R Tutorial-06

T. Scofield

You may click here to access the .qmd file.

In this issue, we investigate various frameworks for plotting data. Specifically, we consider plotting data using routines from

- the R **base** package (auto-loaded with every R session)
- the **ggplot2** package
- the **ggformula** package

Throughout the course I give preference to **ggformula**.

Where to find examples of R base routines in the text

Prior to Chapter 7, examples in the text which include plots of data all employ figures generated using functions in the R **base** package. You can find examples of

- plots of frequency and relative frequency tables (`plot()`; Section 5.2)
- a density plot of data (`plot()` combined with `density()`; Section 5.3)
- a histogram (`hist()`; Section 5.3)
- plots of normal, uniform, chi-square and gamma pdfs (`curve()` combined with `dnorm()`, `dchisq()`, etc.; Section 5.3)

These plotting functions are quite adequate—and the R development team seems uninterested in replacing them—but some R users have sought to replace them with routines that seem more up-to-date.

The ggplot2 package

One popular package providing more up-to-date plotting methods is **ggplot2**. The authors of our text devote Chapter 7 to explaining its use. For those interested in diving in, I commend this chapter to you, but be aware that I intend to avoid the chapter. A side-by-side comparison of one type of plot using the **base** package vs. **ggplot2** may serve to explain why.

The **fosdata** package has a dataset called `houses`. For each house, a number of variables are measured, including `zipcode`, `sqft_lot`, `sqft_living` and `price`. There are a *lot* of houses, so we will create a smaller data frame, filtered from this one, but containing only houses from a particular zipcode:

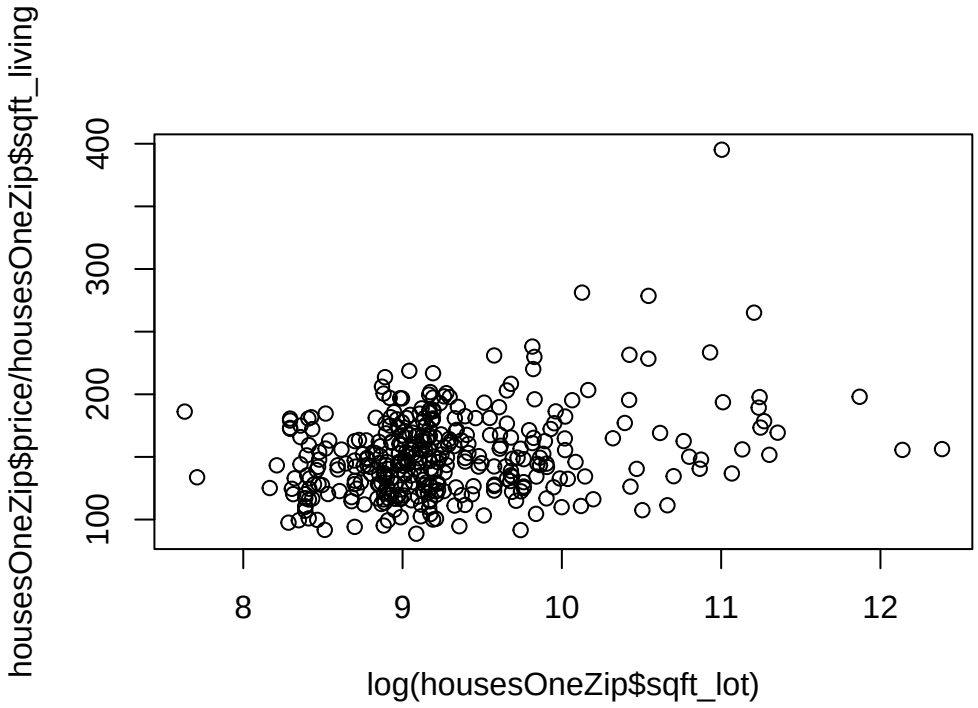
```
housesOneZip = filter(houses, zipcode=="98001")
nrow(housesOneZip)
```

```
[1] 362
```

This took us from 21613 houses down to 362. We might be interested in the relationship between the log(`sqft_lot`) and the *price per square foot*, a value that we get by dividing `price` by `sqft_living`. We produce a **scatterplot**, where each house in the 98001 zipcode is represented by a point with log(`sqft_lot`) as the x-coordinate, and `price / sqft_living` as the y-coordinate.

