Applied Bayesian Analysis for the Social Sciences

Day 3: Introduction to WinBUGS

Ryan Bakker
University of Georgia

BUGS Software for MCMC

- A relatively new convenience: flexible, powerful, but sometimes fragile.
- The commands are relatively `R` -like, but unlike `R` there are relatively few functions to work with.
- Unlike other programming languages, statements are not processed serially, they constitute a full specification.
- Four steps to producing useful MCMC generated inferences in `BUGS` .
  1. Specify the distributional features of the model, and the quantities to be estimated.
  2. Compile the instructions into the run-time program.
  3. Run the sampler that produces Markov chains.
  4. Assess convergence using basic diagnostics in `BUGS` or the more sophisticated suites of programs, `CODA` and `BOA` that run under `R` .

Specifying Models with `BUGS` (cont.)

- The default MCMC transition kernel is the Gibbs sampler and therefore the first step must identify the full conditional distributions for each variable in the model.

- The second major component of this process is the iterative (human) cycling between the third and fourth steps.

- Recall that the Markov chain is conditionally dependent on just the last step, so the only information that needs to be saved in order to restart the chain is the last chain value produced.

- `BUGS` will write these values to a file named "`last.chain.values`" (or some other user-selected file name) with the command `save("last.chain.values")`, and then it is necessary to change the source file to specify these starting values on the following cycle (`inits in "last.chain.values";`).

Specifying Models with `BUGS` (cont.)

- `BUGS` Vocabulary:
  - **node**: values and variables in the model, specified by the researcher.
  - **parent**: a node with direct influence on other nodes.
  - **descendent**: the opposite of a parent node, but also can be a parent.
  - **constant**: a "founder node", they are fixed and have no parents.
  - **stochastic**: a node modelled as a random variable (parameters or data).
  - **deterministic**: logical consequences of other nodes.

Specifying Models with BUGS (cont.)

- Technical notes:
  - The underlying engine is *Adaptive Rejection Sampling* (Gilks 1992, Gilks and Wild 1992), an MCMC implementation of rejection sampling.
  - This means that all priors and all likelihood functions must be either: (1) discrete, (2) conjugate, or (3) log concave.
  - Not a big deal as all GLMs with canonical link functions are well-behaved in this respect.
  - It also means that deterministic nodes must be linear functions.
  - Often these restrictions can be finessed.
  - `BUGS` likes simple, clean model structure. So if pre-processing in `R` can be done, it generally helps.

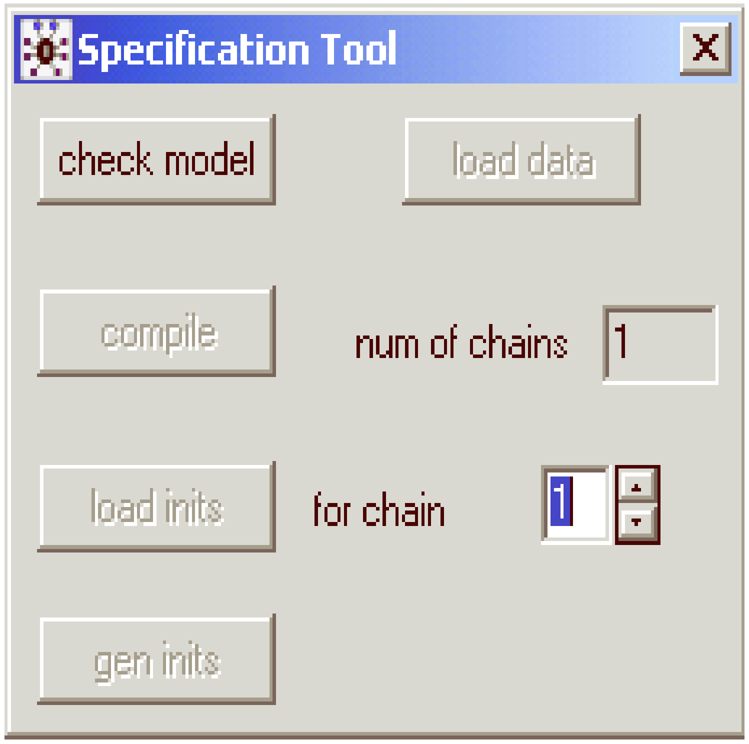
Details on `WinBUGS`

- Minor stuff:
  - `WinBUGS` has lots of "bells and whistles" to explore, such as running the model straight from the doodle.
  - The data from any plot can be recovered by double-clicking on it.
  - Setting the seed may be important to you: leaving the seed as is exactly replicates chains.
  - Things I don't use that you might:
    - Encoding/Doodling documents.
    - Fonts/colors in documents.
    - Log files to summarize time and errors.
    - Fancy windowing schemes.

