M Dwarf Multiplicity Project:

Test Code Write-up

While developing code to fit a log-normal model to point estimates of M Dwarf companion frequency as a function of the log of the semi-major axis, we ran into problems with the resultant fit. The method used in doing this compared frequency values from the data of six different M Dwarf surveys to those drawn from a log-normal model, which is described by three fit parameters: the average ($\log_{10}(\mu)$), standard deviation ($\log_{10}(\sigma)$), and amplitude (A). This comparison was done quantitatively using the reduced chi-squared test, under which a perfect fit returns a reduced chi-squared statistic of one. However, the results of the fit when the full selection of data was used returned an unfavorably high reduced chi-squared statistic: 1.30. This poor result in the goodness of the fit called into question the validity of the modeling code. In order to check whether the code itself was causing the problem (bugs, unfound typos, etc.), we began a side-project: writing code to test our model and fitting process.

The test code was written with the purpose of simulating of the real data used in the original fitting process. Broadly speaking, the process undertaken by the test code can be down into nine steps:

1. Assume a log-normal distribution model
2. Assume values for model parameters $\log_{10}(\mu)$, $\log_{10}(\sigma)$, and A
3. Integrate model over four ranges of semi-major axis to get four frequency values
4. Assume a test parent population size
5. Calculate the number of observed companions for each of four semi-major axis ranges
6. Calculate the Poisson error for each of four semi-major axis ranges
7. Compute new frequency values with noise added in for each of four semi-major axis ranges
8. Use the model and original frequencies to fit the parameters to the new frequency value
9. Check to see if the new fitted parameters converge back to originally assumed values (within error)

The first step in the testing process is to use the same log-normal model used originally. Since the study is interested on how the multiplicity varies with surface density distribution, the probability is computed as a function of semi-major axis, a. The log-normal probability distribution function takes the following form when used in log base 10:

$$\mathrm{P}(a_{min} < \mathrm{a} < a_{max}) = \int_{log_{10}(a_{min})}^{log_{10}(a_{max})} \mathrm{A} * \frac{\mathrm{e}^{-(log_{10}(\mathrm{a}) - \log_{10}(\mu)^2/(2\log_{10}(\sigma)^2)}}{\log_{10}(\sigma)\sqrt{2\pi}}$$

From here, values were assumed for $\log_{10}(\mu)$, $\log_{10}(\sigma)$, and A as 1.3, 1.1, and .5 respectively (Hereafter referred to as the "assumed values"). The assumed values were plugged into the log-normal model and integrated over four ranges of semi-major axis (1-10, 10-100, 100-1000, and 1000-10000) to compute four different simulated frequency values. This process served as a simulation of four real surveys and is the basis of comparison later in the test code.

The next step was to calculate the error in these frequency estimates. For this, a parent population (representing the total number of stars surveyed) was assumed. This number was chosen to be 100 for each of the 4 ranges of semi-major axis. The parent population was then multiplied by the simulated frequencies to obtain an estimate of the number of companions the simulation detected ($f = \frac{\#\ companion}{\#\ parent}$). Error was then calculated for each of the four ranges of semi-major axis as the Poisson counting error in the context of frequency:

$$e = \sqrt{\#\ companions}\ /\ \#\ parents$$

The subsequent step was to check if the fitting process used in the original portion of the study was capable of returning fit parameters that converge back to the originally assumed values of the test code. In order to make this meaningful, noise was added to the simulated frequency values by sampling from a Poisson distribution with an expected value of the old companion numbers. This new randomly sampled companion number was divided by the parent population to compute a new frequency. This process was repeated for each of the four ranges of semi-major axis.

Next, the fitting process was used to compare the randomly sampled frequencies to the simulated frequencies and find best fit parameter values. The fitting process works by looping though ranges of possible values for each of the three fit parameters and plugging every value into the four log-normal integrals, one for each range of semi-major axis. Next, the reduced chi-squared test is performed to compare each of the frequency values calculated by the log-normal model to those found previously after the random sampling. The reduced chi-squared test takes the form:

$$\chi^2_{red} = \frac{1}{\nu}\sum_{i=1}^{4}\frac{(f_{i,model} - f_{i,random\ data})^2}{e_i}$$

where ν is the degrees of freedom:

$$\nu = (number\ of\ data\ points) - (number\ of\ fit\ parameters) = 4 - 3 = 1$$

$f_{i,model}$ is the frequency computed directly from the log-normal distribution using a value for the fit parameters drawn from the range of possibilities, $f_{i,random\ data}$ comes from the Poisson randomly sampling (described above), and e_i is the Poisson error (described above). The summation of one through four represents each of the four ranges of semi-major axis. This process is repeated for every possible parameter value and across all 4 ranges of semi-major axis. The reduced chi-squared statistics are continuously compared against one-another to find the value that is closest to one. The fit parameters associated with the reduced chi-squared statistic closest to one are considered to be the best fit parameters. Along the way, every value of the reduced chi-squared statistic is stored in a 3-dimensional cube with indeces set by the three fit parameters for later analysis.

At this point, the test code has returned the reduced chi-squared statistic that is closest to one as well as the associated values of $\log_{10}$(mu), $\log_{10}$(sigma) and A. (hereafter referred to as the "best fit parameter values"). Next, the best fit parameter values are to be compared to the originally assumed parameter values. The first task in accomplishing this is to marginalize over the best fit parameters to create 1-dimensional probability density functions (PDF's) for each of the three, then to assign error bars as a percentage of area under the PDF's.

Marginalizing over the fit parameters made use of the cube of chi-squared statistic values we stored earlier. First, each of the chi-squared values in the cube were converted to probability values according to the chi-squared distribution:

$$P(\chi^2) = (\chi^2)^{\frac{\nu-2}{2}} \cdot e^{\frac{-\chi^2}{2}}$$

Next, the cube was collapsed three times onto each of its axes by summing over the other two. This left one dimension of probability values for each of the three fit parameters, representing the full probability distribution for each parameter. Error bars were computed using cumulative integration over the PDF's to select a 66% range, +/- 33%, centered around the best fit parameter values. This process was repeated for each of the three fit parameters

If each of the originally assumed parameter values fall within error of their respective best fit parameter value, then the fitting process can be considered a success. If the code is working as intended, the originally assumed parameter values will fall within error of the best fit parameter values 66% of the time. This will ultimately show if the test code has succeeded or failed and will answer the originally posed question about the validity of the code.

The current problem facing this test code is that the fit does not succeed the expected 66% of the time. Using a frequentist approach, the code was run twenty times. A run of the code was considered successful if the fit process returned a reduced chi-squared statistic value within .1 of one and if each of the three originally assumed parameter values fell within error of their respective bet fit parameter values. When this

conducted, the code succeeded only 1 time (5%). This discrepancy between the expected percentage of successes and the actual percentage of successes further calls into question the validity of the code, as we must now carefully examine where something has gone wrong.