Why is computation important?

- Bayesian inference centres around the posterior distribution

$$p(\boldsymbol{\theta}|x) \propto p(x|\boldsymbol{\theta}) \times p(\boldsymbol{\theta})$$

  where $\boldsymbol{\theta}$ is typically a large vector of parameters $\boldsymbol{\theta} = \{\theta_1, \theta_2, ..., \theta_k\}$

- $p(x|\boldsymbol{\theta})$ and $p(\boldsymbol{\theta})$ will often be available in closed form, but $p(\boldsymbol{\theta}|x)$ is usually not analytically tractable, and we want to
  - obtain marginal posterior $p(\theta_i|x) = \int\int ... \int p(\boldsymbol{\theta}|x)\, d\boldsymbol{\theta}_{(-i)}$ where $\boldsymbol{\theta}_{(-i)}$ denotes the vector of θ's excluding θ_i
  - calculate properties of $p(\theta_i|x)$, such as mean $(= \int \theta_i p(\theta_i|x) d\theta_i)$, tail areas $(= \int_T^\infty p(\theta_i|x) d\theta_i)$ etc.

$\rightarrow$ numerical integration becomes vital

Monte Carlo integration

We have already seen that Monte Carlo methods can be used to simulate values from prior distributions and from **closed form** posterior distributions

If we had algorithms for sampling from arbitrary (typically high-dimensional) posterior distributions, we could use Monte Carlo methods for Bayesian estimation:

- Suppose we can draw samples from the joint posterior distribution for $\boldsymbol{\theta}$, *i.e.*

$$(\theta_1^{(1)}, ..., \theta_k^{(1)}), (\theta_1^{(2)}, ..., \theta_k^{(2)}), ..., (\theta_1^{(N)}, ..., \theta_k^{(N)}) \quad \sim \quad p(\boldsymbol{\theta}|x)$$

- Then
  - $\theta_1^{(1)}, ..., \theta_1^{(N)}$ are a sample from the marginal posterior $p(\theta_1|x)$
  - $E(g(\theta_1)) = \int g(\theta_1)p(\theta_1|x)d\theta_1 \approx \frac{1}{N}\sum_{i=1}^{N} g(\theta_1^{(i)})$

$\rightarrow$ this is Monte Carlo integration

$\rightarrow$ theorems exist which prove convergence in limit as $N \rightarrow \infty$ even if the sample is dependent (crucial to the success of MCMC)

How do we sample from the posterior?

- We want samples from joint posterior distribution $p(\boldsymbol{\theta}|x)$
- *Independent* sampling from $p(\boldsymbol{\theta}|x)$ may be difficult
- **BUT** *dependent* sampling from a *Markov chain* with $p(\boldsymbol{\theta}|x)$ as its stationary (equilibrium) distribution is easier
- A sequence of random variables $\theta^{(0)}, \theta^{(1)}, \theta^{(2)}, ...$ forms a Markov chain if $\theta^{(i+1)} \sim p(\theta|\theta^{(i)})$ *i.e.* conditional on the value of $\theta^{(i)}$, $\theta^{(i+1)}$ is independent of $\theta^{(i-1)}, ..., \theta^{(0)}$
- Several standard 'recipes' available for designing Markov chains with required stationary distribution $p(\boldsymbol{\theta}|x)$
  - Metropolis *et al.* (1953); generalised by Hastings (1970)
  - **Gibbs Sampling** (see Geman and Geman (1984), Gelfand and Smith (1990), Casella and George (1992)) is a special case of the Metropolis-Hastings algorithm which generates a Markov chain by sampling from **full conditional distributions**
  - See Gilks, Richardson and Spiegelhalter (1996) for a full introduction and many worked examples.

