HW8

Avinash Ramu

November 9, 2016

Q18.2 I will use the radon model that Andrew used in his demo.

```
library(lme4)
```

```
## Loading required package: Matrix
```

```
library(arm)
```

```
## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

##
## arm (Version 1.9-1, built: 2016-8-21)

## Working directory is /Users/conradlab/src/MultilevelModeling/HW/hw8
```

```
library(rjags); library(R2jags)
```

```
## Loading required package: coda

##
## Attaching package: 'coda'

## The following object is masked from 'package:arm':
##
##     traceplot

## Linked to JAGS 4.2.0

## Loaded modules: basemod,bugs

##
## Attaching package: 'R2jags'

## The following object is masked from 'package:coda':
##
##     traceplot
```

```
## The following object is masked from 'package:arm':
##
##     traceplot
```

```
# Setting up the data (from Gelman & Hill's website)
srrs2 <- read.table ("srrs2.dat", header=T, sep=",")
mn <- srrs2$state=="MN"
radon <- srrs2$activity[mn]
log.radon <- log (ifelse (radon==0, .1, radon))
floor <- srrs2$floor[mn]       # 0 for basement, 1 for first floor
n <- length(radon)
y <- log.radon
x <- floor

# get county index variable
county.name <- as.vector(srrs2$county[mn])
uniq <- unique(county.name)
J <- length(uniq)
county <- rep (NA, J)
for (i in 1:J){
  county[county.name==uniq[i]] <- i
}

srrs2.fips <- srrs2$stfips*1000 + srrs2$cntyfips
cty <- read.table ("cty.dat", header=T, sep=",")
usa.fips <- 1000*cty[,"stfips"] + cty[,"ctfips"]
usa.rows <- match (unique(srrs2.fips[mn]), usa.fips)
uranium <- cty[usa.rows,"Uppm"]
u <- log (uranium)

### The actual JAGS code begins here ###

# Saving the model to an object. This allows us to avoid saving the model to a .bug file and sourcing i
# Make sure it is within quotations!
the.model <- "model{
  for (i in 1:n){
    y[i] ~ dnorm (y.hat[i], tau.y)
    y.hat[i] <- a[county[i]] + b*x[i]
  }
  b ~ dnorm (0, .0001)
  tau.y <- pow(sigma.y, -2)
  sigma.y ~ dunif (0, 100)

  for (j in 1:J){
    a[j] ~ dnorm (a.hat[j], tau.a)
    a.hat[j] <- g.0 + g.1*u[j]
  }
  g.0 ~ dnorm (0, .0001)
  g.1 ~ dnorm (0, .0001)
  tau.a <- pow(sigma.a, -2)
  sigma.a ~ dunif (0, 100)
}"

# Defining the data you will pass into the model -- you *already know* these values
```

```r
radon.data <- list ("n", "J", "x", "y", "county", "u")

# Defining the initial values that your model's parameters (values you *don't* already know)
radon.inits <- function (){list(a=rnorm(J),
                                b=rnorm(1),
                                g.0=rnorm(1),
                                g.1=rnorm(1),
                                sigma.y=runif(1),
                                sigma.a=runif(1))}

# Defining which parameters of your model you want JAGS to return to you
# In the book (page 366), they are missing "g.0" and "g.1"
radon.parameters <- c ("a", "b", "sigma.y", "sigma.a", "g.0", "g.1")

# Now, we can actually run the model with the jags() function
radon.3 <- jags(data=radon.data,
                inits=radon.inits,
                parameters.to.save=radon.parameters,
                model.file=textConnection(the.model), # Note the textConnection() function
                n.chains=3,
                n.iter=5000,
                DIC=F)
```

```
## module glm loaded

## module dic loaded

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 919
##    Unobserved stochastic nodes: 90
##    Total graph size: 3266
##
## Initializing model
```

