

MCMC II

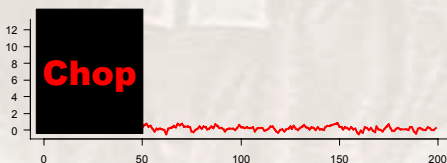
Practicalities

What Is MCMC For?

- Create a Markov chain whose stationary distribution is the same as the target distribution
- If we take a lot of samples from the chain, they should have the correct distribution
- How do we decide if we have reached the distribution?
- How well do the samples approximate the target distribution?

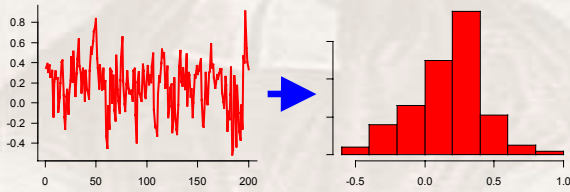
The burn-in

- The theory says the chain will reach the required distribution eventually
...but does not say when
- We remove the first few values, the *burn-in*



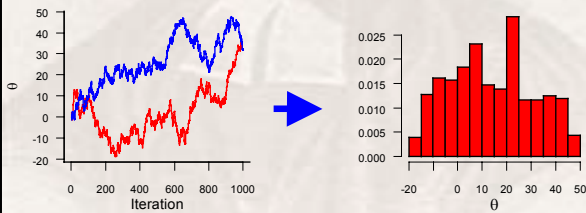
The Rest

- When the posterior has reached the stationary distribution, we say it has *converged*



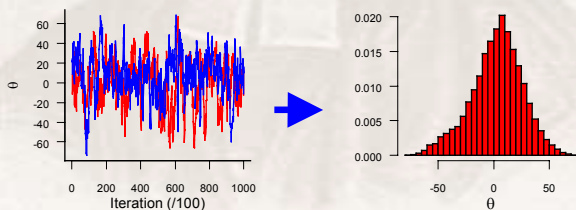
But...

- Even after we have reached convergence, the chain might not be good:



A Solution

- Run the chain for a longer period
 - e.g. 100 times as long
 - Want to get good *mixing*



Long chains

- If we run an chain for a long time, then we get lots of numbers
- But that takes up a lot of memory
- Therefore we only take some iterations
 - e.g. take every 10
- Called *thinning*
- Taking iterations at an even interval reduces the autocorrelation between iterations

How Many Points?

- To get a good coverage of the target distribution, we need a lot of points
 - I typically use 10 000
- The precise number will depend on what you want and what resources you have
 - 1000 iterations might be OK for some problems
- Want to cover most of the distribution
 - more dimensions will need more iterations

How Many Points II

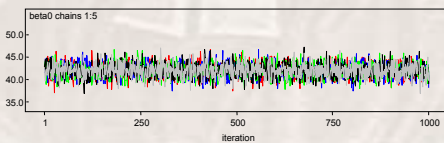
- We can assess how well the estimation is doing by comparing the mean of the samples, and the true posterior mean
 - have to use estimates!
- Called the *MC error*
- Rule of thumb: we are OK if this is less than 5% of the true error
 - i.e. if the error in the estimation swamped by the uncertainty in the true value

Checking The Chain

- We need to check if we have converged
- There are formal methods, mostly based on running several chains
- If we have not converged, then we may have to remove our points, as a burn-in
- Or, if we have converged, but the mixing is not good, then we have to either thin the chain, or improve the sampling

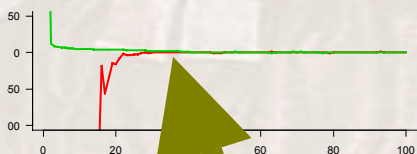
Visuals

- The simplest way to check chains is to look at a graph
 - WinBugs *history*



When shall we 2 meet again?

