1 Comments

"Observe how the extracted parameters fluctuate. How is this result affected if you increase/decrease the number of points in your experiment or increase/decrease the size of the uncertainties? * Compare the mean and standard deviation of the chi2 distribution to expectations"

Increasing sample size reduces noise/fluctuations for the parameter fits. Idk if this is always the best way to look at it, but the Law of Large Numbers says that your fit value will approach the real value as you increase sample size. So if we sample more values before fitting a, b, and c, we will get closer to the true values of these parameters. Getting closer means reducing uncertainty. Increasing uncertainty makes the chi2 distribution broader because we use uncertainty as weights. Larger uncertainty makes smaller weights, so we consider that data point less heavily when doing the fit. Basically with higher uncertainties we have no solid data points where we're like "we know our fit needs to get pretty close to these points."

We expect the mean of the chi2 distribution to be the number of data points minus the number of free parameters, so 12 - 3 = 9. Looking at the 1 and 3, the mean does seem to be around 9, so I think it's a good match. The std dev is sqrt(2*mean) so that's 4.24. We turned the stats boxes off in root, but this can be checked by turning them back on.

The reduced chi2 is basically a normalized chi2 distribution (chi2/mean), so the mean should try to be around 1. Looking at 2 and 4, the means seem to be pretty close to 1. This means the fit can account for the data we randomly generated. A poor model will have the reduced chi2 not be centered near 1 and cannot account for some of the generated data.

As for the distributions of a, b, and c, I wanted to point out that all three of these follow normal curves in the python fit but a and b don't for the root fit. For the root fit we see a and b be pretty noisy and almost uniform while c is pretty strongly gaussian. This could be from how we get our random numbers or some other computer-y differences between root and python. For the 2D histograms, we see the same behavior in root and python: a,b negative correlated, c,a positive correlated, and c,b negative correlated. The correlation is tightest between c and b. Our model is a + b*log(x) + c*log(x)**2 so that's a parabola. a fits the shift, b fits the slope, and c fits the curvature, so these correlations make sense