Gibbs sampling

Let our vector of unknowns $\boldsymbol{\theta}$ consist of k sub-components $\boldsymbol{\theta} = (\theta_1, \theta_2, ..., \theta_k)$

1) Choose starting values $\theta_1^{(0)}$, $\theta_2^{(0)}$, ..., , $\theta_k^{(0)}$

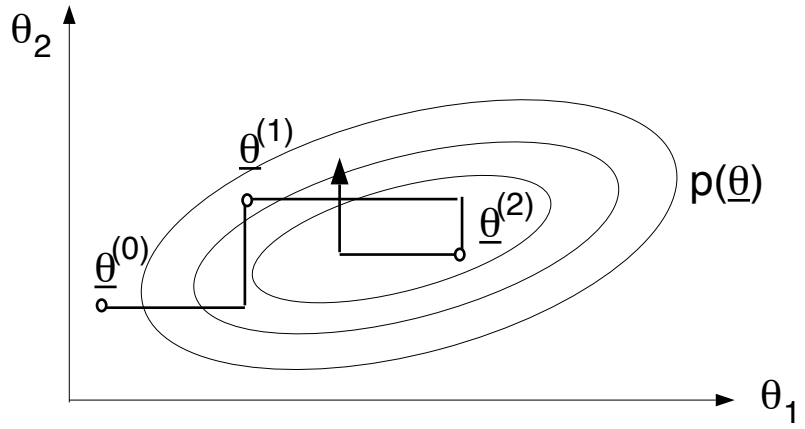
2) Sample $\theta_1^{(1)}$ from $p(\theta_1|\theta_2^{(0)}, \theta_3^{(0)}, ..., , \theta_k^{(0)}, x)$
   Sample $\theta_2^{(1)}$ from $p(\theta_2|\theta_1^{(1)}, \theta_3^{(0)}, ..., , \theta_k^{(0)}, x)$
   .....
   Sample $\theta_k^{(1)}$ from $p(\theta_k|\theta_1^{(1)}, \theta_2^{(1)}, ..., , \theta_{k-1}^{(1)}, x)$

3) Repeat step 2 many 1000s of times
   – eventually obtain sample from $p(\boldsymbol{\theta}|x)$

The conditional distributions are called 'full conditionals' as they condition on all other parameters

Gibbs sampling ctd.

Example with $k = 2$



- Sample $\theta_1^{(1)}$ from $p(\theta_1|\theta_2^{(0)}, x)$
- Sample $\theta_2^{(1)}$ from $p(\theta_2|\theta_1^{(1)}, x)$
- Sample $\theta_1^{(2)}$ from $p(\theta_1|\theta_2^{(1)}, x)$
- .....

$\boldsymbol{\theta}^{(n)}$ forms a Markov chain with (*eventually*) a stationary distribution $p(\boldsymbol{\theta}|x)$.

Using MCMC methods

There are two main issues to consider

- Convergence (how quickly does the distribution of $\boldsymbol{\theta}^{(t)}$ approach $p(\boldsymbol{\theta}|x)$?)
- Efficiency (how well are functionals of $p(\boldsymbol{\theta}|x)$ estimated from $\{\boldsymbol{\theta}^{(t)}\}$?)

Checking convergence

This is the users responsibility!

- Note: Convergence is to target **distribution** (the required posterior), not to a single value.

- Once convergence reached, samples should look like a random scatter about a stable mean value

Convergence diagnosis

- How do we know we have reached convergence?
- *i.e.* How do we the know number of 'burn-in' iterations?
- Many 'convergence diagnostics' exist, but none foolproof
- CODA and BOA software contain large number of diagnostics

Gelman-Rubin-Brooks diagnostic

- A number of runs
- Widely differing starting points
- Convergence assessed by quantifying whether sequences are much further apart than expected based on their internal variability
- Diagnostic uses components of variance of the multiple sequences

Example: A dose-response model

Consider the following response rates for different doses of a drug

dose x_i	No. subjects n_i	No. responses r_i
1.69	59	6
1.72	60	13
1.75	62	18
1.78	56	28
1.81	63	52
1.83	59	53
1.86	62	61
1.88	60	60

