Metropolis-Hastings Algorithm

Strength of the Gibbs sampler

- ○ Easy algorithm to think about.
- ○ Exploits the factorization properties of the joint probability distribution.
- ○ No difficult choices to be made to tune the algorithm

Weakness of the Gibbs sampler

- ○ Can be difficult (impossible) to sample from full conditional distributions.

Idea: Use acceptance-rejection method instead.

Metropolis-Hastings algorithm

Aim: Construct Markov chain $Y^{(t)}$ with stationary distribution $f(y)$.

At time t, generate next value $Y^{(t+1)}$ in two steps:

- ○ *Proposal step:* Sample "candidate " X from the proposal distribution,

$$Z \sim q(z|Y^{(t)}).$$

- ○ *Acceptance step:* With probability

$$\alpha(Y^{(t)}, Z) = \min\left\{1, \frac{f(Z)}{f(Y^{(t)})} \frac{q(Y^{(t)}|Z)}{q(Z|Y^{(t)})}\right\}$$

set

$$Y^{(t+1)} = Z \qquad \text{(acceptance)}$$

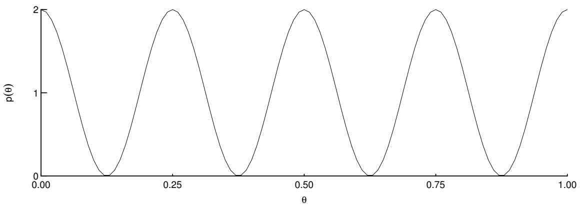
and otherwise set

$$Y^{(t+1)} = Y^{(t)} \qquad \text{(rejection).}$$

Metropolis-Hastings Algorithm

Example: Binomial distribution with non-standard prior

- $Y = (Y_1, \dots, Y_n)^\mathsf{T}$ with $Y_1, \dots, Y_n \overset{\text{iid}}{\sim} \text{Bin}(1, \theta)$
- $S_n = \sum_{i=1}^n Y_i$
- $\pi(\theta) = 2\cos^2(4\pi\theta)$



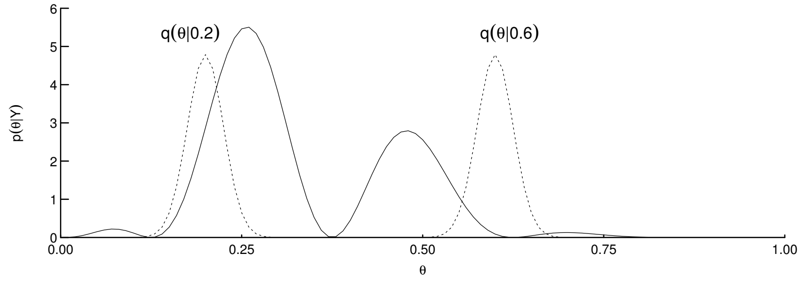
Then the posterior is

$$\begin{aligned}\pi(\theta|Y) &\sim f(Y|\theta)\,\pi(\theta)\\ &= 2\,\theta^{S_n}(1-\theta)^{n-S_n}\cos^2(4\pi\theta)\end{aligned}$$

Metropolis-Hastings Algorithm

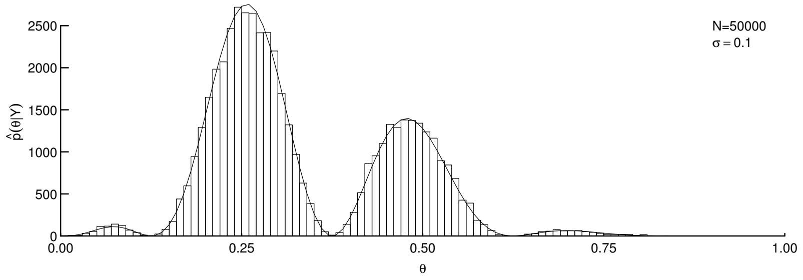
Proposal distribution:

$$q(\theta'|\theta) \sim \exp\left(\frac{1}{2\sigma^2}(\theta - \theta')^2\right)$$



Acceptance probability:

$$\alpha(\theta, \theta') = \min\left\{\frac{\pi(\theta'|Y)\,q(\theta|\theta')}{\pi(\theta|Y)\,q(\theta'|\theta)}\right\} = \min\left\{\frac{\theta'^{S_n}(1-\theta')^{n-S_n}\cos^2(4\pi\theta')}{\theta^{S_n}(1-\theta)^{n-S_n}\cos^2(4\pi\theta)}\right\}$$



