Assignment 3

Eveline Surbakti - 19200629

4/5/2020

EXERCISE 1

(a) Write JAGS code to fit this hierarchical linear model to the Hepatitis data set in hepatitis.dat.

Answer: The model is in hepatitis.model as attached.

```
model {
	for (i in 1:N){
		for(j in 1:M) {
			Y[i,j] ~ dnorm(mu[i,j], tau)
			mu[i,j] <- alpha[i] + beta[i]*(log(minutes[j])-mean(log(minutes[])))
			#unknown alpha and beta with -mean to make it neater
		}
		alpha[i] ~ dnorm(alpha0,taua) #hierarchycal prior construction
		beta[i] ~ dnorm(beta0,taub) #hierarchycal prior construction
	}
	alpha0 ~ dunif(-100,100) #uniform priors
	beta0 ~ dunif(-100,100) #uniform priors
	taua ~ dgamma(1e-3,1e-6)
	taub ~ dgamma(1e-3,1e-6)
	tau ~ dgamma(1e-3,1e-6)
}
```

```
hepatitis.data=list(N=52,M=7,minutes=c(1, 2.5, 5, 7, 10, 15, 20),Y=read.table
("hepatitis.dat",header = TRUE)) # reading the data
```

(b) Give an estimate of the mean slope and mean intercept including a 95% credible interval.

Answer:

```
#the model
hepatitis.model=jags.model("hepatitis.model",hepatitis.data,n.chains=1)

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
```

```
##    Observed stochastic nodes: 364
##    Unobserved stochastic nodes: 109
##    Total graph size: 1231
##
## Initializing model
```

```
#the sample
hepatitis.samps=coda.samples(hepatitis.model,variable.names=c("alpha","beta")
,1e4)
```

Here is the estimate of the mean intercept:

```
mean(colMeans(hepatitis.samps[[1]][,grep("alpha", colnames(hepatitis.samps[[1
]]))]))

## [1] 1.046014
```

Here is its 95% credible interval:

```
Alpha<-apply(hepatitis.samps[[1]][,grep("alpha", colnames(hepatitis.samps[[1]
]))], 1, mean)
quantile(Alpha,prob=c(0.025,0.975))

##      2.5%     97.5%
## 0.9867327 1.1046123
```

Here is the estimate of the mean slope:

```
mean(colMeans(hepatitis.samps[[1]][,grep("beta", colnames(hepatitis.samps[[1]
]))]))

## [1] 2.092684
```

and here is its 95% credible interval:

```
Beta<-apply(hepatitis.samps[[1]][,grep("beta", colnames(hepatitis.samps[[1]])
)], 1, mean)
quantile(Beta,prob=c(0.025,0.975))

##     2.5%    97.5%
## 2.033056 2.152483
```

EXERCISE 2

```
library(MASS)
data(swiss)
```

(a)You will fit a Bayesian Linear Regression model for Fertility as dependent on the other 5 variables.

Answer: We can see that the posterior mean for betas equals (66.92,-0,17,-0.26,-0.87,-0.10,1.08) correct to two decimal place.

```
Y<- swiss$Fertility #the dependent variable
X<- cbind(swiss$Agriculture,swiss$Examination,swiss$Education,swiss$Catholic,
swiss$Infant.Mortality) #the independent variables
n    <- length(Y)
p    <- ncol(X)
allone<-(rep(c(1), times = 47))
X<-cbind(allone,X)

Q=t(X)%*%X
S=solve(Q) #epsilon

beta<-round(S%*%t(X)%*%Y,2)
beta #miu
```

```
##         [,1]
## allone 66.92
##        -0.17
##        -0.26
##        -0.87
##         0.10
##         1.08
```

#b) Fit the model using JAGS and uniform [-100; 100] priors on the regression coeficients and 50000 MCMC iterations. Is the posterior mean you get the same as you found in part (a)?

Answer: Quite the same but few are slightly difference because we now use the random sampling.

