Inference, Models and Simulation for Complex Systems — CSCI 7000-001
Lecture 5 — 8 September 2011
Prof. Aaron Clauset

1 Model comparison

Model comparison, a.k.a. model selection, is typically done by comparing the likelihoods of some data $\{x_i\}$ under model A_1 and under under model A_2 (or, more generally, comparing its likelihood under each of several models $\{A_j\}$). Under a likelihood framework, the interpretation is clear: if the data are more likely under A_1, then A_1 is a better explanation of the data; and conversely for A_2.

The differences between the several likelihood-based approaches for model comparison come down to different philosophical approaches to answering these three questions.

1. How should we adjust the comparison when A_1 and A_2 have different sizes or "complexities"? (Typically size = number of parameters, but sometimes a more subtle notion is required.)
2. How should we account for increasing sample size?
3. How do we account for possible fluctuations in the data, which may lead to fluctuations in our conclusion of which model is "better"?

1.1 The Bayesian approach

The Bayesian approach to model comparison is increasingly popular, and can be very powerful. It can be used both to compare completely orthogonal models or to choose the number of parameters or model "complexity" within a single family of models. For instance, consider fitting a kth order polynomial to some scatter data where k is a free parameter. In this case, k controls the size of the model and if $k = n$ we can fit the data perfectly. Thus, we want an automatic method for choosing a some $k < n$ that does a reasonably good job at trading off between maximizing the likelihood of the observed data and generalizing to unobserved data.

The primary Bayesian assumption is that the data $\{x_i\}$ are fixed.[1] The fundamental quantity in Bayesian model comparison is the *marginal likelihood* (sometimes also called the "evidence"), which is simply the likelihood of the data integrated over *all* parameter choices:

$$\Pr(\{x_i\} \,|\, A) = \int_\theta \Pr(\{x_i\} \,|\, \theta) \Pr(\theta \,|\, A) \, \mathrm{d}\theta \ . \tag{1}$$

[1] This assumptions is different from that of the frequentist method for model plausibility we saw in Section 2, which assumes the data $\{x_i\}$ are random variables, i.e., if you looked again, you would get different values drawn from the same generative model.

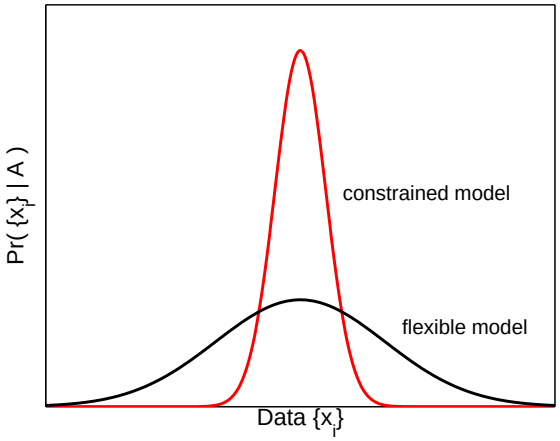

Figure 1: A cartoon of the difference between constrained models (low "complexity") and flexible models (high "complexity") in the range of data they can fit well (produce with high likelihood).

If we have two models A_1 and A_2, then we can compare the marginal likelihoods of each, i.e., compare $\Pr(\{x_i\} \mid A_1)$ to $\Pr(\{x_i\} \mid A_2)$ and ask which is better (larger). Or, if we have more than two models, we can compute the marginal likelihoods of each and ask which among the set is the largest.[2] The whole point of marginalization is to eliminate the effect that different numbers of parameters have—that is, models with more parameters are more "complex" and thus more flexible, but, as a result, they assign lower likelihoods to all the data sets they can generate. In contrast, simpler models assign higher likelihoods to a smaller range of data sets, and thus should win under this kind of model comparison. Thus, marginalization in the Bayesian framework is a kind of formalization of Occam's razor. [3] (This idea is illustrated by a cartoon in Figure 2.)

Further, it can be shown that this approach to model comparison is asymptotically consistent when the true generating process is in the set of models we fit to the data, that is, given that $\{x_i\} \sim M$ and $M \in \{A_j\}$, then as $n \to \infty$, $\sup_j \Pr(\{x_i\} \mid A_j) \to M$.

[2] Marginalization requires a potentially computationally expensive integration step. As a result, the full Bayesian marginalization approach is not as commonly used as computationally cheaper alternatives like the penalized likelihood functions we'll see Section 1.1.1.

[3] Albert Einstein in 1933: "It can scarcely be denied that the supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience." This is often paraphrased as "Models should be made as simple as possible, but no simpler." This notion that accuracy should not be sacrificed for simplicity is slightly different from that of Occam's razor, and so is sometimes called Einstein's razor to distinguish the two.