Metropolis-Hastings Algorithm

Remarks:

- Suppose we want to sample from the posterior distribution

  $$\pi(\theta|Y) = \frac{f(Y|\theta)\,\pi(\theta)}{f(Y)} \qquad \text{with} \quad f(Y) = \int f(Y|\theta)\,\pi(\theta)\,d\theta.$$

  Then

  $$\frac{\pi(\theta'|Y)}{\pi(\theta|Y)} = \frac{f(Y|\theta')\,\pi(\theta')}{f(Y|\theta)\,\pi(\theta)},$$

  that is, the normalising constant is not required to run the algorithm.

- Usually the proposal distribution q is chosen such that it is easy to sample from it.

- If the proposal distribution is symmetric,

  $$q(z|y) = q(y|z)$$

  we obtain the Metropolis algorithm. In this case

  $$\alpha(Y^{(t)}, Z) = \min\Big\{1, \frac{f(Z)}{f(Y^{(t)})}\Big\}.$$

  Interpretation:

  · Proposal state Z with higher probability are always accepted.

  · Change to state with lower probability possible with probability α.

  Special case: Random-walk Metropolis

  $$q(z|y) = q(|z - y|).$$

- Any density q that has the same support should work.

- HOWEVER: Some distributions are better than others.
  ⇝ tuning problem

Metropolis-Hastings Algorithm

Tuning Metropolis-Hastings

We need to find a good proposal distribution

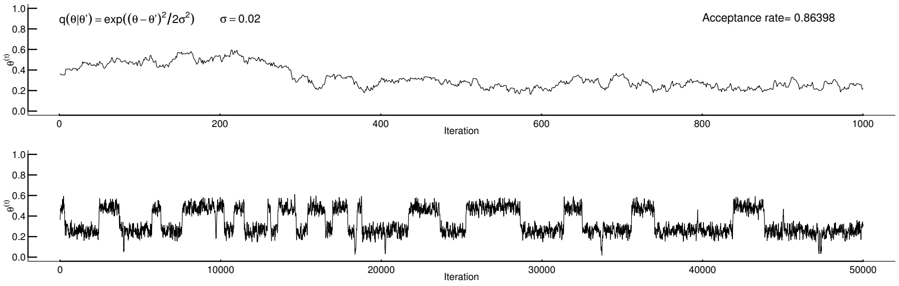
- with high acceptance rate,
- which allows to reach all states frequently (good mixing).

Example: Binomial distribution with non-standard prior

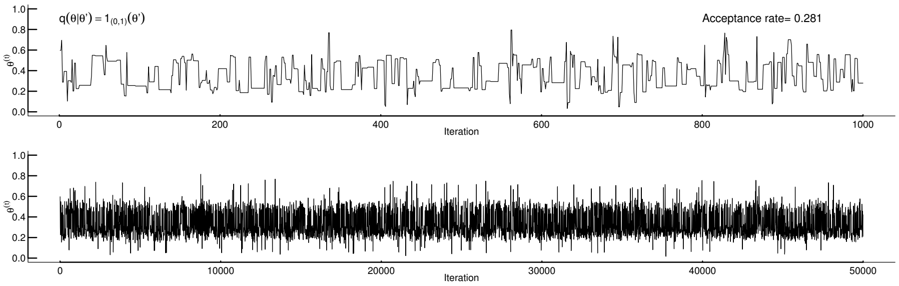
The prososal distribution was

$$q(\theta'|\theta) \sim \exp\left(\frac{1}{2\sigma^2}(\theta-\theta')^2\right).$$

- We can choose the variance σ^2 of the proposal distribution.
- If too small, the chain does not (or only slowly) cover the whole distribution.



