Some Data Visualization with ggplot2, ggalt

Andrea Tu

October 31, 2017

Data science. Big data. Data analysis. These are all buzz words that we've been hearing, especially as universities are developing data science initiatives and curriculum, and companies are increasing the demand for data scientists and analysts. There is data literally everywhere, and it can certainly get very overwhelming to just look at pages upon pages of information or spreadsheets. This is where data visualization comes in.

...AND HERE WE HAVE OUR DATA VISUALIZATION TEAM. THIS IS DAVE OUR PIE CHART SPECIALIST, LENNY IN BAR GRAPHICS, AND SPENCE, OUR SCATTERPLOT DESIGNER.
© 2013 cartertoons.com

Data visualization is important as it helps us examine the data in a visual context, allowing us to see things that we may not have by purely looking at the text and numbers. Patterns and trends are definitely more easily seen visually, which leaves room for more robust data analysis as well. Data visualization techniques using R are now used in both schools and industry, so it is important for us to understand the best and most effective ways of displaying data using visualizations.

Two main systems for data visualization in R are baseR and ggplot2. There is some debate about which system is "better". Both ggplot2 and baseR may have their pros and cons, and the useful-ness of each one may depend on the type of data you have and what kind of analysis you want to perform. In this post I will speak mostly of data visualization using ggplot2 and a couple extension packages, including ggalt.

Basics of ggplot2

ggplot2 is quite a versatile package that allows users (like us) to make beautiful and helpful visualizations of both the simplest of data and also more complex data. It provides similar functions as base R visualizations, but it also has some additional features that are pretty helpful to learn and use.

The first step to using ggplot2 is to install and load the library. I will also install another package "ggalt" which we will be looking at later.

```
install.packages('ggplot2')
install.packages('ggalt')
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(ggalt)
```

Some of the basic functions of ggplot2 can also be done using baseR, so in many cases the use of the two of these is interchangeable. Something that may differentiate them is aesthetics of the visualizations, such as the color of the background, but otherwise many types of graphs can be made using either package (i.e. scatterplots, bar graphs, historgrams)

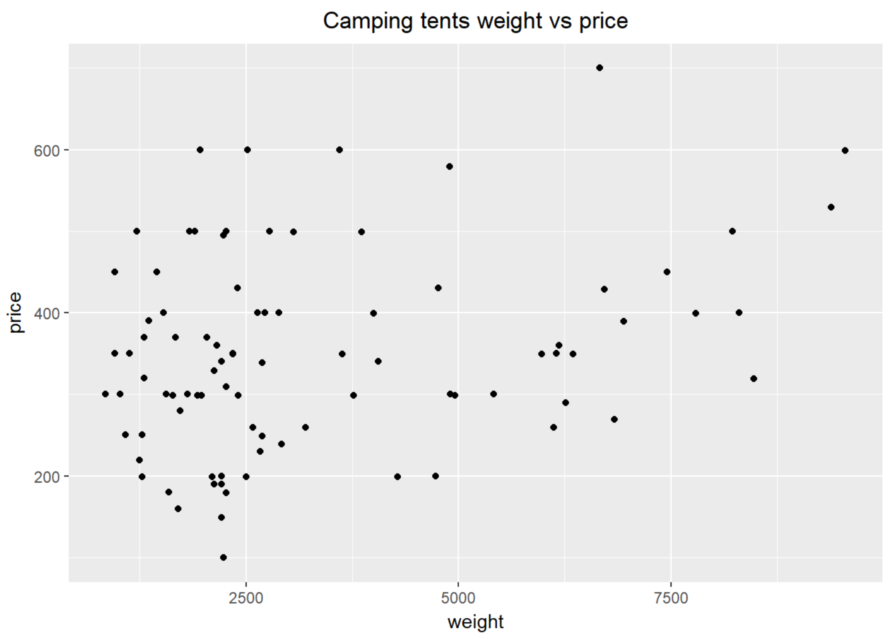
I will be using the camping tents data set provided by Professor Sanchez on the github website for most of my examples.

```
# setting my working directory
setwd("C:/Users/Andrea/OneDrive/Documents/School/stat133/stat133-hws-fall17/post01/code")

# loading data into a table
camping_tents <- read.csv('../data/camping-tents.csv', stringsAsFactors = FALSE)
```

