Biostatistics 682: Applied Bayesian Inference Lecture 11: Introduction to JAGS

Jian Kang

Department of Biostatistics
University of Michigan, Ann Arbor

Background

- BUGS: Bayesian Inference Using Gibbs Sampling
- Examples used are WinBugs, OpenBUGS, and JAGS
- WinBugs
  - Developed in 1997 and only ran on Windows machines.
  - A point and click type program and your results will dump into coda files (G.S. output) which you can then read into R
  - Can be run on Unix or Mac using a windows simulator
- OpenBUGS
  - An open source version of WinBUGS; Runs on all platforms
  - Can be either menu or command line driven. Running from the command line requires BRugs package in R (NOT compatible with Unix or Linux).
  - Since BRugs currently does not work with all platforms, I prefer JAGS.

JAGS

- JAGS: Just Another Gibbs Sampler by Martyn Plummer in 2003

- It is a program for analysis of Bayesian hierarchical models using Markov Chain Monte Carlo (MCMC) simulation
- Created to make more similar to Classic Bugs as well as make improvements.
- Runs on all platforms.
- Can run either in JAGS and read coda into R or run JAGS directly from R.
- Utilizes the Adaptive Rejection Metropolis sampler.

Installation

- Download: `https://sourceforge.net/projects/mcmc-jags/`. Latest version: JAGS-4.3.0
- Mac users: Install the GNU Fortran library from the CRAN tools directory: `https://cran.r-project.org/bin/macosx/tools/`
- After the installation, start the Terminal (Mac) or Console (Windows) and type: jags. The following message indicates your installation is successful!

```
Welcome to JAGS 4.3.0 on Mon Nov 13 23:40:47 2017
JAGS is free software and comes with ABSOLUTELY NO WARRANTY
Loading module: basemod: ok
Loading module: bugs: ok
.
```

- In R, you need to install packages: "R2jags"

How to run JAGS?

Running a model refers to generating samples from the posterior distribution of the model parameters. This takes place in five steps:

- Description of the model
- Definition of the data
- Set initial values and parameters to simulate
- Run model fitting
- Diagnostics

Description of the model in JAGS

Consider a simple linear regression model

$$\begin{aligned} y_i &\sim \mathrm{N}\left\{\alpha + \beta(x_i - \bar{x}), \sigma^2\right\}, \qquad i = 1, \ldots, n \\ \alpha &\sim \mathrm{N}(0, 10^4), \\ \beta &\sim \mathrm{N}(0, 10^4), \\ \sigma^{-2} &\sim \mathrm{G}(0.1, 0, 1). \end{aligned}$$

- The model in JAGS is defined using a dialect of the BUGS language.
- It consists of a series of stochastic relations "$\sim$" and deterministic relation (arrows) "$< -$"

Define model using R2jags

```
linear.model.JAGS = function(){
  for(i in 1:n){
    y[i] ~ dnorm(mu[i],tau2)
    mu[i]<- alpha + beta*(x[i]-x.bar)
  }
  x.bar <- mean(x)
  alpha ~ dnorm(0.0, 1.0E-4)
  beta ~ dnorm(0.0, 1.0E-4)
  sigma2 <- 1.0/tau2
  tau2 ~ dgamma(0.1,0.1)
}
```

Relations

- Each relation defines a node in the model in terms of other nodes that appear on the right hand side.
- These are referred to as the parent nodes.
- Taken together, the nodes in the model (together with the parent/child relationships represented as directed edges) form a directed acyclic graph.
- The very top-level nodes in the graph, with no parents, are constant nodes, which are defined either in the model definition (e.g. 1.0E-3), or in the data file (e.g. x[1]).
- Relations can be of two types.
  - A stochastic relation ($\sim$) defines a stochastic node, representing a random variable in the model.
  - A deterministic relation ($< -$) defines a deterministic node, the value of which is determined exactly by the values of its parents.

