Open Source Software Tools for Parallel Computation of Multiple MCMC Chains with WinBUGS

William R. Gillespie, Marc R. Gastonguay, William Knebel, George Georgalis
Metrum Institute, Tariffville, CT 06081 USA

BACKGROUND

- When used for a Bayesian analysis, Markov chain Monte Carlo (MCMC) simulations generate samples that approximate the joint posterior distribution of the sampled parameters.
- The sequential nature of MCMC simulation limits the benefits of parallel computation when applied to a single chain.
- Parallel computation of multiple chains, however, is an "embarrassingly parallel" problem that can substantially reduce computation time, and is relatively easy to implement using freely available software.

OBJECTIVE

- To develop open source software tools for parallel computation of multiple MCMC chains with WinBUGS [1] running in Microsoft Windows.

METHODS

- bugsParallel is a set of R [2] functions for distributed computing with WinBUGS 1.4.*.
  - It is a modified subset of the R2WinBUGS package [3].
  - Uses the R package Rmpi [4] to implement a network of Windows workstations.
  - Uses the rlecuyer package [5] for parallel random number generation.
    - This assures appropriately independent initial estimates when they are specified via a function that uses random number generation to generate the initial estimates.
  - The seeds used by each WinBUGS chain are generated in bugsParallel by the master node. The master generates a vector of random seeds – one per chain. The vector and chain identifier are passed to each slave that uses the appropriate element of that vector. The seed is passed to WinBUGS via the undocumented set.seed WinBUGS script command.
- Implementation involves installation of the following freely distributed software components:
  - MPICH2 for Windows [6], an implementation of the Message-Passing Interface (MPI) for parallel computation,
  - WinBUGS 1.4.3 [1],
  - R for Windows [2],
  - R packages Rmpi [4] and rlecuyer [5].
- bugsParallel is currently provided as an R script that the user accesses via a "source" command in a user written R script.
- Parallel computation with WinBUGS was previously reported by Girgis et al [7] using commercial computing grid software.

Illustrative Example

- The following R script and WinBUGS model were used to fit a sigmoid Emax model to a set of 125 simulated data points.
- The example was run on a ThinkMate Workstation with two Intel E5345 Quad-Core processors (2.33 GHz) with 16 GB RAM.
- Eight MCMC chains of 100,000 iterations each were computed.
  - The first 10,000 iterations were discarded.
  - The MCMC chains were thinned to retain 1 of each 100 iterations, i.e., 7200 MCMC samples remained for analysis.

Equation 1. Bayesian Model from Illustrative Example

$$E_i \sim N\left(\hat{E}_i, \sigma^2\right)$$

$$\hat{E}_i = \frac{E_{max} c_i^{\gamma}}{EC_{50}^{\gamma} + c_i^{\gamma}}$$

Where E_i is the "observed response."

Equation 2. Prior distributions:

$$E_{max} \sim U(0, 100)$$
$$\log(ED_{50}) \sim N\left(0, 10^6\right)$$
$$\gamma \sim U(0, 10)$$
$$\sigma \sim U(0, 1000)$$

