

Dublin R Workshop on Bayesian Data Analysis

Mick Cooney
mickcooney@gmail.com

February 19, 2014

1. Introducing the Data and the Analysis

In order to learn and explore the use of Bayesian Data Analysis, we need some kind of problem to solve, and data with which to solve it. Before we begin, it is worth looking at the problem we are trying to solve, exploring the data for its structure and layout, and just getting a ‘feel’ for the issues we may have to face.

Our problem is a little contrived, and the data has been generated randomly, but it should be relatively straightforward to generalise this for other problems.

Our dataset is a set of outcomes of cointosses, where the coins have been minted from three separate mints, and we want to get an estimate of the quality of the output from the three mints.

For the purposes of estimating and inferring on mint-quality, we are looking for coins that are as fair as possible i.e. have a θ as close to 0.50 as possible, and can do so as consistently as possible, i.e. the variance of the θ values for the coins is as low as possible.

So, to start, we should first load the data, and then start to explore it.

Exercise 1.1 Load the data into the workspace using the code given in `setup_data.R`.

Exercise 1.2 Using whatever summary methods you like, investigate the mean and variance of the μ for each mint.

Exercise 1.3 Use density estimation to explore the distribution of θ s for each mint.

Hint: The code in `explore_data.R` may be of use to get you started.

2. Using the Hierarchical Model in JAGS

We build our first JAGS model based off the data from a single mint. Our hierarchical model is represented in Figure 1.

One issue with running JAGS models is that it can take time. To help with this issue, two ‘reduced’ versions of the data are produced, and this simple task raises an immediate question of how to do this.

Exercise 2.1 Investigate the setup code for any sampling issues that may arise.

Exercise 2.2 Using a reduced data set, run the hierarchical code for each mint. Compare the results.

Exercise 2.3 Plot the μ and κ for each mint in the same plot.

Exercise 2.4 Rerun the code for the other reduced dataset, and compare with previous results.

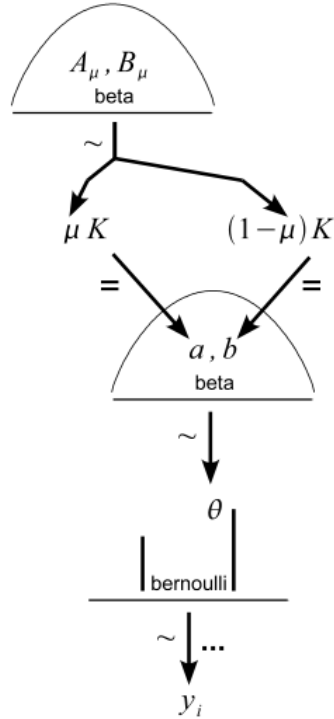


Figure 1: Graphical Representation of the Single Mint, Single Coin Hierarchical model

Exercise 2.5 Run the code with the full dataset, again comparing with the two previous outputs.

Since performance is always an issue, we can investigate the speedup of switching from a Bernoulli distribution at the lowest level, to a Binomial one. This involves a little preprocessing on the data, but nothing too difficult.

Exercise 2.6 Perform the above analysis, but using the binomial distribution at the lowest level. Use `Sys.time()` to time to the two runs.

Hint: The code in `run_bern_singlemint.R` and `run_beta_singlemint.R` may be of help here.

3. Expanding the Model to Compare Mints

While the methods employed in the previous section work fine, a more ‘Bayesian’ approach would be to incorporate the mint data into the model, rather than running a separate run for each mint. As you imagine, this is especially true if there are a large number of separate entities at the highest hierarchy of the model.

Incorporating the mint data for each coin is relatively simple — we just need to index by μ and κ for each mint, as shown in Figure 2.

Exercise 3.1 Using the reduced data, run the full multiple coin, multiple mint hierarchical model, and investigate the output.

Exercise 3.2 Is there any difference in the posterior distributions from this approach to the single mint model?

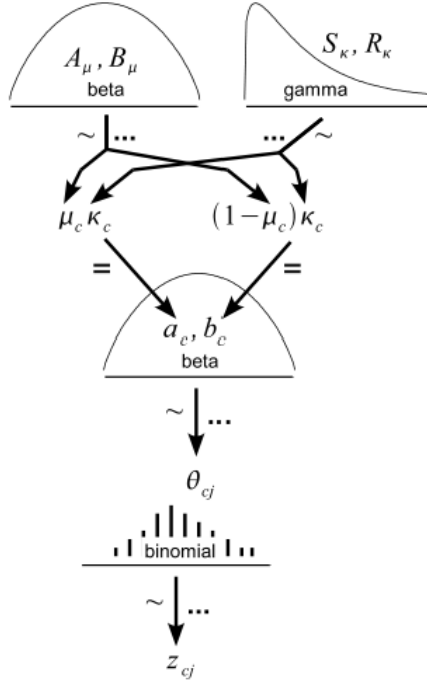


Figure 2: Graphical Representation of the Multiple Mint, Multiple Coin Hierarchical model

Exercise 3.3 How do the above posterior distributions compare to the simple density estimation distributions we originally produced?

Hint: The code in `run_multiplemint.R` may be of help here.

4. Exploring the Effect of the Data

We are going to investigate the effect that the composition of the data has on inference. This can be important if we have a constrained amount of data collection at the lowest level, but have a certain amount of discretion at the higher levels.

So, suppose we have a relatively fixed amount of total coin tosses we can record, but we can choose whether to use more coins per mint, or more tosses per coin. How does this affect the output of our analysis?

Exercise 4.1 Inspect the code in the function `generate.cointoss.data()` and make sure you understand how it works.

Exercise 4.2 Use the function to generate a dataset with a relatively high number of coins per mint, and low number of tosses per coin.

Exercise 4.3 Use the function to generate a dataset with a relatively high number of coins per mint, and low number of tosses per coin.

Exercise 4.4 Use all of the above to investigate the above compositional effects of the data on the outputs of our model?

5. Further Work and Open Questions

Exercise 5.1 Is there ways in which we can improve the data generation to help us understand how this approach handles differences?

Exercise 5.2 How do we incorporate errors in data collection to our model?

Exercise 5.3 What kind of effects would these errors have on our output and our conclusions?

Exercise 5.4 How can we extend this model at the higher levels?