STK 880: Bayesian Statistical Modelling and Computing using R and JAGS

Working with JAGS and R

Dr. Rianne Jacobs

University of Groningen, The Netherlands

3 - 5 October 2019

https://github.com/Stattie/JagsTutorial.git

Run JAGS from R

- Install and load packages

```
> install.packages("runjags")
> install.packages("coda")
> library("runjags")
> library("coda")
```

- Set working directory

```
> setwd("C:/Documents/STK880_2018/Jags_Tutorial/")
```

- Load the data and check

```
> ALP <- read.table("data/ALP.txt", header = TRUE)
> head(ALP)
> str(ALP)
> summary(ALP)
```

Run JAGS from R

- Data transformations

```
> ALP$alkfos_tr <- 100/sqrt(ALP$alkfos)
> hist(ALP$alkfos_tr)
```

- Create model file and check

```
> file.show("osteo.model.txt")
```

- Prepare data and gather in a list

```
> alp <- ALP[ALP$artikel==0,"alkfos_tr"]
> n.obs <- length(alp)
> data.list <- list(y = alp, n.obs = n.obs)
```

Run JAGS from R

- Prepare initial values
  - Explicitly for each chain

```
> inits.list <- list(
+ list(mu = 5.5, tau = 2.5),
+ list(mu = 5.2, tau = 2.2))
```

  - As a function

```
> inits.list <- function()(list(mu = rnorm(1,5.5,1),
+ tau = runif(1,2,3)))
```

- Parameters to monitor - those for which you want a posterior sample

```
> parameters <- c("mu","sig2")
```

Run JAGS from R

- Run the `run.jags` function in R

```
> post.sim <- run.jags(model = "NormalDistribution",
+ data = data.list, inits = inits.list,
+ monitor = parameters,
+ n.chains = 2, burnin = 500, thin = 1, sample = 5000)
```

  - Provide the model, data, initial values and parameter vector to monitor
  - number of chains (n.chains $= 2$), number of burnin iterations (burnin $= 500$), thinning factor (thin $= 1$), number of iterations per chain (sample $= 5000$)

- Produce plots

```
> plot(post.sim, plot.type = c("trace"), vars = "mu")
```

- Produce summary statistics

```
> summary(post.sim)
```

Run JAGS from R

- Convert `run.jags` object to `mcmc` object

```
> samples <- as.mcmc(post.sim)
```

- CODA plots and functions

```
> par(mfrow=c(2,2)) # plot figures in 2x2 format
> traceplot(samples) # trace plots
> cumuplot(samples,ask=FALSE) # running mean plots
> acfplot(samples) # autocorrelation function plot
> autocorr(samples) # autocorrelation values
> crosscorr.plot(samples) # cross-correlation output
> densplot(samples) # density plots of the marginal s
> effectiveSize(samples) # effective size
> HPDinterval(samples) # HPD intervals
```

Run JAGS from R

- CODA convergence diagnostics

```
> gelman.diag(osteo.mcmc)
> gelman.plot(osteo.mcmc,ask=FALSE)
> geweke.diag(osteo.mcmc)
> geweke.plot(osteo.mcmc,ask=FALSE)
```

- Interpret results

Adaptation and burnin

- When a model is initialized, it may be in adaptive mode
- Samplers used by the model may modify their behaviour for increased efficiency
- Sequence generated by an adapting sampler is no longer a Markov chain - not guaranteed to converge to the target distribution
- Sufficient number of iterations must take place after the adaptive phase to ensure successful burnin
- `run.jags` default: adapt $= 1000$, burnin $= 4000$

Choice of priors

- Incorporating prior knowledge
- Unique feature for Bayesian approach
- But might introduce subjectivity - Good to include sensitivity analysis
- Different kinds of priors
  - Conjugate
  - Noninformative
  - Informative

Conjugate priors

Table 5.2 Common members of the exponential family and their associated (natural) conjugate prior.