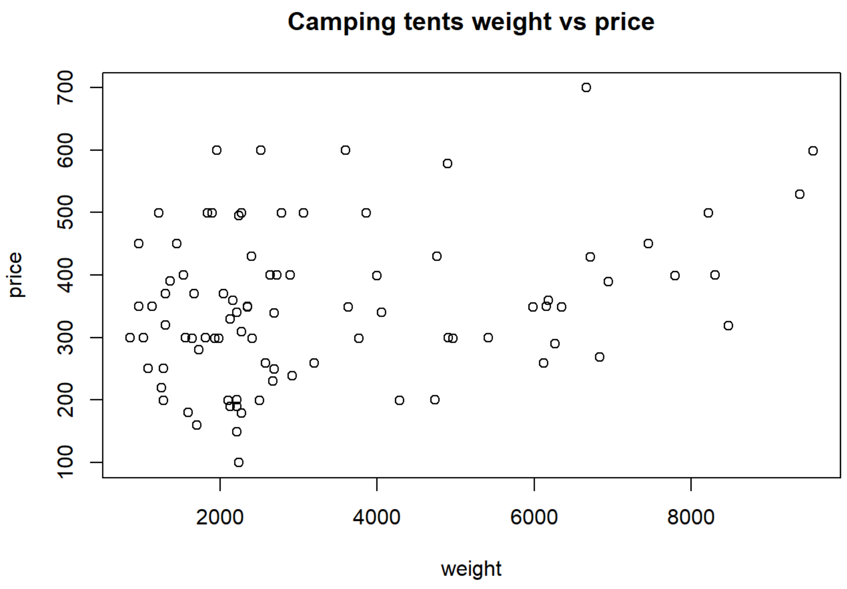
ggplot2 has a useful function, called **qplot**. It creates a pretty basic but complete plot, comparable to that of baseR's scatterplot. It takes a few basic arguments: x and y are what we expect – the data to be plotted on the x- and y-axes; the optional data argument takes a dataframe. Other arguments include plot labels and limits, which can be used depending on what we want to see on our graph.

ggtitle is another useful function that, as you might expect from the name, titles the plot that is generated. The only unfortunate thing about this function (in my opinion) is that it automatically aligns left, but there is an easy fix to that, using theme().

```
# scatterplot using ggplot2
qplot(data = camping_tents, x = weight, y = price, geom = 'point') +
  ggtitle('Camping tents weight vs price') +
  theme(plot.title = element_text(hjust = 0.5))
```



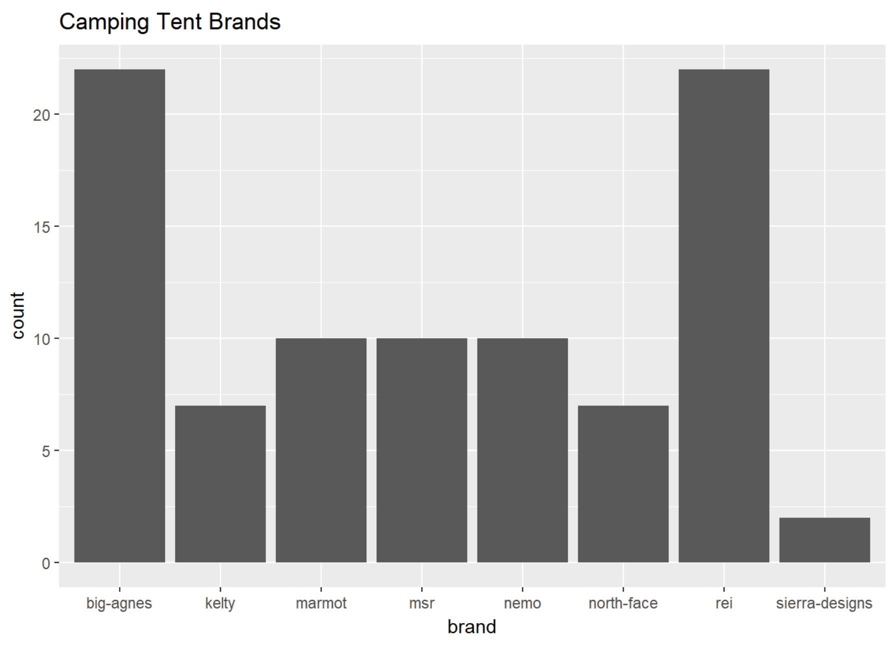
```
# scatterplot using baseR
plot(camping_tents$weight, camping_tents$price,
     main = 'Camping tents weight vs price',
     xlab = 'weight', ylab = 'price')
```



