

A few other functions that you might find useful to improve your plot are `scale_x_continuous`, `scale_y_continuous` and the `theme_` functions.

Save your work

This is a good point to save our work in an image file that can be shared with others.

```
ggsave("dogs.pdf", p1, height = 4, width = 4.5)
ggsave("dogs.png", p1, height = 4, width = 4.5)
```

Question: How does the PDF and PNG differ? When should you save the plot as a PDF, and when should you save as a PNG?

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). Let's see how well this estimate agrees with the data by plotting the line that best fits these data (the "least-squares" estimate).

```
fit <- lm(aod ~ weight, dogs)
coef(fit)
```

Write code to add "best fit" line to the plot (and name the new plot object "p2"):

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

Notice how easy it was to add layers to the existing plot. Also notice that this geom, like most geoms, has properties such as color and size that can be adjusted.

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

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

There's one catch here—there isn't enough real estate on the plot to accommodate all the breed names. Let's try the "smart" function `geom_text_repel` from the ggrepel package, which 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:

```
library(ggrepel)
p4 <- p2 + geom_text_repel(mapping = aes_string(label = "breed"),
                           size = 2.5, color = "gray")
```

Notice that the labels are automatically redrawn as the plot is resized—give it a try.

D. melanogaster

In-class exploration: a surprising subtlety with color

Other "aesthetics"—color, size, shape, *etc*—can also be used to tell a visual story. In the last chapter of our exploratory data visualization, we will visualize the `shortcoat` column with color, and in the process discover a complication.

The `shortcoat` column, you may recall, had values of 0 or 1 (with a few NAs).

```
dogs$shortcoat
```

In principle, varying the color of the points should be straightforward.

```
p5 <- ggplot(dogs, aes_string(x = "weight", y = "aod", label = "breed",
                              color = "shortcoat")) +
  geom_point() +
  theme_cowplot()
```

What is the problem with this plot, and how can we fix it? (And is it really a problem?) Write your code for the improved plot here:

A little more on color

One the few complaints I have with ggplot2 is that the default colors choices are poor. So you will often need to make adjustments to the color scheme. One rule-of-thumb is that "warmer" colors (e.g., orange, red) tends 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", by Bang Wong in *Nature Methods* (2011). For more extensive discussion, see "Fundamentals of data visualization" by Claus Wilke. In our scatterplot, the best color choice is less clear, so I will let you experiment with vdifferent choices. To override the color defaults, add a "scale_color_manual" layer, for example,

```
p6 <- p5 +
  scale_color_manual(values = c("firebrick", "tomato"),
                     na.value = "skyblue")
```

There are several different ways to specify colors. I prefer specifying colours by name; to get the full list of named colors, 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.

Optional exercise: Try varying shape instead of color. Use `scale_shape_manual` to select the shapes. Running `plot(0:23,pch = 0:23)` will give you all the shapes to choose from.

Optional exercise: Try varying point size instead of color. What problem do you run into, and what is the solution?

E. coli

Programming challenge: Mapping the genetic basis of physiological and behavioral traits in outbred mice

In this programming challenge, you will use simple visualizations to gain insight into biological data.

You are working in a lab studying studying the genetics of physiological and behavioral traits in mice. The lab has just completed a large study of mice from an outbred mouse population, "CFW" ("Carworth Farms White"). The aim of the study is to identify genetic contributors to variation in behaviour and musculoskeletal traits.

Note: These challenges are roughly ordered in increasing level of complexity. Do not be discouraged if you have difficulty completing every one.

Collaboration strategy

Before diving into the problems, first agree on a collaboration strategy with your teammates. Important aspects include communication and co-ordination practices, and setting goals and deadlines. How will your team collaborate on code, and share solutions? (Consider online resources such as Etherpad or the UofC-hosted Google Drive.) The aim is not just to complete the challenges, but also to do collaboratively; all team members should be included, and should have the opportunity to contribute and learn from each other.

Instructions

- Locate the files for this exercise on your computer (see "Materials" below).
- Make sure your R working directory is set to the same directory containing the tutorial materials; use `getwd()` to check this.
- Some of the programming challenges require uploading an image file containing a plot. Use `ggsave` to save your plot as a file. Any standard image format (e.g., PDF, PNG) is acceptable.
- No additional R packages are needed beyond what you used in the hands-on exercises above.

Materials

- **pheno.csv:** CSV file containing physiological and behavioral phenotype data on 1,219 male mice from the CFW outbred mouse stock. Data are from Parker *et al*, 2016. Use **readpheno.R** to read the phenotype data from the CSV file into a data frame. After filtering out some of the samples, this script should create a new data frame, `pheno`, containing phenotype data on 1,092 samples (rows).
- **hmdp.csv:** CSV file containing bone-mineral density measurements taken in 878 male mice from the Hybrid Mouse Diversity Panel (HMDP). Data are from Farber *et al*, 2011. To load the data into your R environment, run this code:

  ```
  hmdp <- read.csv("hmdp.csv", stringsAsFactors = FALSE)
  ```

  This will create a data frame, `hmdp`, containing BMD data on 878 mice (rows).