Bayes factors are a slight variation on the general marginalization approach that makes the procedure look a lot like a likelihood ratio test (see Section 1.3 and the first problem set). That is, a Bayes factor is the ratio of marginal likelihood of A_1 to that of A_2:

$$K = \frac{\Pr(\{x_i\} \,|\, A_1)}{\Pr(\{x_i\} \,|\, A_2)} = \frac{\int_{\theta_1} \Pr(\{x_i\} \,|\, \theta_1) \Pr(\theta_1 \,|\, A_1) \,\mathrm{d}\theta_1}{\int_{\theta_2} \Pr(\{x_i\} \,|\, \theta_2) \Pr(\theta_2 \,|\, A_2) \,\mathrm{d}\theta_2} \ . \tag{2}$$

The interpretation of Bayes factors is done by heuristic: if $K \gg 1$ then the result is interpreted as strong support for A_1, while if $K \ll 1$, we rule in favor of A_2. If $K \approx 1$, then we say that we cannot tell which model is better. This is one place where a true likelihood ratio test offers an advantage over the Bayesian approach: in a likelihood ratio test, by assuming that the data are random variables and thus that the likelihoods (or marginal likelihoods) are also random variables, the likelihood ratio test can quantitatively estimate just how close to 1 is "too close to call."

1.1.1 Information Criteria

A computationally cheap alternative to more sophisticated approaches like Bayesian marginalization is to use an "information criterion" to level the playing field between models of different sizes, thereby penalizing larger models and reducing their higher likelihoods. Mathematically, this approach looks like this:

$$\ln \mathcal{L}_\mathrm{c} = \ln \mathcal{L} - f(k, n) \ , \tag{3}$$

where $f(k, n)$ is a convex function of the number of parameters in the model k, and possibly also on the number of observations n. Thus, the more complex or flexible a model, the larger its penalty.

The *Bayesian Information Criterion* (BIC) is defined as

$$f_\mathrm{BIC} = \frac{1}{2} k \ln n \ . \tag{4}$$

There are several ways to mathematically derive the BIC, e.g., by considering an asymptotic Gaussian approximation to the shape of the likelihood function at the MLE's location. Intuitively, the functional form represents a kind of volume measure in the k-dimensional likelihood space, i.e., $\mathrm{e}^{f_\mathrm{BIC}} \propto n^k$, so that models with a more concentrated likelihood function get less of a penalty than "bigger" models where the likelihood is spread out over a larger part of the parameter space (see again Fig. 2).

The *Aikake's Information Criterion* (AIC) is another popular information criterion (named after its creator Aikake, who originally called it "An Information Criterion") and is defined as

$$f_\mathrm{AIC} = k \ . \tag{5}$$

The AIC can also be derived from an asymptotic analysis of likelihood functions. The tiny difference in the functional form between the AIC and BIC points to the fact that there are differing philosophical preferences of what makes a good model underlying the mathematical analysis. For largely historical reasons, AIC is typically used in the ecological sciences, while BIC is used in other parts of biology, and full marginalization is favored in machine learning.

1.2 The minimum description length principle

We won't cover the minimum-description length (MDL) version of model comparison. MDL is an alternative philosophy for choosing among models that is based on notions of model-based compressibility and information theory[4] and it's also framed in terms of penalized likelihood functions. That is, there's some $f_{\mathrm{MDL}}(k, n)$ that you would attach to your log-likelihoods to correct for its flexibility.[5]

The idea behind MDL is that a good model is one that requires a small number of bits to encode the observed empirical data, which is (i) the number of bits required to specify the model itself (its parameter values) plus (ii) the number of bits required to encode the observed data using that model (which simply their log-likelihood under that model).

The form of this penalty term depends strongly on the way we represent encode the parameter values. The simplest case is an integer value from the interval $[1, n]$; this takes exactly $\log_2 n$ bits to write down if each parameter value is equally likely.[6] If there are k such integer parameters, the MDL penalty term takes the form $k \log_2 n$, which looks much like the BIC.

1.3 Likelihood-ratio tests

This approach to model comparison is covered in the first problem set. The main difference between it and the other approaches is that, as a frequentist method, it assumes that the data $\{x_i\}$ are random variables and thus, small differences in likelihoods under two candidate models could be generated by statistical fluctuations. That is, likelihoods are random variables, thus differences between likelihoods are also random variables and thus the sign of the difference might be erroneous, if we happened to be unlucky in our draws of data.

The upside of this assumption is that a likelihood ratio test (LRT) can say "I don't know" when asked which model is best; this is particularly useful if you want to avoid being forced to make

[4] The single best information theory book I can recommend is Thomas M. Cover and Joy A. Thomas, "Elements of Information Theory" (1991).