Figure 1. R script from illustrative example

```
model.name = "FXaExample1" #root name of modeling files
toolsDire = "C:/bugsTools"

library(Rmpi)
library(rlecuyer)
library(coda)
library(lattice)
source(paste(toolsDir,"/bgillespie.utilities.R",sep
source(paste(toolsDir,"/bugsParallelAll.R",sep=""))
source(paste(toolsDir,"/bugs.tools.R",sep="")) # A few BUGS-specific
  utilities

#################################################################
#Data management

xdata = read.csv("fxa.data.avg.csv")

#create WinBUGS data set
bugsdata = list(
  nobs = nrow(xdata),
  cobs = xdata$cavg,
  FXa = xdata$fxa.inh.avg)

#create initial estimates
bugsinit = function() list(
  Emax = runif(1,40,100),
  logEC50 ~ rnorm(1,log(100),0.4),
  gamma = 10*rbeta(1,0.25,5),
  sigma = exp(rnorm(1,log(5),0.2)))

# specify what variables to monitor
parameters =  c("Emax","EC50","gamma","sigma","FxaPred")

# specify the variables for which you want history and density plots
parameters.to.plot = c("deviance","Emax","EC50","gamma","sigma")

#################################################################
# run WinBUGS

n.chains = 4
n.iter = 100000
n.burnin = 10000
n.thin = 100

mpi.spawn.Rslaves() # launches multiple R slave processes on available
  processors
mpi.set.rngstream() # initializes parallel random number generation

system.time(bugs.fit <- bugsParallel(data=bugsdata,inits=bugsinit,
  parameters.to.save=parameter,model.file=paste(getwd(),"\\",model.name,
  ".txt", sep=""),n.chains=n.chains,n.iter=n.iter,n.thin=n.thin,
  n.burnin=n.burnin,refresh=1, clearWD=T,bugs.directory =
  "c:/Program Files/WinBUGS14/"))

# save scripts, data and results to a directory
save.model(bugs.fit,model.name)
```

Figure 1. R Script from Illustrative Example, Continued

```
#convert MCMC results to formats suitable for post-processing
sims.array = aperm(array(unlist(bugs.fit),dim=c(nrow(bugs.fit[[1]],
  ncol(bugs.fit[[1]]),length(bugs.fit)),dimnames=c(dimnames(bugs.fit[[1]]),
  list(NULL))),c(1,3,2))
posterior = array(as.vector(sims.array),dim=c(prod(dim(sims.array)[1:2]),
  dim(sims.array)[3]),dimnames=list(NULL,dimnames(sims.array)[[3]]))

#################################################################
# posterior distributions of parameters

# open graphics device
pdf(file = paste(model.name,"/",model.name,".plots.pdf",sep=""),width=6,
  height=6)

# create history, density and Gelman-Rubin-Brooks plots, and a table of
  summary stats
ptable = parameter.plot.table(sims.array[,,
  unlist(sapply(c(paste("",parameters.to.plot,"$",sep=""),
  paste("",parameters.to.plot,"\\[",sep="")),grep,
  x=dimnames(sims.array)[[3]])))]

write.csb(signif(ptable,3),paste(model.name,"/",model.name,",
  summary.csv",sep=""))

#################################################################
#posterior distributions of parameters

pred = posterior[,grep(FXaPred\\[",dimnames(posterior)[[2]])]
x1 = xdata
x1$type = rep("observed",nrow(x1))
x2 = rblind(x1,x1,x1)
x2$fxa.ing.avg = as.vector(t(apply(pred,2,
  quantile,probs=c(0.95.0.5.0.05))))
x2$type = rep(c("5%ile","median","95%ile"),ea=nrow(x1))
x1= rbind(x1,x2)
x1 = [order(x1$ype,x1$cavg),]

xyplot(fxa.inh.avg~cavg,x1,groups=type,
   pandel=panel.superpose,pch=c(NA,NA,NA,1),
  type=c("l","l","l","p",),lty=c(3,3,1,0),col=c("red","red","blue",
  "black"),lwd=3,scales=list(cex=1),par.strip.text=list(cex=1),
  strip = function(...) strip.default(..., style = 1),
  xlab= list(label="time-averaged plasma drug concentration",cex=1.2),
  ylab=list(label="time-averaged FXa inhibition (%)",cex=1.2))

dev.off() # close graphics device

mpi.close.Rslaves() # stop R slave processes
```

