An introduction to plotting data

Eric D. Black
California Institute of Technology
v2.0

1 Introduction

Plotting data is one of the essential skills every scientist must have. We use it on a near-daily basis for visualizing our data, drawing conclusions from it, and communicating our results to the rest of the world. In the old days people drew plots by hand, with pen and ink, on graph paper. This was very easy to learn and quick to do, but it was also very limited. If you wanted to make anything more than a minor change to your plot, changing the scale of the axis, for example, or (horrors!) changing it from linear to logarithmic, you basically had to start over and draw a new one. Today we have tools to make all these tasks, as well as the analysis of our data, much easier, and there is no better time than now for you to start learning them.

This week we are going to start learning the basics of plotting and analyzing data, but I'm going to deviate from the usual way this is taught in most freshman-level courses. Many of you have come to this class with some background in research and already know how to generate plots using a particular software package. Others will be entirely new to the subject. Of those who already have some skill, moreover, the particular software you know how to use will vary widely. For this reason, I'm not going to mandate a one-size-fits-all approach. Instead, I'm going to focus on *what* to plot more than on *how* to do the plotting, and I'm going to let you choose what program to do it in.

Along with this handout you should also have a second one describing how to do the exercises in a particular software program. As of this writing we have handouts for either *Kaleidagraph* or *Microsoft Excel*, your choice. Kaleidagraph is the program I recommend you use if you are new to the

subject. If you are already fluent in another program, such as *Matlab* or *Mathematica*, and want to use that, you are free to do so.

2 Single-variable plots

Here, we will focus on two-dimensional plots of single-variable data. For the purposes of our discussion, let's assume you have a set of data points of the form,

$$\{x_i, y_i\}$$

where i runs from 1 to N, the total number of data points. Moreover, let's assume you suspect there is a connection between the numbers, *i.e.* that y_i depends on x_i somehow.

In this context, x_i is called the *independent variable* and is the one you can control. You have a knob, for example, that allows you to set a particular value for x_i at will, and then measure the outcome of some experiment that that is connected to it. This outcome is y_i, and we will call this the *dependent variable* because it is dependent on x_i. The obvious thing you will want to do is plot a graph of y_i vs. x_i. This graph will be more than a simple illustration. As Edward Tufte points out [1],

> At their best, graphics are instruments for reasoning about quantitative information. Often the most effective way to describe, explore, and summarize a set of numbers - even a very large set - is to look at pictures of those numbers.

LABORATORY EXERCISE 1: Import the following data set into your program, and make a plot of y vs. x.

X	Y
0.0	3.4
0.5	4.0
1.0	4.2
1.5	5.0
2.0	5.2
2.5	5.4
3.0	5.8
3.5	5.5
4.0	5.8
4.5	5.9
5.0	7.1
5.5	7.1
6.0	6.8
6.5	7.9
7.0	8.4
7.5	8.2
8.0	8.9
8.5	8.8
9.0	8.9
9.5	9.9
10.0	9.7

If you don't want to type it in by hand, you can download it at

http://pmaweb.caltech.edu/∼phy003/labs/Data1.txt

3 Error bars

You should have started on your reading from *Taylor* by now, and you have probably even done the first homework set. That is your introduction to uncertainty and what it means, so I am not going to go over that material again here. This section is about how to graphically represent your uncertainty on your plot with *error bars*, little lines above and below your data points whose lengths show to the uncertainty in the data point itself.

Any decent software will allow you to put error bars on your data points, and it should also allow you to specify them any way you want.

LABORATORY EXERCISE 2: Put error bars on your plot. Use $\sigma = 0.3$ for the uncertainty in the Y value of each point.

4 Comparison with a theory curve

Very often you want to plot a theory curve along with your data and see how well the two agree. All modern software will include an automatic routine that will do this for you, but before you haul off and click that button, I want you to understand how it works. I don't want it to be a black box for you.

LABORATORY EXERCISE 3: Estimate the slope and y-intercept of a line that passes through these data points. Use the form

$$f(x) = Ax + B$$

and add this line to your plot. Adjust A and B as necessary until the agreement looks good.

5 Residuals

One way to judge how well a theory curve fits your data is to plot your *residuals*, the difference between the data and the theory for each point, along with the error bars.

$$R_i = y_i - f(x_i)$$

The error bars on R_i are the same as those on y_i. If you've gotten that far in *Taylor*, this is because the error on $f(x_i)$ is essentially negligible.

LABORATORY EXERCISE 4: Calculate and plot your residuals, with error bars.