[5] See P.D. Grünwald, "the Minimum Description Length principle." MIT Press (2009).

[6] If the different values are not equally likely, then we can do better using classic prefix coding techniques, e.g., Huffman encoding, to assign shorter codewords to the more common values. This is a bit like having a Bayesian "prior" on the parameter values.

some conclusion when the data are essentially ambiguous. However, like all model-comparison techniques, LRTs cannot completely circumvent the fallacy of model comparison (see Section 1.6), i.e., LRTs cannot tell you if all your models are bad.

1.4 Cross-validation

Cross-validation can also be used to compare models and does so without using likelihood functions. This property can be useful when a model is too complicated to express as a likelihood function, but never-the-less makes precise predictions.

Cross-validation (CV) works by simulating an out-of-sample prediction experiment in which we "train" the models on some observed data and then score how well each does at predicting future data. We simulate this process by taking our empirical data $\{x_i\}$ and randomly dividing it into two disjoint sets y and z (where $y \cap z = \emptyset$ and $y \cup z = \{x_i\}$).[7] We fit any necessary model parameters using only y, and we score each model on the likelihood of generating z.

The key step in this process is to choose the partitioning of data between y and z uniformly at random—that is, to assign each observation x_i independently with probability f to the "training" data set—and to repeat the process many times, averaging the results across the many trials. Here's a psuedo-code version of this algorithm. Let $f = |z|/|x|$ be the fraction of data we try to predict out-of-sample and N be a large number:

```
for i=1:N
    for j=1:m                                          % for each of the m models
        {z_i}    = f*n observations from {x_i} chosen uniformly
        {y_i}    = the remaining observations from {x_i}
        fit model A_j to data {y_i}
        L[j,i]   = likelihood of {z_i} under A_j    % or other ‘‘score’’ function
    end
end
mean-score = mean(L,2)
winner     = sup(mean-score)    % inf if the smaller values are ‘‘better’’ scores
```

The leave-one-out (LOO) version of CV, where $f = 1/n$, can be shown mathematically to choose the correct model under a wide variety of conditions.

[7] When the data are ordered, as in a time series, we instead choose a random point in time and train the model on the data up to that point and test it using the data after that point.

1.5 Goodness-of-fit

We won't cover them in detail here, but another set of likelihood-free approaches for model comparison use notions of "closeness" between the model output and the data. If the measure of closeness has certain mathematical properties, then choosing the model that maximizes closeness, e.g., minimizing the KS goodness-of-fit statistic, can be asymptotically accurate.[8] That is, we use the goodness-of-fit statistic as a replacement for the log-likelihood function.

1.6 How to lie with a model comparison

These quantitative model comparison frameworks are both powerful and useful but there's an important caveat. Model comparison techniques can only identify the true generating process M if that model is within the set of models being compared, i.e., if $M \in \{A_i\}$. We hope that our quantitative machinery will always pick the model within that set that is "close" (in some abstract and, typically, unspecified way) to the true generating process. But, if M is not in our set, by construction these techniques will still pick some model, i.e., $\sup_j \Pr(\{x_i\} \mid A_j)$, as being the *best*, but it cannot warn us that the model it chose is not the correct model. Thus, it might be more clear to say that model comparison techniques choose the *least bad* model among the set considered, which is a relative—not absolute—measurement of quality.[9]

To illustrate this point, consider the following thought experiment. Suppose Nature, in her infinite wisdom, uses model C to generate some empirical data $\{x_i\}$, which she then gives to us. But, being distracted by other issues, she neglects to tell us anything about model C. We attempt to recover this information: we think deeply about what we know and develop two good-sounding candidate generating processes A and B. However, neither of these are the true generating process, i.e., $C \notin \{A, B\}$. We then proceed through our favorite model comparison algorithm, which shows that model A wins—the data were more likely to be generated by it than by the alternative. We then write up our results and share with the world.

What's wrong with this picture? The problem is that because model C was not in the set of models being considered, we arrived at a conclusion that doesn't tell us much about the true generating process. That is, our conclusion is technically correct but scientifically unhelpful. Even if model A

[8] See D. Pollard, "Empirical Processes: Theory and Applications." Institute of Mathematical Sciences (1990).

[9] You may have heard of George E. Box, who is famously quoted for saying "essentially, all models are wrong, but some are useful." Although this statement is oft repeated as gospel by modelers, it is based on a false assumption. That is, there is at least one model that is *correct*: the modern quantum mechanical model of physics makes astoundingly accurate predictions about the physical world and, to the best of our ability to demonstrate otherwise, is how the world actually works. Thus, a more defensible version of Box's quotation would read "*some* models are wrong, and some are useful." This rewording makes clear the importance of figuring out if a model is wrong and, potentially separately, if a model is useful. That is, model comparison is not the end of the scientific process because it leaves unanswered the question of whether any of our models are actually reasonable.

