# Distributing OpenBUGs chains in parallel

Tom Werner
Chris Chekal

Department of Computer Science
The University of Iowa
Iowa City, IA

December 2, 2016

# What is OpenBUGs?

- Software for **B**ayesian inference **U**sing **G**ibbs **S**ampling
- Uses Monte Carlo Markov Chains (MCMC) to model posterior distributions

# Why use OpenBUGs?

- Free and open source software
- Integrates with R through R2OpenBugs/Coda packages
- Used in Bayesian analysis class at Iowa

# What is being parallelized?

- Using MCMC requires drawing many samples from a converged chain
- Determining convergence is easier with multiple chains
- If chains converge to different spots, something is wrong
- Each chain is independent of the others, allowing for naive parallelization
- Just need to make sure they're seeded independently

# Easy cluster/CPU parallization in R

- We used the Snow and Snowfall packages
- They let you easily set up a cluster to run R code on
- sfLapply works like regular apply, but distributes it over the cluster

# Snow/Snowfall Example

- data = matrix(rnorm(1000 * 1000), ncol=100)
  library(snow)
  library(snowfall)
  sfInit(parallel=TRUE, cpus=4)
  print(system.time(sfLapply(data, mean)))
  sfStop()
  sfInit(parallel=TRUE, cpus=1)
  print(system.time(sfLapply(data, mean)))
  sfStop()
- Parallel: 2.61 seconds
- Sequential: 6.26 seconds

# Parallel Bugs workflow

1. Define openbugs model as a string, data as an R list, and inits as a list
2. Call parallelBugs with these arguments, and the IP addresses of cluster machines
3. Use Coda or other R packages for analyzing MCMC output

# Parallel Bugs workflow

1. Define openbugs model as a string, data as an R list, and inits as a list

```
model.str = "model
{
  tau ~ dgamma(0.0001, 0.0001)
  mu ~ dflat()
  sigma <- 1 / sqrt(tau)

  for(i in 1:N)
  {
    x.data[i]~dnorm(mu, tau)
  }
}"
```

# