MCMC, R & JAGS (or OpenBUGS)

JAGS course, Part II

Inga Schwabe

January 12, 2014

Outline

- Markov Chain Monte Carlo (MCMC) sampling
- Regression analysis
- R: Short introduction
- Calling JAGS/OpenBugs from R
- Analysis of output

Markov Chain Monte Carlo (MCMC)

- When analytically deriving posterior distribution is tiresome
- For example: Gibbs sampler (used by e.g. JAGS/OpenBugs)
- Iteratively drawing samples from the full conditonal distributions of all unobserved parameters of a model
- Full distribution of a parameter = distribution of that parameter given the current or known values of all parameters in the model
- All steps: Starting values, n so called burn in iterations, then: subsequent draws from the joint posterior distribution ($N - n$ when N = total number of iterations)

Gibbs sampler: Example

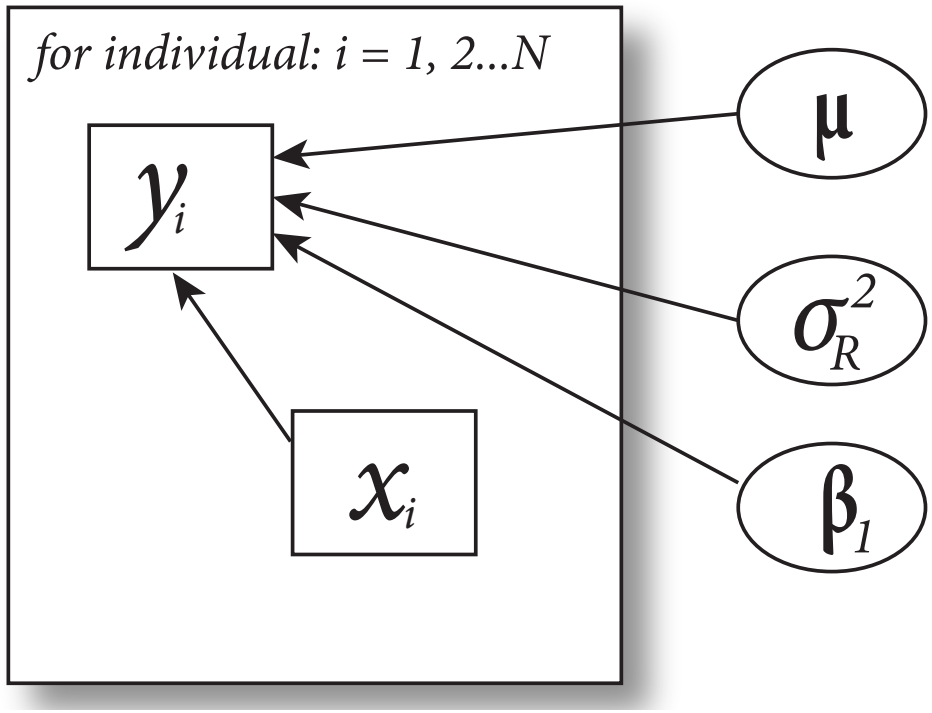
- For example: parameters ν and ξ. Y = data
- Might be difficult to sample from directly, but we can maybe sample directly from the conditional distributions: $P(\nu|\xi, Y)$, $P(\xi|\nu, Y)$
- Gibbs sampler: In iteration N, we first sample $\nu_k \sim P(\nu|\xi_{k-1}, Y)$ and then $\xi_k \sim P(\xi|\nu_{k-1}, Y)$
- After n burn in iterations, we sample from the joint posterior distribution of ν and ξ

Linear regression

$$y_i = \mu + \beta_1 x_i + \epsilon_i$$

$$\epsilon_i \sim N(0, \sigma_R^2)$$
$$y_i \sim N(\mu + \beta_1 x_i, \sigma_R^2)$$

Data: x_i, y_i
Estimation of μ, β_1 & σ_R^2



Model in JAGS and OpenBugs

JAGS:

```
model{
for (i in 1:N){
   y[i] ~ dnorm(mu + b*x[i], tau_r)
}

mu ~ dnorm(0, .1)
b ~ dnorm(0, .1)
tau_r ~ dgamma(1,1)
}
```