Exponential family member		Parameter	Conjugate prior
Univariate case			
Discrete distributions			
Bernoulli	Bern(θ)	θ	Beta(α_0, β_0)
Binomial	Bin(n, θ)	θ	Beta(α_0, β_0)
Negative binomial	NB(k, θ)	θ	Beta(α_0, β_0)
Poisson	Poisson(θ)	θ	Gamma(α_0, β_0)
Continuous distributions			
Normal-variance fixed	N(μ, σ^2)-σ^2 fixed	μ	N(μ_0, σ_0^2)
Normal-mean fixed	N(μ, σ^2)-μ fixed	μ	IG(α_0, β_0) Inv-χ^2(ν_0, τ_0^2)
Normal*	N(μ, σ^2)	μ, σ^2	NIG(μ_0, κ_0, a_0, b_0) N-Inv-χ^2(μ_0, κ_0, ν_0, τ_0^2)
Exponential	Exp(λ)	λ	Gamma(α_0, β_0)

Conjugate priors

Multivariate case			
Discrete distributions			
Multinomial	Mult(n, $\boldsymbol{\theta}$)	$\boldsymbol{\theta}$	Dirichlet($\boldsymbol{\alpha}_0$)
Continuous distributions			
Normal-covariance fixed	N($\boldsymbol{\mu}$, Σ)-Σ fixed	$\boldsymbol{\mu}$	N($\boldsymbol{\mu}_0$, Σ_0)
Normal-mean fixed	N($\boldsymbol{\mu}$, Σ)-$\boldsymbol{\mu}$ fixed	Σ	IW(Λ_0, ν_0)
Normal*	N($\boldsymbol{\mu}$, Σ)	$\boldsymbol{\mu}$, Σ	NIW($\boldsymbol{\mu}_0$, κ_0, ν_0, Λ_0)

- Conjugacy: Posterior same type as prior
- Computational advantage in JAGS
- Interpretation: likelihood of historical data can be easily turned into a conjugate prior

Noninformative priors

- Expresses no knowledge
- non-subjective, objective, default, reference, weak, diffuse, flat, conventional, minimally informative, ...
- Noninformative on what scale?
- Always good to do a sensitivity analysis
- Ignorance cannot be expressed mathematically
- Jeffrey's priors most popular (pgs 115-117 (Lesaffre, Lawson, 2012))
- In practice
  - Locally uniform, vague or weak
  - e.g. $\beta_1 \sim \mathrm{N}(0, 100^2)$
  - e.g. $1/\sigma^2 \sim \mathrm{Gamma}(0.001, 0.001)$

Informative priors

- Examples
  - Based on historical studies
  - Literature
  - Expert knowledge
- Make sure that informative prior has a sound scientific basis

The JAGS model

Series of relations inside a block delimited by curly brackets { and } and preceded by the keyword model

```
model {
  for (i in 1:N) {
    Y[i] ~ dnorm(mu[i], tau)
    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)
  sigma <- 1.0/sqrt(tau)
  tau ~ dgamma(1.0E-3, 1.0E-3)
}
```

The JAGS model

- Each relation defines a node in the model
- Nodes in the model form a directed acyclic graph
- Two types of relations
  - Stochastic relation ($\sim$) defines a stochastic node - random variable in the model
  - Deterministic relation ($<-$) defines a deterministic node - value is determined exactly by the values of its parents
- See JAGS manual for available distributions
- NOTE: $\mathrm{N}(\mu, \tau)$, $\tau = 1/\sigma^2$ - precision
- Use of scalars, vectors, matrices, arrays as in R

The JAGS model

- For loops

```
for (i in 1:3) {
  Y[i] ~ dnorm(mu, tau)
}
```

- is equivalent to

```
Y[1] ~ dnorm(mu, tau)
Y[2] ~ dnorm(mu, tau)
Y[3] ~ dnorm(mu, tau)
```

Compiling the JAGS model

Compilation may fail for a number of reasons

- The model uses a function or distribution that has not been defined in any of the loaded modules
- The graph contains a directed cycle. These are forbidden in JAGS.
- A node is used in an expression but never defined by a relation or supplied with the data.

Other runjags functions and options

- `run.jags`
- `extend.jags` to continue running the model
- `autorun.jags` takes control of the run-length of the MCMC process and runs the chain until convergence
- parallel processing
  - `method = "parallel"`
  - runs each chain on a different core
- modules
  - `modules = glm`
  - The `glm` module implements samplers for efficient updating of generalized linear mixed models.
  - block updating - no need to centre predictor variables

Now let's get down to some programming