Fit a logistic curve with 'centred' covariate $(x_i - \overline{x})$:

$$\begin{aligned} r_i &\sim \text{Bin}(p_i, n_i) \\ \text{logit } p_i &= \alpha + \beta(x_i - \overline{x}) \\ \alpha &\sim \text{N}(0, 10000) \\ \beta &\sim \text{N}(0, 10000) \end{aligned}$$

Checking convergence with multiple runs

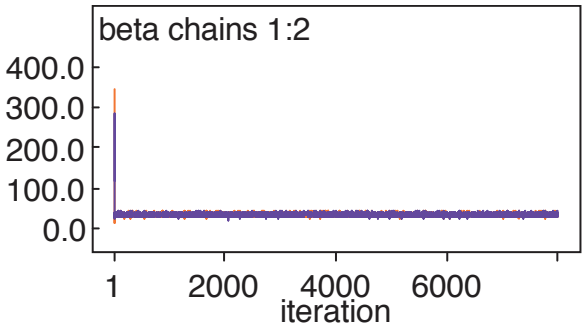
- Set up multiple initial value lists, *e.g.*
  ```
  list(alpha=-100, beta=100)
  list(alpha=100, beta=-100)
  ```
- Before clicking *compile*, set *num of chains* to 2
- Load both sets of initial values
- Monitor from the start of sampling
- Assess how much burn-in needed using the *bgr* statistic

Using the *bgr* statistic

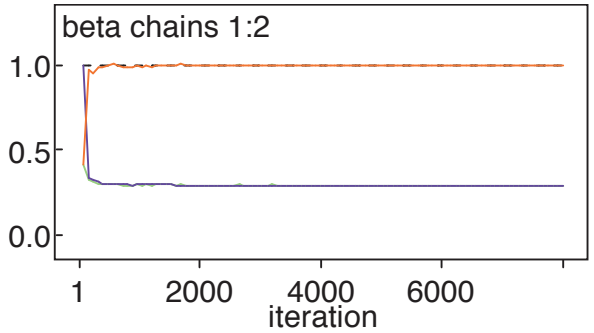
- *Green*: width of 80% intervals of pooled chains: should be stable
- *Blue*: average width of 80% intervals for chains: should be stable
- *Red*: ratio of pooled/within: should be near 1
- Double-click on plot, then *cntl* + right click gives statistics

