15.097: Probabilistic Modeling and Bayesian Analysis

Ben Letham and Cynthia Rudin

Credits: *Bayesian Data Analysis* by Gelman, Carlin, Stern, and Rubin

1 Introduction and Notation

Up to this point, most of the machine learning tools we discussed (SVM, Boosting, Decision Trees,...) do not make any assumption about how the data were generated. For the remainder of the course, we will make distributional assumptions, that the underlying distribution is one of a set. Given data, our goal then becomes to determine *which* probability distribution generated the data.

We are given m data points $y_1, \ldots, y_m$, each of arbitrary dimension. Let $y = \{y_1, \ldots, y_m\}$ denote the full set of data. Thus y is a random variable, whose probability density function would in probability theory typically be denoted as $f_y(\{y_1, \ldots, y_m\})$. We will use a standard (in Bayesian analysis) shorthand notation for probability density functions, and denote the probability density function of the random variable y as simply $p(y)$.

We will assume that the data were generated from a probability distribution that is described by some parameters θ (not necessarily scalar). We treat θ as a random variable. We will use the shorthand notation $p(y|\theta)$ to represent the *family* of conditional density functions over y, parameterized by the random variable θ. We call this family $p(y|\theta)$ a *likelihood function* or likelihood model for the data y, as it tells us how likely the data y are given the model specified by any value of θ.

We specify a *prior* distribution over θ, denoted $p(\theta)$. This distribution represents any knowledge we have about how the data are generated prior to

observing them.

Our end goal is the conditional density function over θ, given the observed data, which we denote as $p(\theta|y)$. We call this the *posterior* distribution, and it informs us which parameters are likely given the observed data.

We, the modeler, specify the likelihood function (as a function of y and θ) and the prior (we completely specify this) using our knowledge of the system at hand. We then use these quantities, together with the data, to compute the posterior. The likelihood, prior, and posterior are all related via Bayes' rule:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} = \frac{p(y|\theta)p(\theta)}{\int p(y|\theta')p(\theta')d\theta'}, \tag{1}$$

where the second step uses the law of total probability. Unfortunately the integral in the denominator, called the *partition function*, is often intractable. This is what makes Bayesian analysis difficult, and the remainder of the notes will essentially be methods for avoiding that integral.

Coin Flip Example Part 1. Suppose we have been given data from a series of m coin flips, and we are not sure if the coin is fair or not. We might assume that the data were generated by a sequence of independent draws from a Bernoulli distribution, parameterized by θ, which is the probability of flipping Heads.

But what's the value of θ? That is, which Bernoulli distribution generated these data?

We could estimate θ as the proportion of the flips that are Heads. We will see shortly that this is a principled Bayesian approach. Let $y_i = 1$ if flip i was Heads, and $y_i = 0$ otherwise. Let $m_H = \sum_{i=1}^{m} y_i$ be the number of heads in m tosses. Then the likelihood model is

$$p(y|\theta) = \theta^{m_H}(1-\theta)^{m-m_H}. \tag{2}$$

1.1 A note on the Bayesian approach

The problem formulation we have just described has historically been a source of much controversy in statistics. There are generally two subfields of statis-

tics: frequentist (or classical) statistics, and Bayesian statistics. Although many of the techniques overlap, there is a fundamental difference in philosophy. In the frequentist approach, θ is an unknown, but *deterministic* quantity. The goal in frequentist statistics might then be to determine the range of values for θ that is supported by the data (called a confidence interval). When θ is viewed as a deterministic quantity, it is nonsensical to talk about its probability distribution. One of the greatest statisticians of our time, Fisher, wrote that Bayesian statistics "is founded upon an error, and must be wholly rejected." Another of the great frequentists, Neyman, wrote that, "the whole theory would look nicer if it were built from the start without reference to Bayesianism and priors." Nevertheless, recent advances in theory and particularly in computation have shown Bayesian statistics to be very useful for many applications. Machine learning is concerned mainly with *prediction* ability. A lot of the methods we discussed do not worry about exactly what the underlying distribution is - as long as we can predict, we are happy, regardless of whether we even have a meaningful estimate for $p(y|\theta)$.

2 Point estimates

Rather than estimate the entire distribution $p(\theta|y)$, sometimes it is sufficient to find a single 'good' value for θ. We call this a *point estimate*. For the sake of completeness, we will briefly discuss two widely used point estimates, the *maximum likelihood* (ML) estimate and the *maximum a posteriori* (MAP) estimate.

2.1 Maximum likelihood estimation

The ML estimate for θ is denoted $\hat{\theta}_{\text{ML}}$ and is the value for θ under which the data are most likely:

$$\hat{\theta}_{\text{ML}} \in \arg\max_{\theta} p(y|\theta). \tag{3}$$

As a practical matter, when computing the maximum likelihood estimate it is often easier to work with the *log-likelihood*, $\ell(\theta) := \log p(y|\theta)$. Because the logarithm is monotonic, it does not affect the argmax:

$$\hat{\theta}_{\text{ML}} \in \arg\max_{\theta} \ell(\theta). \tag{4}$$

The ML estimator is very popular and has been used all the way back to Laplace. It has a number of nice properties, one of which is that it is a *consistent* estimator. Let's explain what that means.

Definition 1 (Convergence in Probability)**.** A sequence of random variables $X_1, X_2, \ldots$ is said to *converge in probability* to a random variable X if, $\forall \epsilon > 0$,

$$\lim_{n\to\infty} \mathbb{P}\left(|X_n - X| \geq \epsilon\right) = 0.$$

We denote this convergence as $X_n \xrightarrow{P} X$.

Definition 2 (Consistent estimators)**.** Suppose the data $y_1, \ldots, y_m$ were generated by a probability distribution $p(y|\theta_0)$. An estimator $\hat{\theta}$ is *consistent* if it converges in probability to the true value: $\hat{\theta} \xrightarrow{P} \theta_0$ as $m \to \infty$.

We said that maximum likelihood is consistent. This means that if the distribution that generated the data belongs to the family defined by our likelihood model, maximum likelihood is guaranteed to find the correct distribution, as m goes to infinity. In proving consistency, we do not get finite sample guarantees like with statistical learning theory; and data are always finite.

Coin Flip Example Part 2. Returning to the coin flip example, equation (2), the log-likelihood is

$$\ell(\theta) = m_H \log\theta + (m - m_H)\log(1-\theta).$$

We can maximize this by differentiating and setting to zero, and doing a few lines of algebra:

$$\begin{aligned} 0 = \frac{d\ell(\theta)}{d\theta}\Big|_{\hat{\theta}} &= \frac{m_H}{\theta} - \frac{m - m_H}{1-\theta}\Big|_{\hat{\theta}} \\ m_H(1 - \hat{\theta}_{\mathrm{ML}}) &= (m - m_H)\hat{\theta}_{\mathrm{ML}} \\ m_H - \hat{\theta}_{\mathrm{ML}} m_H &= m\hat{\theta}_{\mathrm{ML}} - \hat{\theta}_{\mathrm{ML}} m_H \\ \hat{\theta}_{\mathrm{ML}} &= \frac{m_H}{m}. \end{aligned} \tag{5}$$

(It turns out not to be difficult to verify that this is indeed a maximum). In this case, the maximum likelihood estimate is exactly what we intuitively thought we should do: estimate θ as the observed proportion of Heads.

2.2 Maximum a posteriori (MAP) estimation

The MAP estimate is a pointwise estimate with a Bayesian flavor. Rather than finding θ that maximizes the likelihood function, $p(y|\theta)$, we find θ that maximizes the posterior, $p(\theta|y)$. The distinction is between the θ under which the data are most likely, and the most likely θ given the data.

We don't have to worry about evaluating the partition function $\int p(y|\theta')p(\theta')d\theta'$ because it is constant with respect to θ. Again it is generally more convenient to work with the logarithm.

$$\begin{aligned}\hat{\theta}_{\text{MAP}} \in \arg\max_{\theta} p(\theta|y) &= \arg\max_{\theta} \frac{p(y|\theta)p(\theta)}{\int p(y|\theta')p(\theta')d\theta'} \\ &= \arg\max_{\theta} p(y|\theta)p(\theta) = \arg\max_{\theta} \left(\log p(y|\theta) + \log p(\theta)\right). \end{aligned} \tag{6}$$

When the prior is uniform, the MAP estimate is identical to the ML estimate because the $\log p(\theta)$ is constant.

One might ask what would be a bad choice for a prior. We will see later that reasonable choices of the prior are those that do not assign zero probability to the true value of θ. If we have such a prior, the MAP estimate is consistent, which we will discuss in more detail later. Some other properties of the MAP estimate are illustrated in the next example.

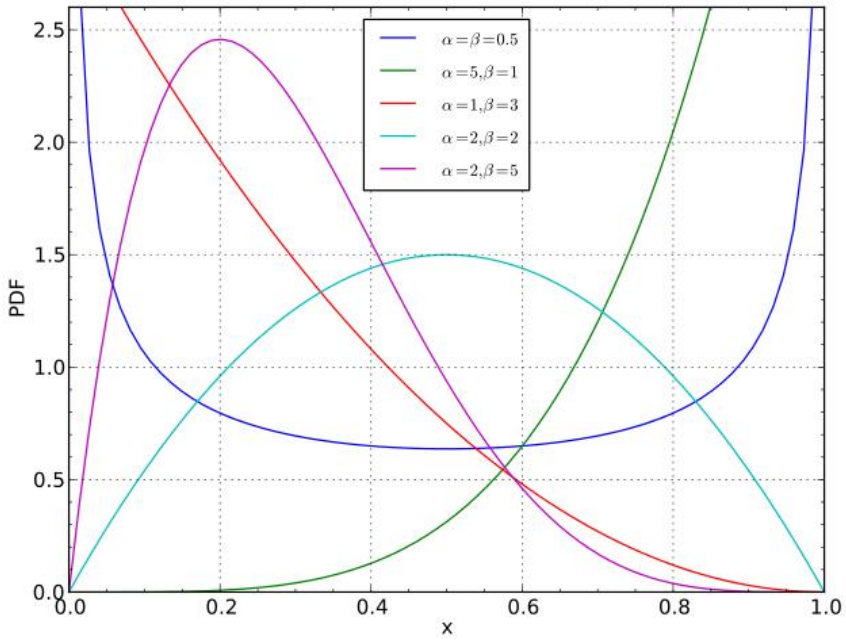
Coin Flip Example Part 3. We again return to the coin flip example. Suppose we model θ using a Beta prior (we will see later why this is a good idea): $\theta \sim \text{Beta}(\alpha, \beta)$. The Beta distribution is:

$$\text{Beta}(\theta; \alpha, \beta) = \frac{1}{B(\alpha, \beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}, \tag{7}$$

where $B(\alpha, \beta)$ is the beta function, and is constant with respect to θ:

$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt. \tag{8}$$

The quantities α and β are parameters of the prior which we are free to set according to our prior belief about θ. By varying α and β, we can encode a wide range of possible beliefs, as is shown in this figure taken from the Wikipedia article on the Beta distribution:



The MAP estimate for θ we can get from the formula for computing $\hat{\theta}_{\text{MAP}}$ in (6), plugging in the formula for the likelihood we found in part 2, and the definition of the Beta distribution for the prior (7):

$$\begin{aligned}\hat{\theta}_{\text{MAP}} &\in \arg\max_{\theta} \left(\log p(y|\theta) + \log p(\theta)\right) \\ &= \arg\max_{\theta} \left(m_H \log\theta + (m - m_H)\log(1-\theta)\right. \\ &\quad \left. +(\alpha-1)\log\theta + (\beta-1)\log(1-\theta) - \log B(\alpha,\beta)\right).\end{aligned}$$

Differentiating and setting to zero at $\hat{\theta}_{\text{MAP}}$,

$$\begin{aligned}\frac{m_H}{\hat{\theta}_{\text{MAP}}} - \frac{m - m_H}{1-\hat{\theta}_{\text{MAP}}} + \frac{\alpha-1}{\hat{\theta}_{\text{MAP}}} - \frac{\beta-1}{1-\hat{\theta}_{\text{MAP}}} &= 0 \\ \hat{\theta}_{\text{MAP}} &= \frac{m_H + \alpha - 1}{m + \beta - 1 + \alpha - 1}. \qquad (9)\end{aligned}$$

This is a very nice result illustrating some interesting properties of the MAP estimate. In particular, comparing the MAP estimate in (9) to the ML estimate in (5) which was

$$\hat{\theta}_{\text{ML}} = \frac{m_H}{m},$$

we see that the MAP estimate is equivalent to the ML estimate of a data set with $\alpha - 1$ additional Heads and $\beta - 1$ additional Tails. When we specify, for example, a prior of $\alpha = 7$ and $\beta = 3$, it is literally as if we had begun the

coin tossing experiment with 6 Heads and 2 Tails on the record. If we truly believed before we started flipping coins that the probability of Heads was around 6/8, then this is a good idea. This can be very useful in reducing the variance of the estimate for small samples.