- If too big, don't accept very often and jump



Note: Proposal distribution does not depend on θ'
$\rightsquigarrow$ independence sampler

Metropolis-Hastings Algorithm

Example: Bivariate normal distribution

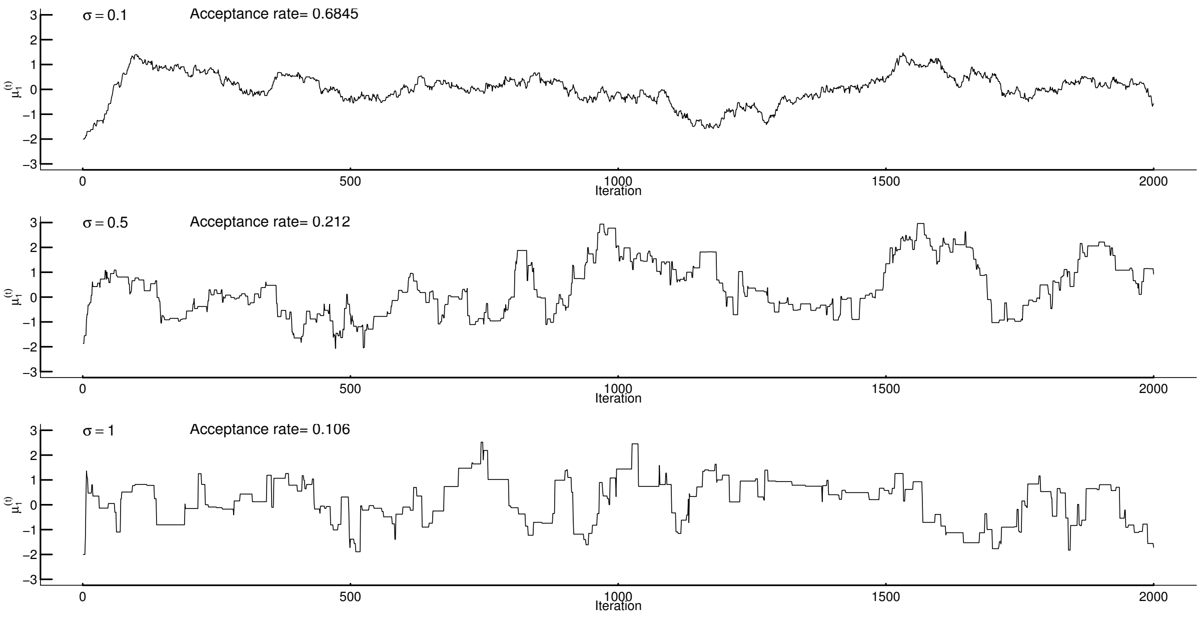
Sample from bivariate normal distribution:

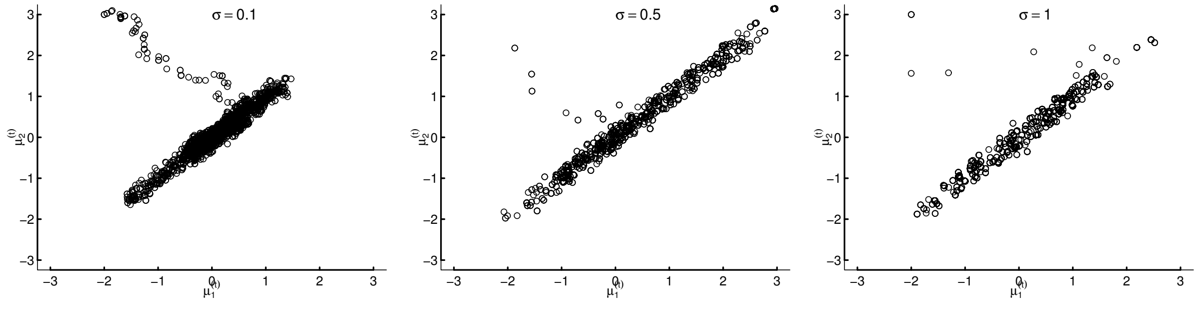
- ○ $Y = (Y_1, Y_2)^{\mathsf{T}} \sim \mathcal{N}(0, \Sigma)$
- ○ $\mathrm{corr}(Y_1, Y_2) = 0.99$

Proposal distribution

$$q(Y, Y') \sim \exp\Big(-\frac{1}{2\sigma^2}|Y - Y'|^2\Big)$$

Results:





Simulated Annealing

Aim: Find maximum y^* of probability distribution/density $f(y)$

Idea: Stochastic optimization - sample from $f(y)$ and approximate y^* by

$$\max\{Y^{(1)}, \ldots, Y^{(n)}\}.$$

Problem: Y takes value y^* with probability $f(y^*)$. This might be small.