Output for 'centred' analysis

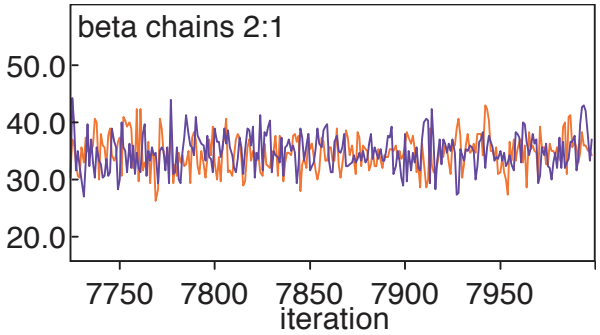
history



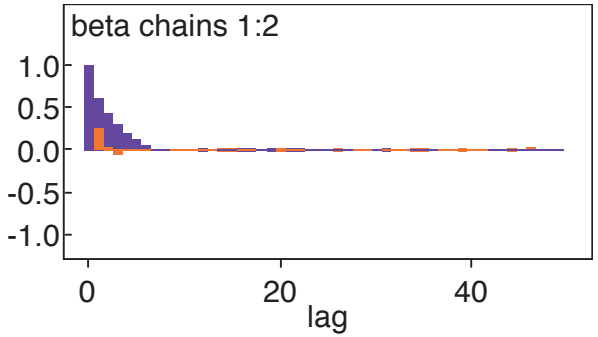
bgr diagnostic



trace



autocorrelation



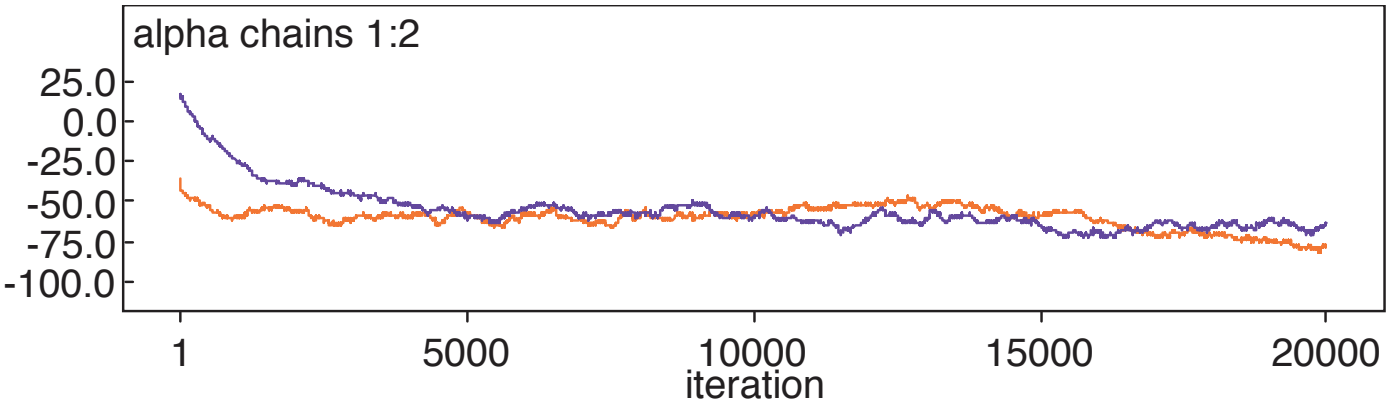
node	mean	sd	MC error	2.5%	median	97.5%	start	sample
alpha	0.7489	0.139	0.00138	0.4816	0.7468	1.026	1001	14000
beta	34.6	2.929	0.02639	29.11	34.53	40.51	1001	14000

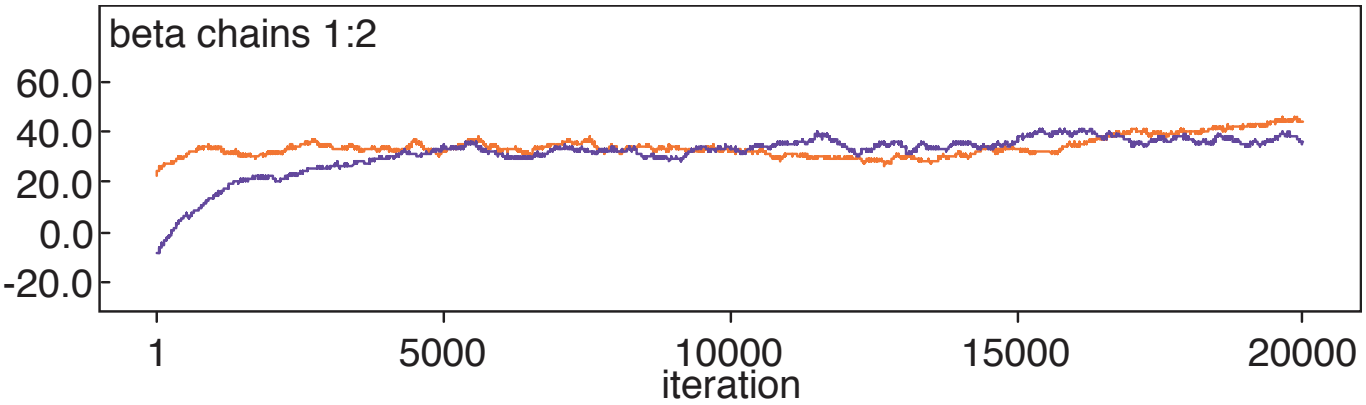
Problems with convergence

Fit a logistic curve with 'un-centred' covariate x:

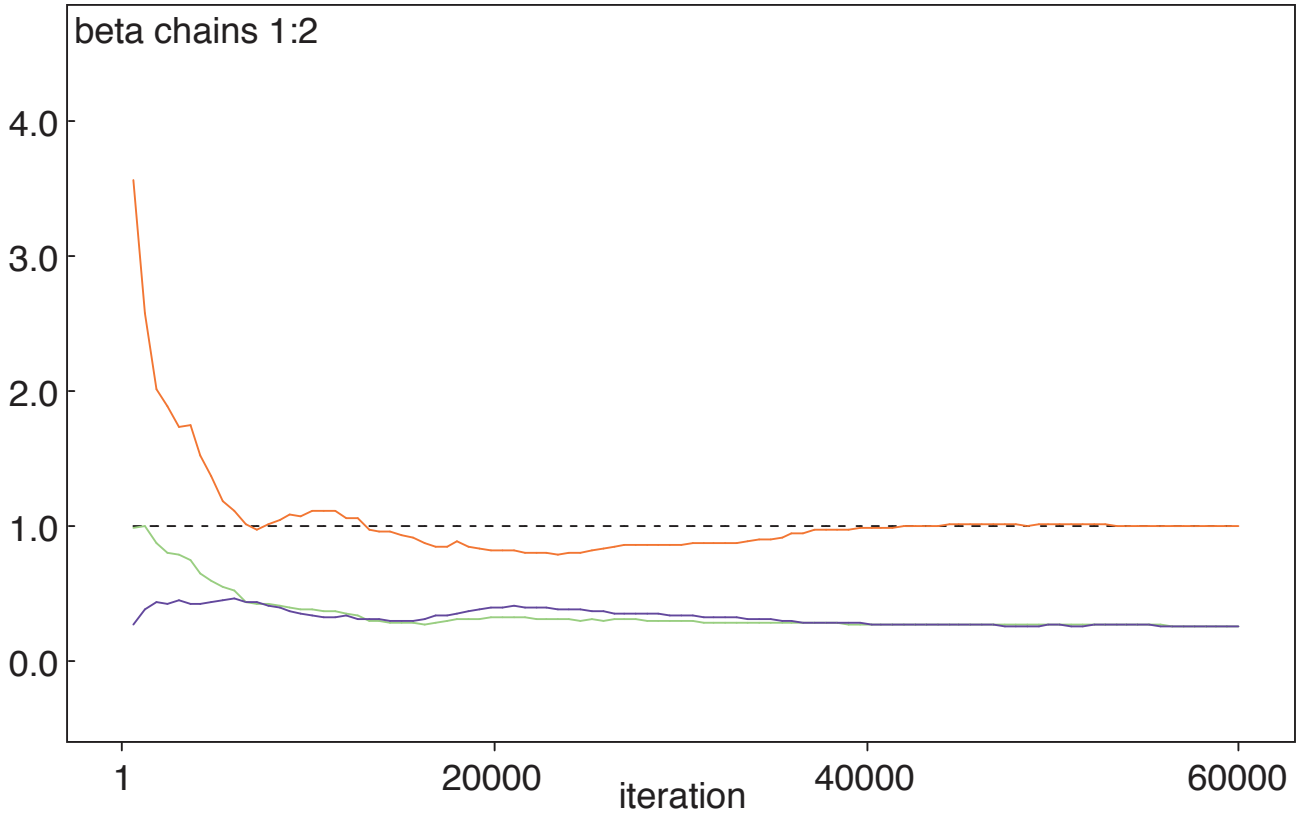
$$\begin{aligned} r_i &\sim \text{Bin}(p_i, n_i) \\ \text{logit } p_i &= \alpha + \beta x_i \\ \alpha &\sim \text{N}(0, 10000) \\ \beta &\sim \text{N}(0, 10000) \end{aligned}$$