For example, suppose the data contain only one coin flip, a Heads. The ML estimate will be $\hat{\theta}_{ML} = 1$, which predicts that we will never flip tails! However, we, the modeler, suspect that the coin is probably fair, and can assign $\alpha = \beta = 3$ (or some other number with $\alpha = \beta$), and we get $\hat{\theta}_{MAP} = 3/5$.

Question How would you set α and β for the coin toss under a strong prior belief vs. a weak prior belief that the probability of Heads was 1/8?

For large samples it is easy to see for the coin flipping that the effect of the prior goes to zero:

$$\lim_{m\to\infty} \hat{\theta}_{\text{MAP}} = \lim_{m\to\infty} \hat{\theta}_{\text{ML}} = \theta_{\text{true}}. \qquad \text{Why?}$$

Recall what know about regularization in machine learning - that data plus knowledge implies generalization. The prior is the "knowledge" part. One could interpret the MAP estimate as a regularized version of the ML estimate, or a version with "shrinkage."

Example 1. (Rare Events) The MAP estimate is particularly useful when dealing with rare events. Suppose we are trying to estimate the probability that a given credit card transaction is fraudulent. Perhaps we monitor transactions for a day, and there are no fraudulent transactions. The ML estimate tells us that the probability of credit card fraud is zero. The MAP estimate would allow us to incorporate our prior knowledge that there is some probability of fraud, we just haven't seen it yet.

2.3 Point estimation and probabilistic linear regression

We will now apply point estimation to a slightly more interesting problem, linear regression, and on the way will discover some very elegant connections between some of the machine learning algorithms we have already seen and our new probabilistic approach. Suppose now that we have m data points $(x_1, y_1), \ldots, (x_m, y_m)$, where $x_i \in \mathbb{R}^d$ are the independent variables and $y_i \in \mathbb{R}$

are the dependent variables. We will use a likelihood model under which y_i depends linearly on x_i, and the y_i's are all independent. Specifically, we model

$$y_i \sim \theta^T x_i + \epsilon,$$

where $\theta \in \mathbb{R}^d$ are the parameters we are interested in, and ϵ represents noise. We will assume that the noise is distributed normally with zero mean and some known variance σ^2: $\epsilon \sim \mathcal{N}(0, \sigma^2)$. This, together with our assumption of independence, allows us to express the likelihood function for the observations $y = \{y_1, \ldots, y_m\}$:

$$\begin{aligned} p(y|x,\theta) &= \prod_{i=1}^{m} p(y_i|x_i,\theta) = \prod_{i=1}^{m} \mathcal{N}(y_i; \theta^T x_i, \sigma^2) \\ &= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y_i - \theta^T x_i)^2\right), \end{aligned}$$

where the first line follows from independence. As usual, it is more convenient to work with the log-likelihood:

$$\begin{aligned} \ell(\theta) &= \log\left(\prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y_i - \theta^T x_i)^2\right)\right) \\ &= -\frac{m}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{m}(y_i - \theta^T x_i)^2. \end{aligned}$$

The ML estimate is

$$\hat{\theta}_{\mathrm{ML}} \in \arg\max_\theta \ell(\theta) = \arg\max_\theta -\frac{1}{2\sigma^2}\sum_{i=1}^{m}(y_i - \theta^T x_i)^2 \tag{10}$$

$$= \arg\min_\theta \sum_{i=1}^{m}(y_i - \theta^T x_i)^2. \tag{11}$$

The ML estimate is equivalent to ordinary least squares!

Now let us try the MAP estimate. We saw in our coin toss example that the MAP estimate acts as a regularized ML estimate. For probabilistic regression, we will use a multivariate normal prior (we will see later why this is a good idea) with mean 0. The covariance matrix will be diagonal, it is I (the

identity matrix) times σ^2/C, which is the known, constant variance of each component of θ. So the prior is:

$$\theta \sim \mathcal{N}(0, I\sigma^2/C) = \frac{1}{(2\pi)^{d/2}|\sigma^2/C|^{1/2}} \exp\left(-\frac{1}{2}\theta^T \left(I\sigma^2/C\right)^{-1} \theta\right).$$

Here we are saying that our prior belief is that the "slope" of the regression line is near 0. Now,

$$\begin{aligned}\hat{\theta}_{\text{MAP}} &\in \arg\max_\theta \log p(y|x,\theta) + \log p(\theta) \\ &= \arg\max_\theta -\frac{1}{2\sigma^2}\sum_{i=1}^m (y_i - \theta^T x_i)^2 + \log p(\theta)\end{aligned}$$

from (10). At this point you can really see how MAP is a regularized ML. Continuing,

$$\begin{aligned}&= \arg\max_\theta -\frac{1}{2\sigma^2}\sum_{i=1}^m (y_i - \theta^T x_i)^2 - \frac{d}{2}\log 2\pi - \frac{1}{2}\log\frac{\sigma^2}{C} - \frac{1}{2}\theta^T(IC/\sigma^2)\theta \\ &= \arg\max_\theta -\frac{1}{2\sigma^2}\sum_{i=1}^m (y_i - \theta^T x_i)^2 - \frac{1}{2\sigma^2}C\theta^T\theta \\ &= \arg\min_\theta \sum_{i=1}^m (y_i - \theta^T x_i)^2 + C\|\theta\|_2^2. \qquad (12)\end{aligned}$$

We see that the MAP estimate corresponds exactly to ℓ_2-regularized linear regression (ridge regression), and that the ℓ_2 regularization can be interpreted as a Gaussian prior. Increasing C corresponds to increasing our certainty that θ should be close to zero.

3 Conjugate priors

Although point estimates can be useful in many circumstances (and are used in many circumstances), our true goal in Bayesian analysis is often to find the full posterior, $p(\theta|y)$. One reason for wanting the posterior is to be able to use the *posterior predictive distribution* of a yet unobserved data point:

$$p(y_{m+1}|y) = \int p(y_{m+1}|y,\theta)p(\theta|y)d\theta = \int p(y_{m+1}|\theta)p(\theta|y)d\theta, \qquad (13)$$

because y_{m+1} and y are conditionally independent given θ by assumption.

We saw earlier that the posterior can be obtained in principle from the prior and the likelihood using Bayes' rule, but that there is an integral in the denominator which often makes this intractable. One approach to circumventing the integral is to use conjugate priors.

The appropriate likelihood function (Binomial, Gaussian, Poisson, Bernoulli,...) is typically clear from the data. However, there is a great deal of flexibility when choosing the prior distribution. The key notion is that if we choose the 'right' prior for a particular likelihood function, then we can compute the posterior without worrying about the integrating. We will formalize the notion of conjugate priors and then see why they are useful.

Definition 3 (Conjugate Prior)**.** Let $\mathcal{F}$ be a family of likelihood functions and $\mathcal{P}$ a family of prior distributions. $\mathcal{P}$ is a *conjugate prior* to $\mathcal{F}$ if for any likelihood function $f \in \mathcal{F}$ and for any prior distribution $p \in \mathcal{P}$, the corresponding posterior distribution p^* satisfies $p^* \in \mathcal{P}$.

It is easy to find the posterior when using conjugate priors because we know it must belong to the same family of distributions as the prior.

Coin Flip Example Part 4. In our previous part of coin flip example, we were very wise to use a Beta prior for θ because the Beta distribution is the conjugate prior to the Bernoulli distribution. Let us see what happens when we compute the posterior using (2) for the likelihood and (7) for the prior:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{\int_0^1 p(y|\theta')p(\theta')d\theta'} = \frac{\theta^{m_H}(1-\theta)^{m-m_H}\theta^{\alpha-1}(1-\theta)^{\beta-1}}{(\text{normalization})}$$
$$= \frac{\theta^{m_H+\alpha-1}(1-\theta)^{m-m_H+\beta-1}}{\int_0^1 \theta'^{m_H+\alpha-1}(1-\theta')^{m-m_H+\beta-1}d\theta'}.$$

We can recognize the denominator as a Beta function from the definition in (8):

$$p(\theta|y) = \frac{1}{B(m_H+\alpha, m-m_H+\beta)}\theta^{m_H+\alpha-1}(1-\theta)^{m-m_H+\beta-1},$$

and we recognize this as being a Beta distribution:

$$p(\theta|y) \sim \text{Beta}\left(m_H+\alpha, m-m_H+\beta\right). \tag{14}$$

As with the MAP estimate, we can see the interplay between the data y_i and the prior parameters α and β in forming the posterior. As before, the exact choice of α and β does not matter asymptotically, the data overwhelm the prior.

If we knew that the Beta distribution is the conjugate prior to the Bernoulli, we could have figured out the same thing faster by recognizing that

$$p(\theta|y) \propto p(y|\theta)p(\theta) = \theta^{m_H+\alpha-1}(1-\theta)^{m-m_H+\beta-1}, \tag{15}$$

and then realizing that by the definition of a conjugate prior, the posterior must be a Beta distribution. There is exactly one Beta distribution that satisfies (15): the one that is normalized correctly, and that is equation (14).

The parameters of the prior distribution (α and β in the case of the Beta prior) are called *prior hyperparameters*. We choose them to best represent our beliefs about the distribution of θ. The parameters of the posterior distribution ($m_H + \alpha$ and $m - m_H + \beta$) are called *posterior hyperparameters*.

Any time a likelihood model is used together with its conjugate prior, we know the posterior is from the same family of the prior, and moreover we have an explicit formula for the posterior hyperparameters. A table summarizing some of the useful conjugate prior relationships follows. There are many more conjugate prior relationships that are not shown in the following table but that can be found in reference books on Bayesian statistics[1].

[1] *Bayesian Data Analysis* by Gelman, Carlin, Stern, and Rubin is an excellent choice.

Likelihood	**Conjugate Prior**	**Prior Hyper-params**	**Posterior Hyper-params**
Bernoulli	Beta	$\alpha,\ \beta$	$\alpha + \sum_{i=1}^{m} y_i,\ \beta + m - \sum_{i=1}^{m} y_i$
Binomial	Beta	$\alpha,\ \beta$	$\alpha + \sum_{i=1}^{m} y_i,\ \beta + \sum_{i=1}^{m} m_i - \sum_{i=1}^{m} y_i$
Poisson	Gamma	$\alpha,\ \beta$	$\alpha + \sum_{i=1}^{m} y_i,\ \beta + m$
Geometric	Beta	$\alpha,\ \beta$	$\alpha + m,\ \beta + \sum_{i=1}^{m} y_i$
Uniform on $[0, \theta]$	Pareto	$x_s,\ k$	$\max\{\max y_i, x_s\},\ k + m$
Exponential	Gamma	$\alpha,\ \beta$	$\alpha + m,\ \beta + \sum_{i=1}^{m} y_i$
Normal unknown mean known variance σ^2	Normal	$\mu_0,\ \sigma_0^2$	$\left(\frac{\mu_0}{\sigma_0^2} + \frac{1}{\sigma^2}\sum_{i=1}^{m} y_i\right) / \left(\frac{1}{\sigma_0^2} + \frac{m}{\sigma^2}\right), \left(\frac{1}{\sigma_0^2} + \frac{m}{\sigma^2}\right)^{-1}$

We will now discuss a few of these conjugate prior relationships to try to gain additional insight. For Bernoulli-Beta, we saw that the prior hyperparameters can be interpreted as starting the tossing with a certain number of Heads and Tails on the record. The posterior hyperparameters then simply add the observed number of heads (m_H) to the prior hyperparameter for number of Heads (α), and the observed number of tails ($m - m_H$) to the prior hyperparameter for number of tails (β).

For Binomial-Beta, we now have m Binomial experiments, each of which consists of a certain number of coin tosses (m_i) and a certain number of Heads (y_i). As before, the first prior hyperparameter corresponds to number of "hallucinated" Heads, and in the posterior we combine the prior hyperparameter α with the total number of observed Heads across all Binomial trials. Similarly, for the second posterior hyperparameter, we compute the total number of Tails observed across all Binomial trials and combine it with the corresponding prior hyperparameter (β).

In the Uniform-Pareto example, the data come from a uniform distribution on $[0, \theta]$, but we don't know θ. We choose a prior for θ that is a Pareto distribution with prior hyperparameters x_s and k. Here, x_s and k can be interpreted as beginning the experiment with k observations, whose maximum value is x_s (so we believe θ is at least x_s). In the posterior, we replace the maximum value x_s with the new maximum value $\max\{\max y_i, x_s\}$ including

the observed data, and update the number of observations to $k + m$.