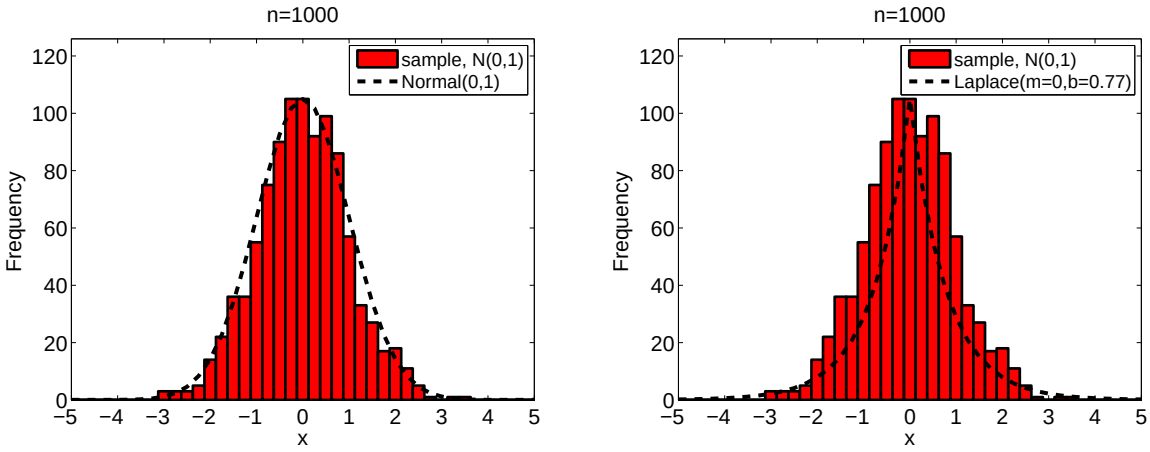

Figure 2: Histogram of $n = 1000$ iid observations drawn from $N(0, 1)$, with the pdfs for the true generating process and the maximum-likelihood Laplace distribution.

is closer to the data than model B, it could still be very far away when compared with the true generating process C. Model comparison only produces *relative* measures of goodness and is unable to tell us when all models under consideration are terrible. By construction, it must choose a winner.

This behavior is not problematic if we're not interested in the true generating process. For instance, this is often the case in industrial settings where we're more concerned with squeezing our error rates as much as possible or where the true generating process is probably too complex to ever codify. However, in scientific settings, this is not the case and interpreting the results of a model comparison exercise as if they were the results of a model plausibility test, i.e., concluding that model A is the correct generating process, can lead to false conclusions.

2 An example of marginalization

To illustrate how Bayesian marginalization works, we will walk through a simple example. Matlab code for everything is included in Section 3.

The models we will consider are the Normal distribution $N(\mu, \sigma^2)$ and the Laplace distribution $L(m, b)$. The pdf for the Laplace disribution is

$$\Pr(x) = \frac{1}{2b} \mathrm{e}^{-\frac{|x-m|}{b}} \quad , \tag{6}$$

where b and m are parameters. This distribution is simply a symmetric exponential function, cen-

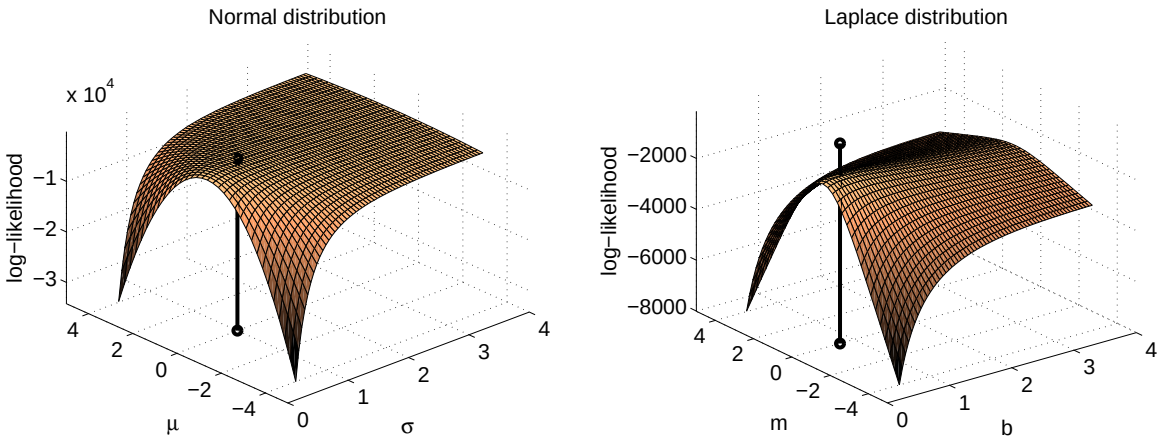

Figure 3: Likelihood surfaces for the Normal and Laplace models. The vertical line shows the location in parameter space of the maximum-likelihood model. Notably, the variance is not equal along the two dimensions: varying μ and m produces much greater changes in the log-likelihood values than variations in σ and b.

