Homework 7: The Bootstrap Method

Caden Gobat

April 7, 2021

1 Introduction

In this assignment, we return to fitting functions to data, but this time using a more advanced technique than last week: bootstrapping. Bootstrapping and Monte Carlo methods in our case rely on the assumption that every data point and corresponding error represent their own normal distribution. When this is the case, we can "resample" data by randomly reassigning the value of each y_i within a parameter space defined by a Gaussian centered on its original value with a width of its associated error, σ_i. Doing this for every point in the dataset many times over and fitting a function each time will give us a number of fits, which we can then analyze as a whole. On the whole, we expect the errors on the data points to more or less negate one another, so the average of our fit results will give us a good idea of the "true" statistical value. Having many results for the fitted parameters can also give us an idea of the error on those results themselves, because we can take the standard deviation.

2 Results

Question 1

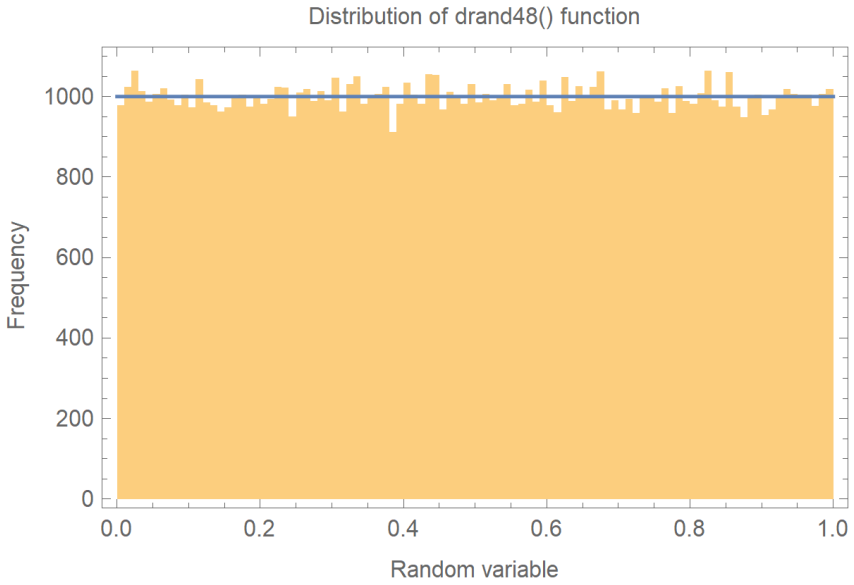

Figure 1: 100-bin histogram (in gold) with 100 bins of 100,000 random numbers generated using C's `drand48()` function. As expected, this conforms relatively closely to a uniform distribution of floating point numbers between 0 and 1, the PDF of which is shown in blue.

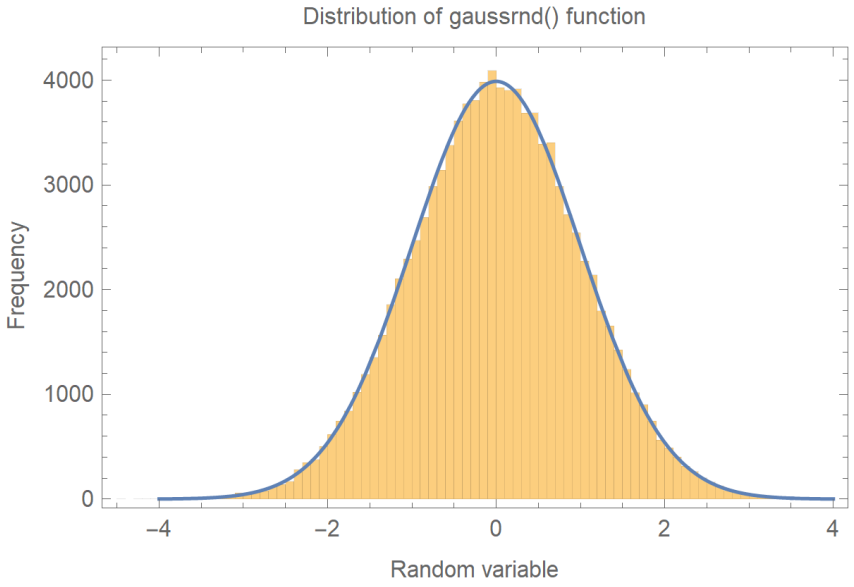

Figure 2: 100-bin histogram (in gold) of 100,000 random numbers generated using our `gaussrnd()` function. As expected, this conforms quite closely to a normal distribution with $\mu = 1$ and $\sigma = 1$, the PDF of which is shown in blue.