- **gwscan.csv:** CSV file containing results of a "genome-wide scan" for abnormal BMD. Association p-values were computed using GEMMA 0.96. To read the results of the genome-wide scan, run the following code:

```
gwscan <- read.csv("gwscan.csv", stringsAsFactors = FALSE)
gwscan <- transform(gwscan, chr = factor(chr, 1:19))
```

  This will create a data frame, `gwscan`. Each row of the data frame is a genetic variant (a single nucleotide polymorphism, or "SNP"). The columns are chromosome ("chr"), base-pair position on the chromosome ("pos"), and the p-value for a test of association between variant genotype and trait value ("abnormalBMD"). The value stored in the "abnormalBMD" column is $-\log_{10}(P)$, where P is the p-value.
- **geno_rs29477109.csv:** CSV file containing estimated genotypes at one SNP (rs29477109) for 1,038 CFW mice. Use the following code to read the genotype data into your R environment:

```
geno <- read.csv("geno_rs29477109.csv", stringsAsFactors = FALSE)
geno <- transform(geno, id = as.character(id))
```

  This will create a new data frame, `geno`, with 1,038 rows (samples). The genotypes are encoded as "dosages"—that is, the expected number of times the alternative allele is observed in the genotype. This will be an integer (0, 1, 2), or a real number between 0 and 2 when there is some uncertainty in the estimate of the genotype. For this SNP, the reference allele is T and the alternative allele is C. Therefore, dosages 0, 1 and 2 correspond to genotypes TT, CT and CC, respectively (genotypes CT and TC are equivalent).
- **wtccc.png:** Example genome-wide scan ("Manhattan plot") from Fig. 4 of the WTCCC paper. The p-values highlighted in green show the regions of the human genome most strongly associated with Crohn's disease risk.

A couple tips

- Some "geoms" you may find useful: `geom_point`, `geom_histogram`, `geom_boxplot`.
- In some cases it may be useful to convert to a *factor*.

Part A: Exploratory analysis of muscle development and conditioned fear data

Your first task is to create plots to explore the data.

1. A basic initial step in an exploratory analysis is to visualize the distribution of the data. It is often convenient if the distribution is normal, or "bell shaped".
   - Visualize the empirical distribution of tibialis anterior (TA) muscle weight (column "TA") with a histogram. Units are mg. *Hint:* Try using function `geom_histogram`.
   - Is the distribution of TA weight roughly normal? Are there mice with unusually large or unusually small values ("outliers")? If so, how many "outliers" are there? (Unusually small or large values can lead to misleading results in some statistical tests.)
2. It is also important to understand relationships among measured quantities. For example, the development of the tibia bone (column "tibia") could influence TA muscle weight. Create a scatterplot (`geom_point`) to visualize the relationship between TA weight and tibia length. (Tibia length units are mm.) Based on this plot, what can you say about the relationship

between TA weight and tibia length? Quantify this relationship by fitting a linear model, before and after removing the outlying TA values. (*Hint*: Use the `lm` and `summary` functions. See also the "r.squared" return value in `help(summary.lm)`.)

3. The "AvToneD3" column contains data collected from a behavioral test called the "Conditioned Fear" test.
    - Visualize the empirical distribution of AvToneD3 ("freezing to cue") with a histogram. Is the distribution of AvToneD3 approximately normal?
    - Freezing to cue is a proportion (a number between 0 and 1). A common way to obtain a more "normal" quantity is to transform it using the "logit" function[1]. Visualize the empirical distribution of the logit-transformed phenotype. Is the transformed phenotype more "bell shaped"? After the transformation, do you observe unusually small or unusually large values?
    - A common concern with behavioral tests is that the testing devices can lead to measurement error. It is especially a concern when multiple devices are used, as the devices can give slightly different measurements, even after careful calibration. Create a plot to visualize the relationship between (transformed) freezing to cue and the device used ("FCbox" column). *Hint:* Try a boxplot (`geom_boxplot`). Based on this plot, does the apparatus used affect these behavioral test measurements?

Part B: Exploratory analysis of bone-mineral density data

Now you will examine data on bone-mineral density (BMD) in mice. This is a trait that is important for studying human diseases such as osteoporosis (units are mg/cm^2).

- Plot the distribution of BMD in CFW mice (see column "BMD"). What is most notable about the distribution?
- Compare these data against BMD measurements taken in a "reference" mouse population, the Hybrid Mouse Diversity Panel. To compare, create two histograms, and draw them one on top of the other. What difference do you observe in the BMD distributions? For a correct comparison, you will need to account for: (1) BMD in CFW mice was measured in the femurs of male mice only; (2) BMD in HMDP mice was recorded in g/cm^2. *Hints:* Functions `xlim` and `labs` from the ggplot2 package, and `plot_grid` from the cowplot package, might be useful for creating the plots. The `binwidth` argument in `geom_histogram` may also be useful.