tered at $x = m$ and with decay rates controlled by b.[10]

First, we will need some "empirical" data. We draw $n = 1000$ iid observations from a standard Normal distribution $N(0, 1)$. That is, $N(0, 1)$ is the true generating process of our data. Figure 1.6a shows the histogram of the data with the pdf for the underlying generative process and Fig. 1.6b shows the maximum-likelihood Laplace distribution.

Before we compute the marginal likelihood, let's take a look at the likelihood surfaces for the two models. Figure 2 shows the two-dimensional likelihood functions. In each case, I've highlighted the location of the maximum likelihood parameterization (black line). Note that varying the "location" parameters, which are μ and m, yields much more variation in the log-likelihood scores, indicating that there is much less uncertainty about the good values of these parameters. In contrast, varying the "scale" parameter, which are σ and b, produces much less variation (except for some choices of the location parameter).

To illustrate the idea of marginalization, let's fix the location parameters at their true value, $\mu = m = 0$, and only consider variations in the scale parameters σ and b. Figure 2 shows the pdfs for several choices of these parameters along with the empirical histogram; in the inset, we show the

[10] It's a simple calculation to derive the maximum likelihood estimator for the Laplace distribution. You should try it.

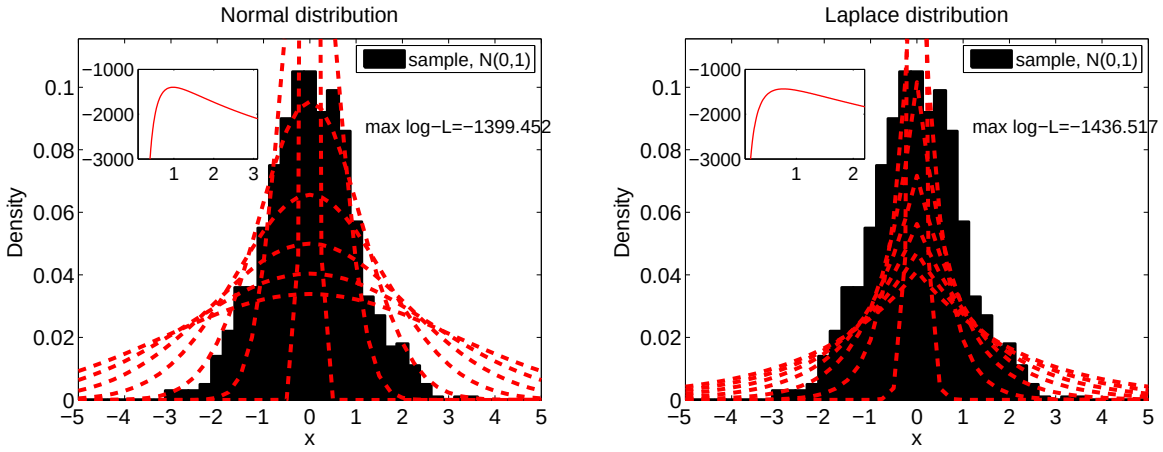

Figure 4: Examples showing the pdfs for various choices of scale parameters σ and b in our two models along with the empirical histogram. The insets show the marginal likelihood as a function of the parameter, and the text on the figure gives the maximum likelihood score.

marginal log-likelihood as a function of the parameter. In both cases there's a clear maximum in the marginal likelihood function, and its location corresponds to the maximum likelihood estimate of the parameter value.

For the Normal distribution, the maximum log-likelihood value is $\mathcal{L}_N = -1399.5$ while the Laplace distribution yields $\mathcal{L}_L = -1436.5$. Because both models have the same number of free parameters, we could simply compare the likelihood scores and observe that the Normal distribution fits the data better. (Can you calculate how *much* better?)

If we prefer a likelihood ratio test, we would first compute the ratio $R = \mathcal{L}_N - \mathcal{L}_L = 37.065$. Then, we would have to decide if this value is sufficiently large that we can trust its positive sign. Going through the process outlined in the first homework, this yields a $p = 0.00$, which means that the sign of R is statistically distinguishable from 0 and we rule in favor of the Normal distribution.