For Normal-Normal, we see that the posterior mean is a weighted combination of the data and the prior, where the weights are the variances and correspond to our certainty in the prior vs. the data. The higher the variance in the data, the less certain we are in them and the longer it will take for the data to overwhelm the effect of the prior. (Might be helpful to think of μ_0 as one prior example rather than a bunch of them.)

3.1 Exponential families and conjugate priors

There is an important connection between exponential families (not to be confused with the exponential distribution) and conjugate priors.

Definition 4. The family of distributions $\mathcal{F}$ is an *exponential family* if every member of $\mathcal{F}$ has the form:

$$p(y_i|\theta) = f(y_i)g(\theta)\exp\left(\phi(\theta)^T u(y_i)\right), \tag{16}$$

for some $f(\cdot), g(\cdot), \phi(\cdot)$, and $u(\cdot)$.

Essentially all of the distributions that we typically work with (normal, exponential, Poisson, beta, gamma, binomial, Bernoulli,...) are exponential families. The next theorem tells us when we can expect to have a conjugate prior.

Theorem 1 (**Exponential families and conjugate priors**). *If the likelihood model is an exponential family, then there exists a conjugate prior.*

Proof. Consider the likelihood of our iid data $y = \{y_1, \ldots, y_m\}$:

$$\begin{aligned} p(y|\theta) &= \prod_{i=1}^{m} p(y_i|\theta) = \prod_{i=1}^{m} f(y_i)g(\theta)\exp\left(\phi(\theta)^T u(y_i)\right) \\ &= \left[\prod_{i=1}^{m} f(y_i)\right] g(\theta)^m \exp\left(\phi(\theta)^T \sum_{i=1}^{m} u(y_i)\right). \end{aligned}$$

Take the prior distribution to be:

$$p(\theta) = \frac{g(\theta)^\eta \exp\left(\phi(\theta)^T \nu\right)}{\int g(\theta')^\eta \exp\left(\phi(\theta')^T \nu\right) d\theta'}, \tag{17}$$

where η and ν are prior hyperparameters. Then the posterior will be

$$\begin{aligned} p(\theta|y) &\propto p(y|\theta)p(\theta) \\ &= \left[\prod_{i=1}^{m} f(y_i)\right] g(\theta)^m \exp\left(\phi(\theta)^T \sum_{i=1}^{m} u(y_i)\right) \frac{g(\theta)^\eta \exp\left(\phi(\theta)^T \nu\right)}{\int g(\theta')^\eta \exp\left(\phi(\theta')^T \nu\right) d\theta'} \\ &\propto g(\theta)^{\eta+m} \exp\left(\phi(\theta)^T \left(\nu + \sum_{i=1}^{m} u(y_i)\right)\right) \quad \text{(dropping non-}\theta\text{ terms)}, \end{aligned}$$

which is in the same family as the prior (17), with the posterior hyperparameters being $\eta + m$ and $\nu + \sum_{i=1}^{m} u(y_i)$.

Although this proof yields the form of the conjugate prior, we may not always be able to compute the partition function. So we can't always gain something in practice from it.

It turns out that in general, the converse to Theorem 1 is true, and exponential families are the only distributions with (non-trivial) conjugate priors.

Coin Flip Example Part 5. Returning again to the coin flip example, let us first verify that the Bernoulli distribution is an exponential family:

$$\begin{aligned} p(y_i|\theta) &= \theta^{y_i}(1-\theta)^{1-y_i} \\ &= \exp\left(y_i \log\theta + (1-y_i)\log(1-\theta)\right) \\ &= \exp\left(y_i \log\theta - y_i \log(1-\theta) + \log(1-\theta)\right) \\ &= (1-\theta)\exp\left(y_i \log\frac{\theta}{1-\theta}\right), \end{aligned}$$

and we see that the Bernoulli distribution is an exponential family according to (16) with $f(y_i) = 1$, $g(\theta) = 1-\theta$, $u(y_i) = y_i$, and $\phi(\theta) = \log\frac{\theta}{1-\theta}$. Thus, according to (17), the conjugate prior is

$$\begin{aligned} p(\theta) &\propto g(\theta)^\eta \exp\left(\phi(\theta)^T \nu\right) \\ &= (1-\theta)^\eta \exp\left(\nu \log\frac{\theta}{1-\theta}\right) \\ &= (1-\theta)^\eta \left(\frac{\theta}{1-\theta}\right)^\nu \\ &= \theta^\nu (1-\theta)^{\eta-\nu}. \end{aligned}$$

Reparameterizing by defining $\nu = \alpha - 1$ and $\eta - \nu = \beta - 1$ gives us the Beta distribution that we expect.

Example 2. (Normal Distribution is an Exponential Family) Consider a normal distribution with known variance but unknown mean:

$$p(y_i|\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y_i - \theta)^2\right).$$

Let $f(y_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-y_i^2/2\sigma^2\right)$, $u(y_i) = y_i/\sigma$, $\phi(\theta) = \theta/\sigma$, and $g(\theta) = \exp\ -\theta^2/2\sigma^2\ $. Then,

$$\begin{aligned} f(y_i)g(\theta)\exp\left(\phi(\theta)u(y_i)\right) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}y_i^2\right) \exp\left(-\frac{1}{2\sigma^2}\theta^2\right) \exp\left(\frac{1}{2\sigma^2}2\theta y_i\right) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y_i - \theta)^2\right) \\ &= p(y_i|\theta). \end{aligned}$$

Thus the normal distribution with known variance is an exponential family.

Example 3. (Exponential Distribution is an Exponential Family) Consider an exponential distribution:

$$p(y_i|\theta) = \theta e^{-\theta y_i}.$$

Let $f(y_i) = 1$, $u(y_i) = y_i$, $\phi(\theta) = -\theta$, and $g(\theta) = \theta$. Then,

$$f(y_i)g(\theta)\exp\left(\phi(\theta)u(y_i)\right) = \theta e^{-\theta y_i} = p(y_i|\theta).$$

Thus the exponential distribution is an exponential family.

4 Posterior asymptotics

Up to this point, we have defined a likelihood model that is parameterized by θ, assigned a prior distribution to θ, and then computed the posterior $p(\theta|y)$. There are two natural questions that arise. First, what if we choose the 'wrong' likelihood model? That is, what if the data were actually generated by some distribution $q(y)$ such that $q(y) \neq p(y|\theta)$ for any θ, but we use $p(y|\theta)$ as our likelihood model? Second, what if we assign the 'wrong' prior? We can answer both of these questions asymptotically as $m \to \infty$. First we

must develop a little machinery from information theory.

A useful way to measure the dissimilarity between two probability distributions is the *Kullback-Leibler (KL) divergence*, defined for two distributions $p(y)$ and $q(y)$ as:

$$D(q(\cdot)||p(\cdot)) := \mathbb{E}_{y\sim q(y)}\left[\log\frac{q(y)}{p(y)}\right] = \int q(y)\log\frac{q(y)}{p(y)}dy.$$

The KL divergence is only defined if $q(y) > 0$ for any y such that $p(y) > 0$. It is sometimes referred to as the KL distance, however it is not a metric in the mathematical sense because in general it is asymmetric: $D(q(\cdot)||p(\cdot)) \neq D(p(\cdot)||q(\cdot))$. It is the average of the logarithmic difference between the probability distributions $p(y)$ and $q(y)$, where the average is taken with respect to $q(y)$. The following property of the KL divergence will be very important for us.

Theorem 2 (Non-negativity of KL Divergence)**.** $D(q(\cdot)||p(\cdot)) \geq 0$ *with equality if and only if* $q(y) = p(y)\,\forall y$.

Proof. We will rely on Jensen's inequality, which states that for any convex function f and random variable X,

$$\mathbb{E}[f(X)] \geq f(\mathbb{E}[X]).$$

When f is strictly convex, Jensen's inequality holds with equality if and only if X is constant, so that $\mathbb{E}[X] = X$ and $\mathbb{E}[f(X)] = f(X)$. Take $y \sim q(y)$ and define the random variable $X = \frac{p(y)}{q(y)}$. Let $f(X) = -\log(X)$, a strictly convex function. Now we can apply Jensen's inequality:

$$\begin{aligned}
\mathbb{E}_{y\sim q(y)}[f(X)] &\geq f(\mathbb{E}_{y\sim q(y)}[X]) \\
-\int q(y)\log\frac{p(y)}{q(y)}dy &\geq -\log\left(\int q(y)\frac{p(y)}{q(y)}dy\right) \\
&\geq -\log\left(\int p(y)dy\right) \\
&\geq -\log 1 = 0 \quad \text{so,} \\
\int q(y)\log\frac{q(y)}{p(y)}dy &\geq 0,
\end{aligned}$$

with equality under the same conditions required for equality in Jensen's inequality: if and only if X is constant, that is, $q(y) = p(y)\,\forall y$.

We will use the KL divergence to find the distribution from the likelihood family that is 'closest' to the true generating distribution:

$$\theta^* = \arg\min_{\theta \in \Theta} D(q(\cdot)||p(\cdot|\theta)). \tag{18}$$

For convenience, we will suppose that the arg min in (18) is unique. The results can be easily extended to the case where the arg min is not unique. The main results of this section are two theorems, the first for discrete parameter spaces and the second for continuous parameter spaces. The intuition for the theorem is that as long as there is some probability in the prior that $\theta = \theta^*$, then as $m \to \infty$ the whole posterior will be close to θ^*.

Theorem 3 (Posterior consistency in finite parameter space)**.** *Let $\mathcal{F}$ be a finite family of likelihood models, with $\Theta = \{\theta : p(\cdot|\theta) \in \mathcal{F}\}$ the finite parameter space. Let $y = (y_1, \ldots, y_m)$ be a set of independent samples from an arbitrary distribution $q(\cdot)$, and θ^* be as in (18). If $p(\theta = \theta^*) > 0$, then $p(\theta = \theta^*|y) \to 1$ as $m \to \infty$.*

Proof. Consider any $\theta \neq \theta^*$:

$$\log \frac{p(\theta|y)}{p(\theta^*|y)} = \log \frac{p(\theta)p(y|\theta)}{p(\theta^*)p(y|\theta^*)} = \log \frac{p(\theta)}{p(\theta^*)} + \sum_{i=1}^{m} \log \frac{p(y_i|\theta)}{p(y_i|\theta^*)} \tag{19}$$

Consider the sum. By the strong law of large numbers, with probability 1,

$$\frac{1}{m} \sum_{i=1}^{m} \log \frac{p(y_i|\theta)}{p(y_i|\theta^*)} \to \mathbb{E}_{y \sim q(y)} \left[\log \frac{p(y|\theta)}{p(y|\theta^*)} \right]. \tag{20}$$

Well,

$$\begin{aligned}
\mathbb{E}_{y \sim q(y)} \left[\log \frac{p(y|\theta)}{p(y|\theta^*)} \right] &= \mathbb{E}_{y \sim q(y)} \left[\log \frac{p(y|\theta)q(y)}{p(y|\theta^*)q(y)} \right] \text{ (1 in disguise)} \\
&= \mathbb{E}_{y \sim q(y)} \left[\log \frac{q(y)}{p(y|\theta^*)} - \log \frac{q(y)}{p(y|\theta)} \right] \\
&= D(q(\cdot)||p(\cdot|\theta^*)) - D(q(\cdot)||p(\cdot|\theta)) \\
&< 0. \quad \boxed{\text{Why?}}
\end{aligned}$$

By (20), with probability 1, as $m \to \infty$,

$$\sum_{i=1}^{m} \log \frac{p(y_i|\theta)}{p(y_i|\theta^*)} \to m\mathbb{E}_{y \sim q(y)} \left[\log \frac{p(y|\theta)}{p(y|\theta^*)} \right] = -\infty.$$

Let's plug back into (19). By supposition, $p(\theta^*) > 0$ (and the prior doesn't change as $m \to \infty$).Thus as $m \to \infty$, (19) obeys

$$\log \frac{p(\theta|y)}{p(\theta^*|y)} = \log \frac{p(\theta)}{p(\theta^*)} + \sum_{i=1}^{m} \log \frac{p(y_i|\theta)}{p(y_i|\theta^*)} \to -\infty,$$

and

$$\log \frac{p(\theta|y)}{p(\theta^*|y)} \to -\infty \text{ implies } \frac{p(\theta|y)}{p(\theta^*|y)} \to 0 \text{ which implies } p(\theta|y) \to 0.$$

This holds for every $\theta \neq \theta^*$, thus $p(\theta^*|y) \to 1$. □

Again, this theorem tells us that the posterior eventually becomes concentrated on the value θ^*. θ^* corresponds to the likelihood model that is 'closest' in the KL sense to the true generating distribution $q(\cdot)$. If $q(\cdot) = p(\cdot|\theta_0)$ for some $\theta_0 \in \Theta$, then $\theta^* = \theta_0$ is the unique minimizer of (18) because $p(\cdot|\theta^*)$ is closest to $p(\cdot|\theta_0)$ when $\theta^* = \theta_0$. The theorem tells us that the posterior will become concentrated around the true value. This is only the case if in the prior, $p(\theta^*) > 0$, which shows the importance of choosing a prior that assigns non-zero probability to every plausible value of θ.