OpenBugs:

```
model{
for (i in 1:N){
   y[i] ~ dnorm(y_hat[i], tau_r)
   y_hat[i] <- mu + b*x[i]
}

mu ~ dnorm(0, .1)
b ~ dnorm(0, .1)
tau_r ~ dgamma(1,1)
}
```

A very short introduction to R syntax

```
setwd("M:/R/") #Set working directory
install.packages("rjags") #Install a package
library(rjgas) #Load package (every time when you start R)
object = c(1,2,3,4) #Make a simple object of numbers
list_of_data = list("data1" = c(1,2,3,4), #Make a list
                    "data2" = "string")
list_of_data$data1 #Call first object of the list
?plot #Open help file for the plot function
x = rnorm(100) #Generate 100 random values from N(0,1)
plot(x) #Plot the generated values
```

Run model in R (JAGS and OpenBugs)

```
install.packages("rjags")
library(rjags)
jagsdata = list("N" = N,
                "x" = x,
                "y" = y)
jags <- jags.model("jags.txt",
                   jagsdata,
                   inits = NULL,
                   n.chains = 1)

update(jags, 10000) #burn in iterations
out <- jags.samples(jags,
                    c("b","mu","tau_r"),
                    20000)
```

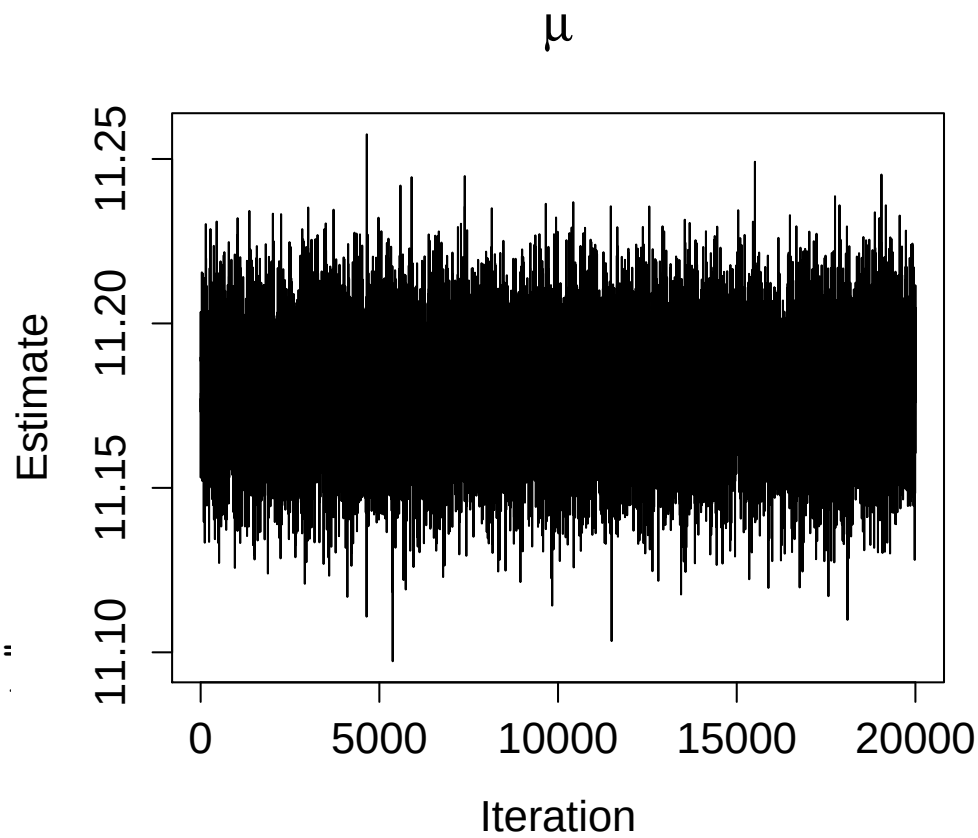
```
install.packages("R2OpenBUGS")
library(R2OpenBUGS)
bugsdata = list("N" = N,
                "x" = x,
                "y" = y)

bugs_out <- bugs(bugsdata,
 inits = NULL,
 model.file = "bugs.txt",
                 parameters =
                  c("mu", "b", "tau_r"),
                 n.chains = 1,
                 n.burnin = 10000,
                 n.iter = 20000)
```

Convergence?