Finally, if we prefer marginalization, we need to numerically integrate each model's likelihood function (*not* the log-likelihood function) over its free parameter. For more complex models, this is typically done using a Markov chain Monte Carlo (MCMC) algorithm. Here, we can do it a quick and dirty way, which is to use a Monte Carlo sampler to estimate the volume under the likelihood function.[11] Marginalizing over σ for the Normal distribution yields a marginal likelihood of $\mathrm{e}^{-1395.5}$.

[11] Integrating likelihood functions is tricky because their value at any particular point in the parameter space is typically too small to represent using standard floating point variables. For instance, the *maximum* likelihood score

Doing the same for the Laplace distribution yields $e^{-1432.6}$, implying that the Normal distribution is favored. (Note that these values are actually *larger* than the maximum likelihood values shown in Figure 2. This is because we integrated over less likely parameterizations.)

The fact that all of our model comparison techniques agree that the Normal distribution is the better model of the data is a good sanity check. If one of the techniques had ruled in the opposite direction, that would be an interesting result—it would indicate that something about the differences in assumptions that these various techniques make is coming into play. In this case, happily, every technique agrees and every technique gets it right.

under the Laplace model is $e^{-1436.517}$, while most computers bottom out at much larger numbers (Matlab can get down to about $e^{-745.133}$). Thus, unless we're smart about it, our computer will round each of these likelihoods to 0, which is not helpful. Using the identity for addition under a logarithm, which is $\ln(x+y) = \ln x + \ln(1+e^{\ln y - \ln x})$, we can do the integration using the easy-to-represent log-likelihoods rather than difficult-to-represent raw likelihoods.

3 Matlab code

3.1 Figure 2

Code for generating the $n = 1000$ iid observations from the standard Normal distribution, for tabulating their histogram and for plotting them against both the Normal and Laplace distributions.

```
% generative model is N(mu,sigma)
[mu sig] = deal(0,1);            % parameter choices
n        = 1000;                 % sample size
x        = (randn(n,1)*sig)+mu;  % uses Matlab's built-in normal deviate generator

figure(1); clf;      % histogram + pdf of generating model
r = (-5:0.25:5);
h = hist(x,r);
g = bar(r,h); hold on;
p = pdf('norm',r,mu,sig);
plot(r,(p./max(p)).*max(h),'k--','LineWidth',3); hold off;
set(g,'BarWidth',1.0,'FaceColor',[1 0 0],'LineWidth',2);
set(gca,'FontSize',16,'YLim',[0 1.2*max(h)],'XLim',[min(r) max(r)]);
set(gca,'XTick',(min(r):1:max(r)))
xlabel('x','FontSize',16);       ylabel('Frequency ','FontSize',16);
title(strcat('n=',num2str(n)),'FontSize',16);
g=legend(strcat('sample, N(',num2str(mu),',',num2str(sig)',')'), ...
        strcat('Normal(',num2str(mu),',',num2str(sig),')'),1);
set(g,'FontSize',14);

figure(4); clf;     % histogram + pdf of maximum-likelihood Laplace model
g = bar(r,h); hold on;
bhat = (1/n).*sum(abs(x));
p    = (1/2*bhat).*exp(-abs(r)./bhat);
plot(r,(p./max(p)).*max(h),'k--','LineWidth',3); hold off;
set(g,'BarWidth',1.0,'FaceColor',[1 0 0],'LineWidth',2);
set(gca,'FontSize',16,'YLim',[0 1.2*max(h)],'XLim',[min(r) max(r)]);
set(gca,'XTick',(min(r):1:max(r)))
xlabel('x','FontSize',16);       ylabel('Frequency ','FontSize',16);
title(strcat('n=',num2str(n)),'FontSize',16);
g=legend(strcat('sample, N(',num2str(mu),',',num2str(sig)',')'), ...
        strcat('Laplace(m=0,b=',num2str(bhat,'%4.2f'),')'),1);
set(g,'FontSize',14);
```