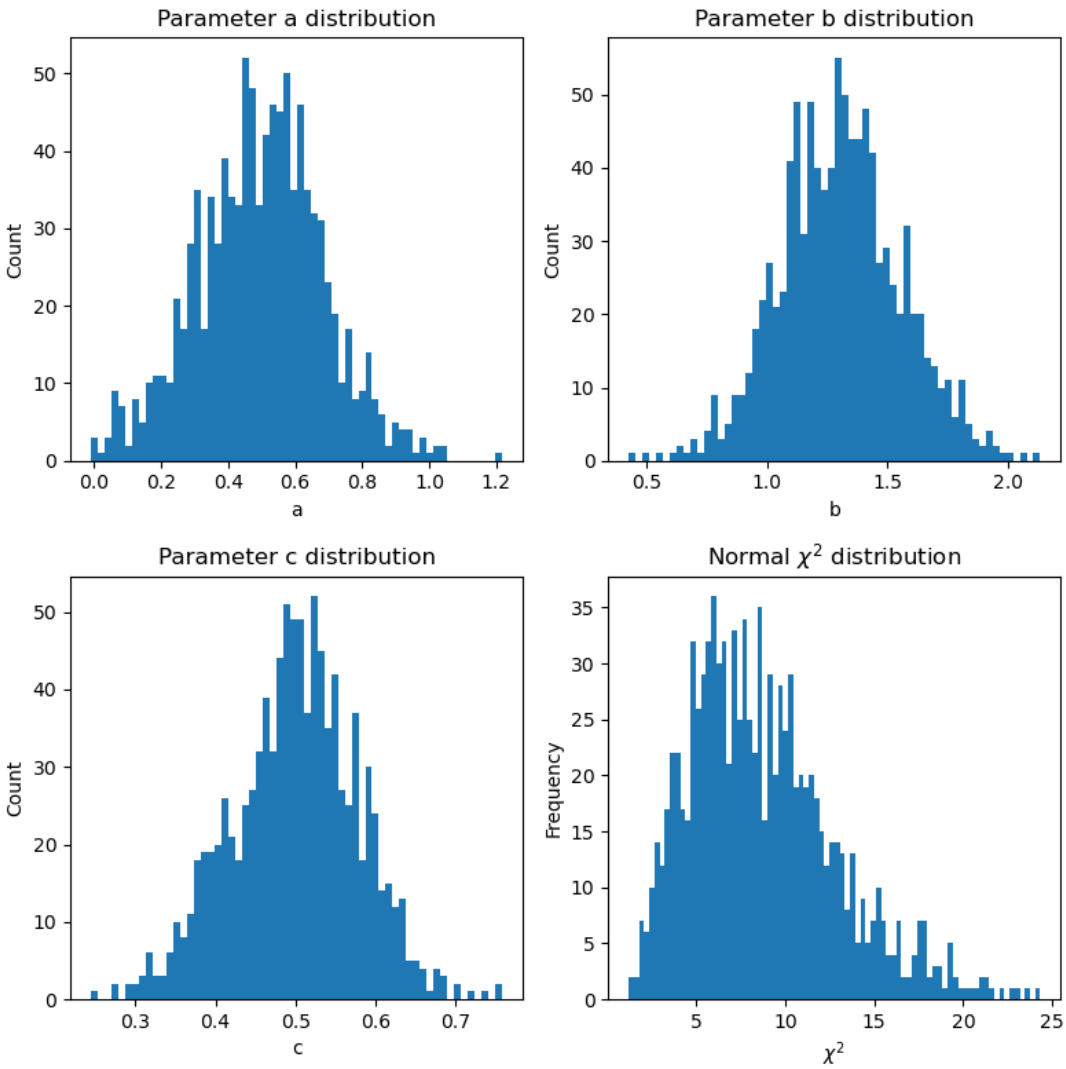

Figure 1: python

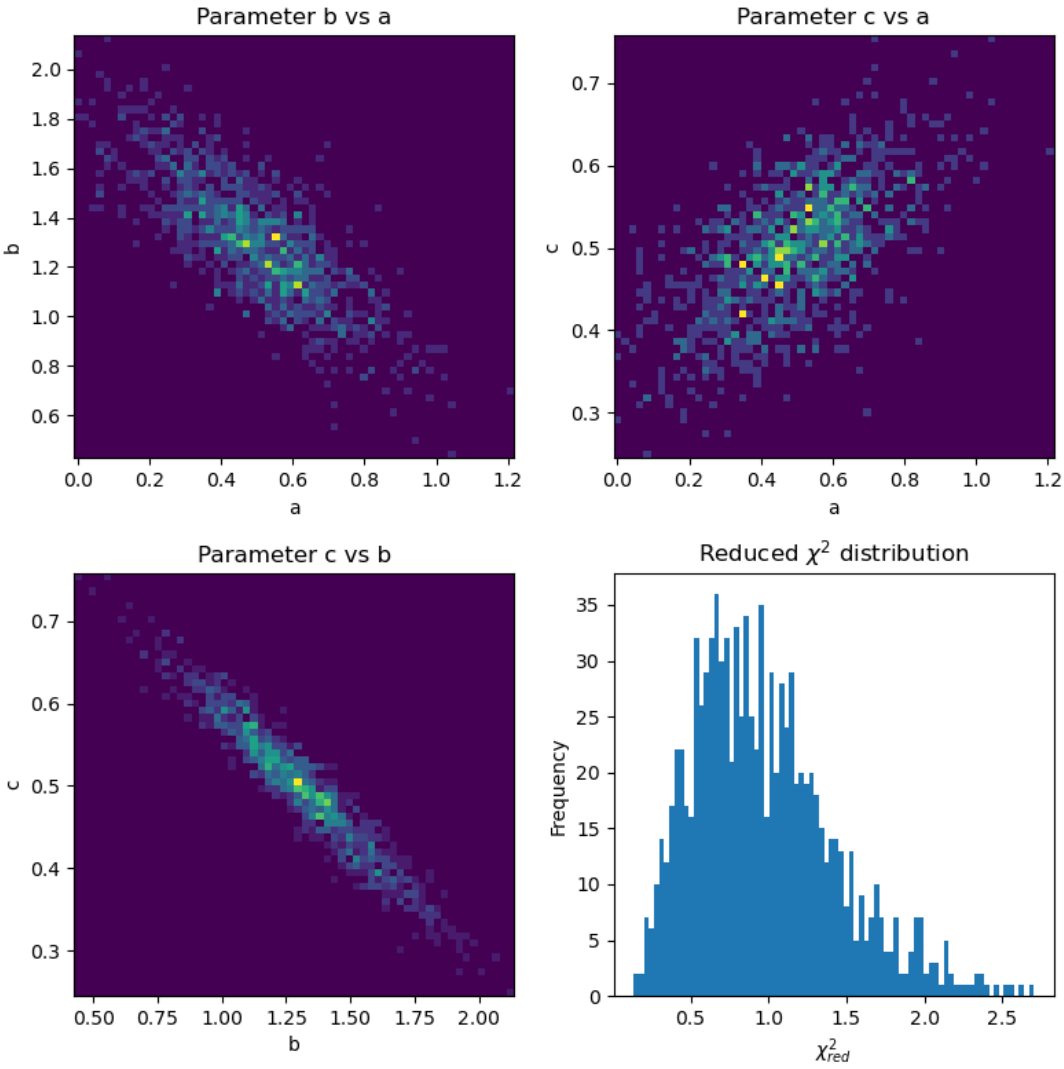

Figure 2: python