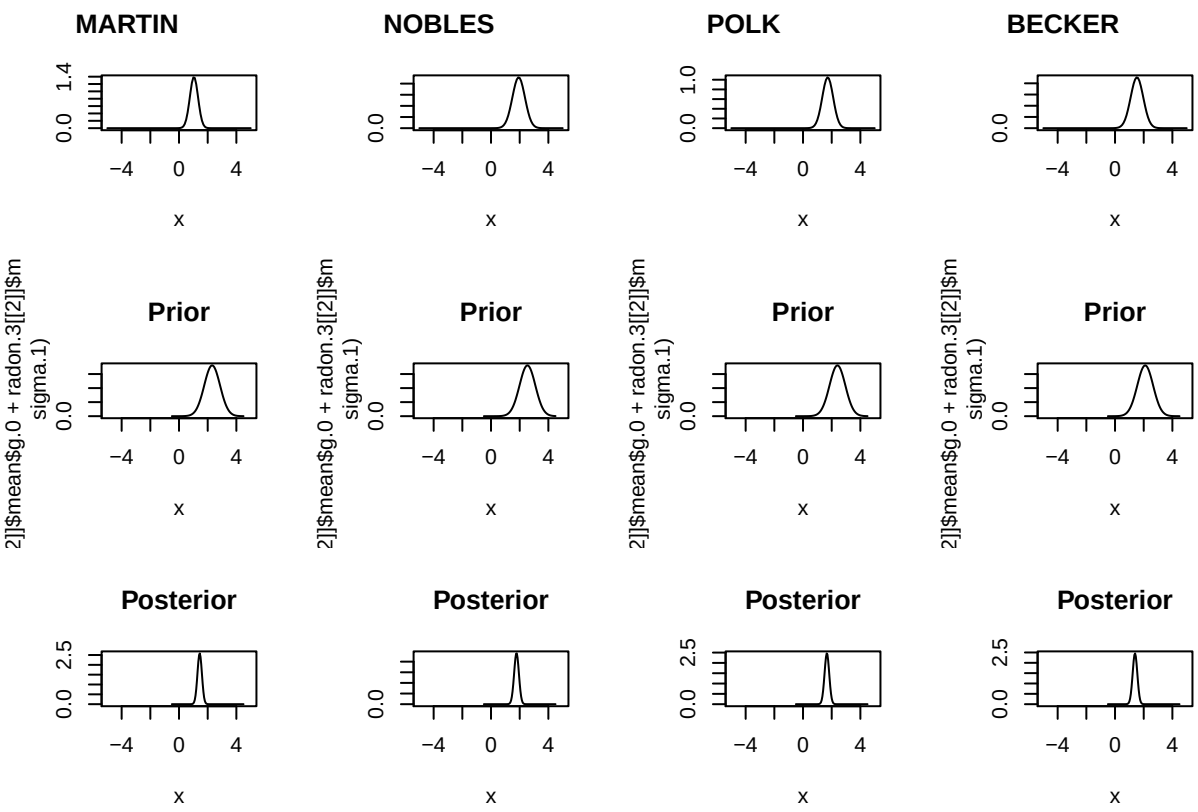
```r
unpooled_model <- lm(y ~ floor + factor(county) - 1)


par(mfrow=c(3, 4))
counties <- sample(1:85, 4)
for(i in counties) {
  curve(dnorm(x, coef(unpooled_model)[i+1], se.coef(unpooled_model)[i+1]), main = uniq[i], ylab="", xlim
}

sigma.1 <- runif(1)
for(i in counties){
  curve(dnorm(x, radon.3[[2]]$mean$g.0 + radon.3[[2]]$mean$g.1 * uranium[i], sigma.1), from = -0.5, to =
}

for(i in counties){
```

```
  curve(dnorm(x, radon.3[[2]]$mean$a[i], radon.3[[2]]$sd$a[i]), from=-0.5, to=4.5, main="Posterior", yla
}
```



The prior, likelihood and posterior have been plotted. The inference remains the same as in the previous homework. The counties with a larger sample size drive the posterior more from the prior towards the likelihood.

Q18.4

Part A

I have written this out on paper.

Part B

```
beauty <- read.csv("ProfEvaltnsBeautyPublic.csv")
attach(beauty)
C <- 3
censored <- courseevaluation < 3
censored_eval <- ifelse(censored, C, courseevaluation)
Loglik <- function (parameter.vector, x, y, C) {
  a <- parameter.vector[1]
  b <- parameter.vector[2]
  sigma <- parameter.vector[3]
  ll.vec <- ifelse (y<C, dnorm (y, a + b*x, sigma, log=TRUE),
  pnorm ((a + b*x - C)/sigma, log=TRUE))
```

```
  return (sum (ll.vec))
}
inits <- runif (3)
mle <- optim (inits, Loglik, lower=c(-Inf,-Inf,1.e-5), method="L-BFGS-B", control=list(fnscale=-1), x =
mle$par
```

```
## [1] 4.0212587 0.1208320 0.5169756
```

The coefficient for beauty is 0.12, for every unit increase in beauty there is a 0.12 increase in the coursevaluation. This could indicate some sort of bias towards course evaluations based on the looks of the instructor.

