36-463/663Multilevel and Hierarchical Models

From Bayes to MCMC to MLMs

Brian Junker
132E Baker Hall
brian@stat.cmu.edu

Outline

- Bayesian Statistics and MCMC
- Distribution of Skill Mastery in a Population
- Digression: What is Markov Chain Monte Carlo (MCMC)
- Estimating the Distribution of Skills
  - JAGS & RUBE for MCMC
  - Checking the MCMC output
  - Results of the MCMC output
  - P3() function
- The JAGS recipe: Prof Smedley's Histograms
- MCMC for Hierarchical Linear Models: Minnesota Radon

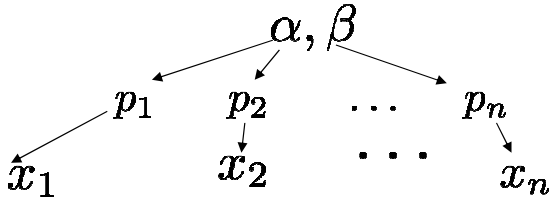
Bayesian Statistics and MCMC

- Our slogan(s)
  - (posterior)$\propto$(likelihood)$\times$(prior)
  - (posterior)$\propto$(level 1)$\times$(level 2)
  - (posterior)$\propto$(level 1)$\times$(level 2)$\times$(level 3)
  - etc.
- Often there are no formulas for posterior, so we have to simulate draws from the posterior
- MCMC is a general method for doing posterior draws
  - Calculate complete conditionals
  - Invent methods for sampling from complete conditionals
  - Stitch together samples into a Markov Chain
  - Summarize Markov Chain with histograms, medians, credible intervals, etc.

Mastery Learning: Distribution of Masteries

- *We want to know the distribution of p, the probability of success, in the population: <u>i.e. we want to estimate α & β !</u>*
  - Level 1: $x_i \sim$ NB(x|r,p_i), i=1, …, n
  - Level 2: $p_i \sim$ Beta(p|α,β), i = 1, …, n
  - Level 3: $\alpha \sim$ Gamma(α|a_1,b_1), $\beta \sim$ Gamma(α|a_2,b_2)

n+2 parameters!



α,β for population of students

p_i = student prob of right

x_i = errors before r rights

Mastery Learning: Distribution of Masteries

- ***Level 1***: If $x_1, x_2, \ldots x_n$ are the failure counts then

$$f(x_1,\ldots,x_n|p_1,\ldots,p_n,r) = \prod_{i=1}^{n}\binom{r+x_i-1}{x_i}p_i^r(1-p_i)^{x_i}$$

- ***Level 2***: If $p_1, p_2, \ldots p_n$ are the success probabilties,

$$f(p_1,\ldots,p_n|\alpha,\beta) = \prod_{i=1}^{n}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}p_i^{\alpha-1}(1-p_i)^{\beta-1}$$

- ***Level 3***:

$$f(\alpha|a_1,b_1) = \frac{b_1^{a_1}}{\Gamma(a_1)}\alpha^{a_1-1}e^{-\alpha b_1}$$

$$f(\beta|a_2,b_2) = \frac{b_2^{a_2}}{\Gamma(a_2)}\beta^{a_2-1}e^{-\beta b_2}$$

Distribution of Mastery probabilities…

- Applying the "slogan" for 3 levels:

$$f(p_1,\ldots,p_n,\alpha,\beta|\text{data})$$

$$\propto \underbrace{f(x_1,\ldots,x_n|p_1,\ldots,p_n,r)}_{\text{(level 1)}} \times \underbrace{f(p_1,\ldots,p_n|\alpha,\beta)}_{\text{(level 2)}} \times \underbrace{f(\alpha|a_1,b_1)f(\beta|a_2,b_2)}_{\text{(level 3)}}$$

$$\propto \prod_{i=1}^{n}\binom{r+x_i-1}{x_i}p_i^r(1-p_i)^{x_i}$$

We can drop these since they only depend on data

$$\times \prod_{i=1}^{n}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}p_i^{\alpha-1}(1-p_i)^{\beta-1}$$

Can't drop these since we want to est α & β…

$$\times \frac{b_1^{a_1}}{\Gamma(a_1)}\alpha^{a_1-1}e^{-\alpha b_1}\frac{b_2^{a_2}}{\Gamma(a_2)}\beta^{a_2-1}e^{-\beta b_2}$$

We need to choose a_1, b_1, a_2, b_2. Then we can drop these

Solution: Markov-Chain Monte Carlo (MCMC)

- MCMC is very useful for multivariate distributions, e.g. f(θ_1,θ_2, …,θ_K)
- Instead of dreaming up a way to make a draw (simulation) of all K variables at once MCMC takes draws one at a time
- We "pay" for this by not getting independent draws. The draws are the states of a *Markov Chain.*
- The draws will not be "exactly right" right away; the Markov chain has to "burn in" to a stationary distribution; the draws after the "burn-in" are what we want!

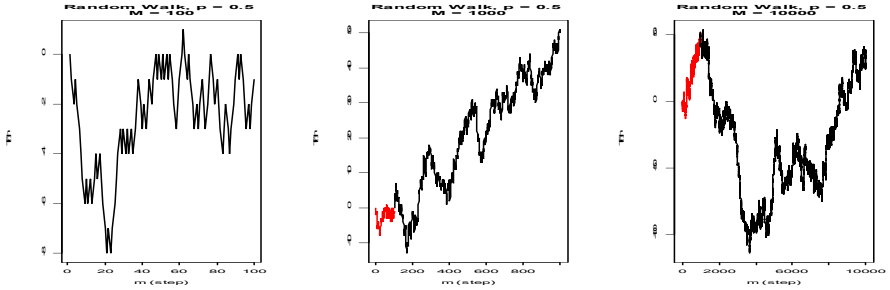
(Digression: What is a Markov Chain?)

- A *Markov Chain* is a stochastic process (36-410!), i.e. it is a sequence of random variables T_1, T_2, T_3, T_4, T_5, …
- The thing that makes it a Markov Chain is the *Markov Property*:
  - T_{m+1} is independent of T_1, … T_{m-1}, given T_m
  - "the future is independent of the past, given the present"
- A *stationary* Markov Chain has a transition probability function f($t_m|t_{m-1}$)…
  - If the T's are discrete rv's, can write f($t_m|t_{m-1}$) in terms of a matrix of probabilities
  - If the T's are continuous rv's, f($t_m|t_{m-1}$) is just a conditional density

(Digression: What is a Markov Chain? …An Example)

- Random Walk
  - T_0 = initial state or "starting point", e.g. 0
  - The transition probability is

$$p(T_m = t_m | T_{m-1} = t_{m-1}) = \begin{cases} p, & \text{if } t_m = t_{m-1} + 1 \\ 1-p, & \text{if } t_m = t_{m-1} - 1 \\ 0 & \text{else} \end{cases}$$



Back to MCMC…

- We want to simulate draws from f(θ_1, …,θ_K).
  - Let $T_m = (\theta_1^{(m)}, \theta_2^{(m)}, \ldots, \theta_K^{(m)})$ be a reasonable initial state
  - Now successively sample each θ_k from its "complete conditional" distribution:

$$\begin{aligned}
\theta_1^{(m+1)} &\sim f(\theta_1 | \theta_2^{(m)}, \theta_3^{(m)}, \ldots, \theta_K^{(m)}) \\
\theta_2^{(m+1)} &\sim f(\theta_2 | \theta_1^{(m+1)}, \theta_3^{(m)}, \ldots, \theta_K^{(m)}) \\
\theta_3^{(m+1)} &\sim f(\theta_3 | \theta_1^{(m+1)}, \theta_2^{(m+1)}, \theta_4^{(m)}, \ldots, \theta_K^{(m)}) \\
&\vdots \quad \vdots \qquad\qquad \vdots \\
\theta_K^{(m+1)} &\sim f(\theta_K | \theta_2^{(m+1)}, \theta_3^{(m+1)}, \ldots, \theta_{K-1}^{(m+1)})
\end{aligned}$$

  and let $T_{m+1} = (\theta_1^{(m+1)}, \theta_2^{(m+1)}, \ldots, \theta_K^{(m+1)})$
  - Now T_1, T_2, …, T_M are MCMC draws "from f"

MCMC generalities…

- The theory of MCMC (e.g. Chib & Greenberg, *American Statistician*, 1995, pp. 327-335) tells us that
  - $T_m = (\theta_1^{(m)}, \theta_2^{(m)}, \ldots, \theta_K^{(m)})$ is a stationary Markov Chain
  - T_m has stationary distribution f(θ_1, …., θ_K)
- So, if we sample M steps, and throw away the first few, the remaining T_m's can be treated like a sample from f(θ_1, …., θ_K)
  - Not an iid sample though! $\sqrt{M}$-law may not apply!

MCMC and BUGS (JAGS) / RUBE

- Working out the complete conditionals (CC's) is fun but error prone
- Working out how to sample from the CC's is fun but error-prone
- BUGS[1]/ JAGS[2] can work out CC's and sample from them, for a large set of models, automagically!
- RUBE is an interface to BUGS or JAGS that
  - checks syntax,
  - provides additional flexibility for writing models,
  - makes nice diagnostic plots, etc.

1. **B**ayesian inference **U**sing **G**ibbs **S**ampling
2. **J**ust **A**nother **G**ibbs **S**ampler

Mastery Probabilities: The JAGS Program

- **Level 1:**

  $x_i \sim NB(x|r,p_i)$, i=1, …, n

- **Level 2:**

  $p_i \sim Beta(p|\alpha,\beta)$,
  i = 1, …, n

- **Level 3:**

  $\alpha \sim Gamma(\alpha|a_1,b_1)$,
  $\beta \sim Gamma(\beta|a_2,b_2)$

Different writers & software tools use different notation – be careful!

R-like syntax for most things

Should specify constant parameters at highest level

```
"model {

  # LEVEL 1
  for (i in 1:n) {
    x[i] ~ dnegbin(p[i],r)
  }

  # LEVEL 2
  for (i in 1:n) {
    p[i] ~ dbeta(alpha,beta)
  }

  # LEVEL 3
  alpha ~ dgamma(1,1)
  beta ~  dgamma(1,1)

}"
```


Running JAGS from R with RUBE

```
> library(R2jags)
> library(rube)
> # 1. Write model for JAGS
> #    (prev slide!)
> # 2. Write the data as an
> #    R list, or data frame
> mastery.data <- list(x =
+ c(4,10,5,7,3), n=5, r=3)
> # 3. Make an "initial
> #    values" function
> mastery.inits<-function()
{ list(alpha=rgamma(1,1,1),
  beta = rgamma(1,1,1))
}
```

```
> # 4. Check JAGS model
> #    with RUBE
> rube(mastery.model,
+ mastery.data,
+ mastery.inits)
> # 5. Perform JAGS simu-
> #    lation with RUBE
> mcmc.3 <-
+ rube(mastery.model,
+ mastery.data,
+ mastery.inits,
+ parameters.to.save =
+ c("alpha","beta","p"))
```

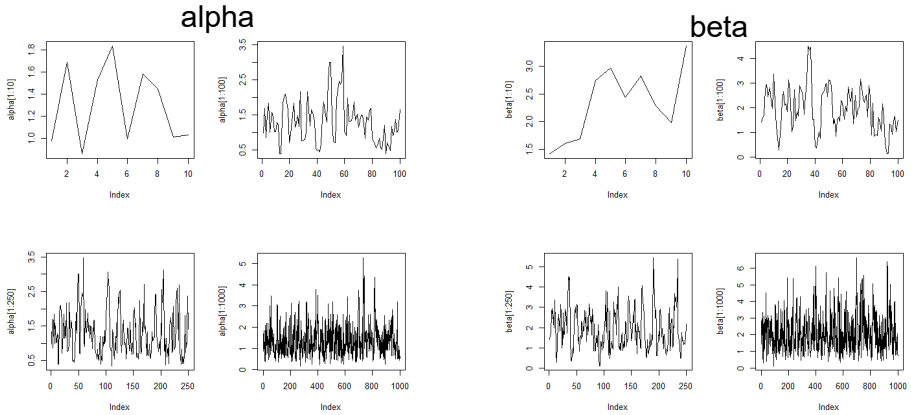
Results of MCMC Simulation

- *Normally, RUBE removes half the MCMC steps as "burn-in", runs 3 chains & produces ~ 300 samples per chain.*
- I adjusted the arguments to rube so that I got 1000 steps with no burn-in, for each chain. I looked carefully at the output for the first chain, for all seven parameters:
  - α, β, p_1, p_2, p_3, p_4 and p_5
- On the next couple of pages are pictures of the raw output, so you can see
  - What it looks like
  - What to look for

  [Can get this and other diagnostics with rube's p3() function!]
- After that, I present some results (CI's, the distribution of P[mastery] in the population, etc.)

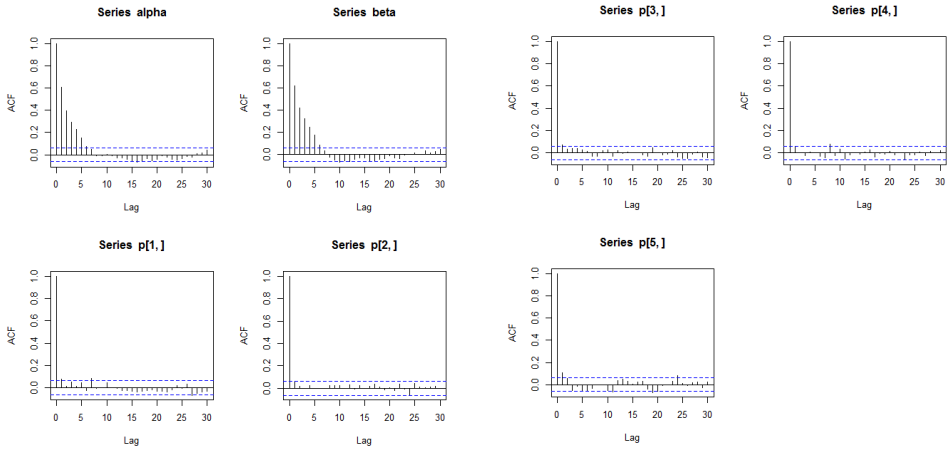
Raw MCMC output

- Two things you should always check:
  1. Graph the "random walk" of each parameter to decide whether the initial value has too much influence, the chain is "sticky" or appears not to be hovering around a good central value, etc.



Raw MCMC output

- Two things you should always check:
  2. Graph the "autocorrelation plot" of each parameter. Too much correlation makes using means, variances, etc. of the posterior sample more difficult



Results of the MCMC Simulation, I

- 95% CI's and (median) point estimates

	2.5%	50%	97.5%
alpha	0.41	1.25	3.21
beta	0.55	1.86	4.65
p1	0.15	0.41	0.72
p2	0.08	0.25	0.50
p3	0.13	0.37	0.67
p4	0.11	0.32	0.58
p5	0.18	0.47	0.78

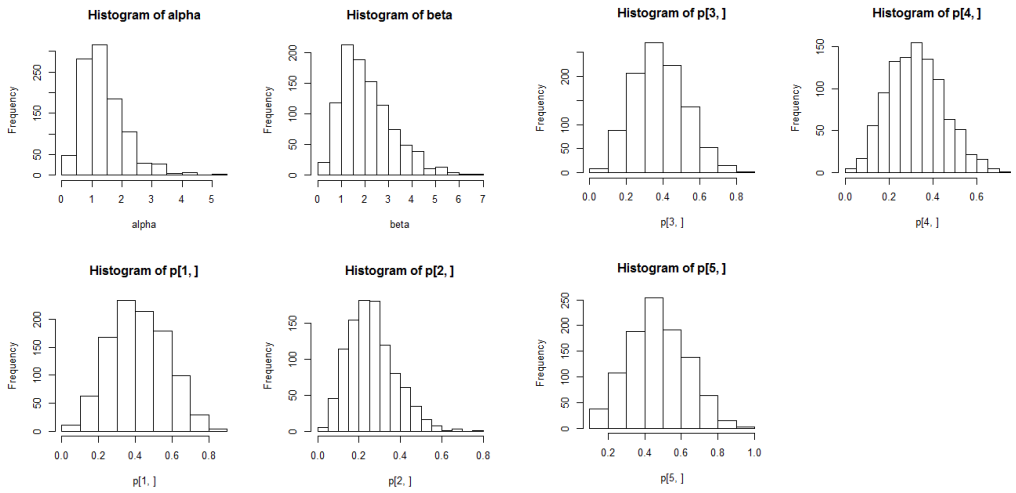
Results of the MCMC Simulation, I (Aside)

- It can be a good idea to check the correlation between parameters, to see whether there really is separable information about every parameter

	alpha	beta	p1	p2	p3	p4	p5
alpha	1.00	0.54	0.10	0.19	0.10	0.16	0.07
beta	0.54	1.00	-0.14	-0.01	-0.11	-0.08	-0.20
p1	0.10	-0.14	1.00	0.03	0.05	0.07	0.07
p2	0.19	-0.01	0.03	1.00	0.05	-0.01	0.04
p3	0.10	-0.11	0.05	0.05	1.00	0.01	0.11
p4	0.16	-0.08	0.07	-0.01	0.01	1.00	0.07
p5	0.07	-0.20	0.07	0.04	0.11	0.07	1.00

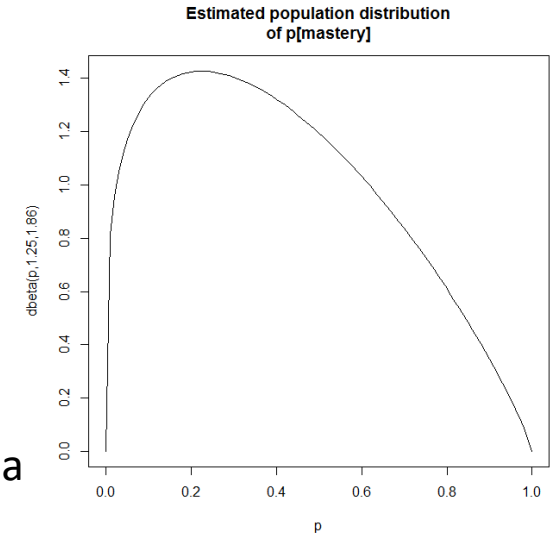
Results of the MCMC Simulation, II

- Histograms of the posterior distributions



Results of the MCMC Simulation, III

- A picture of the population distribution of P[mastery], as estimated from these 5 students
- Recall that there were 4, 10, 5, 7, & 3 failures before the 3rd success
- The estimates of p were 0.41, 0.25, 0.37, 0.32, 0.47
- Estimates of alpha, beta were 1.25 and 1.86