Distributions

- Distributions are used to define stochastic nodes using the "$\sim$" operator.
- Some distributions have restrictions on the valid parameter values,
- If a Distribution is given invalid parameter values when evaluating the loglikelihood, it returns $-\infty$.
- When a model is initialized, all stochastic nodes are checked to ensure that the initial parameter values are valid for their distribution

Univariate Continuous Distributions

Name	Usage	Density	Lower	Upper
Beta	`dbeta(a,b)` $a > 0,\ b > 0$	$\frac{x^{a-1}(1-x)^{b-1}}{\beta(a,b)}$	0	1
Chi-square	`dchisqr(k)` $k > 0$	$\frac{x^{\frac{k}{2}-1}\exp(-x/2)}{2^{\frac{k}{2}}\Gamma(\frac{k}{2})}$	0	
Double exponential	`ddexp(mu,tau)` $\tau > 0$	$\tau\exp(-\tau\lvert x-\mu\rvert)/2$		
Exponential	`dexp(lambda)` $\lambda > 0$	$\lambda\exp(-\lambda x)$	0	
F	`df(n,m)` $n > 0,\ m > 0$	$\frac{\Gamma(\frac{n+m}{2})}{\Gamma(\frac{n}{2})\Gamma(\frac{m}{2})}\left(\frac{n}{m}\right)^{\frac{n}{2}} x^{\frac{n}{2}-1}\left\{1+\frac{nx}{m}\right\}^{-\frac{(n+m)}{2}}$	0	
Gamma	`dgamma(r, lambda)` $\lambda > 0,\ r > 0$	$\frac{\lambda^r x^{r-1}\exp(-\lambda x)}{\Gamma(r)}$	0	
Generalized gamma	`dgen.gamma(r,lambda,b)` $\lambda > 0,\ b > 0,\ r > 0$	$\frac{b\lambda^{br}x^{br-1}\exp\{-(\lambda x)^b\}}{\Gamma(r)}$	0	
Logistic	`dlogis(mu, tau)` $\tau > 0$	$\frac{\tau\exp\{(x-\mu)\tau\}}{[1+\exp\{(x-\mu)\tau\}]^2}$		
Log-normal	`dlnorm(mu,tau)` $\tau > 0$	$\left(\frac{\tau}{2\pi}\right)^{\frac{1}{2}} x^{-1}\exp\left\{-\tau(\log(x)-\mu)^2/2\right\}$	0	
Noncentral Chi-squre	`dnchisqr(k, delta)` $k > 0, \delta \geq 0$	$\sum_{r=0}^{\infty}\frac{\exp(-\frac{\delta}{2})(\frac{\delta}{2})^r}{r!}\frac{x^{(k/2+r-1)}\exp(-\frac{x}{2})}{2^{(k/2+r)}\Gamma(\frac{k}{2}+r)}$	0	
Normal	`dnorm(mu,tau)` $\tau > 0$	$\left(\frac{\tau}{2\pi}\right)^{\frac{1}{2}}\exp\{-\tau(x-\mu)^2/2\}$		
Pareto	`dpar(alpha, c)` $\alpha > 0,\ c > 0$	$\alpha c^{\alpha}x^{-(\alpha+1)}$	c	
Student t	`dt(mu,tau,k)` $\tau > 0,\ k > 0$	$\frac{\Gamma(\frac{k+1}{2})}{\Gamma(\frac{k}{2})}\left(\frac{\tau}{k\pi}\right)^{\frac{1}{2}}\left\{1+\frac{\tau(x-\mu)^2}{k}\right\}^{-\frac{(k+1)}{2}}$		
Uniform	`dunif(a,b)` $a < b$	$\frac{1}{b-a}$	a	b
Weibull	`dweib(v, lambda)` $v > 0,\ \lambda > 0$	$v\lambda x^{v-1}\exp(-\lambda x^v)$	0	