Compound Document Interface in `BUGS`

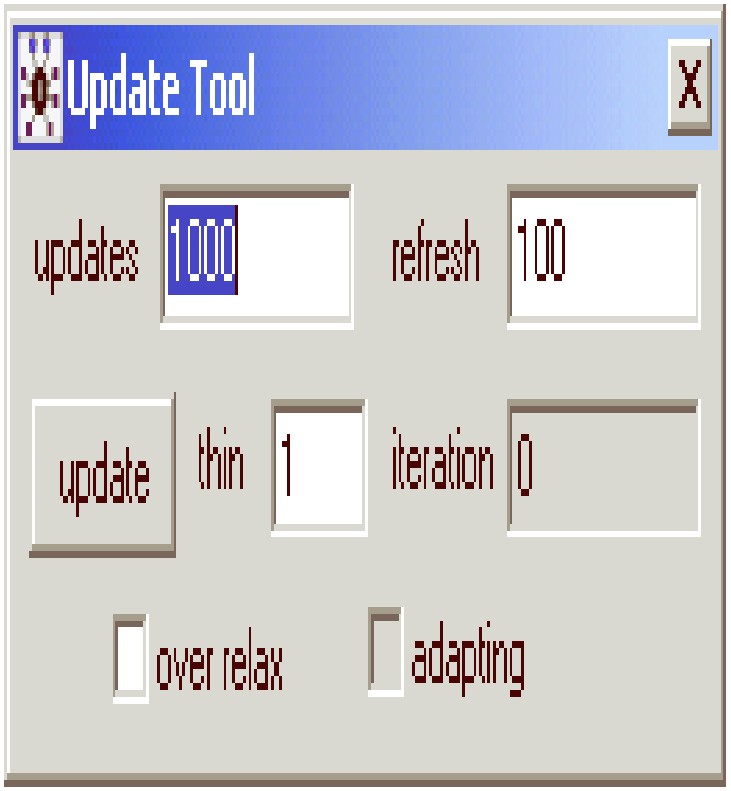
- An omnibus file format that holds: text, tables, code, formulae, plots, data, inits.
- The goal is to minimize cross-applications work.
- If one of the CDI elements are focused, its associated tools are made available.
- Also integrates the built-in editor.
- Online documentation has details about creating CDI files.
- Simplest conversion process: open .bug file as text, modify, save as .odc file, open .dat file, modify, copy into same .odc file, copy .in file into same .odc file.

Specification Tool Window



- Buttons:
  - check model: checks the syntax of your code.
  - load data: loads data from same or other file.
  - num of chains: sets number of parallel chains to run.
  - compile: compiles your code as specified.
  - load inits: loads the starting values for the chain(s).
  - gen inits: lets `WinBUGS` specify initial values.
  - Dismiss Specification Tool window when done.

