MCMC Sampling in R Using JAGS

February 28, 2013

Overview

- Just Another Gibbs Sampler (JAGS)
  - Programming/modeling language based on similar syntax as Bayesian Inference Using Gibbs Sampling (BUGS)
  - Developed by Martyn Plummer for Mac/Unix
  - Adaptive Metropolis sampler for MCMC posterior inference on complex, multiparameter and hierarchical models
- Extensible to R through 'rjags' (or 'R2jags') and 'coda' packages
  - Throughput in R to call jags and collect results
  - WinBUGS or OpenBUGS for non-Mac users

Accessing and Installing JAGS

- JAGS (3.3.0) can be downloaded at:

`http://sourceforge.net/projects/mcmc-jags/files/`

- Consider installing Tcl/Tk libraries from for graphing capabilities:

`http://cran.r-project.org/bin/macosx/tools/`

- Install 'rjags' and 'coda' packages in R

JAGS Basics

- Inputs needed to run a model in JAGS:
  - Create a file containing the model syntax <**model.bug**>
  - Data inputs from R, JAGS data environment or data a file <**data.bug**>
- Steps to run the model
  - Compile and initialize the model, including starting values
  - Update the model for burnin
  - Set the monitor for the relevant parameters in the MCMC sampler
  - Update the model for analysis
  - Collect monitored parameters from the full updates
- Use 'rjags' to input data and perform all the necessary steps – all the action is in writing down the 'right' model

JAGS Models and Relation Objects

- A JAGS model contains a series of stochastic or deterministic relations between variables, each of which defines unique nodes
  - Nodes are evaluated in a directed acyclical graph – from parent to child (without recursion), i.e. from right to left
  - Unlike BUGS, nodes cannot be redefined – think, pointers in one direction
- Stochastic relations (~)
  - Define the probabilistic components – the prior and likelihood densities
- Deterministic relations (<–)
  - Define transformations of data or parameters – as exact transformations of parent nodes
- FOR EMPHASIS, once a node object is defined, it cannot be redefined AND circular child-parent-child definitions are inadmissible

Hierarchical Linear Model in JAGS

- The joint likelihood, $p(y|\mu, \sigma^2)$:

$$y|\mu, \sigma^2 \sim N(y|\mu, \sigma^2)$$

- The joint prior, $p(\mu, \sigma^2, \mu_0, \sigma_0^2, \tau_0^2)$:

$$\begin{aligned} \mu|\sigma^2 &\sim N(\mu_0, \sigma_0^2) \\ \sigma^2 &\sim \text{Inv-}\chi^2(n-1, \tau_0^2) \end{aligned}$$

- The hyperpriors (with known α, β, μ_{-1} and σ_{-1}^2):

$$\begin{aligned} \mu_0|\sigma_0^2 &\sim N(\mu_{-1}, \sigma_{-1}^2) \\ \sigma_0^2 &\sim \text{Inv-Gamma}(\alpha, \beta) \\ \tau_0^2 &\sim \text{Inv-Gamma}(\alpha, \beta) \end{aligned}$$

Hierarchical Linear Model in JAGS

```
model{
  for(i in 1:n){
    y[i] ~ dnorm(mu, sigma)}

  mu ~ dnorm(mu0, sigma0)
  s.sigma ~ dchisqr(tau0)
  sigma <- 1.0/s.sigma

  mu0 ~ dnorm(m1,sigma1)
  s.sigma0 ~ dgamma(alpha,beta)
  sigma0 <- 1.0/s.sigma0

  t.tau0 ~ dgamma(alpha,beta)
  tau0 <- 1.0/t.tau0

  alpha <- 2; beta <- 2; m1 <- 1; sigma1 <- 1
}
```

Hierarchical Linear Regression

- The joint likelihood, $p(y|x\beta, \sigma^2)$:

$$y|x\beta, \sigma^2 \sim N(y|x\beta, \sigma^2)$$

- The joint prior, $p(\beta, \sigma^2, \mu_0, \sigma_0^2, \tau_0^2)$:

$$\begin{aligned} \beta|\sigma^2 &\sim N(\mu_0, \sigma_0^2) \\ \sigma^2 &\sim \text{Inv-}\chi^2(n-1, \tau_0^2) \end{aligned}$$