Question 2

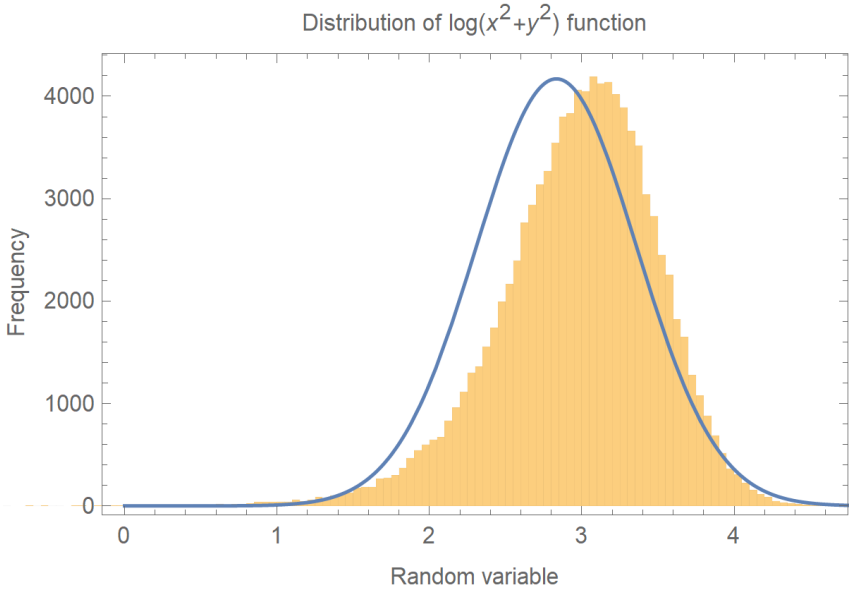

Figure 3: 100-bin histogram (in gold) of 100,000 randomly generated numbers from the distribution $\log(x^2 + y^2)$, where x and y are each uniformly-distributed variables with $\bar{x} = 1$, $\bar{y} = 4$, $\sigma_x = 2$, and $\sigma_y = 1$. The PDF of a normal distribution with $\mu = \log(1^2 + 4^2) = \log(17)$ and $\sigma^2 = \left(\left.\frac{\partial f}{\partial x}\right|_{\bar{x},\bar{y}}\right)^2 \sigma_x^2 + \left(\left.\frac{\partial f}{\partial y}\right|_{\bar{x},\bar{y}}\right)^2 \sigma_y^2 = \frac{16\bar{x}^2}{(\bar{x}^2+\bar{y}^2)^2} + \frac{4\bar{y}^2}{(\bar{x}^2+\bar{y}^2)^2} = \frac{80}{289}$. This PDF is in less agreement with the distribution than in the previous problem because the logarithm throws off the normality of the selection. Although x and y are each normally distributed, taking the logarithm of the sum of their squares creates a sample of numbers that is not normal, so trying to ascribe a Gaussian function of μ and σ in this way is not appropriate.

