OpenBugs

Michael Andreae

December 11, 2015

We are trying to repeat the analysis by Dr. Johnson's for the ICBG evidency synthesis for ACE 151

OpenBugs Example

We followed the Tutorial

Repeating Dr. Johnson ICBG Analysis

```
rm(list=ls())

# Model with old data

model <- function() {

    ##############################
    # Blumenthal                                    #
    ##############################
    for(i in 0:3){
        logL[1,i+1] <- i*log(p[1,1]) +(18-i)*log(1-p[1,1]) - logfact(i) - logfact(18-i) + logfact(18)
        L[1,i+1] <- exp(logL[1,i+1])
        p1[i+1] <- L[1,i+1]/sum(L[1,1:4])
    }
    for(i in 16:18){
        logL[2,i-15] <- i*log(p[1,2]) + (18-i)*log(1-p[1,2]) - logfact(i) - logfact(18-i)+logfact(18)
        L[2,i-15] <- exp(logL[2,i-15])
        p2[i-15] <- L[2,i-15]/sum(L[2,1:3])
    }
    for(i in 1:2){
        d[i] <- 1
        d[i] ~ dbern(LogLike[i])
        LogLike[i] <- mean(L[i,1:(r[i])])
    }
    ######################################

    ##################################
    # Other Studies                                 #
    ##################################
    for(i in 1:3){
        for(j in 1:2){
            x[i,j] ~ dbin(p[i+1,j],N[i,j])
        }
    }
    ##################################
```

```
    ####################################
    # Priors                                                    #
    ####################################
    for(i in 1:4){
        for(j in 1:2){
            logit(p[i,j]) <- gamma[i,j]
        }
        gamma[i,1:2] ~ dmnorm(gamma[5,1:2],Tau[1:2,1:2])
        #gamma[i,1] ~ dnorm(gamma[5,1],
        or[i] <- exp(gamma[i,1]-gamma[i,2])
    }
    gamma[5,1] ~ dnorm(0,0.001)
    gamma[5,2] ~ dnorm(0,0.001)
    or[5] <- exp(gamma[5,1]-gamma[5,2])
    logit(p[5,1]) <- gamma[5,1]
    logit(p[5,2]) <- gamma[5,2]
    nnt <- 1/(p[5,2] - p[5,1])
    Sigma[1] ~  dt(0,3,1)#T(0,)
    Sigma[2] ~ dt(0,3,1)#T(0,)
    rho ~ dunif(-1,1)
    Sigma[3] <- rho*sqrt(Sigma[1]*Sigma[2])
    det <- Sigma[1]*Sigma[2] - Sigma[3] * Sigma[3]
    Tau[1,1] <- Sigma[2]/det
    Tau[2,2] <- Sigma[1]/det
    Tau[1,2] <- -Sigma[3]/det
    Tau[2,1] <- Tau[1,2]
}


# To transfer the model to OpenBUGS, we load the R2OpenBUGS extension
# and write the model to a temporary location using the method
# write.model. We denote the model file location by model.file.

library(R2OpenBUGS)
model.file <- file.path(tempdir(),
                        "model.txt")
write.model(model, model.file)

# We then identify data variables in a list called data.

data = list (r=c(4,3),
             N=structure(.Data=c(30,15,11,11,9,11),.Dim=c(3,2)),
             x=structure(.Data=c(2,5,2,6,0,7),.Dim=c(3,2)),
             Sigma=c(1,1,NA),rho=0)

# Gundes, Brull, Singh #

# And we identify the variable p to be monitored in a vector called
# params.

params <- c("or", "nnt", "p")

# Lastly, we may select some initial parameters for the simulation.
```

```
# A rule of thumb is to choose values as close to the expected result
# as possible. In this case, we initialize p to be 0.5. Notice how we
# wrap the initial values inside a list that is to be returned by a
# function.

#inits <- function() { list(p=0.5) }

# Then we invoke OpenBUGS with the namesake method bugs and save the
# result in a variable out. We select 10,000 iterations per simulation
# chain.

out <- bugs(data, inits = NULL, params,  model.file, n.iter=10000)
out$summary
```

```
##                mean          sd        2.5%        25%      50%       75%
## or[1]     0.02898088  0.03609000  0.00135900  0.0079700  0.01708  0.036795
## or[2]     0.17943262  0.25244090  0.01922000  0.0675700  0.12385  0.215200
## or[3]     0.77981177  1.10537957  0.03368000  0.1955000  0.44330  0.922000
## or[4]     0.35703621  0.54743326  0.03722000  0.1276500  0.24390  0.424200
## or[5]     0.18438659  0.22373237  0.01642975  0.0621100  0.11935  0.226225
## nnt       1.79861033 80.35516247  1.27800000  1.7230000  2.17950  3.060000
## p[1,1]    0.12576713  0.08191095  0.01663000  0.0647075  0.10800  0.171000
## p[1,2]    0.85493579  0.08110129  0.66820000  0.8085000  0.86800  0.916000