3.2 Figure 3

To construct the likelihood surfaces, we need to evaluate the log-likelihood function at each point on a 2d grid in the parameter space. We then pass this matrix of values to a visualizer that connects the dots. In addition to the surface, this code also notes the location of the maximum-likelihood model.

```
% Normal: compute 2d log-likelihood function over the mu-sigma space
tmu = (-4:0.25:4.0);     % parameter range for mu
tsi = (0.5:0.05:3.6);    % parameter range for sigma
n   = length(x);
L   = -Inf.*ones(length(tmu),length(tsi));
for i=1:length(tmu)
    for j=1:length(tsi)
        [m s] = deal(tmu(i),tsi(j));
        L(i,j) = -(n/2).*log(2*pi*s.^2)-(1./(2*s.^2)).*sum( (x-m).^2 );
    end;
end;

figure(2);  % make the surface plot
g=surf(tsi,tmu,L); hold on;
plot3([sig sig],[mu mu],[min(min(L)) 0.1.*max(max(L))],'ko-','LineWidth',3); hold off;
ylabel('\mu','FontSize',16);     xlabel('\sigma','FontSize',16);
zlabel('log-likelihood','FontSize',16);
set(gca,'FontSize',16,'YTick',(-4:2:4),'XTick',(0:1:4));
set(gca,'ZLim',[min(min(L)) 0.1.*max(max(L))]);
title('Normal distribution','FontSize',16);
colormap('copper')

% Laplace: compute 2d log-likelihood function over the m-b space
tm = (-4:0.25:4.0);      % parameter range for m
tb = (0.5:0.05:3.6);     % parameter range for b
L  = -Inf.*ones(length(tm),length(tb));
for i=1:length(tm)
    for j=1:length(tb)
        [m b] = deal(tm(i),tb(j));
        L(i,j) = -n.*log(2*b)-(1./b).*sum( abs(x-m) );
    end;
end;
```

```
figure(5);  % make the surface plot
g=surf(tb,tm,L); hold on;
plot3([bhat bhat],[mu mu],[min(min(L)) 0.1.*max(max(L))],'ko-','LineWidth',3); hold off;
ylabel('m','FontSize',16);     xlabel('b','FontSize',16);
zlabel('log-likelihood','FontSize',16);
set(gca,'FontSize',16,'YTick',(-4:2:4),'XTick',(0:1:4));
set(gca,'ZLim',[min(min(L)) 0.1.*max(max(L))]);
title('Laplace distribution','FontSize',16);
colormap('copper')
```

3.3 Figure 4

Instead of giving you code that simply produces Figure 4, this code constructs a little animation showing the process. For each model, we scan over its free parameter (σ for the Normal and b for the Laplace), plot it against the histogram and compute its log-likelihood. Effectively, this simulation traces out the marginal log-likelihood function along that free dimension.

```
% Normal distributional model N(0,sigma)
m = 0;                    % fix the mu parameter
r = (-5:0.25:5);          % bin values for the histogram
h = hist(x,r);            % make histogram of the empirical data
c = max(h./sum(h))./max(pdf('norm',r,mu,sig));  % recale the pdf to match histogram
tsi  = (0.1:0.1:3*sig);  % range of values for free parameter
logL = zeros(length(tsi),1);

for i=1:length(tsi)
    s = tsi(i);
    p = pdf('norm',r,m,s);   % pdf of the specified Normal
    figure(3); clf;
    g = bar(r,h./sum(h)); hold on;
    plot(r,p.*c,'k--','LineWidth',3); hold off;
    set(g,'BarWidth',1.0,'FaceColor',[1 0 0],'LineWidth',2);
    set(gca,'FontSize',16,'YLim',[0 1.1.*max(h./sum(h))],'XLim',[min(r) max(r)]);
    set(gca,'XTick',(min(r):1:max(r)))
    xlabel('x','FontSize',16);
    ylabel('Density ','FontSize',16);
    title(strcat('n=',num2str(n)),'FontSize',16);
    g=legend(strcat('sample, N(',num2str(mu,'%3.1f'),',',num2str(sig,'%3.1f'),')'), ...
           strcat('Normal(',num2str(m,'%3.1f'),',',num2str(s,'%3.1f'),')'),1);
    set(g,'FontSize',14);
```

```
    % write the current and maximum log-likelihoods on figure
    logL(i) = -(n/2).*log(2*pi*s.^2)-(1./(2*s.^2)).*sum( (x-m).^2 );
    g=text(1.5,max(h./sum(h))/1.5,strcat('log-L=',num2str(logL(i),'%5.3f')));
    set(g,'FontSize',14);
    g=text(1.5,max(h./sum(h))/1.8,strcat('max log-L=',num2str(max(logL(1:i)),'%5.3f')));
    set(g,'FontSize',14);

    % make the inset
    g=axes('position',[0.2 0.6 0.2 0.2]);
    plot(g,tsi(1:i),logL(1:i),'r-','LineWidth',1);
    set(g,'FontSize',14,'XTick',(0:1:s));
    drawnow;    % flush the graphics buffer to the screen
    pause(0.3); % wait 0.3 seconds
end;

% Laplace distributional model L(0,b)
m    = 0;                    % fix the m parameter
r    = (-5:0.25:5);          % bin values for the histogram
h    = hist(x,r);            % make histogram of the empirical data
p    = (1/2*bhat).*exp(-abs(r)./bhat);
c    = max(h./sum(h))./max(p./sum(p));  % recale the pdf to match histogram
tb   = (0.1:0.1:3*bhat);
logL = zeros(length(tb),1);

for i=1:length(tb)
    b = tb(i);
    figure(6); clf;
    p = (1/2*b).*exp(-abs(r)./b); p = p./sum(p);
    g = bar(r,h./sum(h)); hold on;
    plot(r,p.*c,'k--','LineWidth',3); hold off;
    set(g,'BarWidth',1.0,'FaceColor',[1 0 0],'LineWidth',2);
    set(gca,'FontSize',16,'YLim',[0 1.1.*max(h./sum(h))],'XLim',[min(r) max(r)]);
    set(gca,'XTick',(min(r):1:max(r)))
    xlabel('x','FontSize',16);
    ylabel('Density ','FontSize',16);
    title(strcat('n=',num2str(n)),'FontSize',16);
    g=legend(strcat('sample, N(',num2str(mu),',',num2str(sig)',')'), ...
           strcat('Laplace(\mu=0,b=',num2str(b),')'),1);
    set(g,'FontSize',14);
```

```
    % write the current and maximum log-likelihoods on figure
    logL(i) = -n.*log(2*b)-(1./b).*sum( abs(x-m) );
    g=text(1.5,max(h./sum(h))/1.5,strcat('log-L=',num2str(logL(i),'%5.3f')));
    set(g,'FontSize',14);
    g=text(1.5,max(h./sum(h))/1.8,strcat('max log-L=',num2str(max(logL(1:i)),'%5.3f')));
    set(g,'FontSize',14);

    % make the inset
    g=axes('position',[0.2 0.6 0.2 0.2]);
    plot(g,tb(1:i),logL(1:i),'r-','LineWidth',1);
    set(g,'FontSize',14,'XTick',(0:1:b));
    drawnow;    % flush the graphics buffer to the screen
    pause(0.3); % wait 0.3 seconds
end;
```