- We can see when the chains have converged



Thinning

- We can also see if the mixing is good



- Here it is not

Formal Tests

- We can also use more formal tests of convergence
 - several in the Coda package for R
- One test in WinBugs
 - Gelman-Rubin-Brooks
- Multivariate example of the Gelman-Rubin statistic

Gelman-Rubin

- Based on an ANOVA idea
- Look at a single variable
- m chains, each of length n
- Can estimate the variance of a stationary distribution in two ways
 - variance within a single chain, W
 - variance over all chains, B/n

Gelman-Rubin

- If the chains have converged, both estimates are unbiased, i.e. $B=W$
- If the initial values are overdispersed and have not dispersed, then the Between term is an overestimate
- The statistic: $R = B/W$
- If $R > 1$, we have not converged

- we estimate R by $\hat{R} = \frac{m+1}{m} \left(\frac{n-1}{n} + \frac{B}{W} \right) - \frac{n-1}{mn}$

Gelman-Rubin-Brooks

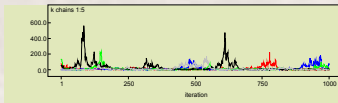
- It may be that the ratio B/W is about 1 by chance
 - if both B and W are still moving
 - for example, if the initial values are not overdispersed, and a chain starts to explore a new part of the parameter space
- B and W should therefore be checked as well
- WinBugs uses a graphical approach

G-R-B: The Graph

- WinBugs plots a bgr diag
- Three components
 - Gelman-Rubin statistic
 - B variance
 - width of the central 80% interval of the pooled runs
 - W variance
 - average width of the 80% intervals within the individual
- W and B scaled to a maximum of 1

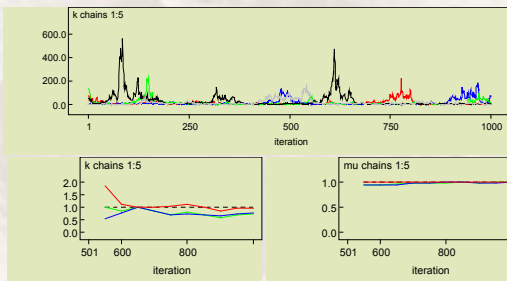
What Should Happen

- We have convergence if:
- **Gelman-Rubin statistic** is about 1
 - **B** and **W** both stabilise around the same value
- For example: the degrees of freedom in a t distribution:



An Example

- The mean (μ) and degrees of freedom (k) in a t distribution:

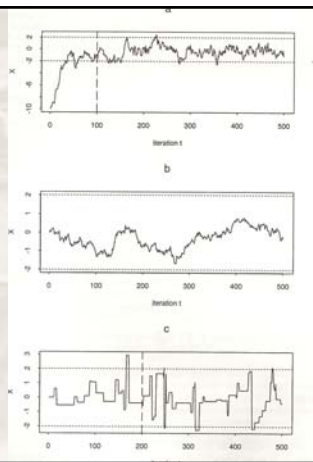


Metropolis-Hastings

- If we are using the Metropolis-Hastings algorithm, then we have to get a good proposal distribution
- WinBugs tries to find a good distribution automatically
- Uses a Normal proposal distribution
 - need to optimise the variance

Tuning M-H

- Can change the variance of the proposal distribution
- If too small, don't cover the whole distribution
- If too big, don't accept very often and jump

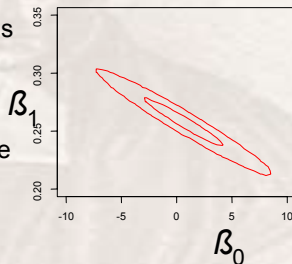


Some Solutions

- If the burn-in seems excessive, then initial values can be set which are closer to the target distribution
 - but you lose some ability to check convergence
- If mixing is bad, run a longer chain and thin
- Or, try a different parameterisation of the model

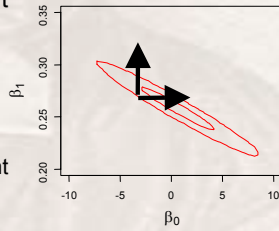
Re-parameterisation

- In several dimensions we can have several parameters being correlated
- e.g. regression: place the intercept 0
- Large correlation
 - $\rho = -0.989$



Problems With Correlations

- Makes sampling difficult
- Difficult to make large jumps
- Solution: re-parameterisation
 - write model in a different form



Problems With Correlations

- In this case, we move the intercept to the mean
 - i.e. remove the mean of the covariate from every covariate
- Removes the correlation