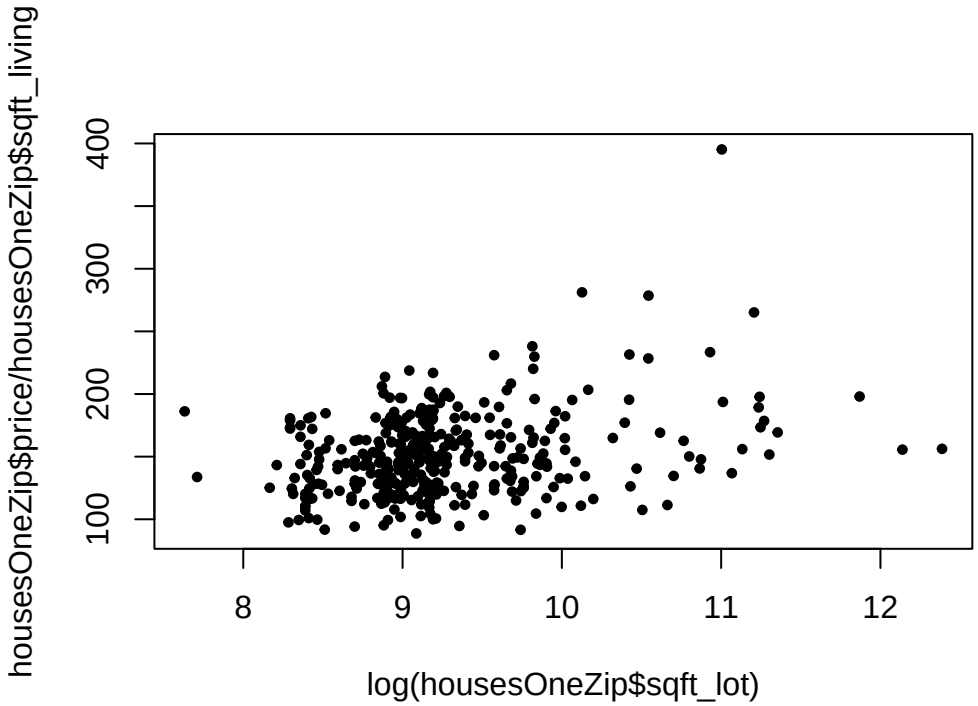
Using **base** graphic plotting routines, the command is

```
plot(log(housesOneZip$sqft_lot), housesOneZip$price / housesOneZip$sqft_living)
```



If you want solid dots, not open circles, you add `pch=19`, and you can set the value of `cex` to control how large the dots are:

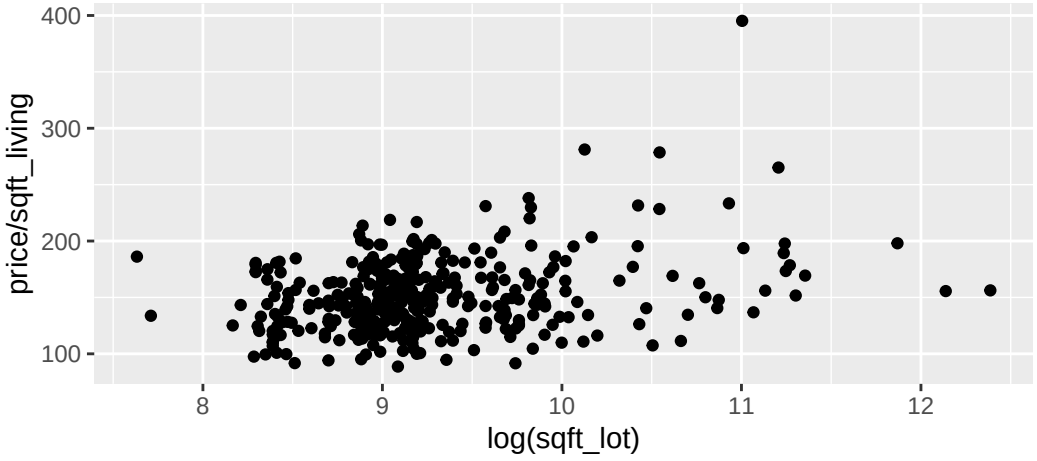
```
plot(log(housesOneZip$sqft_lot), housesOneZip$price / housesOneZip$sqft_living,
     pch=19, cex=.6)
```



Once those extra refinements are added, a relatively simple command has become a bit cumbersome.

This same plot using tools from **ggplot2** is produced as an example in Chapter 7. With a little streamlining, it goes like this:

```
require(ggplot2)
housesOneZip |> ggplot(aes(x = log(sqft_lot), y = price / sqft_living)) + geom_point()
```



This syntax may not be any more taxing to some than the one using **base** tools, particularly if it makes intuitive sense to some of you. Personally, I find it unintuitive to the point that, each of the several dozen times I have directly used **ggplot2** functions, I have spent time looking up examples on the internet to guide me through. But if you want evidence that some effort to learn the use of commands in **ggplot2** pays off, check out the variety of graphics/visualizations at http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html

A compromise: the ggformula package

Still, the authors of the **mosaic** package have written another package, **ggformula** (auto-loaded when you load **mosaic**), as something of a compromise: offering tools that produce **ggplot2**-style graphics but which process inputs more like the **mosaic** functions you have already worked with. The functions `gf_histogram()`, `gf_dhistogram()`, and `gf_dist()`, used already in the course, are from the **gg__formula** package, as are other commands with names beginning as `gf_`functionName`()`.

For direct comparison, I will produce, one more time, the scatterplot above, this time using `gf_point()` from the **ggformula** package:

```
gf_point(price/sqft_living ~ log(sqft_lot), data=housesOneZip)
```