Recipe: Building and Fitting a Bayesian model with JAGS/rube MCMC

1. Write down the model as a hierarchical model
2. Rewrite the model in JAGS notation
3. Write the pieces rube() needs
   A. Model (as an extended text string)
   B. Data List
   C. Initial Values Function
4. Check syntax and modeling assumptions with rube()
5. Decide
   A. parameters.to.save: which parameters do you want simulation for?
   B. n.chains: how many MCMC chains to create?
6. Perform JAGS simulation with rube(), look with p3()

Example: Professor Smedley's Boxplots

- Three randomly-chosen students from Prof. Smedley's class took $x_1 = 3$, $x_2 = 10$ and $x_3 = 8$ days to learn boxplots
- We assume likelihood (level 1) is exponential:

$$f(x_i|\lambda) = \lambda e^{-\lambda x_i}, \ i = 1, \ldots, n$$

- We will assume a Gamma prior distribution (level 2) to start with, with α=1, β=2:

$$f(\lambda|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\lambda\beta}$$

1. Write the model down as a hierarchical model

- Level 1:

$$x_i \overset{iid}{\sim} \text{Expon}(\lambda), i = 1, \ldots, n$$

- Level 2:

$$\lambda \sim \text{Gamma}(\alpha, \beta)$$

2. Rewrite the model in JAGS notation

```
# LEVEL 1
for (i in 1:n) {
    x[i] ~ dexp(lambda)
  }

# LEVEL 2
lambda ~ dgamma(1, 2)
```


3. The pieces rube() needs:
A: model (as extended string)

```
M1 <- "model {

  for (i in 1:n) {
    x[i] ~ dexp(lambda)
  }

  lambda ~ dgamma(1,2)

}"
```

3. The pieces rube() needs: B: data list

```
data.list.1 <- list(x=c(3,10,8), n=3)
```

- Each data variable is one element in the list
- You can (& should) add other constants to the list (like sample size: n=3 in this case)
- When the data is in a data frame "my.data.frame", it usually works to do something like this:

```
my.data.list <- as.list(my.data.frame)
my.data.list$N <- dim(my.data.frame)[1]
my.data.list$J <- 35 # however many groups you
  have!
```


3. The pieces rube() needs: C: initial values function

```
inits.1 <- function () {
  list(lambda=rgamma(1,.5,1))
}
```

- Each parameter should get an initial value
  - If you forget, JAGS will supply a lame initial value
- Typical initial values:
  - Draw from the prior for that parameter (the lame JAGS value)
  - Draw from a distribution with greater variance than the posterior for that parameter ("overdispersed")
- In complicated problems, can take initial value from a simpler model (more on this in later lectures!)

4. Check syntax and assumptions with rube()

```
library(rube)
library(R2jags)
rube(M1,data.list.1,inits.1)
```

- If you do not supply "parameters.to.save" argument, rube() will not run JAGS, it will just check your model
- rube() will summarize
  - constants in your data.list (e.g. n=3)
  - data variables in your data.list (e.g. x=c(3,10,8)
  - distributions for data and parameters (dexp, dgamma)
  - initial values for parameters
- rube() will also try to catch syntax errors for you

5. Decide… A: Which parameters you want random samples of

```
parameters.to.save=c("lambda")
```

- If there is more than one, list each parameter name in quotes, separated by commas (e.g. c("lambda","alpha","beta")
- The more parameters you "save" random samples of, the larger the files you get back from JAGS
  - Eats disk space
  - Eats RAM, may slow down your laptop
- For small problems, this isn't an issue!

5. Decide… B: How many MCMC chains to produce

`n.chains=3`

- For the "first run" of a big/complicated problem, use n.chains=1
  - Something will go wrong
  - You won't have to wait as long to find out and fix it
- For the "last run" of a complicated problem, or if you just have a small problem to start with, use n.chains=3 or more
  - Three chains is usually enough to assess convergence
  - Rhat < 1.1 → MCMC sample is representative of the posterior

