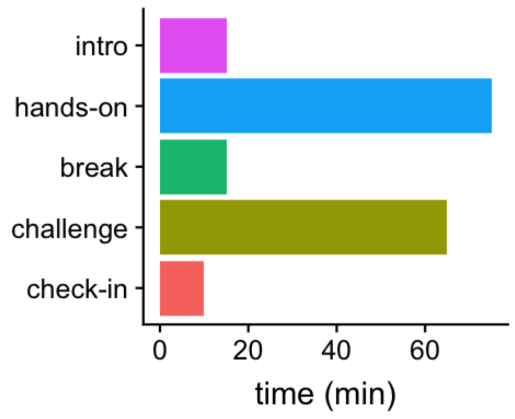
Data visualization tutorial: exploring data and telling stories using ggplot2[*]

Peter Carbonetto *University of Chicago*

In this lesson, you will use ggplot2 to create effective visualizations of data. The ggplot2 package is a powerful set of plotting functions that extend the base plotting functions in R. You will learn that creating effective visualizations *hinges on good data preparation.* (In reality, good data preparation in the lab can take days or weeks, but we can still illustrate some useful data preparation practices.) The main difference with Advanced Computing 1 is that we take a more in-depth look at ggplot2 and plotting strategies.

Outline



Motivation

Why plot? One reason: to gain insight into your data. This is "exploratory data visualization.". Another reason: to tell a story (e.g., for a scientific paper). Both require lots of iteration and refinement to get right. So you need to learn to efficiently create plots, and efficiently improve them.

One solution is the *programmatic approach* to data visualization. This will allow you to:

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

ggplot2 is the preferred approach to creating plots in R.

[*]This document is included as part of the Data Visualization tutorial packet for the BSD qBio Bootcamp, MBL, 2019. **Current version**: July 24, 2019; **Corresponding author**: pcarbo@uchicago.edu. Thanks to John Novembre and Matthew Stephens for their support and guidance.

Setup

If you haven't already done so, install these packages:

```
install.packages("readr")
install.packages("ggplot2")
install.packages("cowplot")
```

Once installed, load the packages into your R environment:

```
library(readr)
library(ggplot2)
library(cowplot)
```

Your first plot

To test that your R environment is properly set up for plotting with ggplot2, plot some fuel economy data from the EPA:

```
epa <- read.csv("epa.csv")
ggplot(mpg, aes(displ, cty, color = class)) + geom_point()
```

Hands-on exercise: "Rising Dough, Rising Neighbourhoods"

The scenario

You have begun a summer internship at the Mansueto Institute for Urban Innovation. The institute has formed a partnership with the City of Chicago to study trends in different neighborhoods across the city.

For your summer project, you will explore ways to study neighbourhoods trends by aggregating publicly available data on pizza restaurants. You will prepare a report summarizing your results. Evocative plots will be critical to a successful report.

Setup

- Make sure you have downloaded the tutorial packet.
- Locate the files for this exercise on your computer (see "Materials" below).
- Make sure your R working directory is the same directory containing the tutorial materials; run `getwd()` to check this.

Materials

- **Food_Inspections.csv.gz**: CSV file containing food inspection data downloaded from the Chicago Data Portal.

Load the data

Use `read_csv` from the **readr** package instead of the standard function, `read.csv`, because `read_csv` is much faster for large data sets.

```
data.file  <- "Food_Inspections.csv.gz"
dat        <- read_csv(data.file)
class(dat) <- "data.frame"
```

Run a few basic commands to inspect the data:

Prepare the data for plotting

Creating sophisticated plots using ggplot2 requires relatively little effort *provided the data are in the right form.* Before creating the plots, you must carefully prepare the data so that it is ready for plotting. *Many analysis mistakes are due to poor data preparation!*

First, remove the columns you don't need (since unnecessary data can be distracting), and change the names of the columns to make the code easier to write. (Note that "DBA" is an abbreviation of "doing business as".)

```
cols     <- c(2:5, 7, 8, 11, 13, 15, 16)
colnames <- c("dba", "aka", "license", "type", "address", "city",
              "date", "results", "latitude", "longitude")
dat        <- dat[cols]
names(dat) <- colnames
```

Now extract the year from the inspection date. This should create a new column, "year".

```
get.third.entry <- function(x) { return(x[3]) }
out      <- strsplit(dat$date, "/")
dat$year <- sapply(out,get.third.entry)
dat$year <- as.numeric(dat$year)
```

Question: How many restaurant inspections were performed in each year?

Now, write code to filter out rows based on these criteria: you want to keep only data on Chicago restaurants, and remove all rows for which one or more of the license and geographic co-ordinates (latitude and longitude) are not available. Write your R code here to accomplish this:

Recall, the goal is to visualize data on pizza restaurants. So you need to extract the rows of the data frame that correspond to pizza restaurants. Write your R code to do this.

You are nearly done preparing the data. Some restaurants were inspected multiple times, and you don't want to overcount the number of restaurants. The "license" column is useful for uniquely identifying restaurants.

First, write code to sort the rows of the table by year of inspection so that the first rows show the earliest inspections (*hint*: use the `order` function):

Question: What is the difference between "sort" and "order"?

Next, write code to remove rows so that there is only one row per pizza location:

Question: How do you double-check that each license appears only once in the data frame?