To get the maximum with higher probability, we create a distribution that pronounces the maximum y^* more: Take

$$f^{(k)}(y) = \frac{f(y)^k}{\sum_{y'} f(y')^k}.$$

Then $f^{(k)}(y^*) \to 1$ as $k \to \infty$.

Implementation:

To generate a sample from $f^{(k)}$, we use the Metropolis-Hastings algorithm with acceptance probability

$$\alpha(y, y') = \min\left\{1, \left(\frac{f(y')}{f(y)}\right)^k \frac{q(y|y')}{q(y'|y)}\right\}.$$

More generally, we can maximize an arbitrary function $g(y)$ by using

$$f(y) = \frac{g(y)}{\sum_{y'} g(y')}.$$

The sum (or integral for continuous sample spaces) in the denominator will cancel out and therefore need not be calculated.

Simulated Annealing

Problem:

For large k

- $f^{(k)}$ concentrates on x^*
- transition between states can be extremely difficult
- chain might become trapped in local mode.

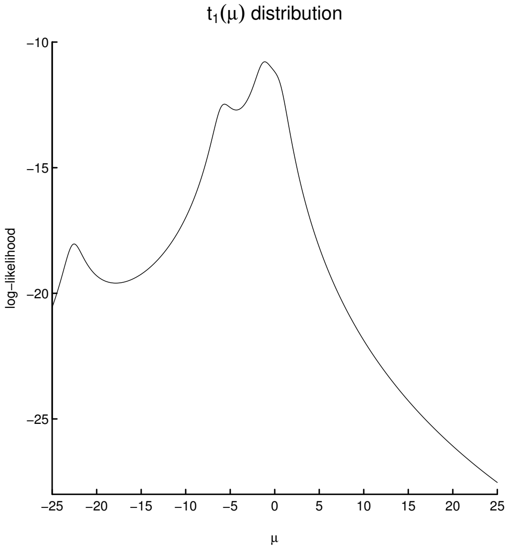
Idea: gradually "cool down" - simulated annealing

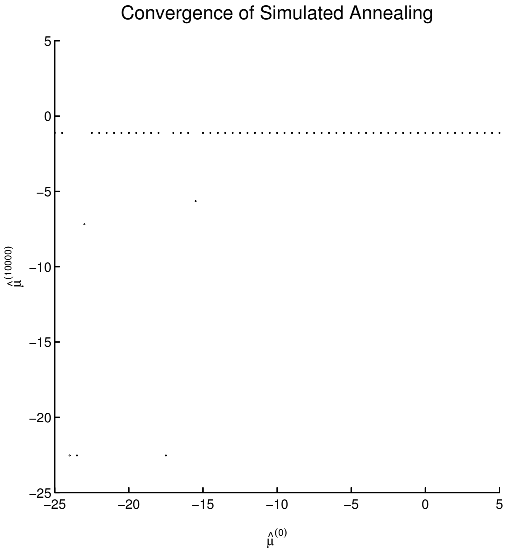
- start with $k < 1$
- increase k slowly ("temperature" $\tau = k^{-1}$ decreases)
- keep track of the best x (since we might leave it and not come back)
- optimal cooling scheme

$$\tau(t) = \frac{c}{\log(1-t)}$$

guarantees that chain converges to maximum: $Y^{(t)} \to y^*$ with probability 1.