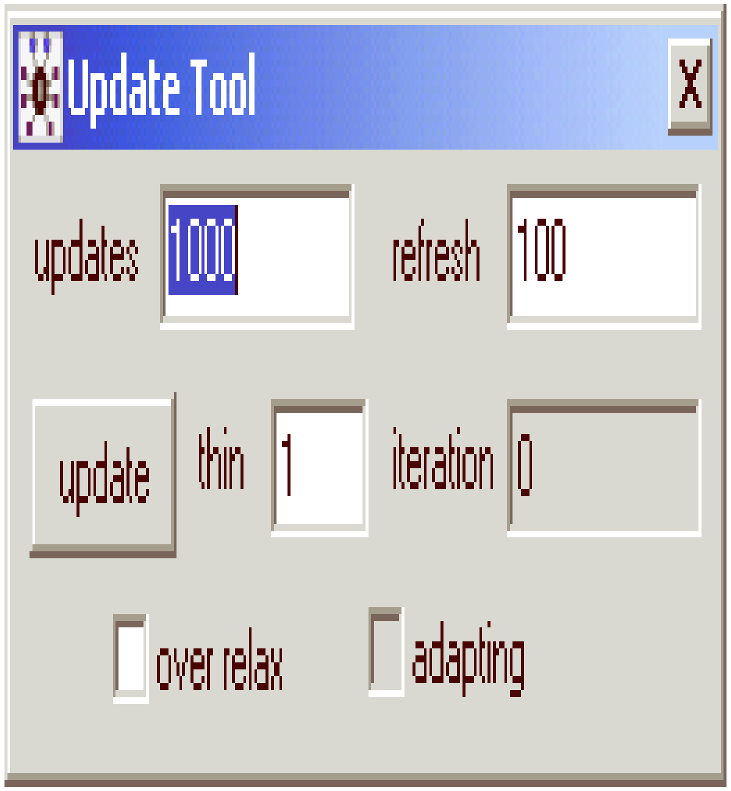
Update Window



updates 1000 refresh 100
thin 1 iteration 0

- Buttons:
  - updates: you specify the number of chain iterations to run this cycle.
  - refresh: the number of updates between screen redraws for traceplots and other displays.
  - update: hit this button to begin iterations.
  - thin: number of values to thin out of chain between saved values.
  - iteration: current status of iterations, by UPDATE parameter.

Update Window (cont.)



Update Tool

updates 1000 refresh 100

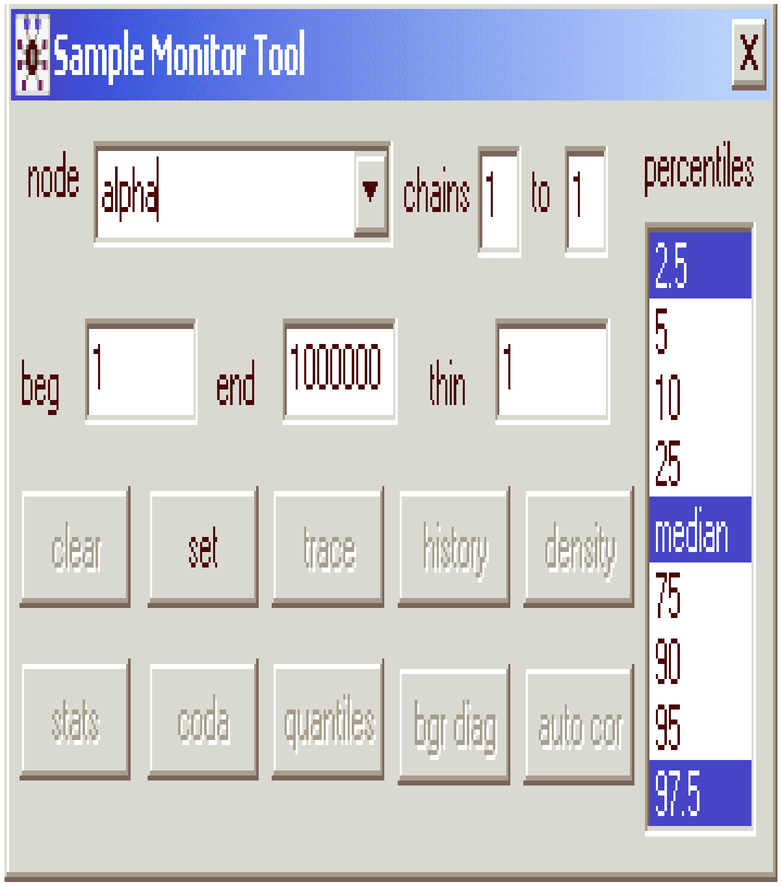
thin 1 iteration 0

[ ] over relax [ ] adapting

- Buttons:
  - over relax: click in the box for option to
    - generate multiple samples at each cycle,
    - pick sample with greatest negative correlation to current value.