We now present the continuous version of Theorem 3, but the proof is quite technical and is omitted.

Theorem 4 (Posterior consistency in continuous parameter space)**.** *If Θ is a compact set, θ^* is defined as in (18), A is a neighborhood of θ^*, and $p(\theta^* \in A) > 0$, then $p(\theta \in A|y) \to 1$ as $m \to \infty$.*

These theorems show that asymptotically, the choice of the prior does not matter as long as it assigns non-zero probability to every $\theta \in \Theta$. They also show that if the data were generated by a member of the family of likelihood models, we will converge to the correct likelihood model. If not, then we will converge to the model that is 'closest' in the KL sense.

We give these theorems along with a word of caution. These are asymptotic results that tell us absolutely nothing about the sort of m we encounter in practical applications. For small sample sizes, poor choices of the prior or likelihood model can yield poor results and we must be cautious.

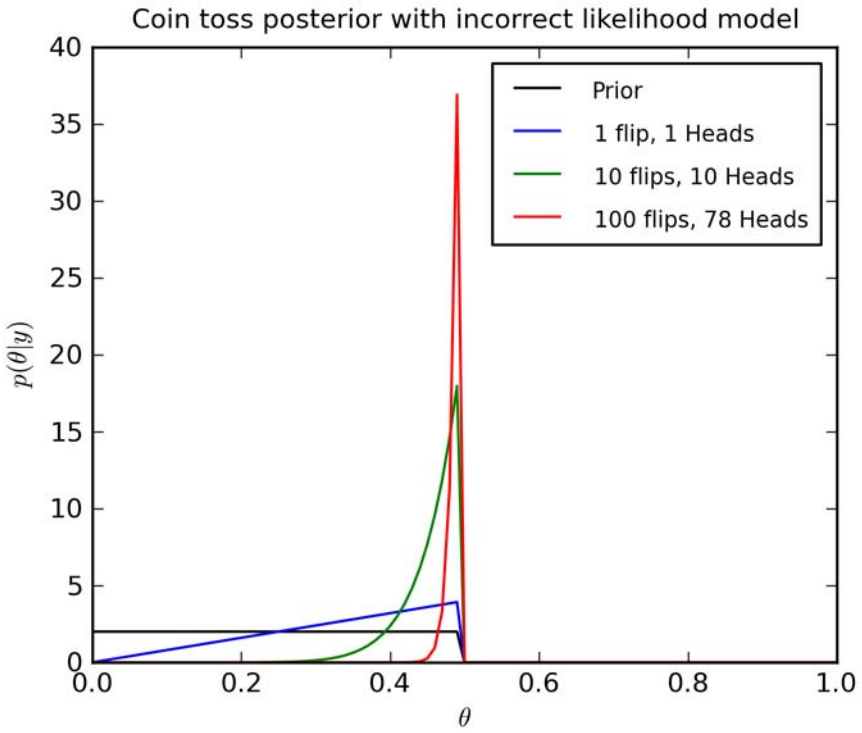
Coin Flip Example Part 6. Suppose that the coin flip data truly came from a biased coin with a 3/4 probability of Heads, but we restrict the likelihood model to only include coins with probability in the interval $[0, 1/2]$:

$$p(y|\theta) = \theta^{m_H}(1-\theta)^{m-m_H}, \quad \theta \in [0, 1/2].$$

This time we will use a uniform prior, $p(\theta) = 2, \theta \in [0, 1/2]$. The posterior distribution is then

$$p(\theta|y) = \frac{\theta^{m_H}(1-\theta)^{m-m_H}2}{\int_0^{1/2} \theta'^{m_H}(1-\theta')^{m-m_H}2d\theta'}.$$

The partition function is an incomplete Beta integral, which has no closed form but can easily be solved numerically. In the following figure, we draw samples iid from the true distribution (3/4 probability of Heads) and show how the posterior becomes increasingly concentrated around $\theta = 0.5$, the closest likelihood function to the true generating distribution.



5 Hierarchical modeling

Hierarchical models are a powerful application of Bayesian analysis. They allow us to reason about knowledge at multiple levels of abstraction. Basically,

the prior parameters at the lower level come from the same distribution at the higher level, and there is a prior at that level, and so on. This is best explained by example, so we will develop a hierarchical model for *topic modeling*, an important information retrieval problem.

5.1 Topic models

Suppose we have been given a collection of m documents and we wish to determine how the documents are related. For example, if the documents are all news articles, the article "Patriots game canceled due to hurricane" is related to the article "New York Giants lose superbowl" because they are both about football. The article "Patriots game canceled due to hurricane" is also related to the article "Record snowfall in May" because they are both about the weather. We will now develop a hierarchical model for finding topic relationships between documents in an unsupervised setting. The method is called *Latent Dirichlet Allocation* (LDA) and it was developed by David Blei, Andrew Ng, and Michael Jordan.

5.1.1 LDA formulation

The model has several components. The data are m documents, with document i consisting of n_i words. Each word in the document will be associated with one of K topics. We let $z_{i,j}$ denote the topic of word j in document i. We model $z_{i,j} \sim \text{Multinomial}(\theta_i)$, where $\theta_i \in \mathbb{R}^K$ describes the topic mixture of document i.

For each topic, we define a multinomial distribution over all possible words. For example, given the topic is "Sports", the probability of having the word "football" might be high; if the topic were "Weather", the probability of having the word "football" might be lower. Other words, like "the" will have a high probability regardless of the topic. If words are chosen from a set of W possible words, then we let $\phi_k \in \mathbb{R}^W$ be the multinomial parameter over words for topic k. Word j of document i, denoted $w_{i,j}$, will be generated by the distribution over words corresponding to the topic $z_{i,j}$: $w_{i,j} \sim \text{Multinomial}(\phi_{z_{i,j}})$.

Finally, we give prior distributions for the parameters θ_i and ϕ_k. The multinomial distribution is a generalization of the binomial distribution, and its

conjugate prior is a generalization of the beta distribution: the Dirichlet distribution. Thus we model the data with the following generative model:

1. For document $i = 1, \ldots, m$, choose the document's topic distribution $\theta_i \sim \text{Dirichlet}(\alpha)$, where $\alpha \in \mathbb{R}^K$ is the prior hyperparameter.

2. For topic $k = 1, \ldots, K$, choose the topic's word distribution $\phi_k \sim \text{Dirichlet}(\beta)$, where $\beta \in \mathbb{R}^W$ is the prior hyperparameter.

3. For document $i = 1, \ldots, m$:

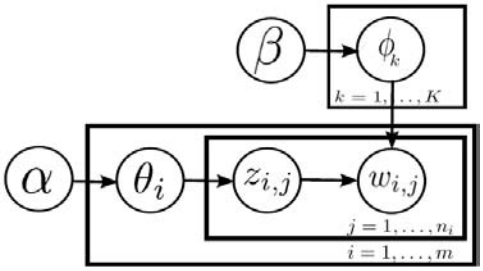
   For word $j = 1, \ldots, n_i$:

   Choose the topic for this word $z_{i,j} \sim \text{Multinomial}(\theta_i)$.

   Choose the word $w_{i,j} \sim \text{Multinomial}(\phi_{z_{i,j}})$.

5.2 Graphical representation

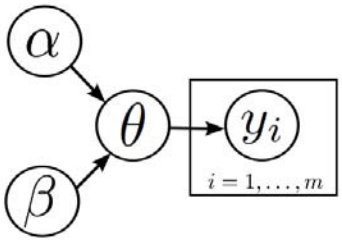
Hierarchical models are illustrated with a node for every variable and arcs between nodes to indicate the dependence between variables. Here's the one for LDA:



Graphs representing hierarchical models must be acyclic. For any node x, we define Parents(x) as the set of all nodes with arcs to x. The hierarchical model consists of, for every node x, the distribution $p(x|\text{Parents}(x))$. Define Descendants(x) as all nodes that can be reached from x and Non-descendants(x) as all other nodes. Because the graph is acyclic and the distribution for each node depends only on its parents, given Parents(x), x is conditionally independent from Non-descendants(x). This is a powerful fact about hierarchical models that is important for doing inference. In the graph for LDA, this means that, for example, $z_{i,j}$ is independent of α, given θ_i.

In addition to the graph structure, we use plates to denote repeated, independent draws from the same distribution. The plates are like the 'for' loops in the pseudocode of how the data is generated in the text description above.

Coin Flip Example Part 7. Even simple models like the coin flip model can be represented graphically. The coin flip model is:



5.3 Inference in hierarchical models

Hierarchical models are useful because they allow us to model interactions between the observed variables (in this case the words in each document) through the use of latent (or hidden) variables.

What are the latent variables for LDA?

Despite introducing a larger number of latent variables than we have observed variables, because of their additional structure hierarchical models are generally not as prone to overfitting as you might expect.

We are interested in infering the posterior distribution for the latent variables. Let $Z = \{z_{i,j}\}_{i=1,\ldots,m, j=1,\ldots,n_i}$, $\theta = \{\theta_i\}_{i=1,\ldots,m}$, $\phi = \{\phi_k\}_{k=1,\ldots,K}$, and $W = \{w_{i,j}\}_{i=1,\ldots,m, j=1,\ldots,n_i}$. Then, by Bayes' rule, ignoring the constant denominator, we can express the posterior as:

$$p(Z, \theta, \phi | w, \alpha, \beta) \propto p(w | Z, \phi, \theta, \alpha, \beta) p(Z, \theta, \phi | \alpha, \beta) \tag{21}$$

We will look at each of these pieces and show that they have a compact

analytical form.

$$\begin{aligned} p(w|Z,\phi,\theta,\alpha,\beta) &= \prod_{i=1}^{m}\prod_{j=1}^{n_i} p(w_{i,j}|Z,\phi,\theta,\alpha,\beta) \quad \text{By iid assumption} \\ &= \prod_{i=1}^{m}\prod_{j=1}^{n_i} p(w_{i,j}|z_{i,j},\phi) \quad \text{By conditional independence} \\ &= \prod_{i=1}^{m}\prod_{j=1}^{n_i} \text{Multinomial}(w_{i,j};\phi_{z_{i,j}}) \quad \text{By definition.} \end{aligned}$$

Also,

$$p(Z,\theta,\phi|\alpha,\beta) = p(Z,\theta|\alpha)p(\phi|\beta)$$

by conditional independence. Again considering each term, by the definition of conditional probability:

$$p(Z,\theta|\alpha) = p(Z|\theta,\alpha)p(\theta|\alpha),$$

where

$$p(Z|\theta,\alpha) = p(Z|\theta) = \prod_{i=i}^{m}\prod_{j=1}^{n_i} p(z_{i,j}|\theta_i) = \prod_{i=i}^{m}\prod_{j=1}^{n_i} \text{Multinomial}(z_{i,j};\theta_i),$$

by conditional independence, iid, and definition as before. Further,

$$p(\theta|\alpha) = \prod_{i=1}^{m} p(\theta_i|\alpha) = \prod_{i=1}^{m} \text{Dirichlet}(\theta_i;\alpha),$$

and

$$p(\phi|\beta) = \prod_{k=1}^{K} p(\phi_k|\beta) = \prod_{k=1}^{K} \text{Dirichlet}(\phi_k;\beta).$$

Plugging all of these pieces back into (21), we obtain

$$\begin{aligned} p(Z,\theta,\phi|w,\alpha,\beta) \propto &\prod_{i=1}^{m}\prod_{j=1}^{n_i} \text{Multinomial}(w_{i,j};\phi_{z_{i,j}}) \prod_{i=i}^{m}\prod_{j=1}^{n_i} \text{Multinomial}(z_{i,j};\theta_i) \times \\ &\prod_{i=1}^{m} \text{Dirichlet}(\theta_i;\alpha) \prod_{k=1}^{K} \text{Dirichlet}(\phi_k;\beta). \end{aligned} \tag{22}$$

For any given $Z, \theta, \phi, w, \alpha$, and β, we can easily evaluate (22). (All the normalizations are known so it's easy.) We will see in the next section that this is sufficient to be able to simulate draws from the posterior.

Even using conjugate priors, in general it will not be possible to recover the posterior analytically for hierarchical models of any complexity. We will rely on (among a few other options) sampling methods like the Monte Carlo Markov Chains (MCMC) that we discuss in the next section. What the statistics community call Bayesian hierarchical models are in the machine learning community often treated as a special case of Bayesian graphical models (specifically, directed acyclic graphs). There is at least one entire course at MIT on inference in Bayesian graphical models (6.438).