Example: t distribution





BUGS

Birats: a bivariate normal hierarchical model

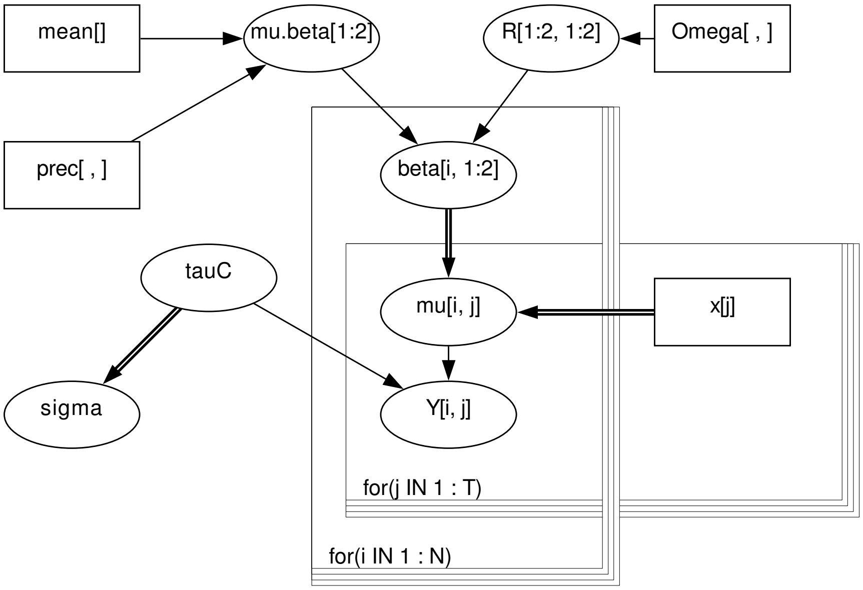
We return to the Rats example, and illustrate the use of a multivariate Normal (MVN) population distribution for the regression coefficients of the growth curve for each rat. This is the model adopted by Gelfand etal (1990) for these data, and assumes *a priori* that the intercept and slope parameters for each rat are correlated. For example, positive correlation would imply that initially heavy rats (high intercept) tend to gain weight more rapidly (steeper slope) than lighter rats. The model is as follows

$$\begin{aligned} Y_{ij} &\sim \text{Normal}(\mu_{ij}, \tau_c) \\ \mu_{ij} &= \beta_{1i} + \beta_{2i} x_j \\ \boldsymbol{\beta}_i &\sim \text{MVN}(\boldsymbol{\mu}_\beta, \boldsymbol{\Omega}) \end{aligned}$$

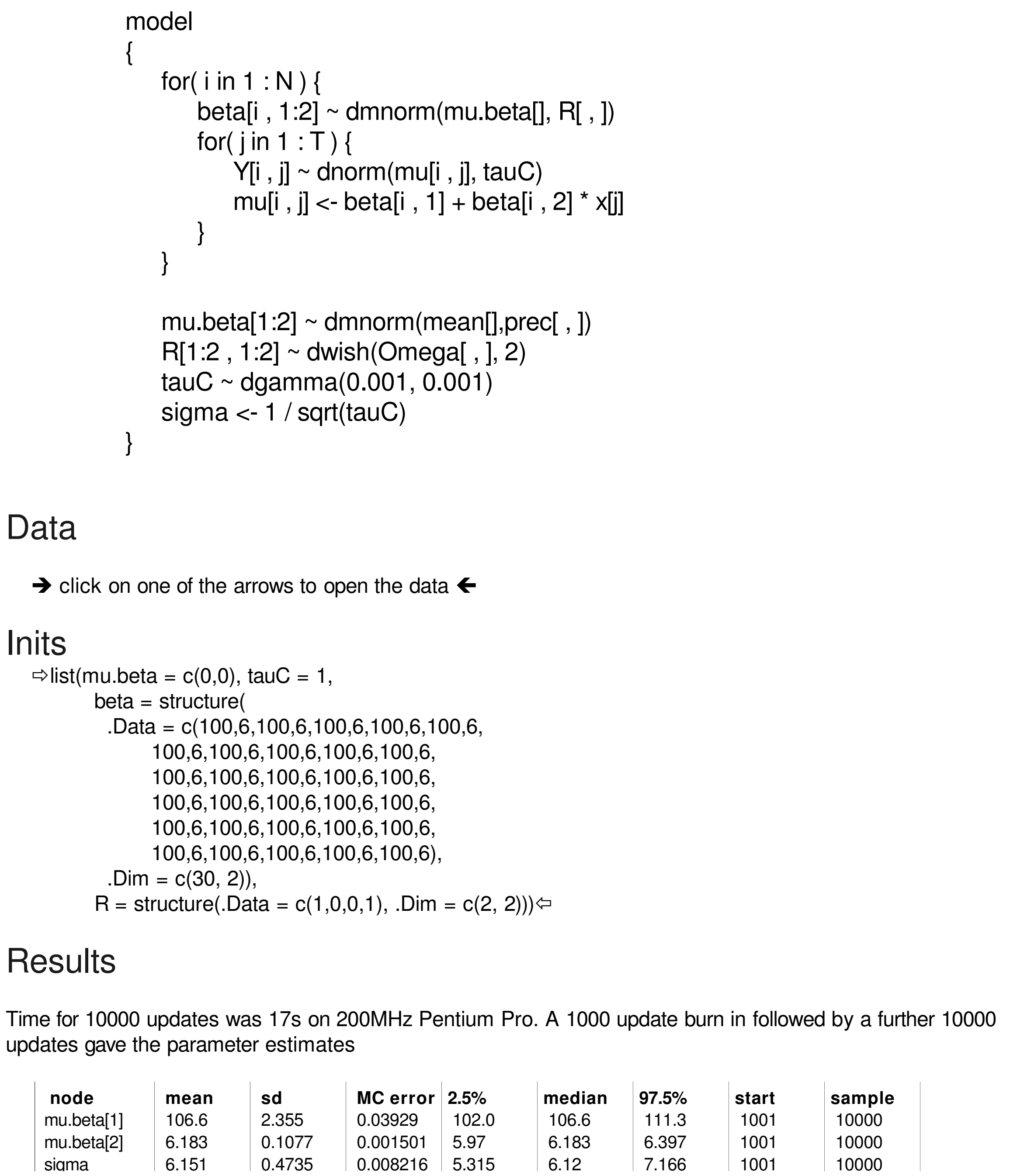
where Y_{ij} is the weight of the ith rat measured at age x_j, and $\boldsymbol{\beta}_i$ denotes the vector (β_{1i}, β_{2i}). We assume 'non-informative' independent univariate Normal priors for the separate components μ_{β_1} and μ_{β_2}. A Wishart(R, ρ) prior was specified for $\boldsymbol{\Omega}$, the population precision matrix of the regression coefficients. To represent vague prior knowledge, we chose the the degrees of freedom ρ for this distribution to be as small as possible (i.e. 2, the rank of $\boldsymbol{\Omega}$). The scale matrix was specified as

