Dublin R Workshop on Bayesian Data Analysis

Mick Cooney
mickcooney@gmail.com

July 29, 2013

1. Conditional Probability

Suppose that in the general population, the probability of having a specific rare disease (the Dreaded Lurgy) is one in a thousand. We denote the true presence or absence of the disease as the value of a parameter, θ, that can have the value 1 if disease is present, or the value 0 if the disease is absent. The base rate of the disease is therefore denoted $p(\theta = 1) = 0.001$. This is our prior belief that a person selected at random has the disease.

Suppose that there is a test for the disease that has a 99% hit rate, which means that if a person has the disease, then the test result is positive 99% of the time. We denote a positive test result as $D = 1$, and a negative test result as $D = 0$. The observed test result is a bit of data that we will use to modify our belief about the value of the underlying disease parameter. The hit rate is expressed as $p(D = 1 \,|\, \theta = 1) = 0.99$.

The test also has a false alarm rate of 5%. This means that 5% of the time when the disease is not present, the test falsely indicates that the disease is present. Wedenote the false alarm rate as $p(D = 1 \,|\, \theta = 0) = 0.05$.

However, what we need to know is $p(\theta = 1 \,|\, D = 1)$, i.e. the probability that the patient has the disease given a positive test result.

We can calculate the above conditional probability given Bayes' Rule and a bit of arithmetic and algebra, and, somewhat counter-intuitively, it says the probability is slightly below 2%.

We will try to estimate this probability using simulation.

Exercise 1.1 The supplied function `generate.disease.test.data()` generates sample data based on the above numbers. Look at the function definition to see how it is used, and generate some sample data, then use this data to estimate the conditional probability. with a reasonable number of datapoints (say 1,000,000) you should get an estimate close to the analytic answer.

Exercise 1.2 Why is this conditional probability so low? Experiment with the dependency of this probability on the three input probabilities.

(Any plotting system will work, though I will use ggplot2 for mine)

Exercise 1.3 How are the probabilities changed if two independant tests are tried. The provided function `generate.disease.twotest.data()` generates random data for this. Using this function, calculate the conditional probabilities of having the disease, given the various combinations of test results.

2. Analytical Approach

Bayesian reasoning is as old as the concept of probabilities, but has only recently started to receive a lot of attention. One likely reason for this is that, apart from a few special cases, it is mot possible to perform the calculations analytically.

In our application we have a prior distribution for our beliefs, $p(\theta)$, and a likelihood for the data, $p(D|\theta)$, and through use of the Chain Rule, we get the posterior distribution, $p(\theta|D)$,

$$p(\theta|D) = \int d\theta \, p(D|\theta) \, p(\theta).$$

In those special cases, the likelihood function has a prior and posterior distribution with the same functional form, i.e. the prior and the posterior are two members of the same 'family' of functions.

For the rest of this workshop we are going to deal with estimating the fairness of a coin, based on the result of multiple coin tosses. We define 'success' as the toss coming up Heads, and denote this probability as θ. Thus,

$$P(y = 1|\theta) = \theta \quad \text{and} \quad P(y = 0|\theta) = 1 - \theta.$$

We can combine the above two into a single expression:

$$P(y|\theta) = \theta^y (1 - \theta)^{(1-y)}$$

.

Now we consider the data y to be fixed, and consider the above as a function of θ. With this approach we call the above equation *the likelihood function of* θ.

The Bernoulli function has a conjugate prior: the *Beta* distribution, $B(a, b)$. R supports the beta distribution natively, via the standard grouping of functions for probability distributions: `rbeta()`, `dbeta()`, `pbeta()`, `qbeta()`.

Exercise 2.1 Plot the density distribution for the Beta distribution using various combinations of parameters a and b.

Exercise 2.2 Load the data in `cointoss10`, and use it and the beta distribution to investigate the effect of the prior on the posterior distribution.

Exercise 2.3 Load the data in `cointoss1000`, and use it and the same priors used in the previous exercise to investigate the effect of the prior on the posterior distribution.

Exercise 2.4 Compare the posterior distributions for the same priors using the two datasets. How do they compare, and why do you think this is?

3. Numerical Solutions using a Discrete Grid

For many applications, the use of simple conjugate priors is not appropriate, and we need to deal with the posterior integral calculation itself. Since analytical solutions do not exist, we use numerical techniques to approximate the integral.

The supplied functions `calculate.data.probability()` and `calculate.posterior.probability()` perform these calculations. The major benefit of this approach is that we can now use arbitrary priors.