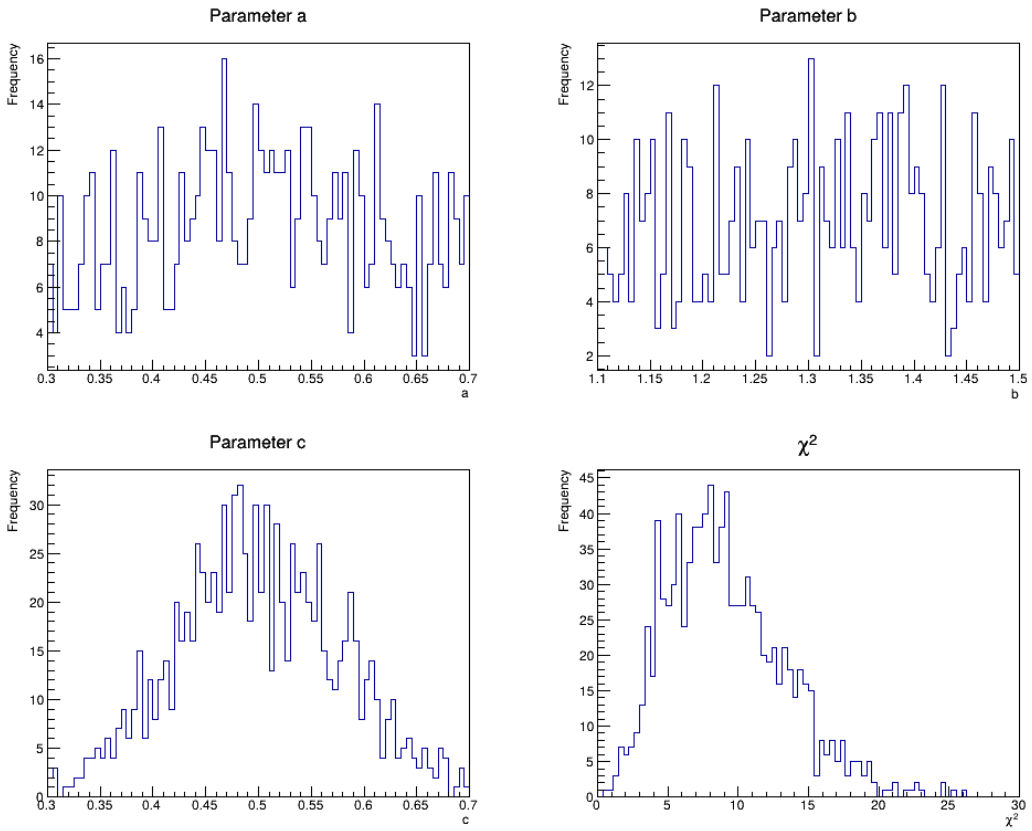

Figure 3: ROOT

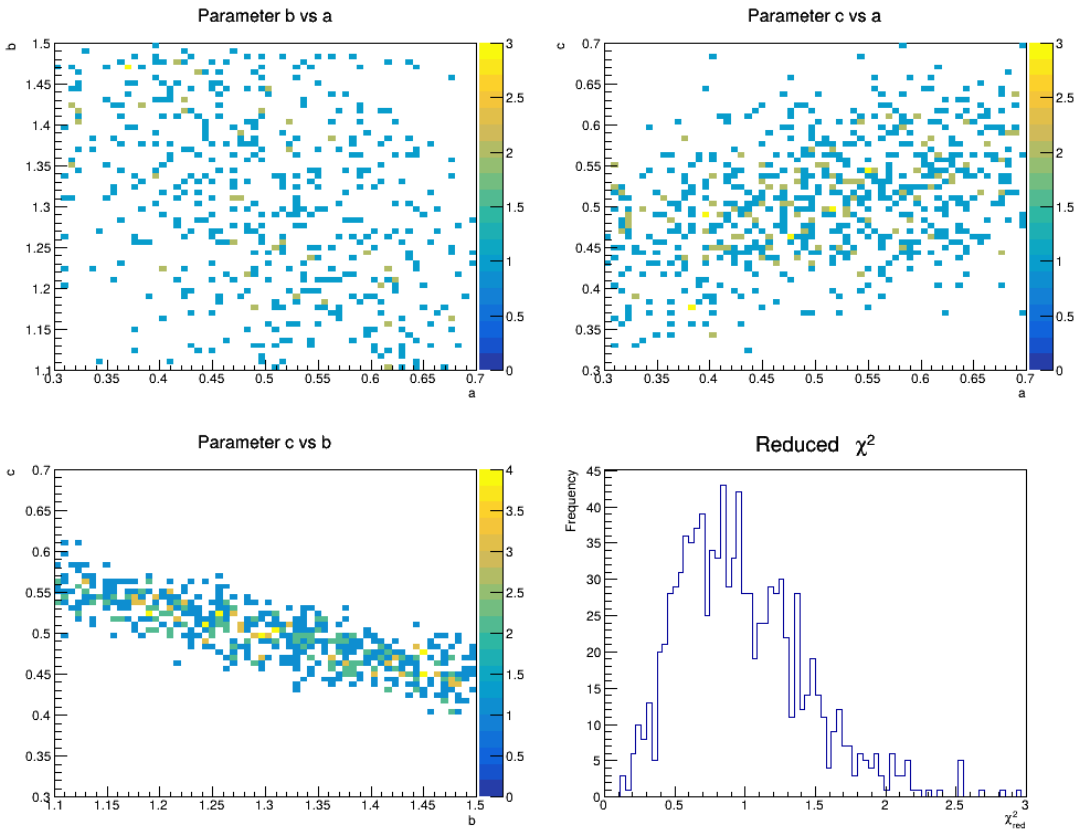

Figure 4: ROOT