History plots for ‘un-centred’ analysis





bgr output for 'un-centred' analysis

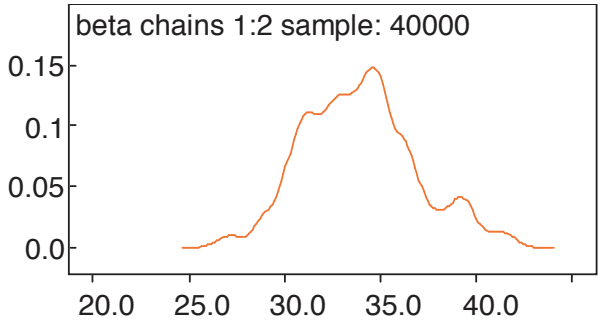


Drop first 40,000 iterations as burn-in

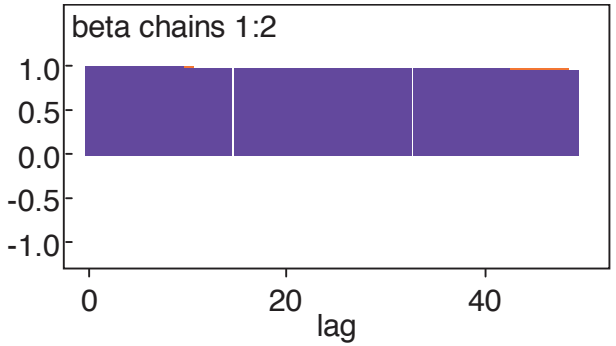
node	mean	sd	MC error	2.5%	median	97.5%	start	sample
beta	33.97	2.955	0.1734	28.7	33.89	40.3	40001	40000

Output for 'un-centred' analysis

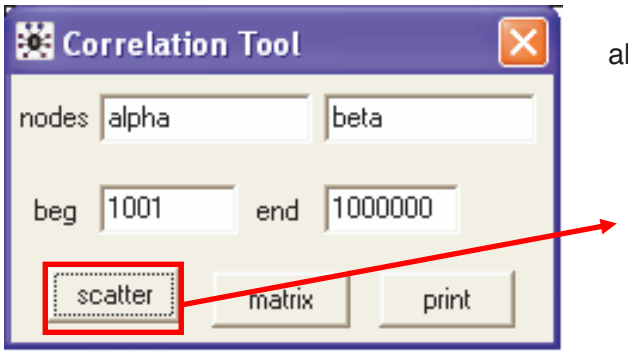
posterior density



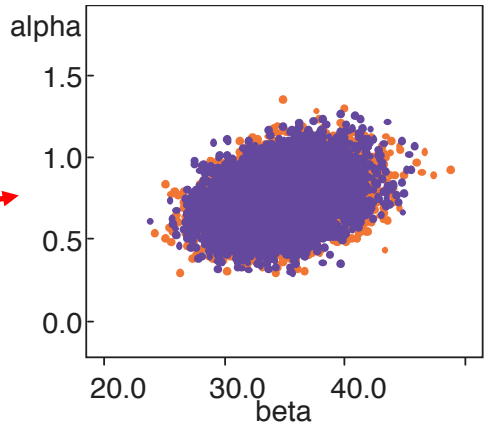
autocorrelation



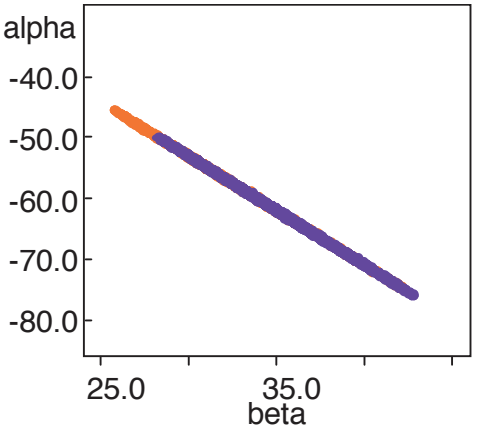
bivariate posteriors



centred



un-centred



How many iterations after convergence?