Your first plot

Your first plot is a bar chart showing an estimate of the number of new pizza restaurants in Chicago by year. Write code that uses the `table` function to create a table containing number of inspected restaurants by year:

To plot these counts in ggplot2, we need to create a data frame. Write code to convert `counts` to a data frame with two columns, "year" and "count":

You now have a data frame with two columns, "year" and "count". Go ahead and plot it! First, specify the data frame, and the mapping between variables (columns) and plotting elements:

```
a <- aes(x = year, y = count)
p <- ggplot(counts, a)
```

Specify *how* to plot the data points:

```
g <- geom_col()
p <- ggplot_add(g, p)
```

Where is the plot? The `print` function tells R to draw the plot to the screen:

```
print(p)
```

It is Best Practice to label your axes clearly. Improve the y-axis label:

```
out <- labs(y = "number of locations")
p   <- ggplot_add(out, p)
```

Here is equivalent code that is more concise (and more commonly used), but also more difficult to understand because it is less clear what the "+" does:

```
p <- ggplot(counts,aes(x = year, y = count)) +
  geom_col(width = 0.5) +
  labs(y = "number of locations")
```

I recommend using the less concise syntax until you are comfortable with ggplot2.

All plots in ggplot2 require these three elements:

1. A data frame.
2. An "aesthetic mapping" that declares how columns are mapped to plot features (axes, shapes, colors, *etc.*).
3. A "geom", short for "geometric object," that specifies the type of plot.

All plots are created by *adding layers.* Layers are added with `ggplot_add` or `+`.

Your second plot

For the second plot, you will create a map of the pizza restaurants. This should be easy *since you have already carefully prepared the data*. Write R code to do this:

Try plotting the restaurants with different shapes—which one works best?

One of the powerful features of ggplot2 is that you can use it to generate a wide variety of plots, and each plot is created in a very similar fashion. For example, modify your code above to draw a contour plot showing the density of pizza restaurants throughout the city:

Based on this plot, which areas of the city have the greatest density of pizza restaurants? Is Hyde Park visible from the contours?

Overlay the contour plot with the points:

Now, try to add color to the points to highlight the "hottest" neighbourhoods:

This is an example where it is important to get the colour scheme right. Write R code that uses function `scale_color_gradient2` to highlight more recent pizza restaurants in warmer (red) colours:

All functions starting with `scale_` modify the behaviour of the geoms.

Now combine the plots. This is easily done with `plot_grid` from the cowplot package:

```r
p12 <- plot_grid(p,p2,labels = c("A","B"))
```

Finally, save your plot using the `ggsave` function. PNG is a commonly used format to share images.

Follow-up programming challenges

Your project advisor says, "Great work. However, I would like a few changes to better present the results. First, create a line chart to better show the trend in pizza restaurants. For the bar chart, color the bars so that they match the colors in the map. Also, a PNG won't work for the report—please send a PDF instead."

These improvements are left as a programming challenge.

1. Your first plot was a bar chart showing an estimate of the number of new pizza restaurants per year in Chicago. Sometimes lines are more effective than bars for showing trends.
   - To show the trend as a line plot, how would you modify the plotting code above to do this? *Hint:* try using `geom_line`.
   - What change do you need to make to the `counts` data frame so that counts can be plotted as lines?
   - Which do you find more effective, the line plot or the bar chart? Identify one benefit of one plot over the other.
   - To submit your response, you will need to upload a file containing your final plot. Use `ggsave` to save your plot as a file.
2. Your project advisor has asked you to colour the bars in the bar chart so that they align with the colours used in the map.
   - What change do you need to make to the `counts` data frame so that the years can be mapped to colors? *Hint:* The code used for creating the line plot may be useful here, too.
   - Submit the *combined* plot (bar chart & map). When submitting your plot, save it as a PDF using `ggsave`.
   - Why did your advisor request a PDF? Why is a PNG file less suitable for a final report or publication?

Main 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 have finished your summer internship at the Mansueto Institute. You are embarking on your first graduate research project 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" (a contraction of "Carworth Farms White", the names of the scientists who bred the first CFW mice). The ultimate 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!

Instructions

- Locate the files for this exercise on your computer (see "Materials" below).
- Make sure your R working directory is 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 examples 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.

General tips

- Some "geoms" you may find useful: `geom_point`, `geom_histogram`, `geom_boxplot`.
- In some cases it may be useful to convert a column to a *factor*.
- Coming up with the answers will involve skills other than programming, so collaborate!

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

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

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". 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?

Notes

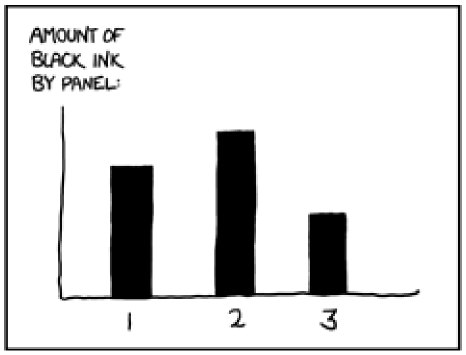
Useful online resources

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

License

Sources

- The CFW phenotype and genotype data were downloaded from the Data Dryad repository.
- The City of Chicago food inspection data were downloaded from the Chicago Data Portal. Specifically, the data were downloaded in CSV format from here on August 3, 2018.


AMOUNT OF BLACK INK BY PANEL:
1
2
3