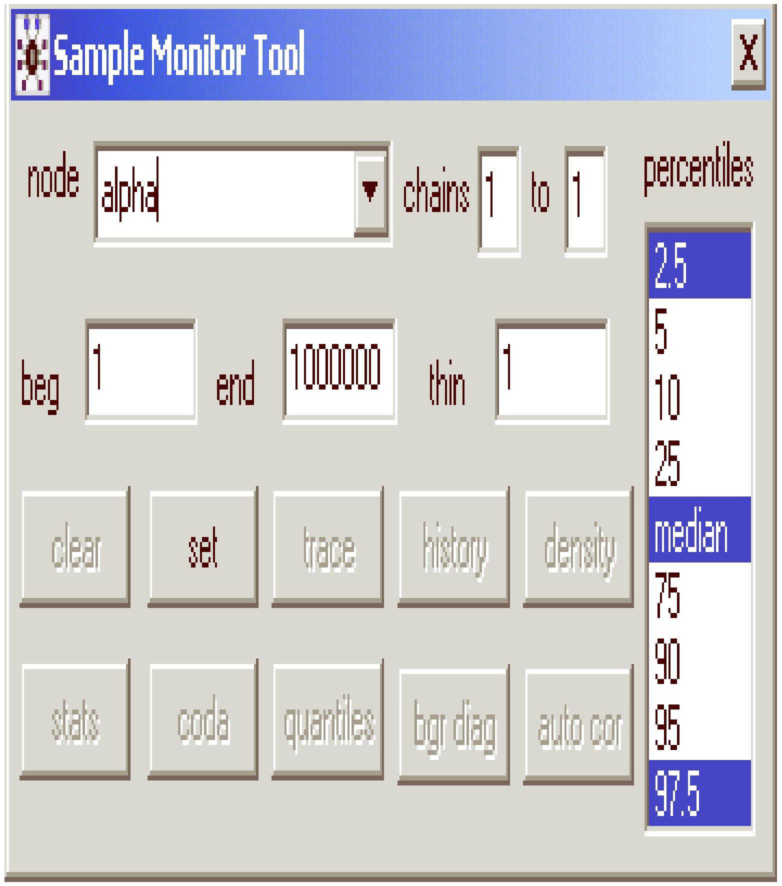
    Trades cycle time for mixing qualities.
  - adapting: box will be automatically clicked while the algorithm for Metropolis or slice sampling (using intentionally introduced auxiliary variables to improve convergence and mixing) still tuning optimization parameters (4000 and 500 iterations, respectively). Other options "greyed out" during this period.

Sampling Window



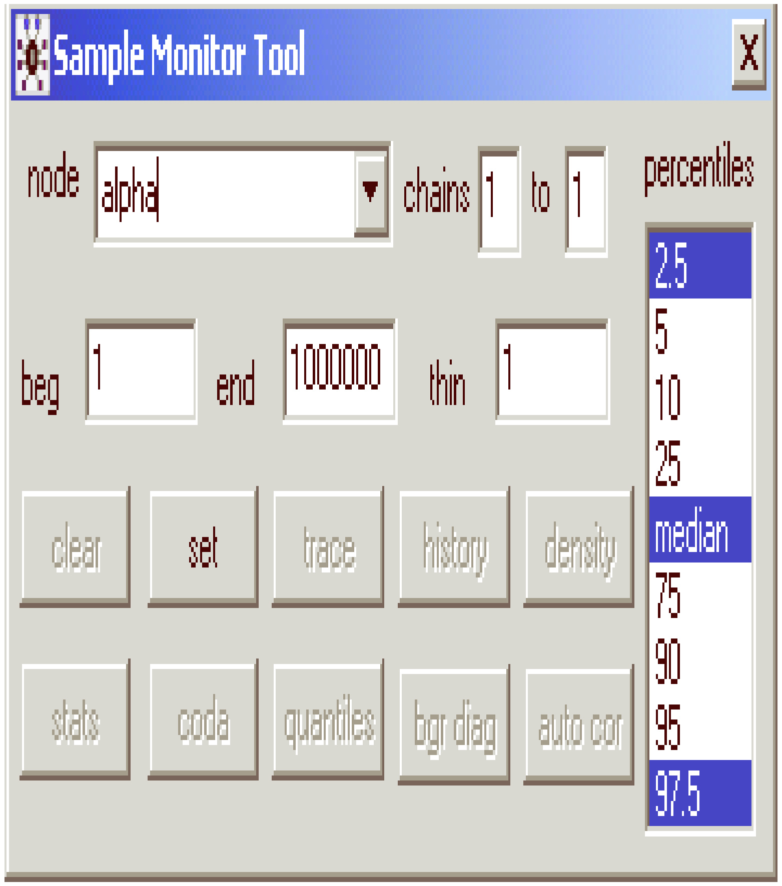
- Buttons:
  - **node**: sets each node of interest for monitoring; type name and click SET for each variable of interest.
  - Use the " * " in the window when you are done to do a full monitor.
  - **chains**: "1 to 10", sets subsets of chains to monitor if multiple chains are being run.

