Mixture Model for Star Locus

The code begins with a maximum likelihood fitting of the parameters μ_1, μ_2, σ_2 of the `ClassStat` Gaussian mixture distribution called `starloglik` using a constrained BFGS method. The model here is

$$\Pr(\hat{c}_b|c) = a\mathcal{N}(\hat{c}_b - c|\mu_1, 1) + (1-a)\mathcal{N}(\hat{c}_b - c|\mu_2, \sigma_2),$$

where $c = 0$ for stars and $\hat{c}_b$ is the observed `ClassStat` value for a given band; he gives no information for how he decided upon his initial parameter guesses. This process is repeated for all four bands; a sample empirical histogram fitted with a distribution given by the chosen parameters by both our code and Henrion's is shown below. It should be noted that the specified function for $\mu'(m)$ was simply chosen to give a good fit; he mentions that it can be changed should one so desire.

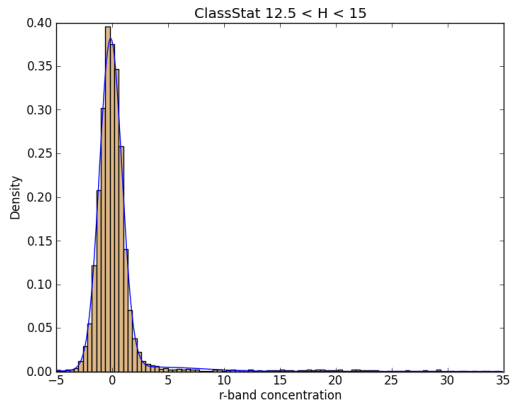

Figure 1: Our Code

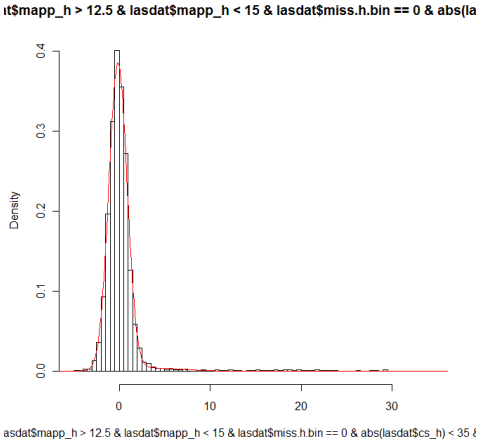

Figure 2: Henrion Code

Estimate Galaxy Locus

Next, the code seeks to describe the distribution for the `ClassStat`, c, distribution for galaxies, which is a more complicated task. First, the empirical mean and variance of the appropriate magnitudes (i.e. those from galaxies) and the median and variance of c values is obtained from a random sample of 20,000 observations. This information is used to then find the optimal $\mu(m)$ and $\sigma(m)$ for the model of c for galaxies, $\Pr(c|\hat{m}, t = g) = \mathcal{L}[c|\mu(m), \sigma(m)]$, where $\mathcal{L}(x|\mu, \sigma)$ is a lognormal distribution. The value of μ' and σ' are found through LS optimization via the BFGS method on the functions

$$\mu'(m) = \left(1 - \frac{m}{m_{max}}\right)\{[\nu_1(m-\nu_4)^2 + \nu_2(m-\nu_4) + \nu_3]^{\nu_5} + \nu_6\}$$

and

$$\sigma'(m) = \eta_1 10^{\eta_2(m-11)+5},$$

respectively, where m_{max} is the upper detection limit in the reference band and $\nu_1, \nu_2, \nu_3, \nu_4, \nu_5, \nu_6, \eta_1$, and η_2 are fitted parameters. The $\mu'(m)$ function corresponds to the function coded as `SSmedCSHyp` and the $\sigma'(m)$ function corresponds to `SSvariCS`. Below are figures, again of our code and Henrions, showing the empirical magnitude values with the $\mu'(m)$ function fitted with the optimized parameters, as well as figures showing $\sigma'(m)$ using the optimized parameters.