Univariate Discrete Distributions

Name	Usage	Density	Lower	Upper
Beta	`dbetabin(a, b, n)`	$\binom{a+x-1}{x}\binom{b+n-x-1}{n-x}\binom{a+b+n-1}{n}^{-1}$	0	n
binomial	$a > 0, b > 0, n \in \mathbb{N}^*$			
Bernoulli	`dbern(p)`	$p^x(1-p)^{1-x}$	0	1
	$0 < p < 1$			
Binomial	`dbin(p,n)`	$\binom{n}{x}p^x(1-p)^{n-x}$	0	n
	$0 < p < 1,\ n \in \mathbb{N}^*$			
Categorical	`dcat(pi)`	$\frac{\pi_x}{\sum_i \pi_i}$	1	N
	$\pi \in (\mathbb{R}^+)^N$			
Noncentral	`dhyper(n1,n2,m1,psi)`	$\frac{\binom{n_1}{x}\binom{n_2}{m_1-x}\psi^x}{\sum_i \binom{n_1}{i}\binom{n_2}{m_1-i}\psi^i}$	$\max(0, n_+ - m_1)$	$\min(n_1, m_1)$
hypergeometric	$0 \le n_i,\ 0 < m_1 \le n_+$			
Negative	`dnegbin(p, r)`	$\binom{x+r-1}{x}p^r(1-p)^x$	0	
binomial	$0 < p \le 1,\ r \ge 0$			
Poisson	`dpois(lambda)`	$\frac{\exp(-\lambda)\lambda^x}{x!}$	0	
	$\lambda > 0$			

Multivariate Distributions

Name	Usage	Density
Dirichlet	`p ~ ddirch(alpha)` $\alpha_j \geq 0$	$\Gamma(\sum_i \alpha_i) \prod_j \frac{p_j^{\alpha_j - 1}}{\Gamma(\alpha_j)}$
Multivariate normal	`x ~ dmnorm(mu,Omega)` Ω $p \times p$ positive definite	$\lvert\Omega\rvert^{\frac{1}{2}} (2\pi)^{-\frac{p}{2}} \exp\{-(x-\mu)^T \Omega (x-\mu)/2\}$
Wishart	`Omega ~ dwish(R,k)` R $p \times p$ pos. def., $k \geq p$	$\frac{\lvert\Omega\rvert^{(k-p-1)/2} \lvert R\rvert^{k/2} \exp\{-\mathrm{Tr}(R\Omega/2)\}}{2^{pk/2}\Gamma_p(k/2)}$
Multivariate Student t	`x ~ dmt(mu,Omega,k)` Ω pos. def.	$\frac{\Gamma\{(k+p)/2\}}{\Gamma(k/2)(n\pi)^{p/2}} \lvert\Omega\rvert^{1/2} \{1 + \frac{1}{k}(x-\mu)^T \Omega (x-\mu)\}^{-\frac{(k+p)}{2}}$
Multinomial	`x ~ dmulti(pi, n)` $\sum_j x_j = n$	$n! \prod_j \frac{\pi_j^{x_j}}{x_j!}$

Functions

- Functions allow deterministic nodes to be defined using the "$< -$" operator.
- Most of the functions in JAGS are scalar functions taking scalar arguments.
- JAGS allows arbitrary vector- and array-valued functions, such as the matrix multiplication operator $\% * \%$ and the transpose function $t()$.