6 Markov Chain Monte Carlo sampling

As we have seen with hierarchical models, even with conjugate priors we are unable to express the posterior analytically. The reason Bayesian statistics is so widely used is because of the development of computational methods for simulating draws from the posterior distribution. Even though we are unable to express the posterior analytically, with a large enough sample of simulated draws we can compute statistics of the posterior with arbitrary precision. This approach is called Monte Carlo simulation. We will describe the two most commonly used Monte Carlo methods, which both fall under the umbrella of Markov Chain Monte Carlo (MCMC) methods: the Metropolis-Hastings algorithm, and Gibbs' sampling.

6.1 Markov chains

The results from MCMC depend on some key results from the theory of Markov chains. We will not do a thorough review of Markov Chains and will rather present the necessary results as fact. A *continuous state Markov chain* is a sequence $\theta^0, \theta^1, \ldots$ with $\theta^t \in \mathbb{R}^d$ that satisfies the *Markov property*:

$$p(\theta^t | \theta^{t-1}, \ldots, \theta^1) = p(\theta^t | \theta^{t-1}).$$

We will be interested in the (unconditioned) probability distribution over states at time t, which we denote as $\pi^t(\theta) := \Pr(\theta^t = \theta)$. Under some

conditions that will be satisfied by all of the chains we are interested in, the sequence of state distributions $\pi^0(\theta), \pi^1(\theta), \ldots$ will converge to a unique distribution $\pi(\theta)$ which we call the *stationary distribution*, or equilibrium distribution or steady-state distribution.

We define the *transition kernel* to be the probability of transitioning from state θ to state θ': $K(\theta, \theta') = p(\theta'|\theta)$. We then have the following important fact.

Fact: if the distribution $\pi(\cdot)$ satisfies the *detailed balance equation*:

$$K(\theta, \theta')\pi(\theta) = K(\theta', \theta)\pi(\theta'), \text{ for all } \theta, \theta', \tag{23}$$

then $\pi(\cdot)$ is the stationary distribution. The interpretation of the detailed balance equation is that the amount of mass transitioning from θ' to θ is the same as the amount of mass transition back from θ to θ'. For a stationary distribution, we cannot have mass going from state to state.

6.2 Metropolis-Hastings algorithm

The goal in MCMC is to construct a Markov Chain whose stationary distribution is the posterior $p(\theta|y)$. We now present the Metropolis-Hastings algorithm. In addition to the distributions we have already used (likelihood and prior), we will need a *proposal distribution* (or jumping distribution) $J(\theta, \theta')$ which will propose a new state θ' given the current state θ.

There are many options when choosing a proposal distribution which we will discuss later. The proposal distribution will yield a random walk over the parameter space, proposing steps $\theta \to \theta'$. We accept or reject each step depending on the relative posterior probabilities for θ and θ'. When we run the random walk for long enough, the accepted values will simulate draws from the posterior.

6.2.1 Some intuition into the Metropolis-Hastings algorithm

Suppose we are considering the transition $\theta \to \theta'$. If $p(\theta'|y)$ is larger than $p(\theta|y)$, then for every accepted draw of θ, we should have at least as many accepted draws of θ' and so we should always accept the transition $\theta \to \theta'$.

If $p(\theta'|y)$ is less than $p(\theta|y)$, then for every accepted draw θ, we should have on average $\frac{p(\theta'|y)}{p(\theta|y)}$ accepted draws of θ'. We thus accept the transition with probability $\frac{p(\theta'|y)}{p(\theta|y)}$. Thus for any transition, we accept the transition with probability

$$\min\left\{\frac{p(\theta'|y)}{p(\theta|y)}, 1\right\}. \tag{24}$$

6.2.2 Steps of the algorithm

We now give the steps of the Metropolis-Hastings algorithm.

Step 1. Choose a starting point θ^0. Set $t = 1$.

Step 2. Draw θ^* from the proposal distribution $J(\theta^{t-1}, \cdot)$. The proposed move for time t is to move from θ^{t-1} to θ^*.

Step 3. Compute the following:

$$\begin{aligned}\alpha(\theta^{t-1}, \theta^*) &:= \min\left\{\frac{p(\theta^*|y)J(\theta^*, \theta^{t-1})}{p(\theta^{t-1}|y)J(\theta^{t-1}, \theta^*)}, 1\right\} \\ &= \min\left\{\frac{p(y|\theta^*)p(\theta^*)J(\theta^*, \theta^{t-1})}{p(y|\theta^{t-1})p(\theta^{t-1})J(\theta^{t-1}, \theta^*)}, 1\right\} \end{aligned} \tag{25}$$

We'll explain this more soon. The fact that we can compute ratios of posterior probabilities without having to worry about the normalization integral is the key to Monte Carlo methods.

Step 4. With probability $\alpha(\theta^{t-1}, \theta^*)$, accept the move $\theta^{t-1} \to \theta^*$ by setting $\theta^t = \theta^*$ and incrementing $t \leftarrow t + 1$. Otherwise, discard θ^* and stay at θ^{t-1}.

Step 5. Until stationary distribution and the desired number of draws are reached, return to **Step 2.**

Equation (25) reduces to what we developed intuition for in (24) when the proposal distribution is symmetric: $J(\theta, \theta') = J(\theta', \theta)$. We will see in the next theorem that the extra factors in (25) are necessary for the posterior to be the stationary distribution.

6.2.3 Convergence of Metropolis-Hastings

Because the proposal distribution and $\alpha(\theta^{t-1}, \theta^*)$ depend only on the current state, the sequence $\theta^0, \theta^1, \ldots$ forms a Markov chain. What makes the Metropolis-Hastings algorithm special is the following theorem, which shows that if we simulate the chain long enough, we will simulate draws from the posterior.

Theorem 5. *If $J(\theta, \theta')$ is such that that the Markov chain $\theta^0, \theta^1, \ldots$ produced by the Metropolis-Hastings algorithm has a unique stationary distribution, then the stationary distribution is $p(\theta|y)$.*

Proof. We will use the fact given in (23) that if the posterior $p(\theta|y)$ satisfies the detailed balance equation, then it is the stationary distribution. Thus we wish to show:

$$K(\theta, \theta')p(\theta|y) = K(\theta', \theta)p(\theta'|y), \text{ for all } \theta, \theta'.$$

where the transition kernel comes from Metropolis-Hastings:

$$\begin{aligned} K(\theta, \theta') = &\quad (\text{probability of proposing } \theta') \times \\ &\quad (\text{probability of accepting } \theta' \text{ given it was proposed}) \\ = &\quad J(\theta, \theta')\alpha(\theta, \theta'). \end{aligned}$$

To show that the detailed balance equation holds, take any θ and θ', and without loss of generality, suppose that α is less than or equal to 1 for the transition θ to θ', which means

$$J(\theta, \theta')p(\theta|y) \geq J(\theta', \theta)p(\theta'|y),$$

so that

$$\alpha(\theta, \theta') = \frac{J(\theta', \theta)p(\theta'|y)}{J(\theta, \theta')p(\theta|y)},$$

and $\alpha(\theta', \theta) = 1$. Now let's plug:

$$\begin{aligned} K(\theta, \theta')p(\theta|y) =& J(\theta, \theta')\alpha(\theta, \theta')p(\theta|y) \\ =& J(\theta, \theta')p(\theta|y)\frac{J(\theta', \theta)p(\theta'|y)}{J(\theta, \theta')p(\theta|y)} \\ =& J(\theta', \theta)p(\theta'|y) \quad (\text{cancel terms}) \\ =& J(\theta', \theta)\alpha(\theta', \theta)p(\theta'|y) \quad (1 \text{ in disguise}) \\ =& K(\theta', \theta)p(\theta'|y). \end{aligned}$$

And that's the detailed balance equation. □

We have now proven that the Metropolis-Hastings algorithm simulations will eventually draw from the posterior distribution. However, there are a number of important questions to be addressed. What proposal distribution should we use? How many iterations will it take for the chain to be sufficiently close to the stationary distribution? How will we know when the chain has reached its stationary distribution? We will discuss these important issues after we introduce the Gibbs' sampler.

6.3 Gibbs' Sampler

The Gibbs' sampler is a very powerful MCMC sampling technique for the special situation when we have access to conditional distributions. It is a special case of the Metropolis-Hastings algorithm that is typically much faster, but can only be used in special cases.

Let us express $\theta \in \mathbb{R}^d$ as $\theta = [\theta_1, \ldots, \theta_d]$. Suppose that although we are not able to draw directly from $p(\theta|y)$ because of the normalization integral, we are able to draw samples from the conditional distribution

$$p(\theta_j|\theta_1, \ldots, \theta_{j-1}, \theta_{j+1}, \ldots, \theta_d, y).$$

In fact, it is often the case in hierarchical models that their structure allows us to determine analytically these conditional posterior distributions. This can be done for LDA, and the derivation is quite lengthy but can be found on the Wikipedia article for LDA. The Gibbs' sampler updates the posterior variables $\theta_1, \theta_2, \ldots, \theta_d$ one at a time. At each step, all of them are held constant in their current state except for one (j), which is then updated by drawing from its (known) conditional posterior distribution, $p(\theta_j|\theta_1, \ldots, \theta_{j-1}, \theta_{j+1}, \ldots, \theta_d, y)$. We then hold it in its state and move on to the next variable to update it similarly. When we do this iterative updating for long enough, we eventually simulate draws from the full posterior. The full algorithm is:

Step 1. Initialize $\theta^0 = [\theta_1^0, \ldots, \theta_d^0]$. Set $t = 1$.

Step 2. For $j \in \{1, \ldots, d\}$, sample θ_j^t from $p(\theta_j|\theta_1^t, \ldots, \theta_{j-1}^t, \theta_{j+1}^{t-1}, \ldots, \theta_d^{t-1}, y)$.

Step 3. Until stationary distribution and the desired number of draws are reached, increment $t \leftarrow t + 1$ and return to **Step 2.**

In each iteration of the Gibbs' sampler, we sequentially update each component of θ^t. We could do that updating in any order, it does not have to be $1, \ldots, d$.

The Gibbs' sampler is a special case of Metropolis-Hastings where the proposal distribution is taken to be the conditional posterior distribution. In fact, it is easy to show (but notationally extremely cumbersome) that when we use these conditional posterior distributions as proposal distributions in Metropolis-Hastings, the probability of accepting any proposed move is 1, hence in the Gibbs' sampler we accept every move.

6.4 Practical considerations

Now that we have seen the general idea of MCMC algorithms and some theory behind them, let us dive into some details.

6.4.1 Proposal distributions

To guarantee existence of a stationary distribution, all that is required (with rare exceptions) is for the proposal distribution $J(\cdot, \cdot)$ to be such that there is a positive probability of eventually reaching any state from any other state. A typical proposal distribution is a random walk $J(\theta, \theta') = \mathcal{N}(\theta, \sigma^2)$ for some σ^2. There are several important features of proposal distributions that work well in practice. First, we must be able to sample from it efficiently. Second, we must be able to compute the ratio $\alpha(\theta, \theta')$ in an efficient way. Third, the jumps should not be too large or we will reject them frequently and the chain will not move quickly. Fourth, the jumps should not be too small or it will take a long time to explore the whole space. The balance between 'too small' and 'too large' is the subject of hundreds of papers on 'adaptive MCMC', but there is really no good way to know which proposal distribution to use. In practice, we often try several proposal distributions to see which is most appropriate, for example, by adjusting σ^2 in the above proposal distribution.

6.4.2 On reaching the stationary distribution

Unfortunately, it is impossible to know how many iterations it will take to reach the stationary distribution, or even to be certain when we have arrived.

This is probably the largest flaw in MCMC sampling. There are a large number of heuristics which are used to assess convergence that generally involve looking at how θ^t varies with time. In general, the initial simulations depend strongly on the starting point and are thrown away. This is referred to as *burn-in*, and often involves discarding the first half of all draws.

To reduce the effect of autocorrelations between draws, we typically only store a small fraction of them, for example we store only 1 out of every 1000 accepted draws. This process is called *thinning*.

Finally, we can assess convergence by comparing the distribution of the first half of stored draws to the distribution of the second half of stored draws. If the chain has reached its stationary distribution, these should have similar distributions, as assessed by various statistics. These (and similar) approaches have been shown to be successful in practice. Unfortunately, there is no way to be entirely certain that we are truly drawing from the posterior and we must be very cautious.