6 Reduced chi-squared ($\widetilde{\chi}^2$) test

Up to now you have been just looking at your plot and judging by eye whether or not the fit looked right. Now we need a quantitative measure of how good the agreement between your theory curve and data points is. The standard way to do that is with a quantity called the *reduced chi squared*, or $\widetilde{\chi}^2$. You will see more of this in your *Taylor* homework, but for now we will simply define it as the mean-squared deviation of the data points from the theory curve, measured in units of the error bars.

$$\widetilde{\chi}^2 \approx \frac{1}{N} \sum_{i=1}^{N} \left[\frac{y_i - f(x_i)}{\sigma} \right]^2 \tag{1}$$

The distance between a data point and a theory curve that fits its set should be on the order of the uncertainty σ, so the reduced chi squared should be around one for a curve that fits the data. If its value is much greater than one, that usually means that the theory is way off from the data. I say "usually" because it is also possible to get a large $\widetilde{\chi}^2$ by underestimating your error bars, but that seldom happens to careful scientists. More often than not, if you make a mistake in estimating your error bars, you do so by assuming they are bigger than they really are. This makes your $\widetilde{\chi}^2$ much less than one.

Something very much like $\widetilde{\chi}^2$ is what computers use to do automated fitting of theory curves to data sets. Every theory curve has a certain number of *parameters* that can be adjusted to change the curve. In our linear case there were two, the slope A and y-intercept B. When you ask a computer to do a curve fit for you, it will essentially just adjust the parameters to minimize the $\widetilde{\chi}^2$. (It can do this even if you don't supply error bars, but the value of the final, minimized quantity won't be quite the same as $\widetilde{\chi}^2$, the main difference being that it won't necessarily be close to one.)

LABORATORY EXERCISE 5: Calculate the reduced chi squared for your best-looking theory curve. Does it indicated a good fit? Does it indicate appropriately-chosen error bars?

7 Least-squares fitting

In the case where your theory curve is a straight line, it is simple to calculate the two parameters A and B that minimize $\widetilde{\chi}^2$ and therefore optimize the fit. Our expression for the reduced chi squared (Equation 1) becomes

$$\widetilde{\chi}^2 \approx \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - Ax_i - B}{\sigma}\right]^2$$

To find the parameters A and B that minimize this, we just take the derivatives with respect to A and B, and set them to zero.

$$\frac{\partial\widetilde{\chi}^2}{\partial A} = 0 \text{ and } \frac{\partial\widetilde{\chi}^2}{\partial B} = 0$$

This gives us

$$\sum_{i=1}^{N}(y_i - Ax_i - B)x_i = 0$$

and

$$\sum_{i=1}^{N}(y_i - Ax_i - B) = 0$$

These two equations are easy to solve for A and B, giving us straightforward formulae for the optimal coefficients.

$$A = \frac{N\sum_{i=1}^{N} x_i y_i - \left(\sum_{i=1}^{N} x_i\right)\left(\sum_{i=1}^{N} y_i\right)}{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2} \tag{2}$$

and

$$B = \frac{\left(\sum_{i=1}^{N} x_i^2\right)\left(\sum_{i=1}^{N} y_i\right) - \left(\sum_{i=1}^{N} x_i\right)\left(\sum_{i=1}^{N} x_i y_i\right)}{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2} \tag{3}$$

This is known as *least-squares fitting* because it finds the parameters that produce the minimum value for the chi squared. (The careful reader may have noted that I have gone back to using an equals sign = where I had been using an approximately-equals sign before $\approx$. That's because the approximation was in the $1/N$ term in front of the sum, which does not matter for zeroing the derivative.)

LABORATORY EXERCISE *[OPTIONAL]*: Using Equations 2 and 3, calculate the parameters A and B for your data. How close are they to the ones you estimated in Section 4?

7.1 More general cases

This procedure can be extended to any theoretical function that is a linear combination of individual functions. The usual example is a polynomial of order n.

$$f(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n$$

Minimizing the chi squared between this function and a set of data points would yield a set of $n + 1$ linear equations, which can be treated by matrix methods to find the parameters a_0, a_1, etc. Note that the terms do not have

to be polynomials, only a linear combination of things, so the procedure would work just as well for a function of the form

$$f(x) = A\sin(2\pi x) + Be^x + C\ln(x)$$

where the parameters to be adjusted for the fit are A, B, and C. The procedure only breaks down if one of your fit parameters falls *inside* the argument of one of your nonlinear functions. For cases like that more sophisticated methods exist, but you already learned the most basic one in Section 4: adjust the parameters by hand until the fit appears to be a good one. That procedure is often referred to as *chi-by-eye*, and good experimentalists use it far more than they will usually admit.

8 Automated fitting

Remember how I told you I didn't want you using the curve fitting routines built into the software just yet? Well, now you know enough to understand how they work. Try it out, and see what happens.

LABORATORY EXERCISE 6: Using the built-in curve-fitting routine in your software, fit a line to your data, and see how the slope and intercept compare with your own estimates.

References

[1] Edward R. Tufte, *The Visual Display of Quantitative Information*, Graphics Pr; 2nd edition (2001).