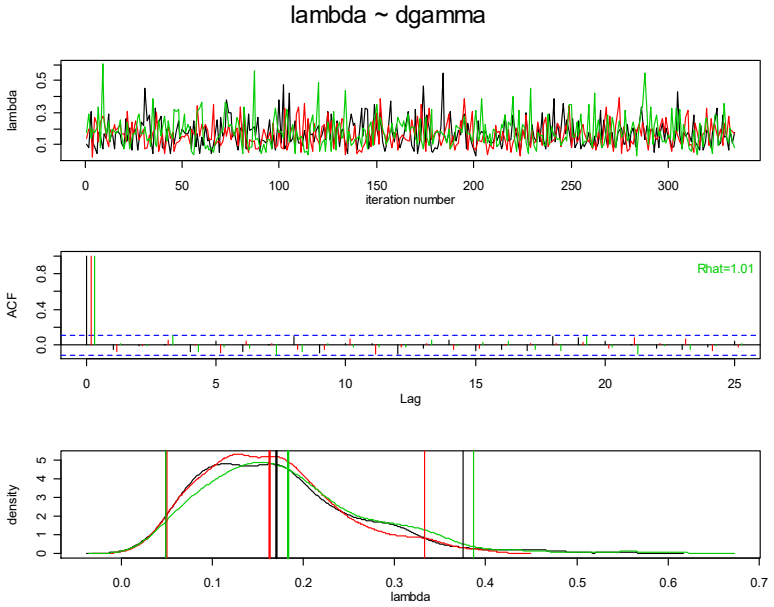
6. Perform JAGS simulation with rube(), look with p3()

```
> M1.fit <- rube(M1,data.list.1,inits.1,
  parameters.to.save=c("lambda"), n.chains=3)
> M1.fit

Rube Results:
Run at 2010-11-10 11:32 and taking 2.82 secs
          mean    sd MCMCerr    2.5%   25%    50%    75%  97.5% Rhat n.eff
lambda    0.172 0.085  0.0411  0.0496  0.11  0.163  0.216  0.361 1.01   310
deviance 18.529 1.203  0.0027 17.6800 17.77 18.045 18.820 21.696 1.02   290
DIC = 19.269

> p3(M1.fit)
```

6. Perform WinBUGS simulation with rube(), look with p3()



What is $\hat{R}$?

- Suppose we have M chains:

Chains				Means	Variances
$\theta^{(1;1)}$,	$\theta^{(1;2)}$,	$\ldots$,	$\theta^{(1;N)}$	$\overline{\theta}_1$	W_1
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
$\theta^{(M;1)}$,	$\theta^{(M;2)}$,	$\ldots$,	$\theta^{(M;N)}$	$\overline{\theta}_M$	W_M
			Grand mean	$\overline{\overline{\theta}}$	

- Define

$$W = \frac{1}{M(N-1)} \sum_{m=1}^{M} \sum_{n=1}^{N} (\theta^{(m;n)} - \bar{\theta}_n)^2 = \frac{1}{M} \sum_{m=1}^{M} W_m$$
$$= \text{Average within-chain variance}$$

$$B = \frac{M}{M-1} \sum_{m=1}^{M} (\bar{\theta}_m - \bar{\theta})^2$$
$$= \text{Between-chain variance, inflated for sample size}$$

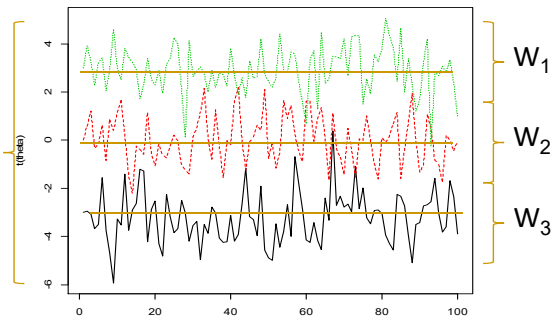
$$V = \frac{M-1}{M} W + \frac{1}{M} B$$
$$= \text{Pooled variance estimate,}$$

What is $\hat{R}$?

- Separated chains:

$$
\begin{aligned}
W &= \tfrac{1}{N}\textstyle\sum_{n=1}^{N} W_n &&= 1.02 \\
B &= \tfrac{M}{N-1}\textstyle\sum_{n=1}^{N}(\bar{\theta}_n - \bar{\theta})^2 &&= 938.53 \\
V &= \tfrac{M-1}{M}W + \tfrac{1}{M}B &&= 10.40
\end{aligned}
$$

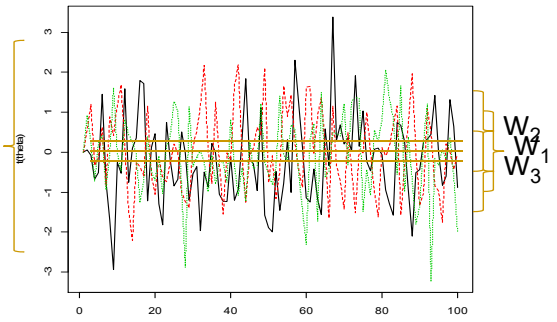
$$\hat{R} = \sqrt{V/W} = 3.19$$



- Converged chains:

$$
\begin{aligned}
W &= \tfrac{1}{N}\textstyle\sum_{n=1}^{N} W_n &&= 1.02 \\
B &= \tfrac{M}{N-1}\textstyle\sum_{n=1}^{N}(\bar{\theta}_n - \bar{\theta})^2 &&= 1.32 \\
V &= \tfrac{M-1}{M}W + \tfrac{1}{M}B &&= 1.03
\end{aligned}
$$

$$\hat{R} = \sqrt{V/W} = 1.00$$



Example: Minnesota Radon – Intercept Only

- MLM:

$$
\begin{aligned}
y_i &= \alpha_{0j[i]} + \epsilon_i, \\
&\quad \epsilon_i \sim N(0, \sigma^2) \\
\alpha_{0j} &= \beta_0 + \eta_j, \\
&\quad \eta_j \sim N(0, \tau^2)
\end{aligned}
$$

