

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

Authors: William R. Gillespie* (1), Marc R. Gastonguay (1), William Knebel (1), George Georgalis (1)

Institutions: (1) Metrum Institute, Tariffville, CT, USA

Objectives: 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 uses the R package Rmpi [3] to implement a network of Windows workstations. bugsParallel is a modified subset of the R2WinBUGS package [4]. bugsParallel 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. 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. It is passed to WinBUGS via the undocumented set.seed WinBUGS script command.

Implementation involves installation of the following open source software components: MPICH2 for Windows [6], an implementation of the Message-Passing Interface (MPI) for parallel computation, WinBUGS 1.4.3, R for Windows, and the R packages Rmpi and rlecuyer. bugsParallel is currently provided as an R script that the user accesses via a “source” command in a user written R script. To illustrate the use of bugsParallel the following R script and WinBUGS model were used to fit a sigmoid Emax model to a set of 125 simulated data points (see Figure 3). The sigmoid Emax 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.

```
model.name = "FXaExample1" # root name of modeling files
toolsDir = "C:/bugsTools"
library(Rmpi); library(rlecuyer)
source(paste(toolsDir, "/bugsParallelAll.R", sep=""))

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

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

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

bugsinit = function() list( # create initial estimates
  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") # variables to monitor

##### Run WinBUGS #####

n.chains=8; n.iter=100000; n.thin=100; n.burnin=10000

mpi.spawn.Rslaves(nslaves=n.chains) # launches multiple R slave processes on available processors
mpi.setup.rngstream() # initializes parallel random number generation

bugs.fit <- bugsParallel(data=bugsdata, inits=bugsinit,
  parameters.to.save=parameters, 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/")

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

Figure 1. R script for sigmoid Emax model example

```

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

  }

  Emax ~ dunif(0,100)
  logEC50 ~ dnorm(0,0.000001)
  log(EC50) <- logEC50
  gamma ~ dunif(0,10)
  sigma ~ dunif(0,1000)
  tau <- 1/(sigma*sigma)
}

```

Figure 2. WinBUGS model for sigmoid Emax model example

Results: Computation of 8 MCMC chains of 100,000 iterations each required an elapsed time of 243 seconds when distributed over 8 processors versus 1806 seconds on a single processor. MCMC history plots, posterior marginal densities of the model parameters and posterior predictions compared to “observed” data are shown in Figure 3.

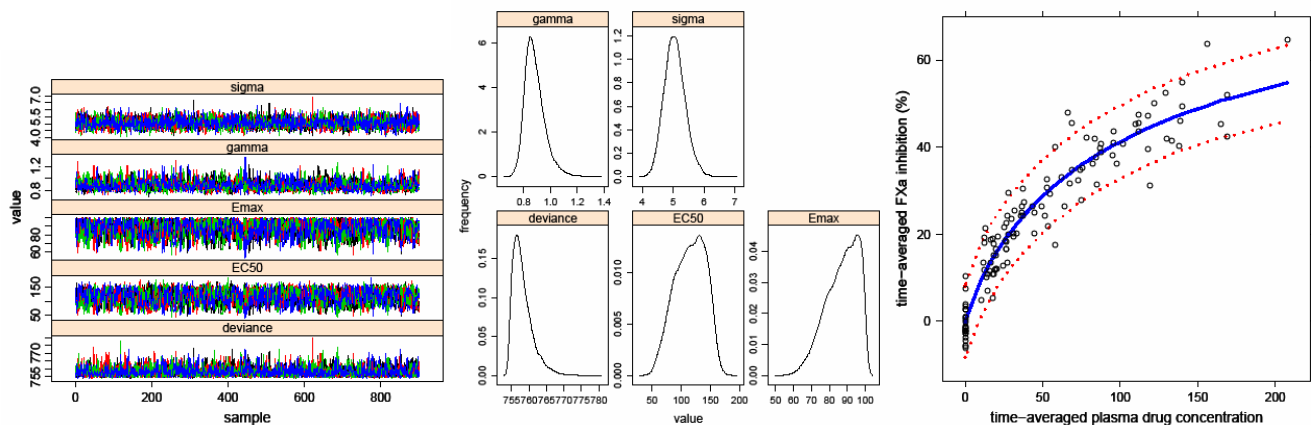


Figure 3. (a) MCMC history plots. (b) Posterior marginal densities of model parameters. (c) Posterior predictions (posterior medians and 90% credible intervals) compared to data.

Conclusions: bugsParallel provides a practical open source option for parallel computation of multiple MCMC chains using WinBUGS in a MS Windows environment. The current version of bugsParallel is freely available from Metrum Institute (<http://metruminstitute.org/>). The remaining software components required for implementation are also freely available at the sites cited below.

References:

[1] Lunn, D.J., Thomas, A., Best, N., and Spiegelhalter, D. (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/Rmpi.html> and <http://www.stats.uwo.ca/faculty/yu/Rmpi/>

[4] <http://cran.wustl.edu/src/contrib/Descriptions/R2WinBUGS.html> and <http://www.stat.columbia.edu/~gelman/bugsR/>

[5] <http://cran.wustl.edu/src/contrib/Descriptions/rlecuyer.html>