Convergence plot:

```
#For example with JAGS:
plot(out$mu, type = "l",
      xlab = "Iteration",
      ylab = "Estimate",
       main = expression(mu))
```

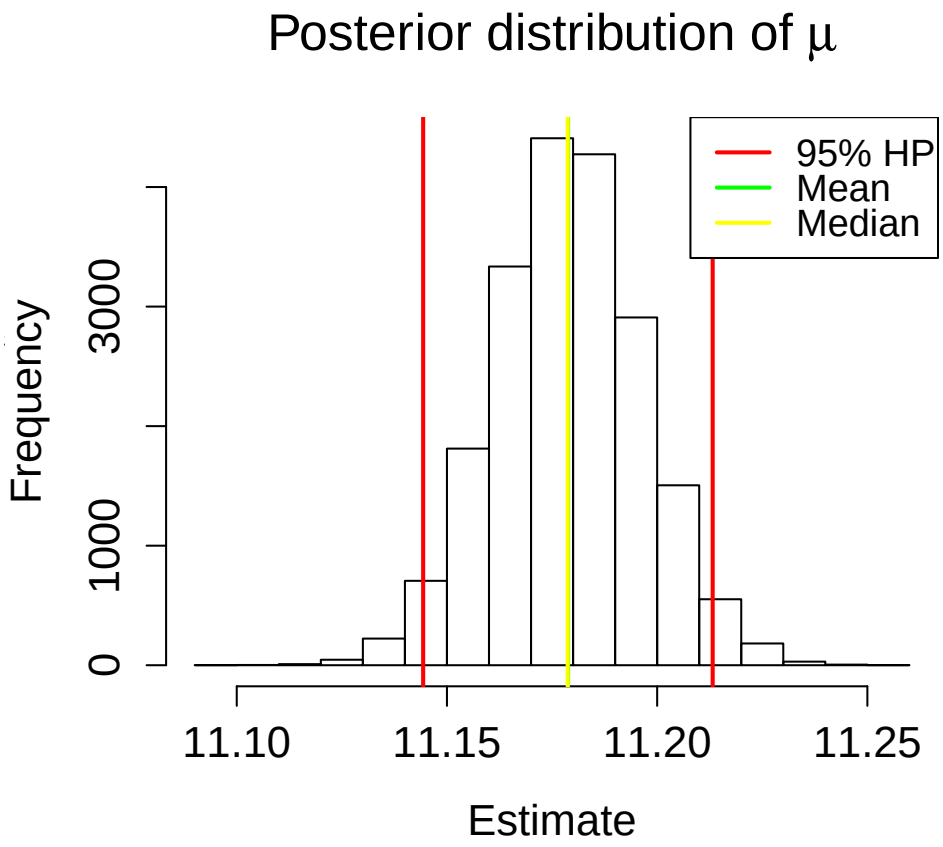


Gelman and Rubin's convergence diagnostic

```
inits1 <- list(list(mu = 2000, b =
                    list(mu = 0.0002,
jags <- jags.model("jags.txt",
   jagsdata, inits = inits1,
   n.chains = 2) #2 or more chains
update(jags, 10000)
out <- coda.samples(jags, c("mu","b",), 20000)
gelman.diag(out) #upper CI must be < 1
```

Results

```
1  out$mu
2  sd(out$mu)
3  plot(hist(out$mu))
4  plot(density(out$mu, adjust = 5))
5  #Density plot met coda package
6  library(coda)
7  out_coda_jags = coda.samples(jags
8  c("b", "mu", "tau_r"),
9  20000)
10 densityplot(out_coda_jags)
```



Differences betweens JAGS and OpenBugs

- Very small syntax differences
- JAGS can *easily* be used on a Mac machine
- OpenBugs is not developed anymore
- Another program: Stan (faster, but more difficult syntax!)