3.4 Marginalization via Monte Carlo integration

To compute and compare the marginal likelihoods of the two models, we need to numerically integrate the likelihood functions (*not* the log-likelihood functions—do you see why?) over the free parameter space.

Unfortunately, because we're dealing with very very small numbers, we cannot simply pass the likelihood function directly to a standard numerical integration procedure. If we did, we would almost surely get back the answer that the marginal likelihood is precisely zero (due to round-off errors in representing such tiny numbers). If we wrote our own numerical integration procedure, we could modify it to integrate using only log-likelihoods. The key insight here is that $\ln(x+y) = \ln x + \ln(1+\mathrm{e}^{\ln y - \ln x})$; however, if $\ln y - \ln x$ is sufficiently large, then we'll run into the floating point problem again, so we need to check whether the new value we're adding is sufficiently large that we can ignore the previous accumulated sum.

A quick and dirty way to approximate the integral, without writing our own numerical integrator, is to use Monte Carlo integration. The idea is to sample points on the domain of the function uniformly at random and use an accumulator to add up their contributions to the integral. In the limit of large sample sizes, this technique yields the correct result. This is the technique we'll use here.

```
N  = 1000000;                % a very large number of samples
[Sn Sb] = deal(-Inf,-Inf);   % initial log-likelihoods
for i=1:N

    % Monte Carlo sample for Normal distribution N(0,sigma)
    s = (rand(1)*1000)+0.1;     % uniform on [0.1,1000] range
    L = -(n/2).*log(2*pi*s.^2)-(1./(2*s.^2)).*sum( (x-m).^2 );
    if L>Sn
        Sn = L  + log(1+exp(Sn-L)); % addition using logarithms
    else
        Sn = Sn + log(1+exp(L-Sn)); % addition using logarithms
    end;

    % Monte Carlo sample for Laplace distribution L(0,b)
    b = (rand(1)*1000)+0.1;     % uniform on [0.1,1000] range
    L = -n.*log(2*b)-(1./b).*sum( abs(x-m) );
    if L>Sb
        Sb = L  + log(1+exp(Sb-L)); % addition using logarithms
    else
        Sb = Sb + log(1+exp(L-Sb)); % addition using logarithms
    end;

end;

fprintf('\nfix mu=0, integrate over sigma (via Monte Carlo sampling)\n');
fprintf('(log) marginal likelihood, N(0,s) = %6.4f\n\n',Sn);
fprintf('\nfix u=0, integrate over b     (via Monte Carlo sampling)\n');
fprintf('(log) marginal likelihood, L(0,b) = %6.4f\n\n',Sb);
```