Scalar Functions

Usage	Description	Value	Restrictions on arguments
`abs(x)`	Absolute value	Real	
`arccos(x)`	Arc-cosine	Real	$-1 < x < 1$
`arccosh(x)`	Hyperbolic arc-cosine	Real	$1 < x$
`arcsin(x)`	Arc-sine	Real	$-1 < x < 1$
`arcsinh(x)`	Hyperbolic arc-sine	Real	
`arctan(x)`	Arc-tangent	Real	
`arctanh(x)`	Hyperbolic arc-tangent	Real	$-1 < x < 1$
`cos(x)`	Cosine	Real	
`cosh(x)`	Hyperbolic Cosine	Real	
`cloglog(x)`	Complementary log log	Real	$0 < x < 1$
`equals(x,y)`	Test for equality	Logical	
`exp(x)`	Exponential	Real	
`icloglog(x)`	Inverse complementary log log function	Real	
`ifelse(x,a,b)`	If x then a else b	Real	
`ilogit(x)`	Inverse logit	Real	
`log(x)`	Log function	Real	$x > 0$
`logfact(x)`	Log factorial	Real	$x > -1$
`loggam(x)`	Log gamma	Real	$x > 0$
`logit(x)`	Logit	Real	$0 < x < 1$
`phi(x)`	Standard normal cdf	Real	
`pow(x,z)`	Power function	Real	If $x < 0$ then z is integer
`probit(x)`	Probit	Real	$0 < x < 1$
`round(x)`	Round to integer away from zero	Integer	
`sin(x)`	Sine	Real	
`sinh(x)`	Hyperbolic Sine	Real	
`sqrt(x)`	Square-root	Real	$x >= 0$
`step(x)`	Test for $x \geq 0$	Logical	
`tan(x)`	Tangent	Real	
`tanh(x)`	Hyperbolic Tangent	Real	
`trunc(x)`	Round to integer towards zero	Integer	

Distribution, Density, Quantile Functions

Distribution	Density	Distribution	Quantile
Bernoulli	dbern	pbern	qbern
Beta	dbeta	pbeta	qbeta
Binomial	dbin	pbin	qbin
Chi-square	dchisqr	pchisqr	qchisqr
Double exponential	ddexp	pdexp	qdexp
Exponential	dexp	pexp	qexp
F	df	pf	qf
Gamma	dgamma	pgamma	qgamma
Generalized gamma	dgen.gamma	pgen.gamma	qgen.gamma
Noncentral hypergeometric	dhyper	phyper	qhyper
Logistic	dlogis	plogis	qlogis
Log-normal	dlnorm	plnorm	qlnorm
Negative binomial	dnegbin	pnegbin	qnegbin
Noncentral Chi-square	dnchisqr	pnchisqr	qnchisqr
Normal	dnorm	pnorm	qnorm
Pareto	dpar	ppar	qpar
Poisson	dpois	ppois	qpois
Student t	dt	pt	qt
Weibull	dweib	pweib	qweib

Scalar value with general input functions

Function	Description	Restrictions
`inprod(x1,x2)`	Inner product	Dimensions of $x1$, $x2$ conform
`interp.lin(e,v1,v2)`	Linear Interpolation	e scalar, $v1, v2$ conforming vectors
`logdet(m)`	Log determinant	m is a symmetric positive definite ma
`max(x1,x2,...)`	Maximum element among all arguments	
`mean(x)`	Mean of elements of x	
`min(x1,x2,...)`	Minimum element among all arguments	
`prod(x)`	Product of elements of x	
`sum(x)`	Sum of elements of x	
`sd(x)`	Standard deviation of elements of x	

Vector or Matrix value Functions

Usage	Description	Restrictions
`inverse(a)`	Matrix inverse	a is a symmetric positive definite matrix
`rank(v)`	Ranks of elements of v	v is a vector
`order(v)`	Ordering permutation of v	v is a vector
`sort(v)`	Elements of v in order	v is a vector
`t(a)`	Transpose	a is a matrix
`a %*% b`	Matrix multiplication	a, b conforming vector or matrices

Arrays

- Nodes defined by a relation are embedded in named arrays.
- Array names may contain letters, numbers, decimal points and underscores, but they must start with a letter.
- The node array “mu” is a vector of length “n” containing n nodes (mu[1], $\cdots$, mu[n]).
- The node array “alpha” is a scalar. Hence the array “alpha” contains a single node “alpha[1]”(The same as `R`). The same for “tau2”, “beta” and “sigma2”.
- Node arrays can be traveled with for loops.