Coin Flip Example Part 8. Let us return to our coin flip example. We will draw from the posterior using the Metropolis-Hastings algorithm. Our model has a scalar parameter $\theta \in [0, 1]$ which is the probability of heads. The proposal distribution $J(\theta^{t-1}, \theta^*)$ will be uniform on an interval of size r around θ^{t-1}:

$$J_t(\theta^{t-1}, \theta^*; r) = \begin{cases} \frac{1}{r} & \text{if } \theta^* \in [\theta^{t-1} \ominus \frac{r}{2}, \theta^{t-1} \oplus \frac{r}{2}] \\ 0 & \text{otherwise} \end{cases}$$

where $\oplus$ and $\ominus$ represent modular addition and subtraction on $[0, 1]$, *e.g.*, $0.7 \oplus 0.5 = 0.2$. Notice that this proposal distribution is symmetric:

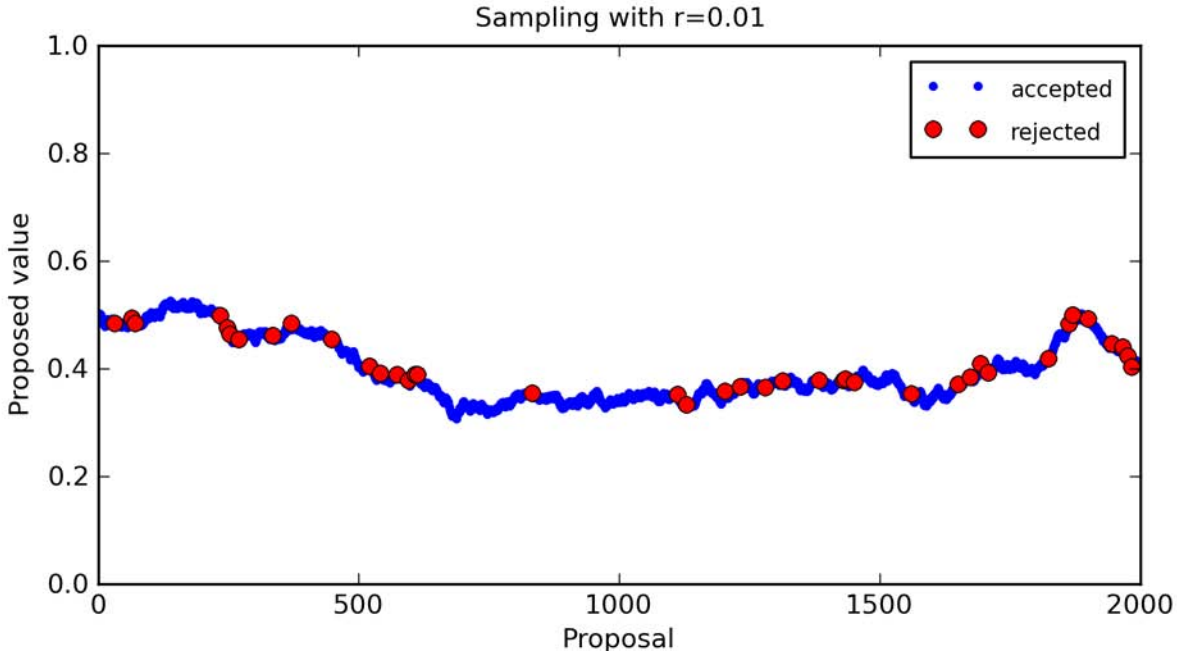
$$J(\theta^{t-1}, \theta^*; r) = J(\theta^*, \theta^{t-1}; r).$$

We accept the proposed θ^* with probability

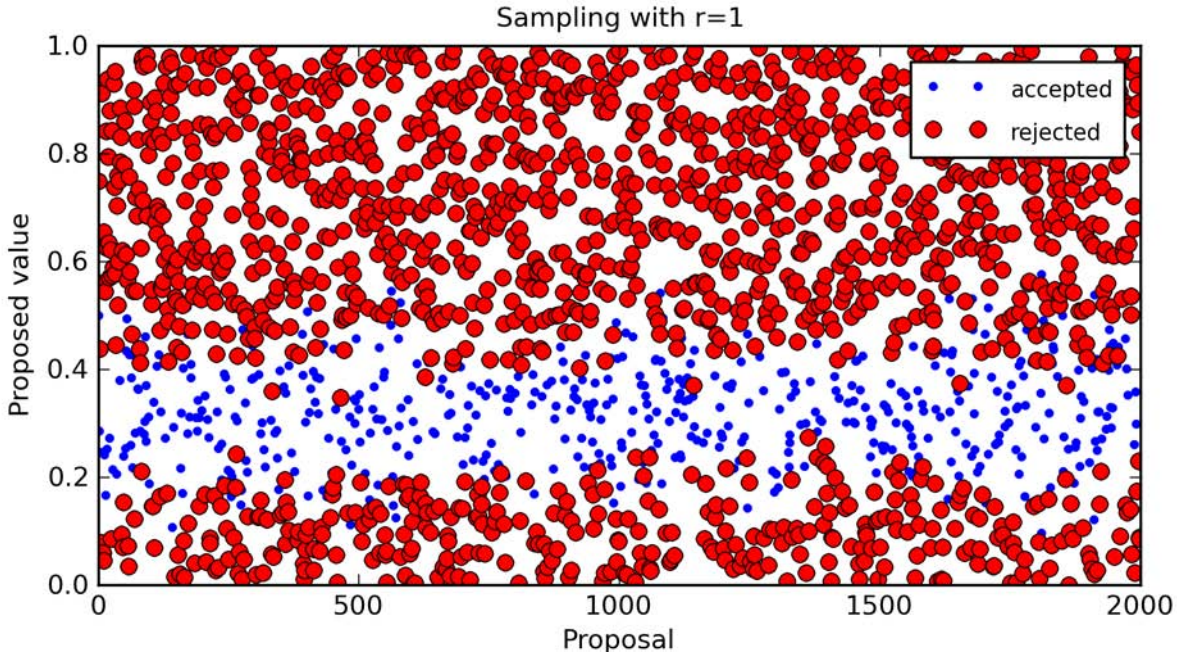
$$\begin{aligned} \alpha(\theta^{t-1}, \theta^*) &= \min\left\{ \frac{p(y|\theta^*)p(\theta^*)J(\theta^*, \theta^{t-1}; r)}{p(y|\theta^{t-1})p(\theta^{t-1})J(\theta^{t-1}, \theta^*; r)}, 1 \right\} \\ &= \min\left\{ \frac{p(y|\theta^*)p(\theta^*)}{p(y|\theta^{t-1})p(\theta^{t-1})}, 1 \right\} \\ &= \min\left\{ \frac{(\theta^*)^{m_H+\alpha-1}(1-\theta^*)^{m-m_H+\beta-1}}{(\theta^{t-1})^{m_H+\alpha-1}(1-\theta^{t-1})^{m-m_H+\beta-1}}, 1 \right\}, \end{aligned}$$

which we can easily compute. This formula and a uniform random number generator for the proposal distribution are all that is required to implement the Metropolis-Hastings algorithm.

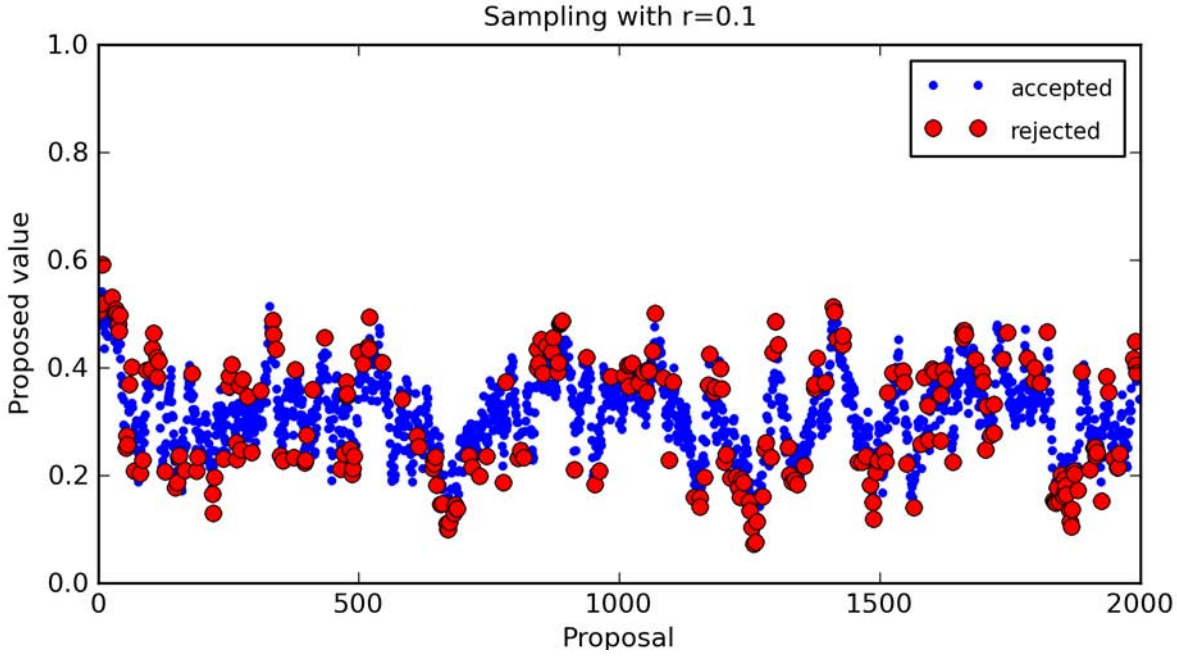
Consider the specific case of $m = 25$, $m_H = 6$, and $\alpha = \beta = 5$. The following three figures show the time sequence of proposals θ^* for chains with $r = 0.01$, $r = 0.1$, and $r = 1$ respectively, with the colors indicating whether each proposed θ^* was accepted or not, and time along the x-axis. In this first figure we see that with $r = 0.01$, the step sizes are too small and after 2000 proposals we have not reached the stationary distribution.



In the next figure, with $r = 1$ the steps are too large and we reject most of the proposals. This leads to a small number of accepted draws after 2000 proposals.



The chain with $r = 0.1$ is the happy medium, where we rapidly reach the stationary distribution, and accept most of the samples.

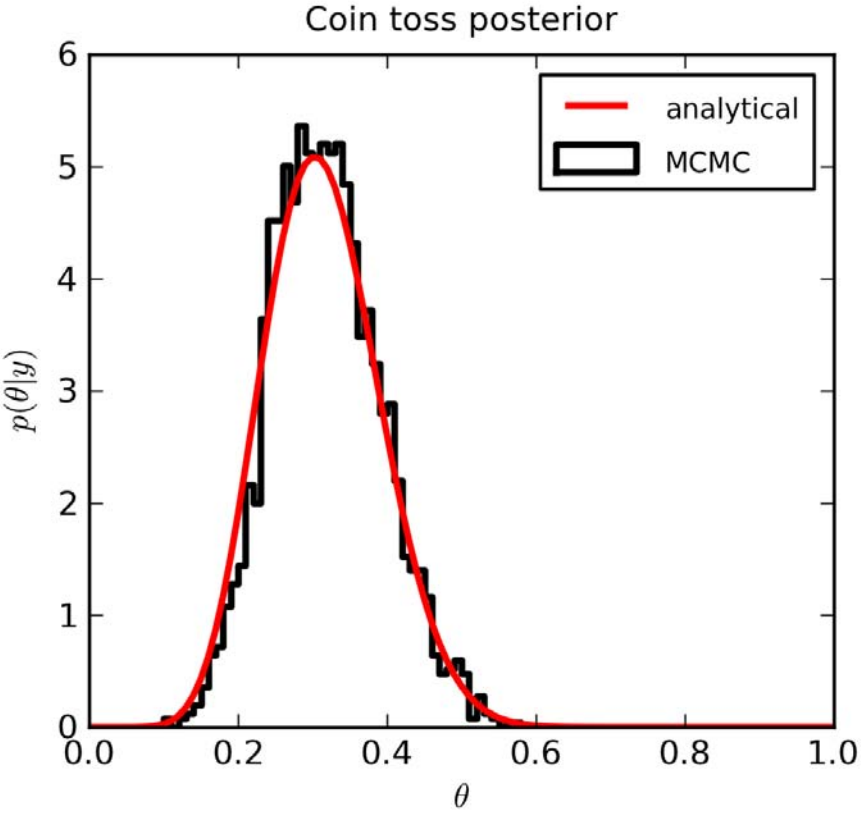


To compare this to the analytic distribution that we obtained in (14), namely

$$p(\theta|y) \sim \text{Beta}\left(m_H + \alpha, m - m_H + \beta\right).$$

we run a chain with $r = 0.1$ until we have collected 25,000 accepted draws. We then discard the initial 200 samples (burn-in) and keep one out of every 100 samples from what remains (thinning). The next figure shows a normalized

histogram of the resulting draws, along with the analytical posterior. The running time to generate the MCMC draws was less than a second, and they are a reasonable approximation to the posterior.