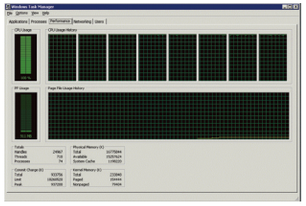
Figure 2. WinBUGS Model for Sigmoid Emax Model Example

```
model{

   for(i in 1:nobs){
     FXa[i] ~ dnorm(FXaHat[i],tau)       #likelihood
     FXaPred[i] ~ dnorm(FXaHat[i],tau    #simulation to generate posterior
                                         #predictive distributions
    FXaHat[i] <- Emax * pow(cobs[i],gamma) / (pow(EC50,gamma) +
     pow(cobs[i],gama))

   }

   Emax ~ dunif(0,100)
   logEC50 ~ dnorm(0,0.000001)
   log(EC50) <- logEC50
   gamma ~ dunif(0,10)
   sigma ~ dunif(0,10)
   tau <- 1/(sigma*sigma)

}
```

RESULTS

- Elapsed time for computation of 8 MCMC chains of 100,000 iterations
  - 8 processors: 243 seconds
  - 1 processor: 1806 seconds

Figure 3. Screen Shot of Windows Task Manager



Screenshot of Windows Task Manager showing 100% CPU usage of 8 processors during execution of example problem.

Figure 4. MCMC History Plot

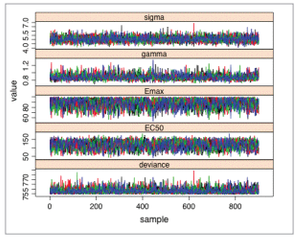


Figure 5. Marginal Posterior Distributions of Model Parameters

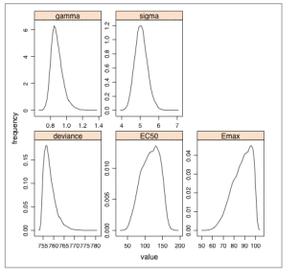
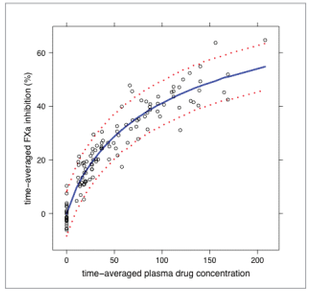


Figure 6. Data and Model Predictions (Posterior Median and 90% Credible Intervals)



DISCUSSION

- bugsParallel provides a practical open source option for parallel computation of multiple MCMC chains using WinBUGS in a MS Windows environment.
- bugsParallel is freely available from Metrum Institute (http://metruminstitute.org/).
- The remaining software components required for implementation are also freely available at the sites listed under "References."
- Integration with R permits data management, MCMC and analysis of MCMC results within a single software environment.
- Works across multiple machines or multiple processors within a single machine.
- Limitations of current version of bugsParallel
  - Limited to a grid of Windows computers/processors.
  - Lack of substantial "bullet-proofing" makes it poorly suited for a large-scale grid.
    - E.g., it does not implement "bullet-proofing" such as identifying and aborting "stuck" processes and cleaning up after crashed processes.

REFERENCES

1. DJ Lunn, A Thomas, N Best, D Spiegelhalter. (2000) WinBUGS -- a Bayesian modelling framework: concepts, structure, and extensibility. Statistics and Computing, 10:325—337. (http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml)
2. R Development Core Team. (2007) R: A Language and Environment for Statistical Computing (http://www.r-project.org/)
3. http://cran.wustl.edu/src/contrib/Descriptions/R2WinBUGS.html and http://www.stat.columbia.edu/~gelman/bugsR/
4. http://cran.wustl.edu/src/contrib/Descriptions/Rmpi.html and http://www.stats.uwo.ca/faculty/yu/Rmpi/
5. http://cran.wustl.edu/src/contrib/Descriptions/rlecuyer.html
6. http://www.mcs.anl.gov/research/projects/mpich2/
7. IG Girgis, T Schaible, P Nandy, F De Ridder, J Mathers, S Mohanty. (2007) Parallel Bayesian methodology for population analysis. PAGE Meeting, June 13-15, 2007, Copenhagen, Denmark (http://www.page-meeting.org/pdf_assets/6232-PARALLEL%20Girgis_POSTER_final.pdf).

 copies of this poster are available at www.metruminstitute.org/publications