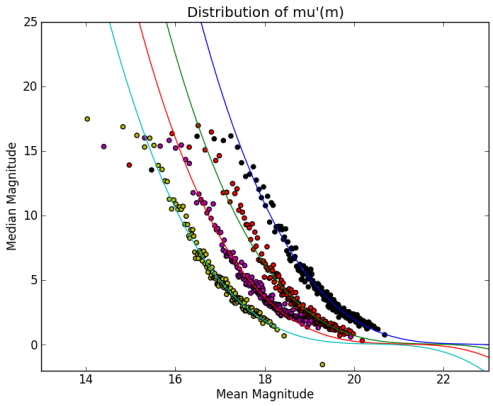

Figure 3: Our Code

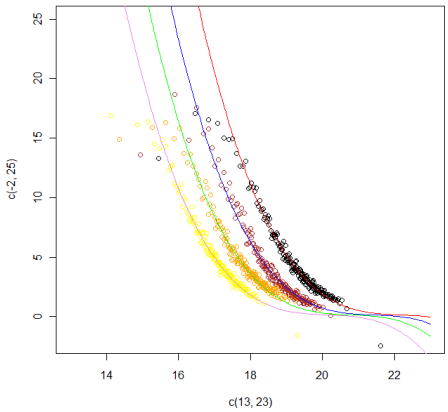

Figure 4: Henrion Code

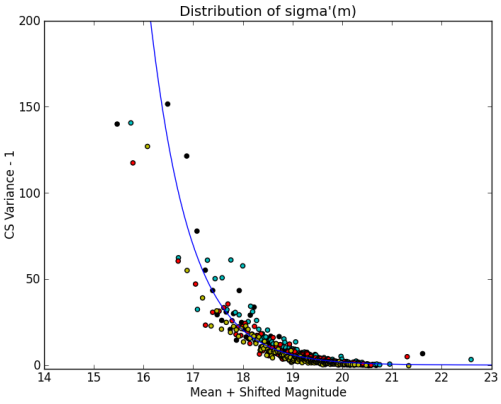

Figure 5: Our Code

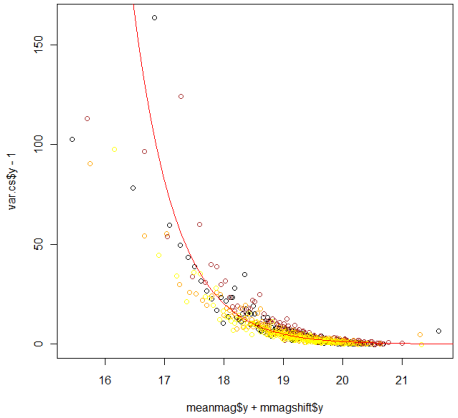

Figure 6: Henrion Code

Fit Magnitude Distribution

This part has been the bane of my existence this week. As I noted in the email, nothing further in the code (with the exception of the next section) can be compiled without the parameters that come from the function `FitMagInit`. This function is supposed to fit a model of observed number counts to magnitude data. The fact that this is not working is starting to make me have doubts about the quality of the BFGS method in the scipy package; if you remember several weeks ago I noted sometimes I got parameters that did not exactly match his. I have tried changing the optimization to other method (e.g. Nelder-Mead, Powell) but have thus far been without success. I even tried to code the function he wants optimized from scratch, which did not work either. `FitMagInit` is a function of functions and I have painstakingly made sure that the results of every one of mine match his results; the only problem comes with the actual optimization part (the most important part!) and that is why I am at a loss and the only possibility in my mind right now is the BFGS method in scipy is the culprit. I tried a couple different versions of BFGS that I downloaded but they were all beta version and did not work at all.

Number Counts

The number counts for stars and galaxies are what provide the prior used to classify sources for which the image is ambiguous. Intrinsic number counts are of the form

$$\rho_t(Y) = \frac{dN_t}{dY} = \alpha_t \log(10) N_t 10^{\alpha_t(Y-Y_0)},$$

where N_t is the number of sources of type t brighter than the reference magnitude Y_0 and α_t is the type-dependent logarithmic slope (he notes that Y_0 is set to be the Y-band magnitude limit). In order to fit these parameters, it is necessary to consider the incompleteness of each band, which is denoted as

$$\Pr(det|m_b) = \frac{1}{2}\,\mathrm{erfc}\left(\frac{m_b - m_{lim,b}}{\Delta m_b}\right),$$

where the magnitude limit $m_{lim,b}$ and incompleteness range Δm_b are fitted in each band for both stars and galaxies. Here, $\mathrm{erfc}(x) = 2\int_{\sqrt{2}x}^{\infty}\mathcal{N}(x'|0,1)$ (ie, the complementary error function). These parameters are fitted by optimizing

$$[\log(N_t) - \log(10^{\alpha_t(Y-Y_0)}] * \Pr(det|Y),$$

where $N_t \triangleq Y$. After the parameters have been chosen, fitting $dN_t/dY\Pr(det|Y)$ to the observed counts produces the figures below. Having the relative numbers of stars and galaxies at given magnitudes give pretty accurate prior probabilities. The bars in the histograms are obviously empirical counts and the functions in the code that were optimized to give us the parameters we needed for stars and galaxies are `SSst` and `SSga`, respectively.