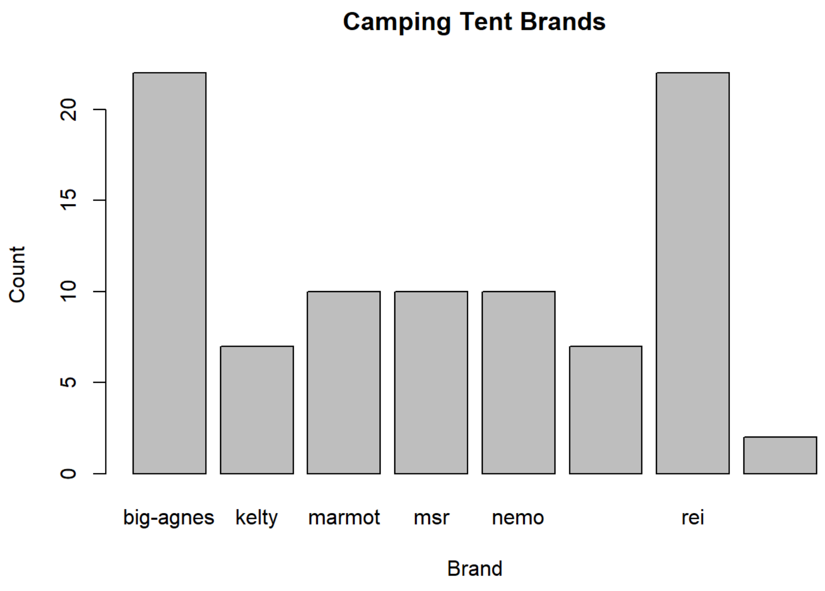
ggplot2 uses the same basic **ggplot** function to start each plot. Like qplot, there are a few basic arguments that it has. ‘data’ is an argument that sets a default dataset to use for the plot. In my examples, I am using the camping_tents dataset, which I downloaded and converted from Professor Sanchez’s github. ggplot also takes a mapping argument, which specifies the aesthetics for the plot. This includes the values that go on the x- and y-axes, denoted by x = and y = .

We can also add on extra functions to specify the type or other aspects of the graph that we want to be displayed. In the following code, we show how to make a bar chart using ggplot2 by adding “geom_bar”. This is different from baseR, where there is a completely different function for making the bar graph: barplot().

```
# barplot using ggplot2
ggplot(data = camping_tents, aes(x = brand)) +
  geom_bar() +
  ggtitle('Camping Tent Brands')
```