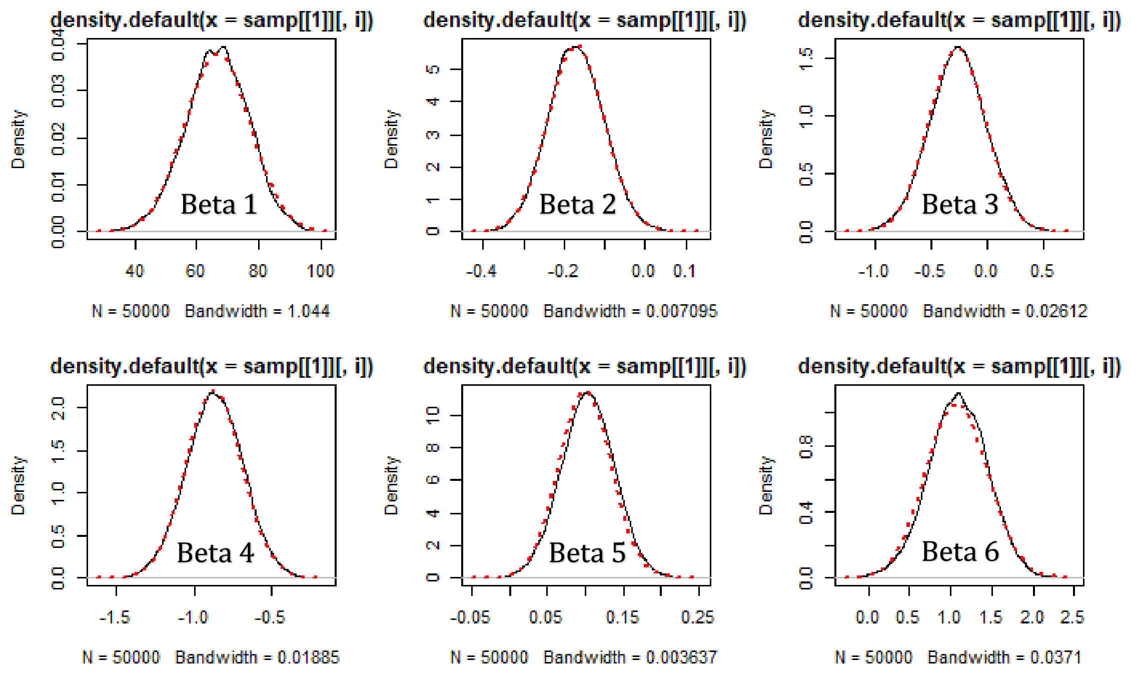
```
model_string <- "model{
  # Likelihood
  for(i in 1:n){
    Y[i]   ~ dnorm(mu[i],0.02)
    mu[i] <- X[i, 1]*beta[1] + X[i, 2]*beta[2] + X[i, 3]*beta[3]+
    X[i, 4]*beta[4]+X[i,5] *beta[5]+X[i, 6]*beta[6]
  }
  # Prior for beta
  for(j in 1:6){
    beta[j] ~ dunif(-100,100)
  }
}"

model <- jags.model(textConnection(model_string),
                     data = list(Y=Y,n=n,X=X))
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 47
##    Unobserved stochastic nodes: 6
##    Total graph size: 558
##
## Initializing model

set.seed(123)
samp <- coda.samples(model,
                     variable.names=c("beta"),
                     n.iter=50000)
first<-apply(samp[[1]],2,sd) #save this for comparison later

round(colMeans(samp[[1]]),2)

## beta[1] beta[2] beta[3] beta[4] beta[5] beta[6]
##   66.59   -0.17   -0.26   -0.87    0.10    1.09
```

(c)Now examine the marginal distributions for the 6 regression coeficients. The jth marginal of a Multivariate Normal is simply a Normal distribution with mean (miuj) and variance (epsilonjj). Create a plot for each coeficient showing the sample marginal of the posterior and the theoretical marginal.

Answer: we can see both lines are closer to each other.

```
tau=0.02
par(mfrow=c(2,3))
for(i in 1:6) {
  plot(density(samp[[1]][,i])) #sample marginal of the posterior distribution
  curve(dnorm(x,beta[i],sqrt(S[i,i]*1/tau)),add = TRUE, col=2,lty=3,lwd=2) }
#theoretical
```



(d) Now use a new .model file that sets tau to unknown and use a reference prior p(tau) = 1/tau. What happens to the marginal posteriors for the regression coeficients? Why?

Answer: All coefficients when tau is unknown are increase, because now the sampling will take any random value of tau from the distribution rather than use the fixed number. The previous value of tau was really small and there is a greater probability that R will take greater value than 0.02.

The calculation of the percentage change in posterior marginal standard deviation for each regression coeficient can change everytime we run the code, but the percentage of coefficient will always increase.

```
model_string <- "model{
  # Likelihood
  for(i in 1:n){
    Y[i]   ~ dnorm(mu[i],tau)
    mu[i] <- X[i, 1]*beta[1] + X[i, 2]*beta[2] + X[i, 3]*beta[3]+
    X[i, 4]*beta[4]+X[i,5] *beta[5]+X[i, 6]*beta[6]
  }
  tau ~ dgamma(1e-3,1e-6)
  # Prior for beta
  for(j in 1:6){
    beta[j] ~ dunif(-100,100)
  }
}"
model <- jags.model(textConnection(model_string),
                    data = list(Y=Y,n=n,X=X))

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 47
##    Unobserved stochastic nodes: 7
##    Total graph size: 560
## Initializing model

samp2 <- coda.samples(model,
                      variable.names=c("beta"),
                      n.iter=50000)

second<-apply(samp2[[1]],2,sd)
diff=((second-first)/(first))*100
diff

##    beta[1]    beta[2]    beta[3]    beta[4]    beta[5]    beta[6]
##  10.255592   5.144321   2.046724   3.203879   2.521352  13.888277
```

Final notes: Some values in this report are the result from sampling and then they will appear in the body of the text and will be updated each time we run the code.