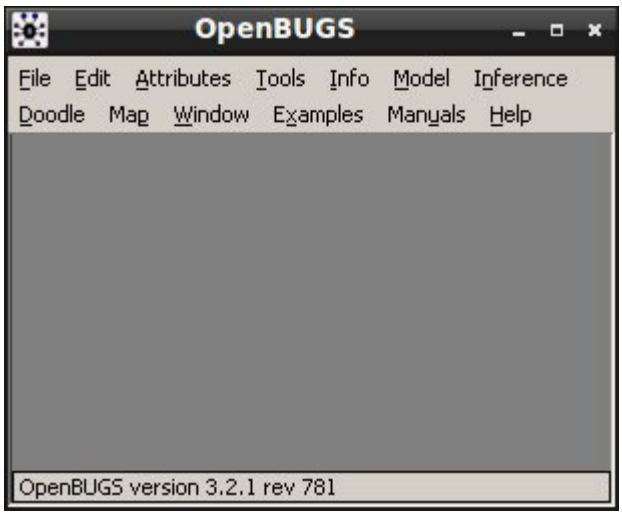
7 MCMC with OpenBUGS

There is a great deal of "art" to MCMC simulation, and a large body of research on the "right" way to do the simulations. OpenBUGS is a nice software package that has built-in years of research in MCMC and can draw posterior samples in a fairly automated way. It is great for doing Bayesian analysis without having to get your hands too dirty with MCMC details. Here we demonstrate OpenBUGS using two examples.

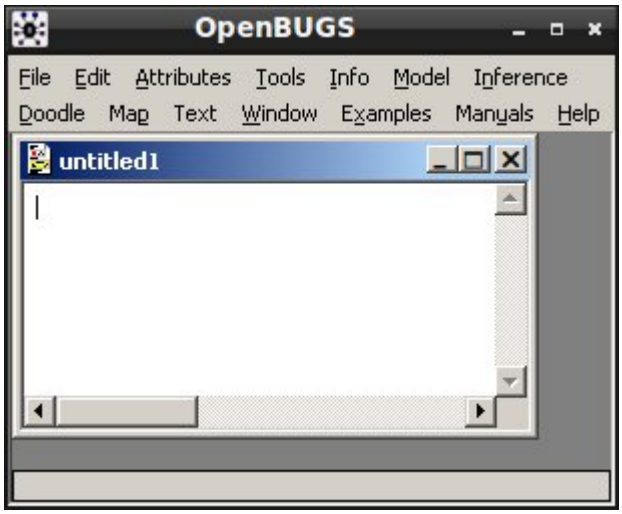
OpenBUGS is old and is infrequently updated, but is still functional and powerful. The graphical interface version of OpenBugs is available only as a Windows executable, but for Linux and Mac users, we had no difficulties running the Windows executable with Wine.

7.1 OpenBUGS example 1: Coin flip example

By this point it should come as no surprise that our first example will be the coin flip example.



When you first open OpenBUGS, you will be greeted with a screen that looks something like this.



`File -> New` will open what is essentially a blank text file. There are three items that must be specified to do simulations in OpenBUGS: The model, the data, and initial conditions. All of these are to be specified in this text file.

Specifying the model in OpenBUGS is very natural, and uses two main symbols: "`<-`" means deterministic assignment, as in R, and "$\sim$" means distributional assignment. OpenBUGS includes many different distributions, a full list of how to specify them is at:

http://mathstat.helsinki.fi/openbugs/Manuals/ModelSpecification.html#ContentsAI

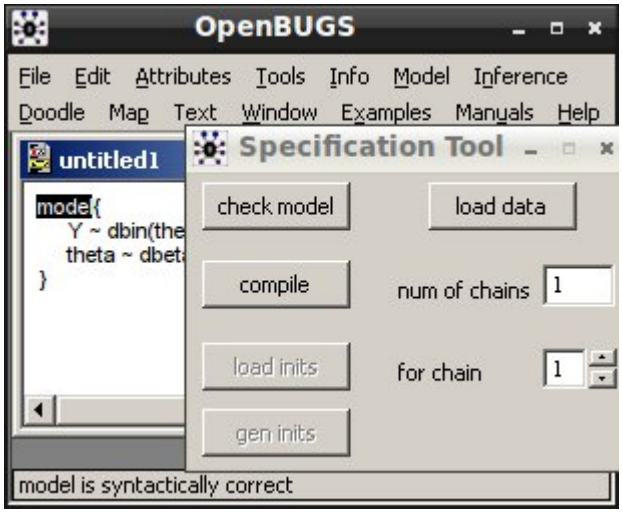
We specify the coin flip model as:

```
model{
Y ~ dbin(theta,m)
theta ~ dbeta(alpha,beta)
```

}

Here dbin and dbeta are the binomial and beta distributions respectively. Notice that in the model we do not specify the constants m, α, and β.

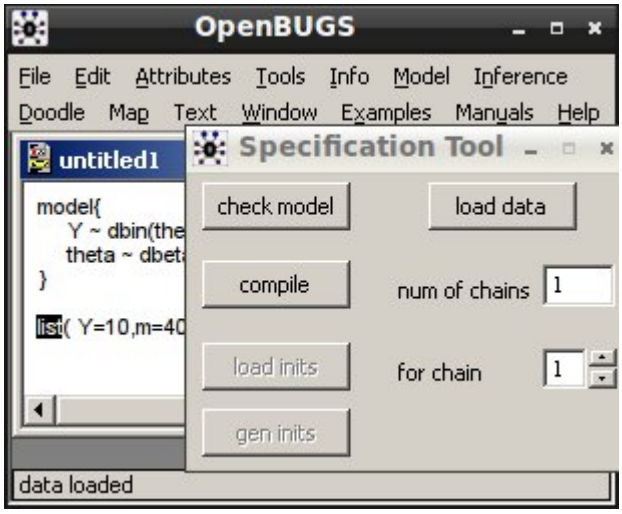
To load the model into OpenBUGS, we first enter it into the text entry space in OpenBUGS. Then from the "`Model`" menu, we go `Model -> Specification`. This opens the model specification window, seen to the right. To set the model as the current working model, we highlight the word "`model`" that we typed in, as in the image on the right, and while it is highlighted we select "`check model`." When we do this, the text "`model is syntatically correct`" appears along the bottom.



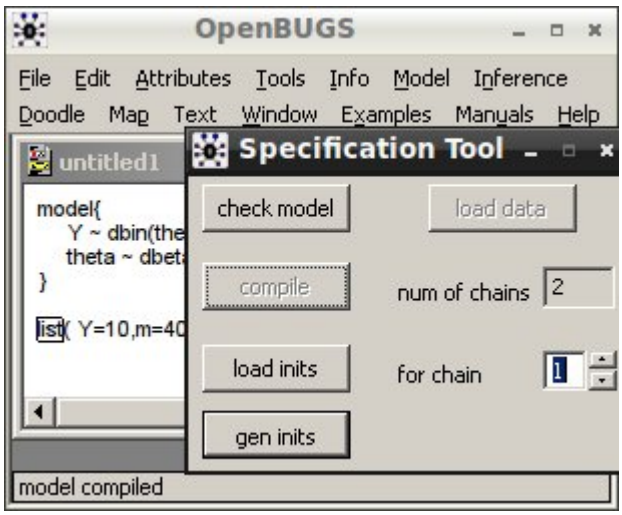
We then specify the data, including all constants. Data is specified in OpenBUGS as an R list. Here, for 10 Heads out of 40 flips and priors $\alpha = \beta = 5$, we write:

```
list(Y=10,m=40,alpha=5,beta=5)
```

To load the data into OpenBUGS, we first type the data list into the text entry space in OpenBUGS. We then highlight the word "`list`", as in the image on the right, and while it is highlighted we click "`load data`" in the model specification window. When we do this, "`data loaded`" appears along the bottom of the screen.



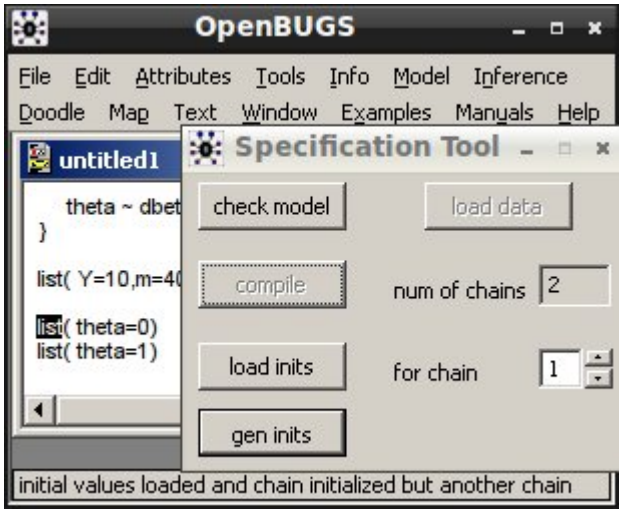
We are now ready to specify the number of chains and compile the model. We use multiple chains from multiple initial conditions to test convergence; it is usually sufficient to use 2 chains. We input 2 chains and press "`compile`" in the model specification window. Then, "`model compiled`" appears along the bottom. If we did not specify one of the variables (for instance, if we had mistakenly left beta out of the data list) then it will give an error when we try to compile.



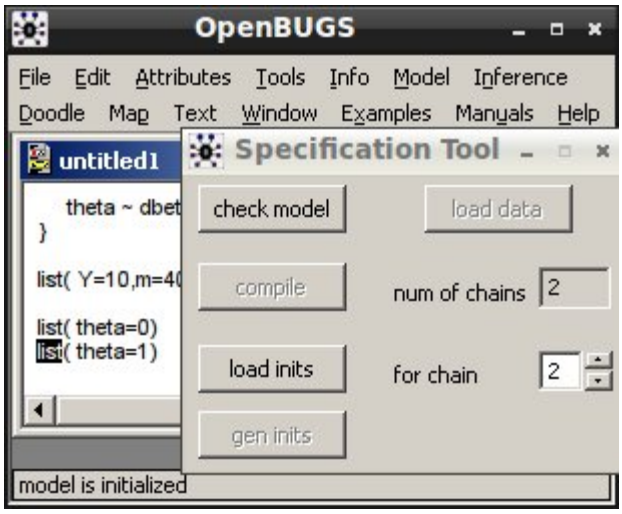
Finally, we specify the initial parameter values for each chain, each as an R list. Here we will start one chain from $\theta = 1$, and the other from $\theta = 0$.

```
list(theta=0)
list(theta=1)
```

For the first initial value we highlight the word "`list`", and click "`load inits`" in the model specification. The number scroll to the right of the "`load inits`" button indicates that we are loading the initial values for chain 1. When we do this, it says on the bottom of the screen "`initial values loaded and chain initialized but another chain`," referring to the fact that chain 2 still has not been initialized.



To initialize chain 2, we simply follow the same procedure with the second list, and with the number scroll set to 2. At the bottom of the screen it then says "model initialized."

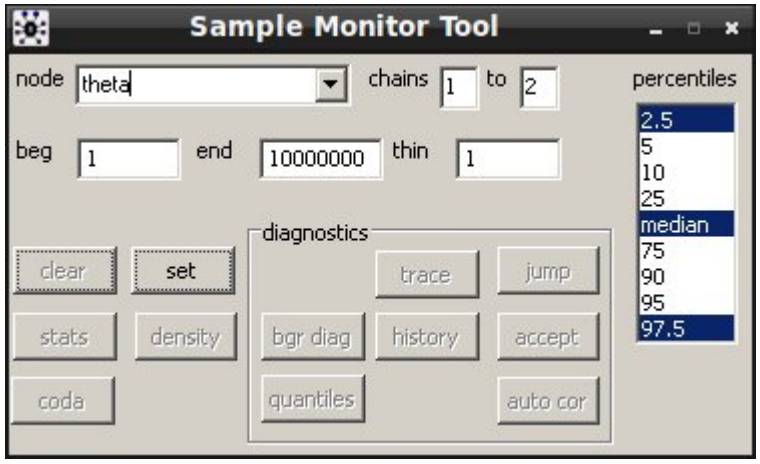


Alternatively, instead of loading intial values for both chains, we could have pressed the button "`gen inits`" in the model specification window, which generates random initial values for both chains.

We may now close the model specification window.

OpenBUGS will only store whatever results we tell it to. Before simulating the chains, we must tell OpenBUGS to store the values of theta that it simulates.