- After convergence, further iterations are needed to obtain samples for posterior inference.
- More iterations = more accurate posterior estimates.
- Efficiency of sample mean of θ as estimate of theoretical posterior expectation $E(\theta)$ usually assessed by calculating Monte Carlo standard error (MC error)
- MC error = standard error of posterior sample mean as estimate of theoretical expectation for given parameter
- MC error depends on
  - true variance of posterior distribution
  - posterior sample size (number of MCMC iterations)
  - autocorrelation in MCMC sample
- Rule of thumb: want MC error $< 1 - 5\%$ of posterior SD

Inference using posterior samples from MCMC runs

A powerful feature of the Bayesian approach is that all inference is based on the joint posterior distribution

$\Rightarrow$ can address wide range of substantive questions by appropriate summaries of the posterior

- Typically report either mean or median of the posterior samples for each parameter of interest as a point estimate

- 2.5% and 97.5% percentiles of the posterior samples for each parameter give a 95% posterior credible interval (interval within which the parameter lies with probability 0.95)

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
beta	34.60	2.929	0.0239	29.11	34.53	40.51	1001	14000

So point estimate of `beta` would be 34.60, with 95% credible interval (29.11, 40.51)

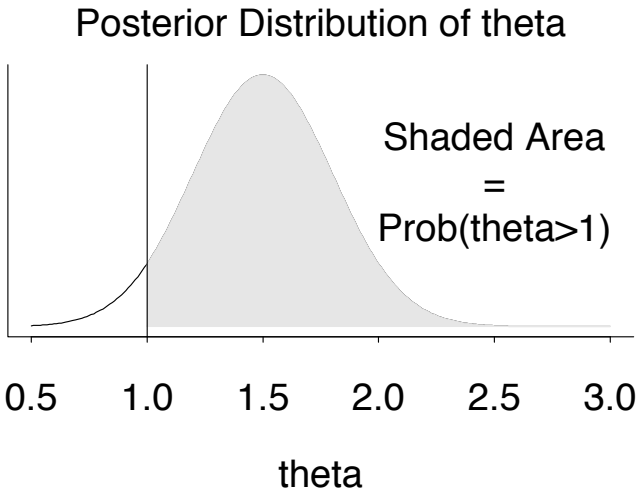
Probability statements about parameters

- Classical inference cannot provide probability statements about parameters (e.g. p-value is not Pr(H_0 true), but probability of observing data as or more extreme than we obtained, given that H_0 is true)

- In Bayesian inference, it is simple to calculate e.g. Pr($\theta > 1$):

  = Area under posterior distribution curve to the right of 1

  = Proportion of values in posterior sample of θ which are > 1



- In `WinBUGS` use the `step` function:
  `p.theta <- step(theta - 1)`

- For discrete parameters, may also be interested in Pr($\delta = \delta_0$):
  `p.delta <- equals(delta, delta0)`

- Posterior means of `p.theta` and `p.delta` give the required probabilities

Complex functions of parameters

- Classical inference about a function of the parameters $g(\theta)$ requires construction of a specific estimator of $g(\theta)$. Obtaining appropriate error can be difficult.

- Easy using MCMC: just calculate required function $g(\theta)$ as a logical node at each iteration and summarise posterior samples of $g(\theta)$

In dose-response example, suppose we want to estimate the $ED95$: that is the dose that will provide 95% of maximum efficacy.

$$\begin{aligned} \text{logit}\,0.95 &= \alpha + \beta(ED95 - \overline{x}) \\ ED95 &= (\text{logit}\,0.95 - \alpha)/\beta + \overline{x} \end{aligned}$$

Simply add into model

```
ED95 <- (logit(0.95) - alpha)/beta + mean(x[])
```

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
ED95	1.857	0.007716	8.514E-5	1.843	1.857	1.874	1001	10000