Exercise 3.1 Using the `cointoss10` data, and a $B(1, 1)$ prior, use the grid approximation to the integral to calculate the posterior density. Compare the output of this to the analytical solution

Exercise 3.2 Repeat the previous exercise of comparing the influence of priors on the posterior density for both the 10 coin toss and 1,000 coin toss data. Match this output to the analytics solutions you derived earlier.

Exercise 3.3 Suppose we think the coin has a 3/1 bias, but we do not know for which side. Create a prior that represents this and investigate the posterior for both sets of data.

Exercise 3.4 Estimate the posterior density for the bias in the coin assuming you have an equally weighted prior belief of the coin being fair, or biased 3/1 for either side.

Exercise 3.5 Investigate the influence of the size of the dataset on the posterior for arbitrary priors.

4. Introducing Monte Carlo Sampling

The numerical integration approach works fine when the dimensionality of the problem is low, but quickly becomes infeasible computationally once we add start adding more parameters to the problem.

In these cases, we instead look for ways in which we can use simulation to approximate the integral, only calculating the integral at places on the grid where there is a significant contribution to the solution.

For these sampling techniques, we first look at the Metropolis algorithm:

Initialise $\mathbf{x}_0$;
Set $i = 1$;
while $i <$ *SampleSize* **do**
  Choose new $\mathbf{x}_i$;
  if $p(\boldsymbol{x}_i) > p(\boldsymbol{x}_{i-1})$ **then**
    Accept $\mathbf{x}_i$;
  else
    Accept $\mathbf{x}_i$ with probability $\frac{p(\mathbf{x}_i)}{p(\mathbf{x}_{i-1})}$;
  end
  Increment i;
end

Once again, we will work with a very simple example, and build up from there.

Suppose we have a politician who is seeking election. He wants to visit remote islands, spending time on each island in proportion with their population. He does not know these population figures in advance, but he knows that the locals do. To make things simple, we will assume the islands are in a long chain, so you can only move to an adjacent island.

If he follows the Metropolis algorithm, using the population ratios as his probabilities, and hops from island to island, in the long run he will visit each island in rough proportion to their population.

Exercise 4.1 Using the supplied function `do.metropolis.island.sampling()`, generate a 1000 datapoint sample for the politician based on a four island chain where the islands have all equal population.

Exercise 4.2 Think about ways in which you might verify that the sample is a reasonable sample from the target distribution.

Exercise 4.3 Suppose we have a seven-island chain, where each island has population proportional to its number. Sample from this distribution using our code. Check that the sample is reasonable.

Exercise 4.4 Re-run the above code with the `proposal.prob` parameter set to 0.7. With this value, the algorithm is more likely to propose moving to islands further up the chain. Is there any difference in the samples from previously?

Exercise 4.5 Now set the parameter to 0.3. Does the sample output change qualitatively?

Exercise 4.6 Think about what the consequences of the choice of this parameter value.

Exercise 4.7 Redo all of the above exercises, but generating a dataset of size 10,000 datapoints.

Investigate what implications exist for sampling larger datasets.

5. Introducing Hierarchical Models

We can now extend this idea by using a hierarchical model. Previously, we were considering the coin by itself, and the goal was to estimate the parameters of the model. We could extend that model by using multiple coins, but our initial intuition would be to treat multiple coins as being independent of one another.

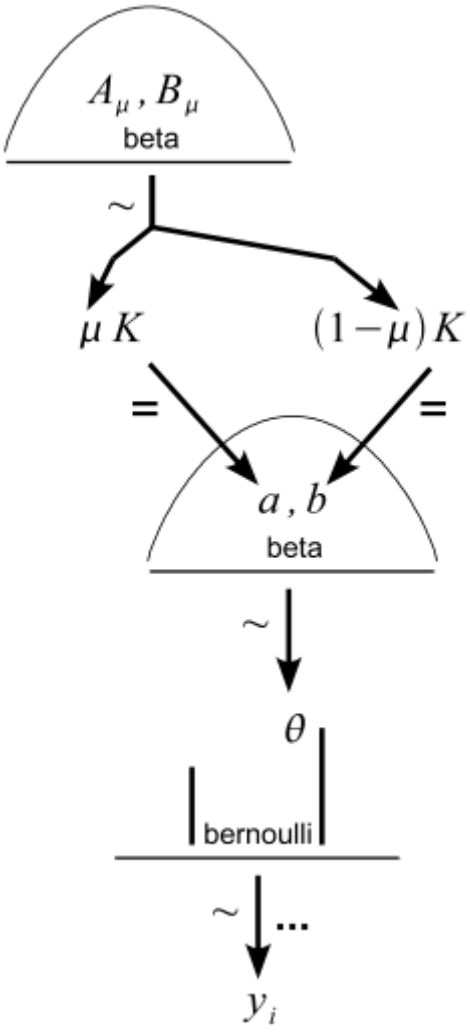

Figure 1: Graphical Representation of the Single Mint, Single Coin Hierarchical model

However, what if we were instead to also consider the fact that a coin is minted, and the bias of the coin is probably influenced by the manufacturing processes and quality of its origin. In this case, we could incorporate prior beliefs about the mint as well the coin, and then use our data to update those beliefs.