Part C: Mapping the genetic basis of osteopetrotic bones

A binary trait, "abnormal BMD", was defined that signals whether an individual mouse had "abnormal", or osteopetrotic, bones. It takes a value of 1 when BMD falls on the "long tail" of the distribution (BMD greater than 90 mg/cm^2), otherwise zero.

GEMMA was used to carry out a "genome-wide association study" (GWAS) for this trait; that is, support for association with abnormal BMD was evaluated at 79,824 genetic variants (single nucleotide polymorphisms, or "SNPs") on chromosomes 1–19. At each SNP, a *p*-value quantifies the support for an association with abnormal BMD.

1. Your first task is to get an overview of the association results by creating a "Manhattan plot".

[1] R code: `logit <- function(x) log((x + 0.001) / (1 - x + 0.001))`

Follow as closely as possible the provided prototype, **wtccc.png**, which shows a genome-wide scan for Crohn's disease. (Don't worry about highlighting the strongest p-values in green.) *Hints:* Replicating some elements of this plot may be more challenging than others, so start with a simple plot, and try to improve on it. Recall the adage that creating plots requires relatively little effort *provided the data are in the right form*—consider adding appropriate columns to the `gwscan` data frame. Functions from the ggplot2 package that you may find useful for this exercise include `geom_point`, `scale_color_manual` and `scale_x_continuous`.

- In your plot, you should observe that the most strongly associated SNPs cluster closely together in small regions of the genome. This is common—it is due to a genetic phenomenon known as linkage disequilibrium (LD). It is a consequence of low recombination rates between markers in small populations. How many SNPs have "strong" statistical support for association with abnormal BMD, specifically with a $-\log_{10} p$-value > 6? How many distinct regions of the genome are strongly associated with abnormal BMD at this p-value threshold?
- What p-value does a $-\log_{10} p$-value of 6 correspond to?
- Using your plot, identify the "distinct region" (this is called a "quantitative trait locus", or QTL) with the strongest association signal. What is, roughly, the size of the QTL in Megabases (Mb) if we define the QTL by base-pair positions of the SNPs with $-\log_{10} p$-value > 6? Using the UCSC Genome Browser, get a rough count of the number of genes that are transcribed in this region. Within this QTL, Parker *et al*, 2016 identified *Col1a1* as a candidate BMD gene. Was this gene one of the genes included in your count? *Hint:* All SNP positions are based on NCBI Mouse Genome Assembly 38 (mm10, December 2011).

2. Your next task is to visualize the relationship between genotype and phenotype. From the genome-wide scan of abnormal BMD, you should find that rs29477109 is the SNP most strongly associated with abnormal BMD. Here you will look closely at the relationship between BMD and the genotype at this SNP. In developing your visualization, consider that:
   - The samples listed in the phenotype and genotype tables are not the same. So you will need to align the two tables to properly show analyze the relationship. *Hint:* Function `match` could be useful for this.
   - The genotypes, stored in file **geno_rs29477109.csv**, are encoded as "dosages" (numbers between 0 and 2). You could start with a scatterplot of BMD vs. dosage. But ultimately it is more effective if the genotypes (CC, CT and TT) are plotted instead. *Hints:* In effect, what you need to do is convert from a continuous variable (dosage) to a discrete variable (genotype). One approach is to create a `factor` column from the "dosage" column. For dosages that are not exactly 0, 1 or 2, you could simply round to the nearest whole number. A boxplot is recommended; see function `geom_boxplot`.

   Based on your plot, how would describe (in plain language) the relationship between the genotype and BMD?

M. mulatta

Notes

An interactive plot

Here is an example of a plot that allows the data to be explored *interactively*:

```
library(plotly)
library(htmlwidgets)
p <- plot_ly(data = dogs, type = "scatter", mode = "markers",
             x = ~weight, y = ~aod, color = ~shortcoat,
             text = ~sprintf("%s\n (%0.1f lbs, %0.1f years)", breed, weight, aod),
             marker = list(line = list(color = "white", width = 1), size = 9),
             hoverinfo = "text", width = 600, height = 500)
p <- layout(p,
            xaxis = list(title = "weight (lbs)"),
            yaxis = list(title = "longevity (years)"),
            hoverlabel = list(bgcolor = "white", bordercolor = "black"))
saveWidget(p, "dogs.html", selfcontained = TRUE)
```

Plotly is a powerful package for creating interactive plots, with an interface similar to ggplot2.

Useful online resources

- ggplot2 reference, where you will also find a ggplot2 cheat sheet. (This cheat sheet is also included in the tutorial packet.)
- Fundamentals of Data Visualization by Claus Wilke.

Sources

- The dogs breeds data were downloaded from the *Genetics* journal website.
- The CFW phenotype and genotype data were downloaded from Data Dryad.

License

X. laevis