- Hierarchical:

$$\textbf{Level 1: } y_i \sim N(\alpha_{0j[i]}, \sigma^2)$$

$$\textbf{Level 2: } \alpha_{0j} \sim N(\beta_0, \tau^2)$$

- Demonstration in R and rube()/JAGS…
- (comparison with lmer also)

Multilevel Models in JAGS

- MLM form

$$\begin{aligned} y_i &= \alpha_{0j[i]} + \epsilon_i, \\ &\quad \epsilon_i \sim N(0, \sigma^2) \\ \alpha_{0j} &= \beta_0 + \eta_j, \\ &\quad \eta_j \sim N(0, \tau^2) \end{aligned}$$

- Hierarchical form

Level 1: $y_i \sim N(\alpha_{0j[i]}, \sigma^2)$

Level 2: $\alpha_{0j} \sim N(\beta_0, \tau^2)$

- BUGS/rube form

```
model {
  # LEVEL 1
  for (i in 1:N) {
    log.radon[i] ~
       dnorm(a0[county[i]],prec.y)
  }
  # LEVEL 2
  for (j in 1:MAX.COUNTY=85) {
    a0[j] ~ dnorm(b0,prec.cty)
  }

  # PRIORS/LEVEL 3:
  b0 ~ dnorm(0,PREC.b0=1e-6)
  prec.y ~ dgamma(ALPHA=0.001,BETA=0.001)
  prec.cty ~ dgamma(ALPHA=0.001,BETA=0.001)

  # CONVERT PRECISION TO VARIANCES
  var.y <- 1/prec.y
  var.cty <- 1/prec.cty
}
```


Multilevel Models in JAGS

- MLM form

$$\begin{aligned} y_i &= \alpha_{0j[i]} + \epsilon_i, \\ &\quad \epsilon_i \sim N(0, \sigma^2) \\ \alpha_{0j} &= \beta_0 + \eta_j, \\ &\quad \eta_j \sim N(0, \tau^2) \end{aligned}$$

- Hierarchical form

Level 1: $y_i \sim N(\alpha_{0j[i]}, \sigma^2)$

Level 2: $\alpha_{0j} \sim N(\beta_0, \tau^2)$

- BUGS/rube form

```
model {
  # LEVEL 1
  for (i in 1:N) {
    log.radon[i] ~
       dnorm(a0[county[i]],prec.y)
  }
  # LEVEL 2
  for (j in 1:MAX.COUNTY=85) {
    a0[j] ~ dnorm(b0,prec.cty)
  }

  # PRIORS/LEVEL 3:
  b0 ~ dnorm(0,PREC.b0=1e-6)
  prec.y ~ dgamma(ALPHA=0.001,BETA=0.001)
  prec.cty ~ dgamma(ALPHA=0.001,BETA=0.001)

  # CONVERT PRECISION TO VARIANCES
  var.y <- 1/prec.y
  var.cty <- 1/prec.cty
}
```

Multilevel Models in JAGS

- MLM form

$$\begin{aligned} y_i &= \alpha_{0j[i]} + \epsilon_i, \\ &\quad \epsilon_i \sim N(0, \sigma^2) \\ \alpha_{0j} &= \beta_0 + \eta_j, \\ &\quad \eta_j \sim N(0, \tau^2) \end{aligned}$$

- Hierarchical form

Level 1: $y_i \sim N(\alpha_{0j[i]}, \sigma^2)$

Level 2: $\alpha_{0j} \sim N(\beta_0, \tau^2)$

- BUGS/rube form

```
model {
  # LEVEL 1
  for (i in 1:N) {
    log.radon[i] ~
       dnorm(a0[county[i]],prec.y)
  }
  # LEVEL 2
  for (j in 1:MAX.COUNTY=85) {
    a0[j] ~ dnorm(b0,prec.cty)
  }

  # PRIORS/LEVEL 3:
  b0 ~ dnorm(0,PREC.b0=1e-6)
  prec.y ~ dgamma(ALPHA=0.001,BETA=0.001)
  prec.cty ~ dgamma(ALPHA=0.001,BETA=0.001)

  # CONVERT PRECISION TO VARIANCES
  var.y <- 1/prec.y
  var.cty <- 1/prec.cty
}
```


Multilevel Models in JAGS

- MLM form

$$\begin{aligned} y_i &= \alpha_{0j[i]} + \epsilon_i, \\ &\quad \epsilon_i \sim N(0, \sigma^2) \\ \alpha_{0j} &= \beta_0 + \eta_j, \\ &\quad \eta_j \sim N(0, \tau^2) \end{aligned}$$