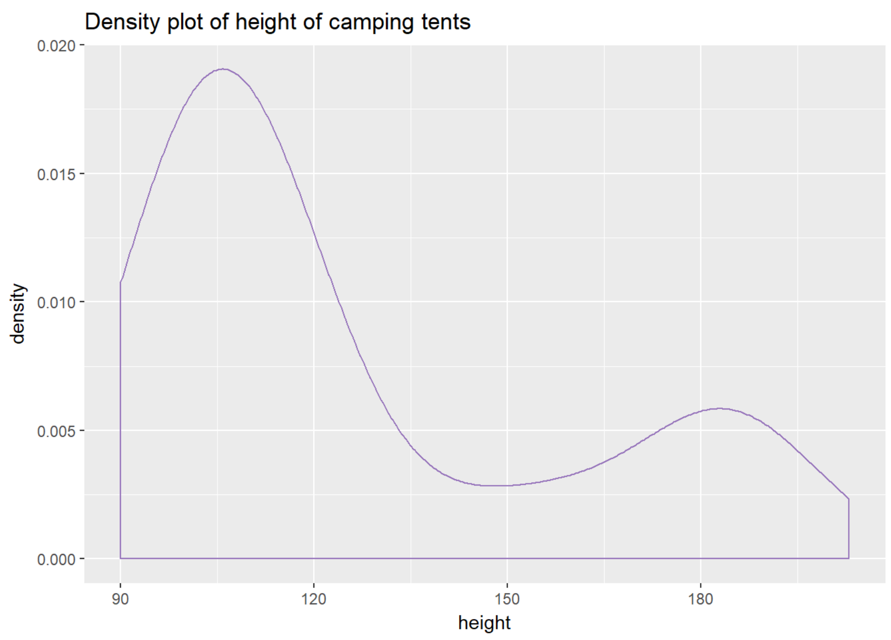
```
# barplot using baseR
brands <- table(camping_tents$brand)
barplot(brands, main = 'Camping Tent Brands',
        xlab = 'Brand', ylab = 'Count')
```



As you can tell, these graphs are pretty simple, and tell us the same information. In this example, we can see from both graphs that the camping tent brands that are most represented in our dataset are big-agnes and rei. The major difference here is the graphic aesthetics, and so the choice of which to use is based on each person's preference. I personally think the ggplot graphs look nicer and have a cleaner feel.

(Other graph types include histogram using "geom_histogram" and density plot using "geom_density". You can also specify features such as color and size in these functions.)

```
# density plot. color has been changed to a pale purple.
ggplot(data = camping_tents, aes(x = height)) +
  geom_density(color = '#9876bc') +
  ggtitle('Density plot of height of camping tents')
```

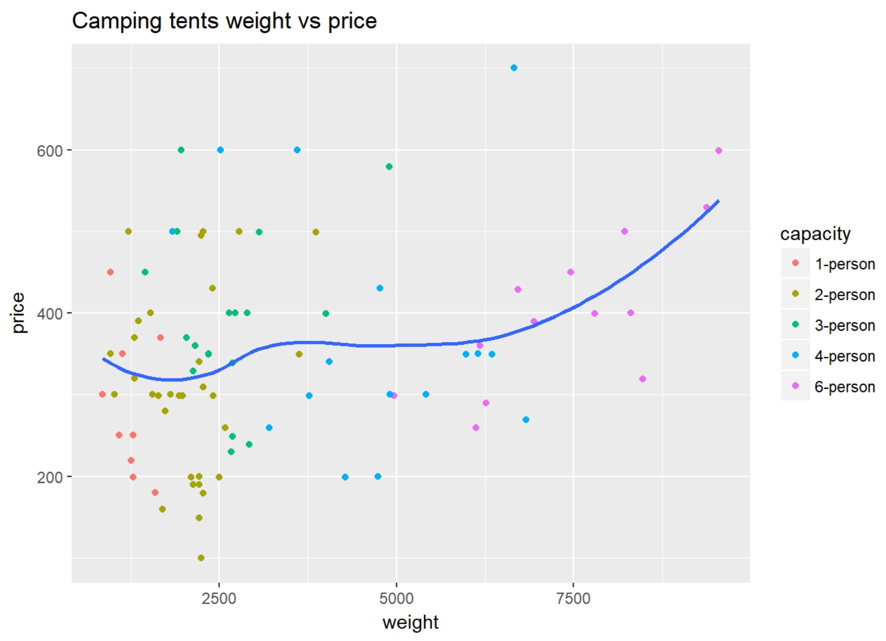


Now that we've done some basic work with ggplot2 and baseR, let's look at some features of ggplot2 that make the visualizations more robust with information and easier to read (as well as more fun to look at!).