Sampling Window



- Buttons, cont.:
  - **beg**, **end**: the beginning and ending chain values current to be monitored. BEG is 1 unless you *know* the burn-in period.
  - **thin**: yet another opportunity to thin the chain.
  - **clear**: clear a node from being monitored.

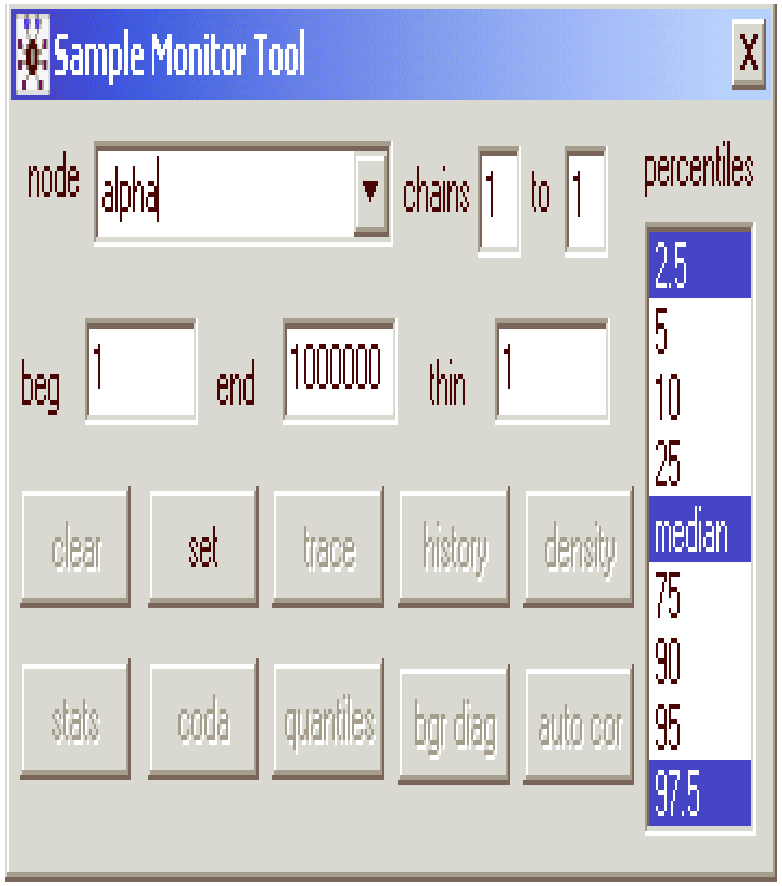
Sampling Window



node alpha
chains 1 to 1
percentiles: 2.5, 5, 10, 25, median, 75, 90, 95, 97.5
beg 1 end 1000000 thin 1

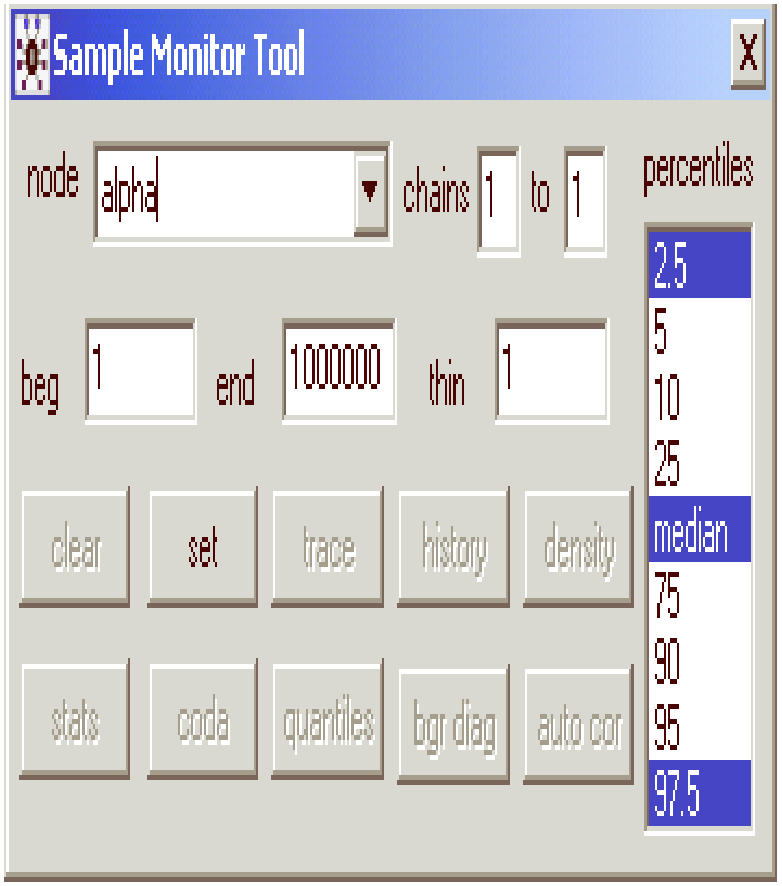
- Buttons, cont.:
  - **trace**: do dynamic traceplots for monitored nodes.
  - **history**: display a traceplot for the complete history.
  - **density**: display a kernel density estimate.