Part C

```
beauty <- read.csv("ProfEvaltnsBeautyPublic.csv")
attach(beauty)
```

```
## The following objects are masked from beauty (pos = 3):
##
##     age, beautyf2upper, beautyflowerdiv, beautyfupperdiv,
##     beautym2upper, beautymlowerdiv, beautymupperdiv, blkandwhite,
##     btystdave, btystdaveneg, btystdavepos, btystdf2u, btystdfl,
##     btystdfu, btystdm2u, btystdml, btystdmu, btystdvariance,
##     class1, class10, class11, class12, class13, class14, class15,
##     class16, class17, class18, class19, class2, class20, class21,
##     class22, class23, class24, class25, class26, class27, class28,
##     class29, class3, class30, class4, class5, class6, class7,
##     class8, class9, courseevaluation, didevaluation, female,
##     formal, fulldept, lower, minority, multipleclass, nonenglish,
##     onecredit, percentevaluating, profevaluation, profnumber,
##     students, tenured, tenuretrack
```

```
model1 <- "model {
  for (i in 1:n){
    isCensored[i] ~ dinterval(y[i], censorLimitVec)
    y[i] ~ dnorm(yhat[i], tau.y)
    yhat[i] <- a + b*x[i]
  }
  a ~ dnorm(0, 1E-6)
  b ~ dnorm(0, 1E-6)
  tau.y <- pow(sigma.y, -2)
  sigma.y ~ dunif (0, 100)
}"
C <- 3
N <- length(courseevaluation)
censored <- courseevaluation < 3
y1 <- ifelse (censored, NA, courseevaluation)
censorLimitVec = 3
beauty_data <- list (x = btystdave, y = y1, n = N, isCensored = as.numeric(censored), censorLimitVec = c
beauty_inits <- function() { list(a=rnorm(1), b=rnorm(1), sigma.y=runif(1)) }
params <- c ("a", "b", "sigma.y")
```

```
radon.3 <- jags(data = beauty_data,
                inits = beauty_inits,
                parameters.to.save = params,
                model.file = textConnection(model1), # Note the textConnection() function
                n.chains = 3,
                n.iter = 5000,
                DIC = F)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 908
##    Unobserved stochastic nodes: 21
##    Total graph size: 1591
##
## Initializing model
```

```
radon.3
```

```
## Inference for Bugs model at "6", fit using jags,
##  3 chains, each with 5000 iterations (first 2500 discarded), n.thin = 2
##  n.sims = 3750 iterations saved
##         mu.vect sd.vect  2.5%   25%   50%   75% 97.5%  Rhat n.eff
## a         4.058   0.023 4.014 4.042 4.058 4.073 4.105 1.002  1600
## b         0.086   0.030 0.026 0.067 0.087 0.107 0.144 1.001  3800
## sigma.y   0.491   0.016 0.460 0.481 0.492 0.502 0.523 1.003   940
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
```

The BUGs model assigns a similar intercept as the MLE model. The coefficient on beauty is a little smaller 0.08 compared to 0.12 from the MLE model. This could reflect the Bayesian aspect of the model with respect to censoring, this is a little unintuitive. Perhaps there is a problem with my model specification. The coefficient is still positive and indicates a 0.08 increase in course evaluation for a unit increase in beauty.

Part D

Regression without censoring

```
unpooled_model <- lm(courseevaluation ~ btystdave)
unpooled_model
```

```
##
## Call:
## lm(formula = courseevaluation ~ btystdave)
##
## Coefficients:
## (Intercept)    btystdave
##       4.010        0.133
```

The intercept and coefficient from Part D i.e the uncensored model look very similar to Part B, the MLE estimate with censoring. The coefficient on beauty is 0.13 which is slightly higher than part B (0.12) and part C (0.08). This is the increase in courseevaluation for a unit increase in beauty.