geom_smooth

geom_smooth is a function that can be added onto a ggplot that creates a smoothed line of conditional means. The default method is 'loess', but here I typed it out as an argument anyways. LOESS stands for Locally Weighted Smoothing, and it allows us to see a relationship between variables when doing regression analysis. It's like a trend line, but we don't have to specify if it's linear or quadratic or anything – it helps especially if the data is noisy or messy and correlations are weak.

```
ggplot(data = camping_tents, aes(x = weight, y = price)) +
  geom_point(aes(color = capacity)) + geom_smooth(method = 'loess', se = F)+
  ggtitle('Camping tents weight vs price')
```



If you noticed, the dots are also now each different colors, depending on the (advertised) capacity of the camping tents! This is another cool feature of ggplot2, where the plot points can be made different colors based on a certain categorical feature. We can now clearly see that for the most part, the tents that hold less people generally weigh less, and are also priced lower. This isn't much of a discovery in this case for this data, as we probably could've guessed that, but it could be useful when the relationship between variables are less obvious.

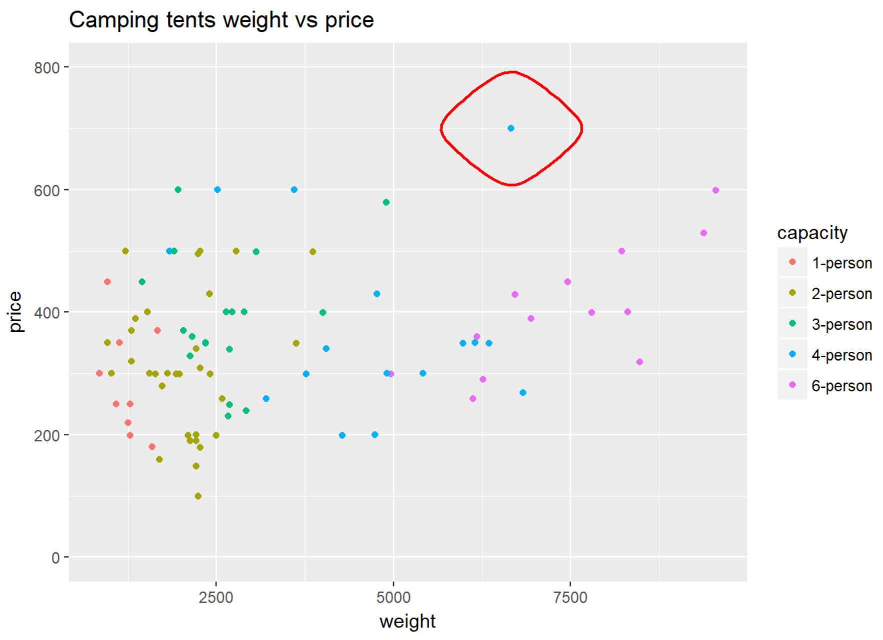
I also just wanted to point out, another nice feature of ggplot2 is that it automatically creates a legend for you! It's even nicely off to the side where it's not interfering with reading the plot, but the info is still very clearly available.

geom_encircle (using ggalt)

This next function requires us to install a new package called **ggalt**, which I did in the beginning of this post. The ggalt package is an extension for ggplot2 that has extra coordinate systems and other transformation capabilities.