Sampling Window



- Buttons, cont.:
  - **quantiles**: displays running mean with running 95% CI by iteration number.
  - **auto cor**: plots of autocorrelations for each node with lags 1 to 50.
  - **coda**: display chain history in window in `CODA` format, another window appears with `CODA` ordering information.

Sampling Window



node: alpha
chains 1 to 1
beg 1
end 1000000
thin 1
percentiles: 2.5, 5, 10, 25, median, 75, 90, 95, 97.5

- Buttons, cont.:
  - **stats**: summary statistics on each monitored node using: mean, sd, MC error, current iteration value, starting point of chain and percentiles from PERCENTILES window.
  - Notes on **stats**: `WinBUGS` regularly provides both: naive SE $=$ sample variance$/\sqrt{n}$ and: MC Error $=$ $\sqrt{\text{spectral density var}}/\sqrt{n}$ $=$ asymptotic SE.

Basic Structure of the BHM

- Start with the most basic model setup for producing a posterior:

$$\pi(\theta|\mathbf{D}) \propto L(\theta|\mathbf{D})p(\theta).$$

- Now suppose that θ is conditional on another quantity ψ, so that the calculation of the posterior becomes:

$$\pi(\theta,\psi|\mathbf{D}) \propto L(\theta|\mathbf{D})p(\theta|\psi)p(\psi).$$

- θ still has a prior distribution, $p(\theta|\psi)$, but it is now conditional on another parameter that has its own prior, $p(\psi)$, called a *hyperprior*, which now has its own *hyperparameters*.

- Inference for either parameter of interest can be obtained from the marginal densities:

$$\pi(\theta|\mathbf{D}) = \int_\psi \pi(\theta,\psi|\mathbf{D})d\psi$$
$$\pi(\psi|\mathbf{D}) = \int_\theta \pi(\theta,\psi|\mathbf{D})d\theta.$$

Basic Structure of the BHM (cont.)

- We can continue stringing hyperpriors to the right:

$$\pi(\theta, \psi, \zeta|\mathbf{D}) \propto L(\theta|\mathbf{D})p(\theta|\psi)p(\psi|\zeta)p(\zeta).$$

- Now ζ is the highest level parameter and therefore the only hyperprior that is unconditional.

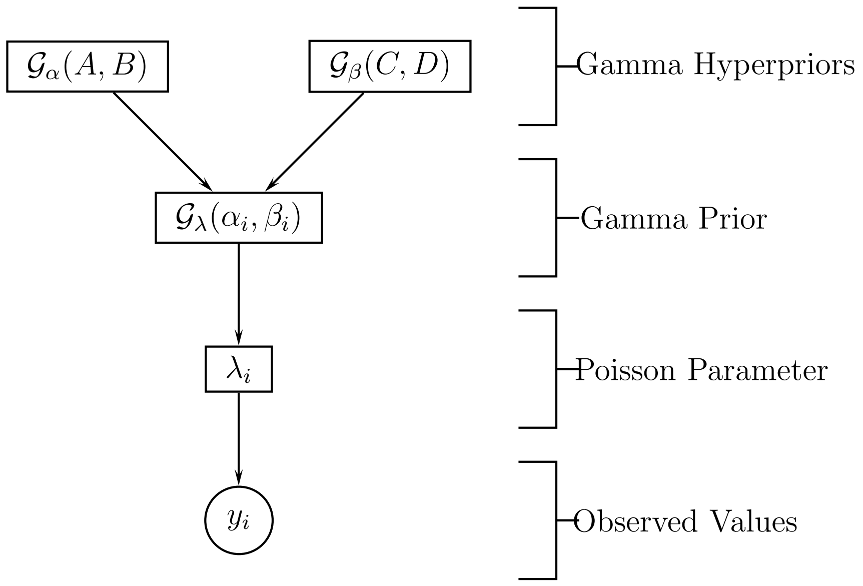
- Marginal for the new conditional obtained the same way:

$$\pi(\psi|\mathbf{D}) = \int_{\zeta} \pi(\psi, \zeta|\mathbf{D})d\zeta.$$

- No real restriction to the number of levels but increasing levels have decreasing *conditional amount of sample information*, and lower model utility. See Goel and Degroot (1981) or Goel (1983) for a formalization of the *conditional amount of sample information* (CASI).

A Poisson-Gamma Hierarchical Model

- Consider the following model with priors and hyperpriors:



A Poisson-Gamma Hierarchical Model (cont.)

- Notes on this model:
  - The intensity parameter of the Poisson distribution is now indexed by i since it is no longer assumed to be a fixed effect.
  - Now: $\lambda_i \sim \mathcal{G}(\alpha, \beta)$.
  - Model expressed in "stacked" notation:

$$\begin{aligned} y_i &\sim \mathcal{P}(\lambda_i) \\ \lambda_i &\sim \mathcal{G}(\alpha, \beta) \\ \alpha &\sim \mathcal{G}(A, B) \\ \beta &\sim \mathcal{G}(C, D), \end{aligned}$$

  where the y_i are assumed conditionally independent and α and β are assumed independent.

A Poisson-Gamma Hierarchical Model (cont.)

- The joint *posterior* distribution of interest is:

$$
\begin{aligned}
p(\lambda, \mathbf{y}, \alpha, \beta) &= \prod_{i=1}^{n} p(y_i|\lambda_i)p(\lambda_i|\alpha, \beta)p(\alpha|A, B)p(\beta|C, D) \\
&= \prod_{i=1}^{n} \Bigg[(y_i!)^{-1}\lambda_i^{y_i}\exp(-\lambda_i)\beta^{\alpha}\Gamma(\alpha)^{-1}\exp(-\lambda_i\beta)\lambda_i^{\alpha-1} \\
&\qquad \times B^A\Gamma(A)^{-1}\exp(-\alpha B)\alpha^{A-1}D^C\Gamma(C)^{-1}\exp(-\beta D)\beta^{C-1} \Bigg],
\end{aligned}
$$

so:

$$p(\lambda_i, \mathbf{y}, \alpha, \beta) \propto \lambda_i^{\sum y_i+\alpha-1}\alpha^{A-1}\beta^{\alpha+C-1}\Gamma(\alpha)^{-1}\exp[-\lambda_i(1+\beta) - \alpha B - \beta D].$$

which is unpleasant.

A Poisson-Gamma Hierarchical Model (cont.)

- Recall that if we can get the full conditional distributions for each of the coefficients in the posterior, we can run the Gibbs sampler and obtain the marginal posterior distributions with MCMC.

$$\begin{aligned}\pi(\lambda|\alpha,\beta) = \frac{p(\lambda,\alpha,\beta|\mathbf{y})}{p(\alpha,\beta|\mathbf{y})} = \frac{p(\lambda,\alpha,\beta)}{p(\alpha,\beta)} = \frac{p(\lambda,\alpha,\beta)}{p(\alpha)p(\beta)} \\ \propto \lambda^{\sum y_i + \alpha - 1} \exp\left[-\lambda\left(\beta+n\right)\right] \\ \sim \mathcal{G}\left(\sum y_i + \alpha, n + \beta\right).\end{aligned}$$

- So given (interim) values for α and β, we can sample from the distribution for λ.

- The easy part: $p(\alpha|\beta,\lambda) = p(\alpha)$, and $p(\beta|\alpha,\lambda) = p(\beta)$, by the initial assumptions of the hierarchical model.