- The hyperpriors (with known a, b, μ_{-1} and σ_{-1}^2):

$$\begin{aligned} \mu_0|\sigma_0^2 &\sim N(\mu_{-1}, \sigma_{-1}^2) \\ \sigma_0^2 &\sim \text{Inv-Gamma}(a, b) \\ \tau_0^2 &\sim \text{Inv-Gamma}(a, b) \end{aligned}$$

Hierarchical Linear Regression

```
model{
  for(i in 1:n){
    y[i] ~ dnorm(mu[i], sigma)
    mu[i] <- alpha + beta*x[i]}

  alpha ~ dnorm(mu0, sigma0)
  beta ~ dnorm(mu0, sigma0)
  s.sigma ~ dchisqr(tau0); sigma <- 1.0/s.sigma

  mu0 ~ dnorm(mu1,sigma1)
  s.sigma0 ~ dgamma(a,b)
  sigma0 <- 1.0/s.sigma0

  t.tau0 ~ dgamma(a,b)
  tau0 <- 1.0/t.tau0

  a <- 2; b <- 2; mu1 <- 1; sigma1 <- 1
}
```

Defining Data in JAGS

- Data must be specified for JAGS to utilize
  - Data defines any nodes that are unassigned in the model (or data) brackets

```
model{ ...
    y[i] ~ dnorm(mu[i], sigma)
    mu[i] <- alpha + beta*x[i]} ...
```

- JAGS utilizes a data file like `dump()` in R
  - 'rjags' automatically creates this file – can only parse scalars, arrays and matrices data structures
  - Manually the file (**data.bug**) looks like:

```
'y'<-
 c(1,2,3,4,5)
'x'<-
 c(-1,3,-3,-4,5)
```

Defining Data in JAGS

```
model{ ...
    y[i] ~ dnorm(mu[i], sigma)
    mu[i] <- alpha + beta*x[i]} ...
```

- What if we did not include values for y? What is the sampling doing?
  - The y (child) node child is defined by its right hand side (parent) nodes
  - Producing a random variable given μ and σ
- What about for x?
  - The model breaks since there is no information to infer about x from a parent node

Data Transformations in JAGS

- Data transformations are different in JAGS than in BUGS
  - Directed acyclical graph prevents nodes being defined twice

```
model{ ...
    z[i] ~ dnorm(mu[i], sigma)
    z[i] <- y[i] - mean(y)
```

  - ... is not allowed!!!
- Must utilize the data environment

```
data{
    z[i] <- y[i] - mean(y)}
model{
    x[i] ~ dnorm(mu[i], sigma)}
```

- Any node defined in `data{}` will be interpreted as fixed data in `model{}`

Compile and Initialize

- Once the data and model are defined, must compile everything and provide initial values and random number seeds (**initial.bug**)
  - Can be done automatically (i.e., blindly)
  - Useful to specify own starting values and seeds to ensure replication

```
".RNG.name" <- "base::Super-Duper"
".RNG.seed" <- 1005
"alpha" <- 0
"beta" <- 1
"gamma" <- c(1,1,1)
```

- From there the model can be updated until convergence, thinned, and rerun, etc

Running JAGS in R

- JAGS was written to be very extensible in R – use this feature to combine JAGS sampling and R data analysis
  - 'rjags' for the interface between R and JAGS
  - 'coda' for diagnostics tools

Runnng JAGS in R

- Compiling a JAGS model in R

```
model <- jags.model(
   file="/jags_models/linearBayes.bug",
   data=list("y"=y,"x"=x,"n"=n),
   n.chains=2, n.adapt=500)
```

- Updating the model and monitorning posterior samples

```
samples <- coda.samples(
   model,
   variable.names=c("beta","alpha"),
   n.iter=500,thin=10)
```

Some Useful Diagnostics in R

- With the model run, it is useful to do some diagnostics on convergence

```
# trace plot
plot(samples$mu)

# autocorrelation plot
autocorr.plot(samples$mu)

# check Gelman-Rubin convergence
gelman.diag(samples[,1:2])
```

Additional JAGS Resources

- Some additional resources:

`http://ftp.iinet.net.au/pub/FreeBSD/distfiles/mcmc-jags/jags_user_manual.pdf`

`http://jkarreth.myweb.uga.edu/bayes/jags.tutorial.pdf`

`http://www.jstatsoft.org/v36/c01/paper`

  - BUGS code for IRT and scaling models