Definition of Data (R2jags)

```
#simulate data
n = 100
x = rnorm(n,3.0)
y = 1.0 + 2.0*(x−3.0) + rnorm(n,sd=0.5)
#define data
dat.JAGS = list(y = y, x = x, n = n)
```

Set initial values (R2jags)

```
#set initial values
inits.JAGS = list(list(alpha=0.0,beta=0.0,tau2=1.0))

#multiple initial values
inits.JAGS = list(list(alpha=0.0,beta=0.0,tau2=1.0),
                  list(alpha=10.0,beta=10.0,tau2=10.0),
                  list(alpha=-10.0,beta=-10.0,tau2=100.0))

#random initial values
inits.JAGS = function(){
  return(list(alpha=rnorm(1),beta=rnorm(1),tau2=rgamma(0.1,0.1)))
}
```

Set parameters to simulate (R2jags)

```
#set parameters to simulate
para.JAGS = c("alpha","beta","tau2","sigma2")
```

Run JAGS (R2jags)

```
fit.JAGS = jags(data=dat.JAGS,
                inits=inits.JAGS,
                parameters.to.save = para.JAGS,
                n.chains=1,
                n.iter=9000,
                n.burnin=1000,
                model.file=linear.model.JAGS)


Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 100
   Unobserved stochastic nodes: 3
   Total graph size: 511

Initializing model

  |**************************************************| 100%
```

Diagnostics

- Summarize Results
- Trace Plots
- Autocorrelation Plots
- Gelman-Rubin Diagnostic

Summarize Results

```
> print(fit.JAGS)

Inference for Bugs model at "xxxx.txt", fit using jags,
 1 chains, each with 9000 iterations (first 1000 discarded), n.thin = 8
 n.sims = 1000 iterations saved
         mu.vect sd.vect    2.5%     25%     50%     75%   97.5%
alpha      1.260   0.049   1.163   1.229   1.261   1.291   1.351
beta       2.009   0.054   1.910   1.973   2.008   2.046   2.112
sigma2     0.256   0.038   0.191   0.230   0.253   0.279   0.337
tau2       3.996   0.582   2.972   3.590   3.953   4.357   5.233
deviance 145.130   2.386 142.352 143.414 144.566 146.176 151.937

DIC info (using the rule, pD = var(deviance)/2)
pD = 2.8 and DIC = 148.0
DIC is an estimate of expected predictive error (lower deviance is better).
```

Deviance information criterion (DIC)

- An information criterion for hierarchical modeling
- Extension of the Akaike information criterion (AIC) and the Bayesian information criterion (BIC)
- Deviance $D(\theta) = -2\log\{\pi(y \mid \theta)\} + 2\log\{\pi(y)\}$
- The effective number of parameters in the model

$$p_D = \mathrm{E}\{D(\theta) \mid y\} - D(\tilde{\theta}),$$

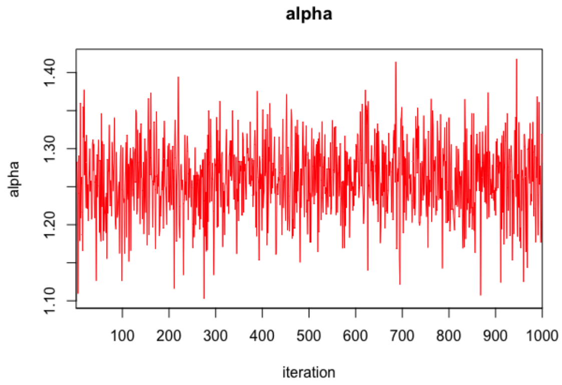
  where $\tilde{\theta} = \mathrm{E}(\theta \mid y)$.
- Deviance information criterion:

$$\mathrm{DIC} = -2\log\{\pi(y \mid \tilde{\theta})\} + 2p_D$$

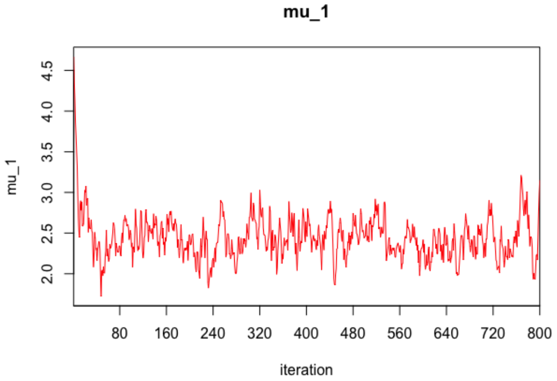
Trace plots

A trace plot is a time series plot of the parameter that we monitor as the Markov chain proceeds.

```
> traceplot(fit.JAGS)
```



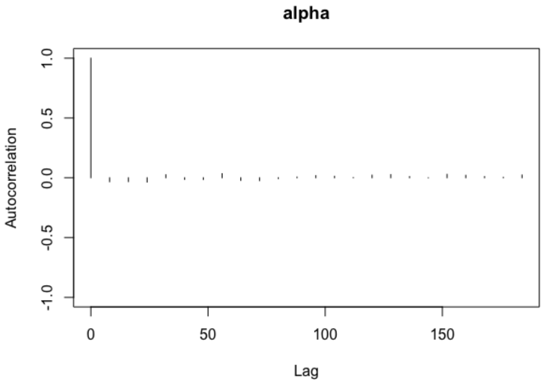
Trace plots (Mixing slow)



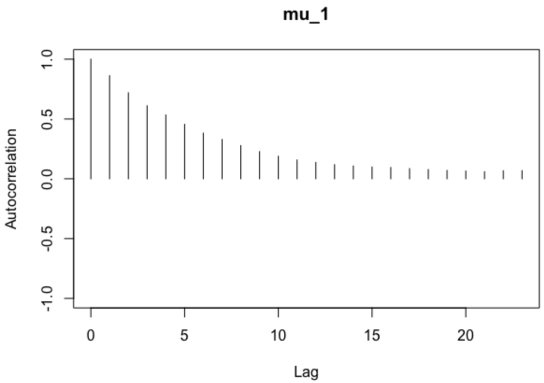
Autocorrelation Plot

An autocorrelation plot graphically measures the correlation between X_1 and each X_{k+1} variable in the chain, for $k = 1, \ldots,$

```
> fit.JAGS.mcmc = as.mcmc(fit.JAGS)
> autocorr.plot(fit.JAGS.mcmc,auto.layout = FALSE)
```



Autocorrelation Plot (Mixing slow)



Gelman-Rubin Diagnostic

- Compute m independent Markov chains
- Compares variance of each chain to pooled variance
- Provides estimate of how much variance could be reduced by running chains longer
- $\hat{R}$ is approaching one as the chain runs longer

$$W = \frac{1}{m}\sum_{j=1}^{m} s_j^2 \qquad \bar{\bar{\theta}} = \frac{1}{m}\sum_{j=1}^{m} \bar{\theta}_j$$

$$B = \frac{n}{m-1}\sum_{j=1}^{m}(\bar{\theta}_j - \bar{\bar{\theta}})^2 \qquad s_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\theta_{ij} - \bar{\theta}_j)^2$$

$$\hat{\mathrm{Var}}(\theta) = (1 - \frac{1}{n})W + \frac{1}{n}B \qquad \hat{R} = \sqrt{\frac{\hat{\mathrm{Var}}(\theta)}{W}}$$

Gelman-Rubin Diagnostic in R

- Run multiple chains, e.g. `n.chain=3` with random initial values
- Using the function `gelman.diag` in package `coda`

```
> gelman.diag(fit.JAGS.mcmc)
Potential scale reduction factors:

         Point est. Upper C.I.
alpha          1.00       1.00
beta           1.00       1.01
deviance       1.01       1.01
sigma2         1.00       1.00
tau2           1.00       1.00

Multivariate psrf

1
```