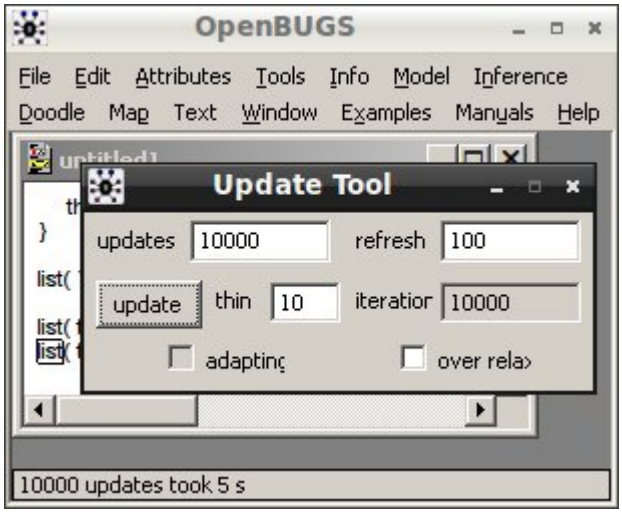
To specify which variables to store, go `Inference -> Samples`. Here we set "`nodes`," which is the OpenBUGS term for stored variables. To tell it to store theta, we type "`theta`" into the node entry box, and press "`set`" below. The other entry boxes will be used for things like burn-in and thinning, which we will worry about later.



Now we are ready to simulate the chains! We may close the sample monitor

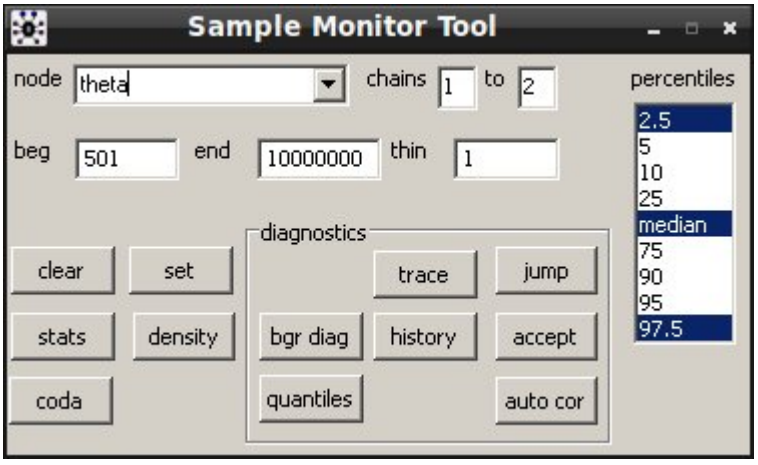
tool for now.

To simulate the chains, we go `Model -> Update`. This opens the update tool which we use to run the simulations. We simply enter the desired number of draws (we choose 10,000), and the amount of thinning (we enter 10, meaning keep only one out of every 10 draws). The "`refresh`" entry refers to how often it will refresh the "iteration" count during the simulation and is not very important. When we have entered in the desired parameters, we press "`update`". The iteration count increases from 0 to 10,000, and finally at the bottom of the screen it reports "`10000 updates took 5 s.`"

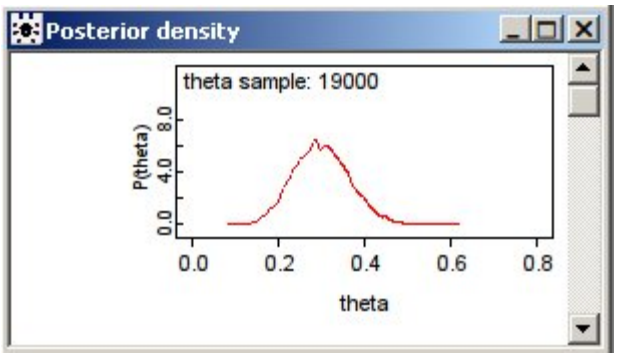


To view the results, we close the update tool and return to the sample monitor tool (`Inference -> Samples`). Here there are a few useful quantities to look at.

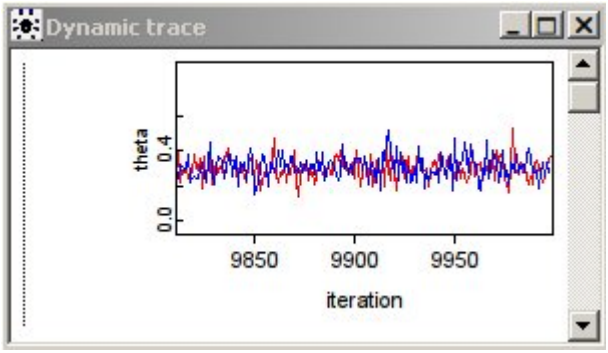
We type in "`theta`" as the node we are interested in. We set "`501`" in the "`beg`" entrybox to specify that we will begin inference at interation 501, thus ignoring the first 500 iterations as burn-in. The buttons we are most interested in are density, coda, trace, history, and accept.



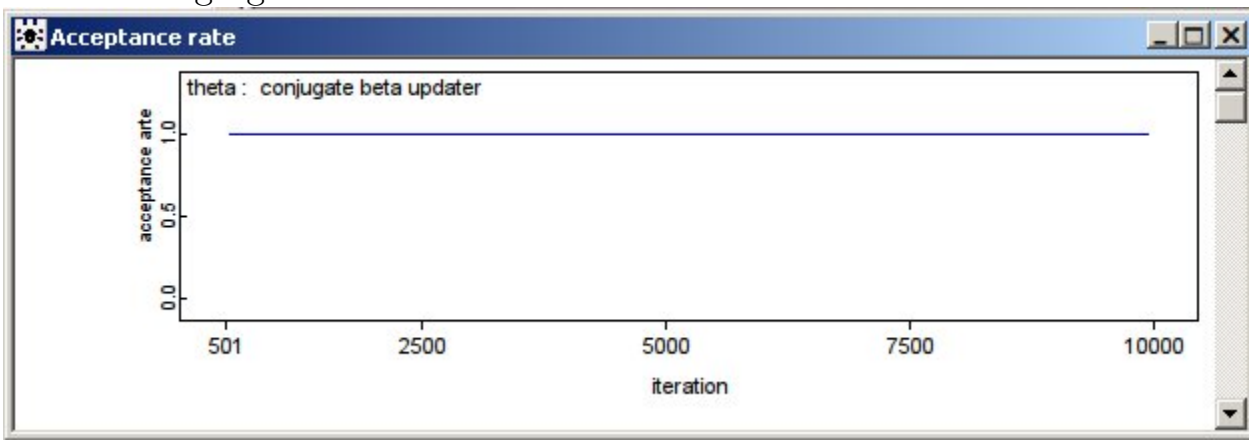
Pressing "`density`" produces the MCMC estimte of the posterior density, what we are most interested in. It specifies that the density was created using 19,000 samples (10,000 for each chain minus 500 burn-in).



Pressing "`trace`" shows the values for the last 200 draws for each chain, so we can verify that the chains are stationary and that they are similar as a check of convergence. This image is shown on the right. Pressing "`history`" provides a similar plot, with all 10,000 iterations instead of only the last 200. Pressing "`coda`" dumps the values at each iteration to a text file which you can then load into R for further analysis.



Pressing "`accept`" shows the acceptance rate vs. iteration number, which is the following figure.



OpenBUGS is smart enough to notice the Binomial-Beta conjugacy and uses a Gibbs' sampler, hence the acceptance rate is uniformly 1.

7.2 OpenBUGS example 2: estimating pump failure rates

The following example is from the OpenBUGS manual, and can be found at:

http://www.openbugs.info/Examples/Pumps.html

This example is a hierarchical model for pump failure. There are 10 pumps, pump i being in operation for a certain amount of time t_i and having had a certain number of failures x_i. We wish to estimate the failure rate for each pump, and we will do so with a Bayesian hierarchical model. We suspect that the failure rates are related for all of the pumps, so we include a common prior. We give flexibility to the prior by sampling it from a diffuse distribution. The generative model for the observed number of pump failures is then:

1. Choose the first prior hyperparameter $\alpha \sim \text{Exponential}(1)$.
2. Choose the second prior hyperparameter $\beta \sim \text{Gamma}(0.1, 1.0)$.
3. For pump $i = 1, \ldots, 10$:

   Choose this pump's failure rate $\theta_i \sim \text{Gamma}(\alpha, \beta)$.

   Choose the number of failures $x_i \sim \text{Poisson}(\theta_i t_i)$.

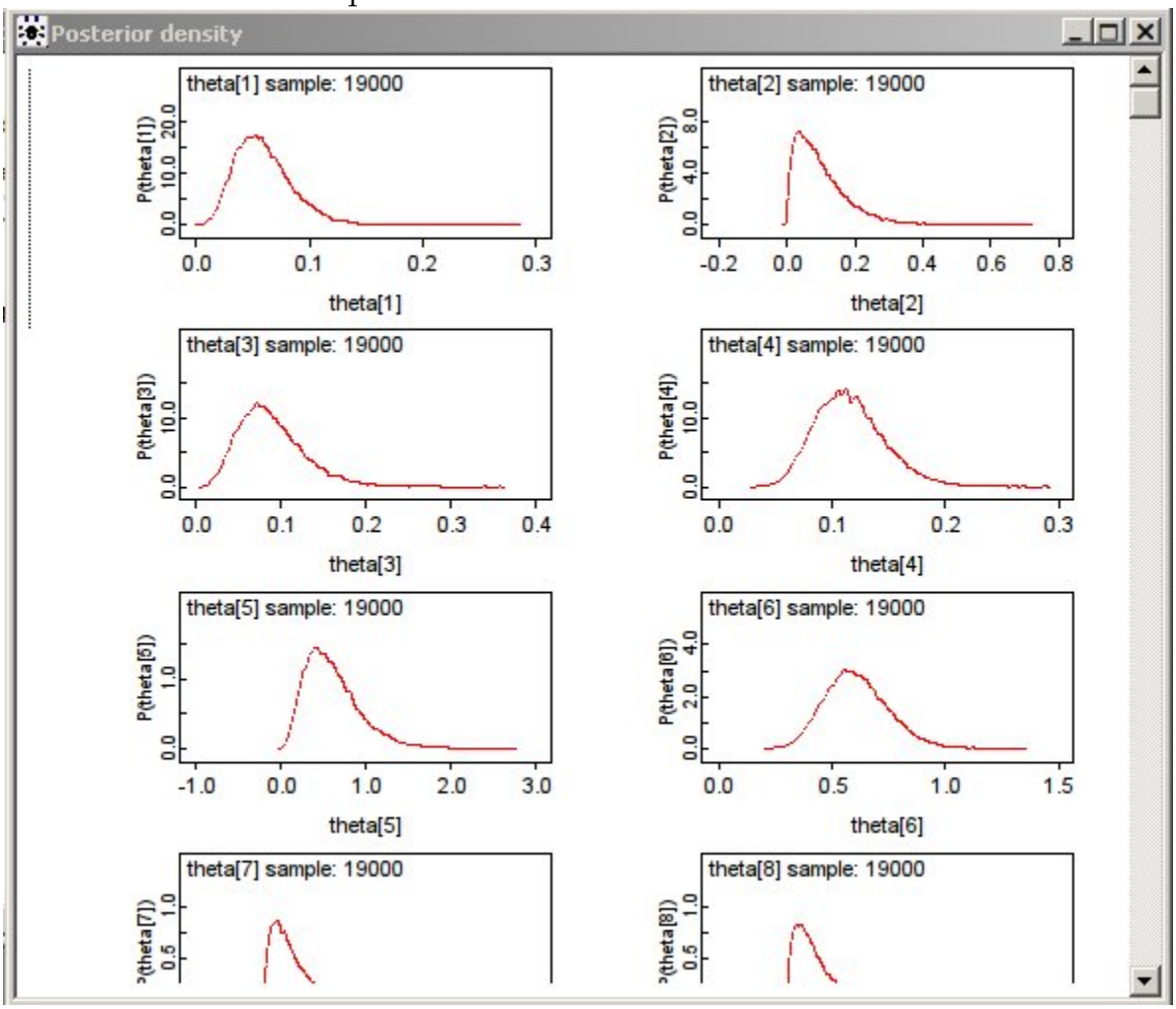
We input this model into OpenBUGS as

```
model{
alpha ~ dexp(1)
beta ~ dgamma(0.1, 1.0)
for (i in 1 :  10) {
theta[i] ~ dgamma(alpha, beta)
lambda[i] <- theta[i]*t[i]
x[i] ~ dpois(lambda[i])
}
}
```

Notice in the above that OpenBUGS does not allow mathematical operations inside of calls to distributions like `dpois(theta[i]*t[i])`, so we had to use a deterministic relationship to specify the new variable `lambda[i]`. The data to be specified are "`t`" and "`x`", and we do this with an R list:

```
list(t = c(94.3, 15.7, 62.9, 126, 5.24, 31.4, 1.05, 1.05, 2.1, 10.5),
x = c( 5, 1, 5, 14, 3, 19, 1, 1, 4, 22))
```

We will allow OpenBUGS to generate the initial values for us randomly. We simulate two chains exactly as in the coin flip example, and easily recover the posterior densities for the failure rates for each pump. In the following figure we show some of the posterior distributions:



15.097 Prediction: Machine Learning and Statistics
Spring 2012