The particular function I want to show you is called **geom_encircle**. This allows us to encircle certain points or groups of points on a plot so that they may be pointed out. It could be especially useful when pointing out outliers, or if we simply want to pay attention to a single point in our data. The data argument for this function should be a dataframe that contains only the points that we are interested in encircling. We can also change the size (thickness) of the circle and how far it expands out from the points themselves.

```
# this camping tent is especially pricey. i want to point it out.
pricey_camping_tents <- filter(camping_tents, price > 600)

ggplot(data = camping_tents, aes(x = weight, y = price)) +
  geom_point(aes(color = capacity)) +
  ylim(c(0, 800)) +
  geom_encircle(data = pricey_camping_tents, aes(x = weight, y = price), color = "red", size = 2, expand = 0.005)
+
  ggtitle('Camping tents weight vs price')
```



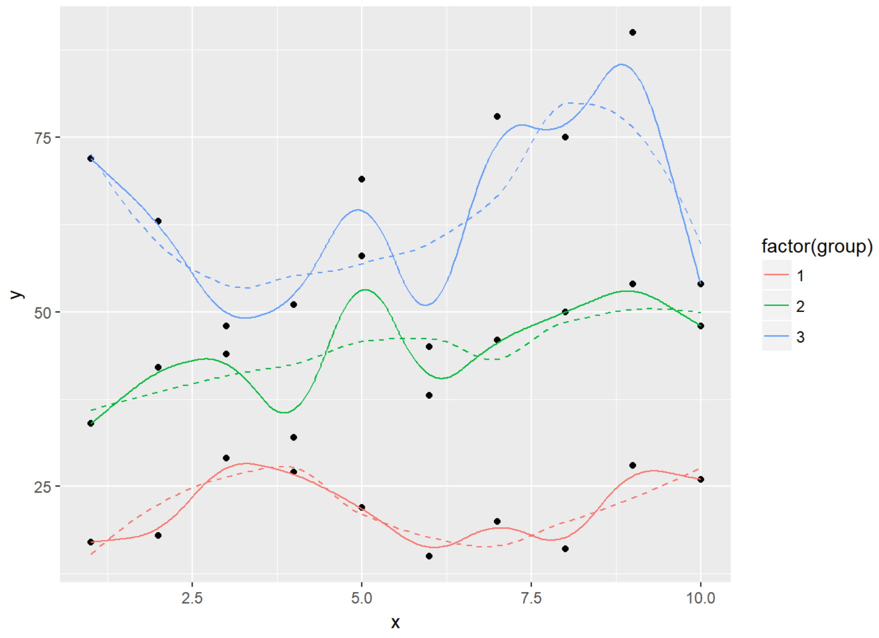
Would you look at that! Now you can really tell where the most pricey camping tent is – the approximate weight as well as the capacity!

geom_xspline (using ggalt)

This is the final function that I want to explore in this post. **geom_xspline** is a function in the ggalt package that draws a line “relative to control points”. This line may go through each point, or it might just approach the point and simply follow the general trend of the data. Some arguments it takes include size (referring to the width of the line) and spline_shape (a number between -1 and 1 that controls the shape of the line relative to the control points).