- Hierarchical form

Level 1: $y_i \sim N(\alpha_{0j[i]}, \sigma^2)$

Level 2: $\alpha_{0j} \sim N(\beta_0, \tau^2)$

- BUGS/rube form

```
model {
  # LEVEL 1
  for (i in 1:N) {
    log.radon[i] ~
       dnorm(a0[county[i]],prec.y)
  }
  # LEVEL 2
  for (j in 1:MAX.COUNTY=85) {
    a0[j] ~ dnorm(b0,prec.cty)
  }

  # PRIORS/LEVEL 3:
  b0 ~ dnorm(0,PREC.b0=1e-6)
  prec.y ~ dgamma(ALPHA=0.001,BETA=0.001)
  prec.cty ~ dgamma(ALPHA=0.001,BETA=0.001)

  # CONVERT PRECISION TO VARIANCES
  var.y <- 1/prec.y
  var.cty <- 1/prec.cty
}
```

Look at the Distributions section (pp 56ff in manual14.pdf)

Multilevel Models in JAGS

- MLM form

$$\begin{aligned} y_i &= \alpha_{0j[i]} + \epsilon_i, \\ &\quad \epsilon_i \sim N(0, \sigma^2) \\ \alpha_{0j} &= \beta_0 + \eta_j, \\ &\quad \eta_j \sim N(0, \tau^2) \end{aligned}$$

- Hierarchical form

Level 1: $y_i \sim N(\alpha_{0j[i]}, \sigma^2)$

Level 2: $\alpha_{0j} \sim N(\beta_0, \tau^2)$

- BUGS/rube form

```
model {
  # LEVEL 1
  for (i in 1:N) {
    log.radon[i] ~
       dnorm(a0[county[i]],prec.y)
  }
  # LEVEL 2
  for (j in 1:MAX.COUNTY=85) {
    a0[j] ~ dnorm(b0,prec.cty)
  }

  # PRIORS/LEVEL 3:
  b0 ~ dnorm(0,PREC.b0=1e-6)
  prec.y ~ dgamma(ALPHA=0.001,BETA=0.001)
  prec.cty ~ dgamma(ALPHA=0.001,BETA=0.001)

  # CONVERT PRECISION TO VARIANCES
  var.y <- 1/prec.y
  var.cty <- 1/prec.cty
}
```


Multilevel Models in JAGS

- MLM form

$$\begin{aligned} y_i &= \alpha_{0j[i]} + \epsilon_i, \\ &\quad \epsilon_i \sim N(0, \sigma^2) \\ \alpha_{0j} &= \beta_0 + \eta_j, \\ &\quad \eta_j \sim N(0, \tau^2) \end{aligned}$$

- Hierarchical form

Level 1: $y_i \sim N(\alpha_{0j[i]}, \sigma^2)$

Level 2: $\alpha_{0j} \sim N(\beta_0, \tau^2)$

- BUGS/rube form

```
model {
  # LEVEL 1
  for (i in 1:N) {
    log.radon[i] ~
       dnorm(a0[county[i]],prec.y)
  }
  # LEVEL 2
  for (j in 1:MAX.COUNTY=85) {
    a0[j] ~ dnorm(b0,prec.cty)
  }

  # PRIORS/LEVEL 3:
  b0 ~ dnorm(0,PREC.b0=1e-6)
  prec.y ~ dgamma(ALPHA=0.001,BETA=0.001)
  prec.cty ~ dgamma(ALPHA=0.001,BETA=0.001)

  # CONVERT PRECISION TO VARIANCES
  var.y <- 1/prec.y
  var.cty <- 1/prec.cty
}
```

Have to add priors to all free parameters

What's new?

- summary(rube.object): point estimates and CI's for "some" parameters. Others available in
  - rube.object$mean
  - rube.object$sd
  - rube.object$median
  - rube.object$sims.list
  - etc
- p3(rube.object): interactive graphical summaries

What's new?

- rube()/WinBUGS/JAGS automatically
  - Runs 3 separate MCMC chains
  - Runs each MCMC chain for 2000 steps, and throws away the first 1000 steps as "burn-in"
  - Thins each chain by 1/3 to reduce autocorrelation
  - *You can change this when you run rube(); see pp. 25ff. of the "rube.pdf" documentation.*
- rube()/WinBUGS/JAGS reports an "Rhat" statistic for each parameter estimated
  - Rhat is a ratio of between-chain to within-chain variation
  - *When the chain is converged, Rhat < 1.2. Otherwise, the chain hasn't run long enough yet.*

Outline

- Bayesian Statistics and MCMC
- Distribution of Skill Mastery in a Population
- Digression: What is Markov Chain Monte Carlo (MCMC)
- Estimating the Distribution of Skills
  - JAGS & RUBE for MCMC
  - Checking the MCMC output
  - Results of the MCMC output
  - P3() function
- The JAGS recipe: Prof Smedley's Histograms
- MCMC for Hierarchical Linear Models: Minnesota Radon