

The WinBUGS Software

Different points of view

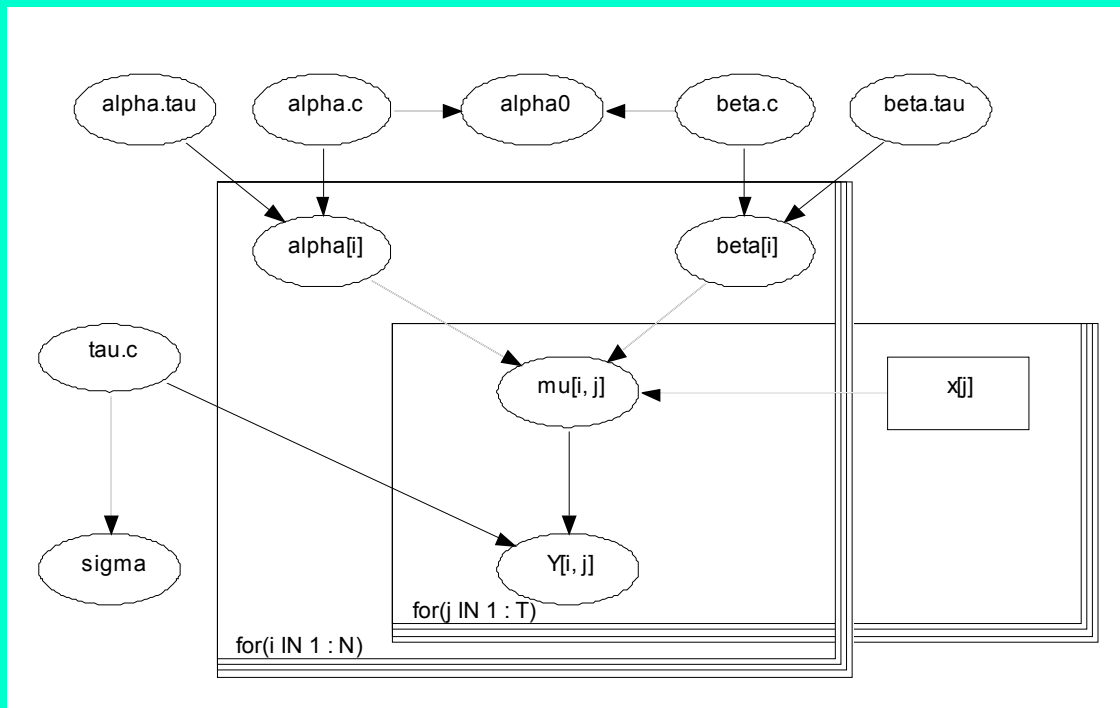
- Software insides / Software outsides
- Making software / Using software
- What's happening / What's happening?
- Something's wrong / Stupid error message!!!

Constructing Software

- Plan – what to construct
- Tools – what to use to do the construction
- Material – what to do the construction out off

The Plan

- Use the graphical model



The Tools

- Smart tools, know what to do, take plan and material and produce Bayesian software.
- Tools are also software, a kind of compiler). Tools need to be developed (this is a major effort).

The Material

- Software build out of material.
- Material can be considered at various levels of aggregation.
- OO level, material is objects, each object has a “type” or “class”.
- Each “type” or “class” needs designing.

Software Insides

- DAG of objects used to represent the Bayesian model.
- Updater objects sit above nodes in DAG and do the MCMC simulation.
- Monitor objects store values from nodes in the DAG.

Physical Software Organization

- Modular design, each module compiles to a separate unit (DLL).
- Each module has well defined interface.
- Consistency checked across modules at compile time and load time.
- WinBUGS consists of over 200 modules.
- New modules can be added.

Software Outsides

- Tools for specifying graphical model -graphics editors (DoodleBUGS) or text editors.
- Tools for monitoring chains of random variables.
- Tools for analysing chains of random variables.

User Interfaces

- Two different approaches – GUIs and script languages.
- GUIs good for learning how to use the software and exploring the model, but bad for repeat use.
- Script languages good for repeat use and slowly updating models, but bad for error feedback.
- GUIs and script languages can be mixed.

Making/Using Software

- With WinBUGS you both make and use software.
- Each model causes a new software system to be build.
- Building consists of assembly of building blocks according to a plan.
- Software user free to build models. Software maker can make new building blocks

What's happening

- Combine model specification and data set(s) into graph of objects.
- One language representation is transformed into another lower language.
- Lower level language can do computation.
- During this transformation expert knowledge is added.

MCMC algorithms

- WinBUGS uses graphical model to calculate full conditionals distributions.
- Examine each factor in the full conditional and classify its functional form.
- Combine functional form of each factors to get functional form of full conditional.
- Choose suitable MCMC simulation algorithm.

Conjugacy

- Sometimes the functional form of the full conditional can be describe by a few parameters.
- If these parameters can be calculated it is easy to sample from the full conditional.
- Example $P(\alpha|Y[i]) = N(\alpha|0, 1E-3)N(Y|\alpha, \tau)$ is a normal distribution with certain mean and precision parameters.

Conjugacy continued

- Normal – normal, uniform prior plus normal, MVN, log normal likelihood.
- Gamma – gamma, exponential, chisq, uniform prior plus normal, gamma, weibull likelihood.
- Beta – beta, uniform prior plus bernoulli, binomial, geometric, negative binomial likelihood.

Conjugacy continued

- MVN - MVN prior plus MVN, normal, log normal likelihood.
- Wishart – wishart prior plus MVN likelihood.
- Dirichlet – dirichlet prior plus categorical, multinomial likelihood.

Log concave distributions

- A function is log concave if the second derivative of its log is always negative. Log concave functions have a single mode.
- Example the normal distribution $N(Y|\mu,\tau)$ is a log concave function in Y , μ and τ .
- Product of log concave functions is log concave.
- Some full conditionals log concave.

Log concave distributions continued

- Example

$$P(\theta|r) = N(\theta|0, 1E-3) \prod \text{Bin}(p|r[i], n[i])$$

where $\text{logit}(p) \leftarrow \theta$.

- Generalized linear models give log concave full conditional distributions.

Adaptive sampling algorithms

- MCMC algorithms generate a new random number only using the current value of the stochastic variables.
- Sometimes it is helpful to use the past history of the chain to choose how to generate the new random number.
- This is NOT markov, can not be used for Bayesian inference. But still useful.

Adaptive metropolis sampling

- Generate a candidate point $x.new$ from the current point x by sampling

$$x.new \sim N(x, \tau)$$

Accept the candidate point $x.new$ if

$$\min(P(x.new|\dots)/P(x|\dots), 1) > U(0,1)$$

else reject new point and stay at x .

- Problem how to choose τ .

Adaptive metropolis sampling continued

- If τ is too small candidate point is often accepted but chain is highly autocorrelated.
- If τ is too big candidate point is often rejected and chain “sticks” in one place.
- Solution adapt value of τ to get in between acceptance rates (40%).
- Stop changing (adapting) value of τ when acceptance rate steady at 40%.

Something's wrong...

- Software decides how to do things.
- If something goes wrong software has to say what it is doing and what is going wrong.
- Not easy to explain to user what is happening.
- Error messages difficult to write AND UNDERSTAND.