$$R = \begin{vmatrix} 200, & 0 \\ 0, & 0.2 \end{vmatrix}$$

This represents our prior guess at the order of magnitude of the *covariance* matrix $\boldsymbol{\Omega}^{-1}$ for $\boldsymbol{\beta}_i$ (see Classic BUGS manual (version 0.5) section on Multivariate normal models), and is equivalent to the prior specification used by Gelfand et al. Finally, a non-informative Gamma(0.001, 0.001) prior was assumed for the measurement precision τ_c.



BUGS



```
model
{
    for( i in 1 : N ) {
        beta[i , 1:2] ~ dmnorm(mu.beta[], R[ , ])
        for( j in 1 : T ) {
            Y[i , j] ~ dnorm(mu[i , j], tauC)
            mu[i , j] <- beta[i , 1] + beta[i , 2] * x[j]
        }
    }

    mu.beta[1:2] ~ dmnorm(mean[],prec[ , ])
    R[1:2 , 1:2] ~ dwish(Omega[ , ], 2)
    tauC ~ dgamma(0.001, 0.001)
    sigma <- 1 / sqrt(tauC)
}
```

Data

➔ click on one of the arrows to open the data ⬅

Inits

```
⇨list(mu.beta = c(0,0), tauC = 1,
        beta = structure(
          .Data = c(100,6,100,6,100,6,100,6,100,6,
                100,6,100,6,100,6,100,6,100,6,
                100,6,100,6,100,6,100,6,100,6,
                100,6,100,6,100,6,100,6,100,6,
                100,6,100,6,100,6,100,6,100,6,
                100,6,100,6,100,6,100,6,100,6),
          .Dim = c(30, 2)),
        R = structure(.Data = c(1,0,0,1), .Dim = c(2, 2)))⇦
```

Results

Time for 10000 updates was 17s on 200MHz Pentium Pro. A 1000 update burn in followed by a further 10000 updates gave the parameter estimates

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
mu.beta[1]	106.6	2.355	0.03929	102.0	106.6	111.3	1001	10000
mu.beta[2]	6.183	0.1077	0.001501	5.97	6.183	6.397	1001	10000
sigma	6.151	0.4735	0.008216	5.315	6.12	7.166	1001	10000

MCMC Summary

Software

BUGS (Bayesian inference Using Gibbs Sampling)

`http://www.mrc-bsu.cam.ac.uk/bugs/`

Strength of MCMC

- Freedom in modelling
- Freedom in inference
- Oppurtunities for simultaneous inference
- Allows sensitivity analysis
- Model comparison/criticism/choice

Weaknesses of MCMC

- Order $N^{-\frac{1}{2}}$ precision
- Possibility of slow convergence
- Difficulty in detecting slow convergence