## p[2,1]    0.16076467  0.05038020  0.08062000  0.1242000  0.15340  0.190200
## p[2,2]    0.59128411  0.16980111  0.24519750  0.4710000  0.60025  0.718200
## p[3,1]    0.23178505  0.08354132  0.09494000  0.1715000  0.22370  0.282900
## p[3,2]    0.42025329  0.21109977  0.09066872  0.2503000  0.39720  0.566900
## p[4,1]    0.31048797  0.08805502  0.16160000  0.2477000  0.30135  0.364100
## p[4,2]    0.63488947  0.16317124  0.29157997  0.5219000  0.64610  0.763825
## p[5,1]    0.19925542  0.09000219  0.06530950  0.1325750  0.18490  0.251725
## p[5,2]    0.64125676  0.15598838  0.31359750  0.5353000  0.65455  0.758500
## deviance 50.98876800 11.67956069 36.74000000 43.1700000 48.16000 55.830000
##                97.5%     Rhat n.eff
## or[1]     0.1375000 1.003711  4400
## or[2]     0.6339025 1.001270  5400
## or[3]     3.6120000 1.000967 15000
## or[4]     1.2690250 1.001142  8200
## or[5]     0.7362225 1.000973 15000
## nnt      10.4212500 1.262702  8800
## p[1,1]    0.3434000 1.002248 10000
## p[1,2]    0.9710000 1.003973  1200
## p[2,1]    0.2760000 1.001123  8900
## p[2,2]    0.8880050 1.001455  4100
## p[3,1]    0.4200150 1.000978 15000
## p[3,2]    0.8672025 1.000919 15000
## p[4,1]    0.5055075 1.001015 15000
## p[4,2]    0.9025000 1.000956 15000
## p[5,1]    0.4109000 1.001061 12000
## p[5,2]    0.9020050 1.000914 15000
## deviance 82.6204999 1.001458  3600
```

```
# Better to invoke the CODA option to get mcmc.list as output.
```

```r
out <- bugs(data, inits = NULL, params, model.file, codaPkg=TRUE,
            n.iter=10000)
out.coda <- read.bugs(out)
```

```
## Abstracting deviance ... 5000 valid values
## Abstracting nnt ... 5000 valid values
## Abstracting or[1] ... 5000 valid values
## Abstracting or[2] ... 5000 valid values
## Abstracting or[3] ... 5000 valid values
## Abstracting or[4] ... 5000 valid values
## Abstracting or[5] ... 5000 valid values
## Abstracting p[1,1] ... 5000 valid values
## Abstracting p[1,2] ... 5000 valid values
## Abstracting p[2,1] ... 5000 valid values
## Abstracting p[2,2] ... 5000 valid values
## Abstracting p[3,1] ... 5000 valid values
## Abstracting p[3,2] ... 5000 valid values
## Abstracting p[4,1] ... 5000 valid values
## Abstracting p[4,2] ... 5000 valid values
## Abstracting p[5,1] ... 5000 valid values
## Abstracting p[5,2] ... 5000 valid values
## Abstracting deviance ... 5000 valid values
## Abstracting nnt ... 5000 valid values
## Abstracting or[1] ... 5000 valid values
## Abstracting or[2] ... 5000 valid values
## Abstracting or[3] ... 5000 valid values
## Abstracting or[4] ... 5000 valid values
## Abstracting or[5] ... 5000 valid values
## Abstracting p[1,1] ... 5000 valid values
## Abstracting p[1,2] ... 5000 valid values
## Abstracting p[2,1] ... 5000 valid values
## Abstracting p[2,2] ... 5000 valid values
## Abstracting p[3,1] ... 5000 valid values
## Abstracting p[3,2] ... 5000 valid values
## Abstracting p[4,1] ... 5000 valid values
## Abstracting p[4,2] ... 5000 valid values
## Abstracting p[5,1] ... 5000 valid values
## Abstracting p[5,2] ... 5000 valid values
## Abstracting deviance ... 5000 valid values
## Abstracting nnt ... 5000 valid values
## Abstracting or[1] ... 5000 valid values
## Abstracting or[2] ... 5000 valid values
## Abstracting or[3] ... 5000 valid values
## Abstracting or[4] ... 5000 valid values
## Abstracting or[5] ... 5000 valid values
## Abstracting p[1,1] ... 5000 valid values
## Abstracting p[1,2] ... 5000 valid values
## Abstracting p[2,1] ... 5000 valid values
## Abstracting p[2,2] ... 5000 valid values
## Abstracting p[3,1] ... 5000 valid values
## Abstracting p[3,2] ... 5000 valid values
## Abstracting p[4,1] ... 5000 valid values
## Abstracting p[4,2] ... 5000 valid values
```

```
## Abstracting p[5,1] ... 5000 valid values
## Abstracting p[5,2] ... 5000 valid values
```

```
# Analyse using shinystan

library(shinystan)
library(coda)

# After verification that the opject is now a mcmc.list
# convert to shinstan object and call shinystan

is.mcmc.list(out.coda)
```

```
## [1] TRUE
```

```
out.shiny <- as.shinystan(out.coda)
# launch_shinystan(out.shiny)
```