To start with, we will work from a single coin coming from a single mint. Each coin is produced with a bias of θ, but these are also random variables drawn from another Beta distribution with mean μ. This means that outcomes of the coin tosses, y, are the result of a hierarchical random process. At the top, we have a distribution for μ, and we call this the *hyperdistribution*. Accordingly, μ is a *hyperparameter*.

We then use the values of the hyperparameters as inputs to the distribution for *theta*. In turn, with the values of θ, we then determine values for the data, y. A graphical representation of this model is shown in Figure 1.

So, the hyperparameter μ is used to determine values A_μ, and B_μ for the hyper-beta-distribution. To do this, we also need a value K, which for the moment we will assume is a constant, and is a representation of how closely the value of θ depends on μ. For high values of K, θ will cluster tightly around μ.

Expressing this formally, we have

$$p(y|\theta) = \theta^y(1-\theta)^{(1-y)},$$

just like before, but we now also have θ as a random variate,

$$p(\theta|\mu) = \text{beta}(\theta|\mu K, (1-\mu)K)$$

with μ itself being drawn from a hyperdistribution (in this case, the beta distribution):

$$\mu \backsim \text{beta}(A_\mu, B_\mu).$$

Note that this model will required values for A_μ and B_μ, and these will reflect our prior beliefs about the mint. For this, we will probably just use low constant values, representing weak prior beliefs on the mint itself.

So, the above seems like a very interesting approach, but how do we go about implementing the approach? It is unlikely that an analytic solution would be either tractable or practical, but like the drunk looking for his keys, this is where the light is.

So, we know from Bayes' Rule that

$$p(\theta, \mu|y) = \frac{p(y|\theta, \mu)p(\theta, \mu)}{p(y)}.$$

From our model, we can see that, conditional on θ, the likelihood model does not depend on μ, and so we have

$$p(y|\theta, \mu) = p(y|\theta).$$

Also, from another application of Bayes' Rule,

$$p(\theta|\mu) = \frac{p(\theta, \mu)}{p(\mu)} \implies p(\theta, \mu) = p(\theta|\mu)p(\mu)$$

,
resulting in the following expression for the posterior distribution

$$p(\mu, \theta|y) = p(y|\theta)p(\theta|\mu)p(\mu)/p(y)$$

So, if we create a grid for both θ and μ, we can approximate the posterior from numerical integration. We take our models for $p(y|\theta)$, $p(\theta|\mu)$ and $p(\mu)$, and create our posterior by integration.

Exercise 5.1 Use a very weak prior for μ, but a value of 5 for K, and generate the posterior density for this model using the `cointoss10` data. You can use the supplied function `calculate.hierarchical.-posterior()` to do this. How would you visualise this output in a meaningful way?

Exercise 5.2 How does the data affect the posterior distribution for μ and θ?

Exercise 5.3 Repeat the above exercise but with a K value of 100. Do you notice a difference in the posterior distributions of μ and θ?

Exercise 5.4 Repeat the above for $K = 1,000$. How are the posterior densities affected now?

Exercise 5.5 Repeat all of the above using the `cointoss1000` data.

Exercise 5.6 What impact does the size of the dataset have on the various posterior densities?

6. Introducing Gibbs Sampling for Hierarchical Models

It should be quickly apparent that using the discrete grid approach to approximating these posterior densities will not be feasible computationally once we start adding more than two or three parameters over anything more than 100 to 1000 discrete steps in each dimension. The grids become very large very quickly, and even the incredible improvements in computational power have not been able to keep up.

Instead, we will use a technique known as Gibbs sampling to sample from the posterior distribution. To do this, we use JAGS (Just Another Gibbs Sampler), as this has taken overs from the original tool, BUGS (Bayesian inference Using Gibbs Sampling). To interface with JAGS, there is an R package available, `rjags`.

There are a few steps involved in setting up the sampler, the first of which involves specifying the model used by JAGS. This is usually done by writing it into a separate file. This has already been done, and the model definition is available in `singlemint_singlecoin.jag`.