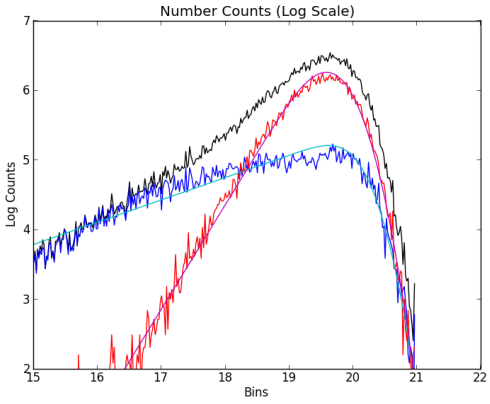

Figure 7: Our Code

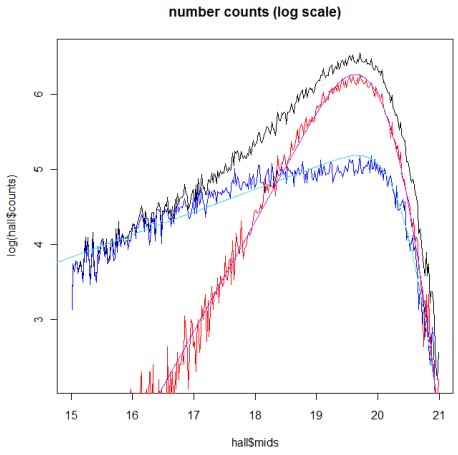

Figure 8: Henrion Code

The next graph shows the proportion of stars to all objects modeled against the empirical proportion of star counts. The black line represents the function

$$\frac{10^{\alpha_t(m_s-Y_{s,0})}\cdot 0.5\cdot \mathrm{erfc}((m_s - m_{s,lim,b})/\Delta m_s)}{10^{\alpha_t(m-Y_{s,0})}\cdot 0.5\cdot \mathrm{erfc}((m_s - m_{s,lim,b})/\Delta m_s) + 10^{\alpha_t(m_g-Y_{g,0})}\cdot 0.5\cdot \mathrm{erfc}((m_g - m_{g,lim,b})/\Delta m_g)},$$

which is referred to as `MxCfGl` in the code. The red is the empirical proportion of stars.

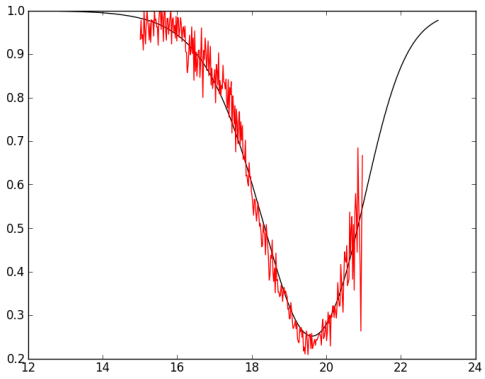

Figure 9: Our Code

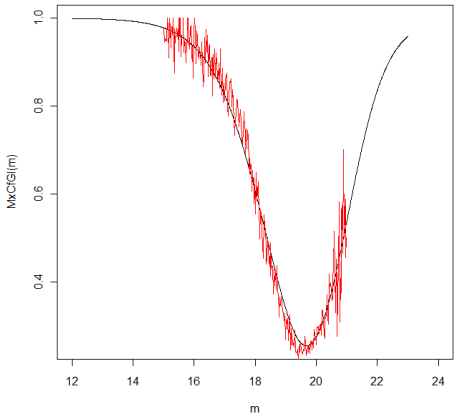

Figure 10: Henrion Code

Rest of the Code

Due to me wasting days in a vain attempt to fix the problem I have been having with `FitMagInit` I have not been able to actually code the classifier. This really frustrates me because given what I have shown above, our prior information looks good and while he has many images, plots, and figures that would take me a long time to figure out how to code, I feel that coding the actual classifier should not be too incredibly difficult.