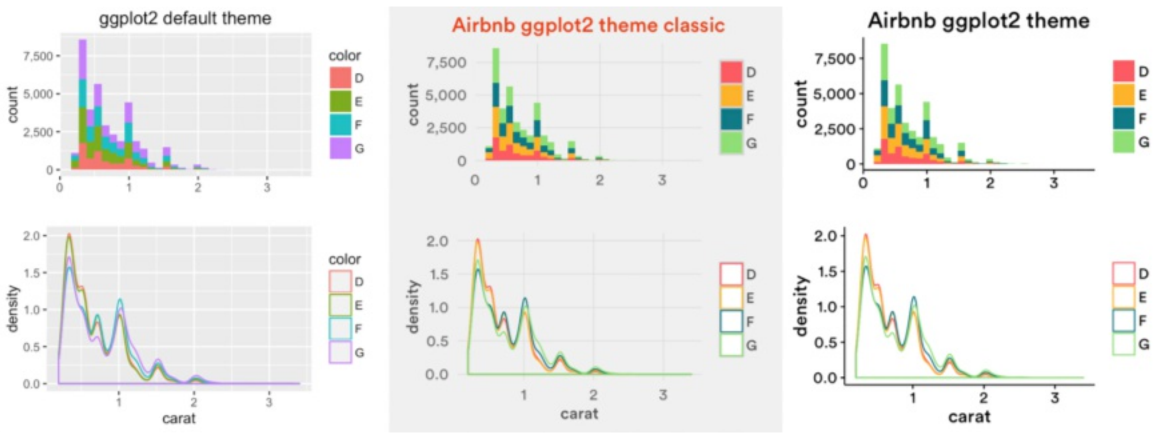
```
# using a different dataset to make the graph and use of geom_xspline more clear
dat <- data.frame(x = c(1:10, 1:10, 1:10),
                  y = c(sample(15:30, 10), 2*sample(15:30, 10), 3*sample(15:30, 10)),
                  group = factor(c(rep(1, 10), rep(2, 10), rep(3, 10)))
)

ggplot(dat, aes(x, y, group = group, color = factor(group))) +
  geom_point(color = 'black') +
  geom_smooth(linetype = "dashed", size = 0.5, se = FALSE) +
  geom_xspline(spline_shape = 0.5, size = 0.5)
```

```
## `geom_smooth()` using method = 'loess'
```

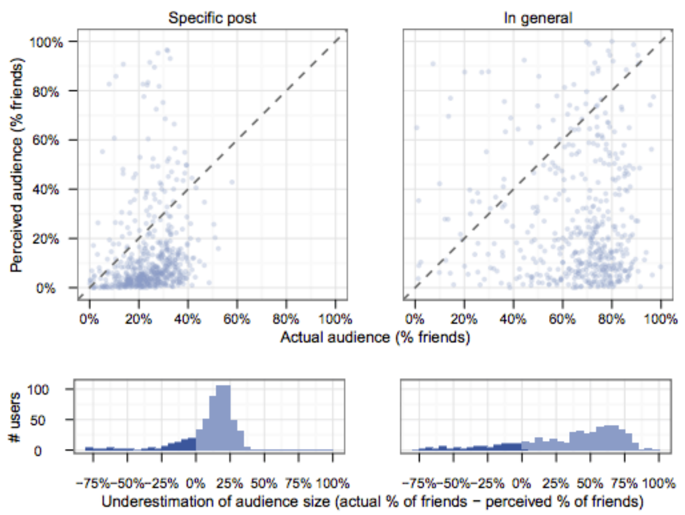


In the above example, the geom_smooth line that we created is dotted, and it is a loess line, which was explained earlier in this post (it's similar to a trend line). The solid lines are the xspline lines – as you can see, they follow the actual points more closely. These are useful because they show a little more than just a general trend line… but are also a little more general than just connecting the dots, which often doesn't tell us too much. This way, we can better understand the trend of the data, and the graph also looks smoother and aesthetically pleasing.

These are just a few of the many amazing features that ggplot2 offers, in addition to some of the ones we've learned in class, such as making box plots and faceting grids. As you can see, the data visualization options are abundant and quite aesthetically pleasing. There is definitely more to explore, and if you're interested, there are so many resources out there! I just want to leave you with this: ggplot2 isn't something that we use ever again, as we might think about many of the things we learn in school. Many companies use it, including Facebook, Twitter, and AirBnb. This is one way that AirBnB has used ggplot2 (they have a custom version though).



ggplot2 has also been put to great use by Facebook data scientists in their analysis.



Take some time to explore the functions and capabilities of ggplot2 and its extensions if you're interested/have some time on your hands! It's really a useful and practical tool in data analysis and will be useful to you not just here at Berkeley, but even more so into the future in industry.

References

What is data visualization?

ggplot2 cheatsheet

Why use ggplot2

Camping Tents data

Some cool ggplot2 visualizations

ggalt examples

Data Science using R at AirBnB