The setup of the sampler in R is fairly straightforward, and the sample code is shown in the file `sample_rjags_template.R`. Use this file as a template.

Exercise 6.1 What inferences can be made on μ from this model? Think about why this is the case.

Exercise 6.2 Repeat all the exercises from the equivalent grid approximation of this model, this time using `rjags` code to produce your distributions.

Exercise 6.3 Compare the prior and posterior distributions for θ.

Exercise 6.4 The above exercises involve changing some of the prior values. Could we redo the model to make it more efficient to change these priors?

Exercise 6.5 Discuss the validity of treating this scenario as a hierarchical model. What does μ represent in this case?

We now have all the tools we need to build hierarchical models, so let us extend the above scenario in more interesting ways. What if we have multiple coins minted from the same mint? What types of inferential models can be built from this set up.

The JAGS model for a single mint, multiple coin model is supplied in the file `singlemint_multiplecoin.jag`. Open the file and make sure you understand the model. Data for multiple coin tosses with two different coins is given in the file `singlemint_twocoin.rds`. This data is a two-row column, with the first row being the label for the coin, and the second row being the result of the coin toss.

Exercise 6.6 Create a prior and posterior sample for the above scenario. What kind of methods can we use to visualise this data? Note that these hierarchical models quickly prove problematic in terms of visualisation, so this is a real issue.

Exercise 6.7 Investigate how different values for K change the inferences. Check the differences in how the data influences the posterior distribution for any given prior.

Exercise 6.8 Think about how you would extend this model to tosses of three coins, and then five coins? How feasible is it to analyse data from the tosses of twenty coins?

7. Expanding Hierarchical Models

The true power of the Bayesian approach is that you can encapsulate any and all uncertainty in your

modelling via your priors. Previously, we did have one parameter that we were 'guessing' at the value of, K, which controls the strength of the dependency of the coin mint bias μ on the bias of any individual coin, θ. For higher values of K, the values of θ are more closely clustered around the μ value for the mint.

To illustrate the influence of this value, we were running our sampler with different values of K. However, in reality, we give this value a prior distribution and allow the data to perform some inference on this value as well. The distribution we use for this one is the *Gamma distribution*, which is a common distribution for values $x \geq 0$.

This distribution is closely related to the *Gamma function*, $\Gamma(s) = \int_0^\infty dt\, t^{s-1}e^{-t}$, a generalisation for factorials.

Gamma distribution has two parameters controlling it, the *rate* and the *shape*, and we set both these values so that our prior is broad across the possible values for κ.

Exercise 7.1 The JAGS file for the fully Bayesian model is in `singlemint_full.jag`. Using the model specified in this file with the data contained in `singlemint_twocoin.rds` to sample from the prior and posterior distributions.

Exercise 7.2 How does using a fully Bayesian model change the inferences we make on the parameters of the model based on the data?

Exercise 7.3 Create a new dataset for five coins with different counts of coin tosses for each coin. Using different values of θ for each coin, but have them be relatively similar.

Exercise 7.4 Run the fully Bayesian model on your data. Make inferences on the posterior distribution. How do these inferences match up with the knowledge you already have from generating the data.

Exercise 7.5 Create another dataset for the five coins. The dataset should be similar to the previous one, but this time use very different values for θ.

Exercise 7.6 Run the model with the new data. How do your inferences change between the two datasets, and does this make sense in light of how the data was generated?

Exercise 7.7 Use the supplied function The accompanying code file contain the function `generate.hierarchical.coin.data()`. This function will generate coin toss data for coins generated from a mint with μ and κ, and for a varying number of coins and total coin tosses. Use this function to generate coin toss data for for 5 coins and 50 tosses and then for 50 coins and 5 tosses. The code has been vectorised, so make sure you understand how it works.

Exercise 7.8 Use the full Bayesian model with both datasets to generate the posterior distributions for the various parameters associated with both datasets. Examine the posterior distributions to decide which dataset is better for making inferences on the mean value of the mint μ. Does your answer make sense? What conclusions can we draw about the tradeoff between the number of coins versus the number tosses per coin in this case? How do you think this can be generalised?

8. Further Work

In what ways might our model be further expanded? What if we had multiple mints, with coins coming from different mints, how would we change our model specification to incorporate this into the hierarchy?

We could start with the assumption that all mints are independent of one another, then possibly adding in some kind of hierarchy for the mints themselves.

Note that despite getting more complex, we are still able to estimate the values for the parameters simultaneously.