Question 3

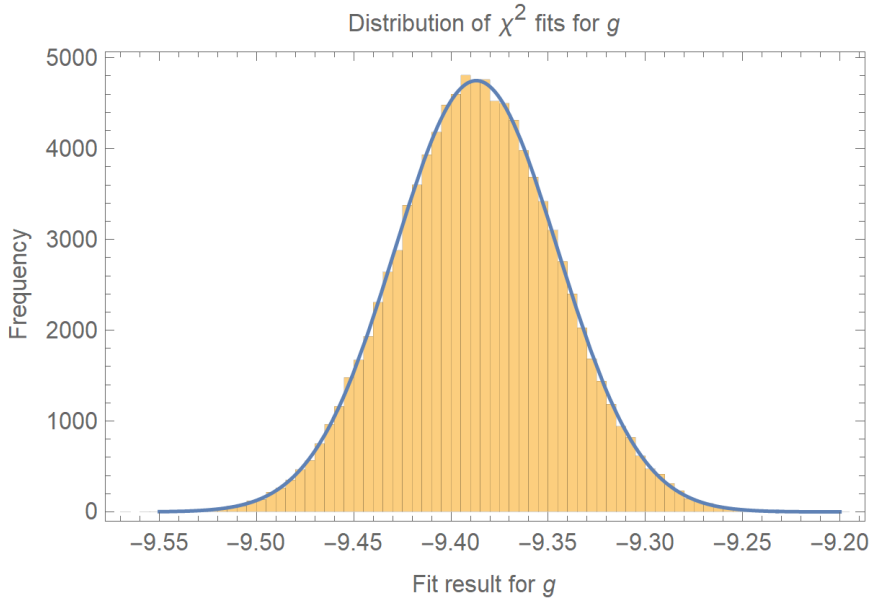

Figure 4: 100-bin histogram (in gold) of the χ^2 fit results for g (the acceleration due to gravity) over 100,000 bootstrapped trials. The average result was $\bar{g} \cong -9.387$, with a standard deviation of $\sigma_g \cong 0.042$ (the corresponding Gaussian PDF is shown in blue). Thus our routine has produced a result of $g = -9.387 \pm 0.042$, which interestingly does *not* overlap with the accepted value of approximately -9.81 m/s^2.

Question 4

Task 1:

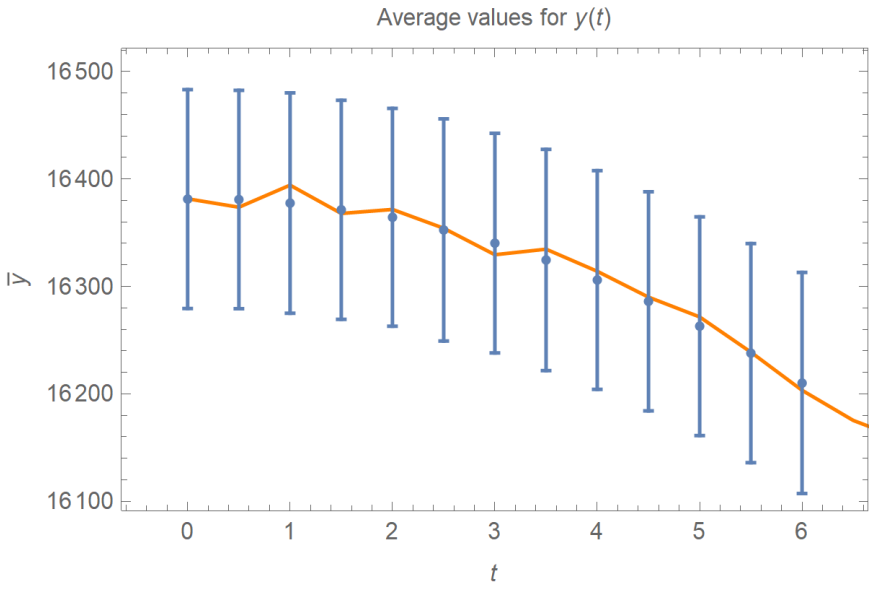

Figure 5: Plot of average values from `bootstrap.dat` of $y(t)$ and their error for each time point t (blue). As expected, this is quite similar to the trajectory in the first 6 seconds of `cannonball.dat`, which is plotted (without its error) in orange.

Task 2:

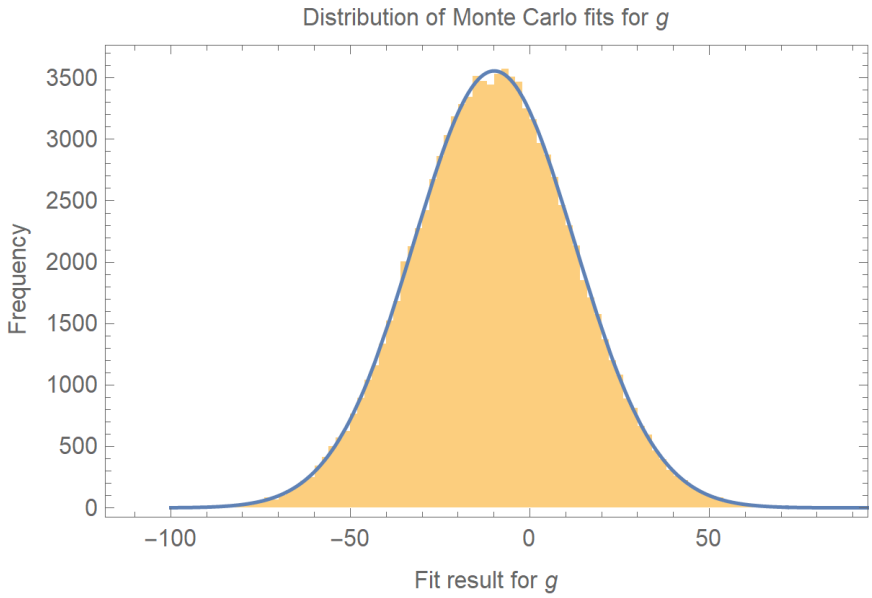

Figure 6: 100-bin histogram (in gold) of Monte Carlo fit results for g, with a corresponding Gaussian PDF plotted over it in blue. This computation produced a result of $g = -9.925 \pm 22.428$, which is closer to the actual value of g (and overlaps), but has a much higher uncertainty due to the large errors on the bootstrap data, as seen in Fig. 5.

Task 3:

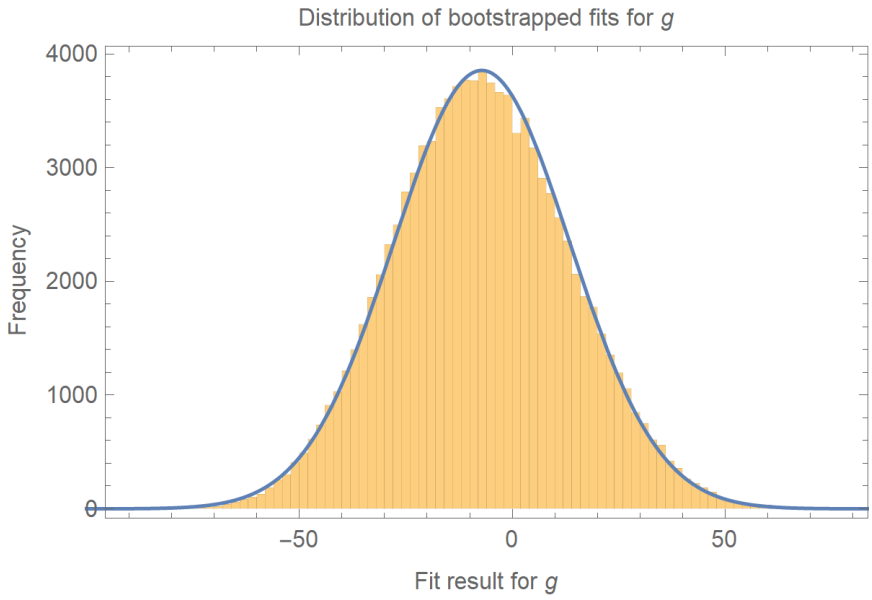

Figure 7: 100-bin histogram (in gold) of bootstrap fit results for g, with a corresponding Gaussian PDF plotted over it in blue. This computation produced a result of $g = -7.16 \pm 20.69$. Although this technically overlaps with the correct value of g due to its large error margin, this result is the least accurate nominal value yet. Again, this can be attributed to the large errors on the data itself. No fitting method or algorithm—no matter how clever—can overcome noisy or error-prone data.

Another possible reason for the large fit uncertainties in the last two cases is the fact that there are, in some sense, only 12 data points, so the resampling process has a much higher capacity to produce vastly different coefficient fit results than when there are 117 (as in the original `cannonball.dat`).

3 Conclusions

Although a bit tricky on the coding side, I found this assignment to be a nice look into these techniques that are so common in scientific data analysis. The first few problems were a good conceptual introduction to the kind of thinking necessary to complete the more advanced ones.

I do wonder if there are more advanced methods one can employ that are more effective at dealing with large errors in datasets, and would be interested to learn more about this.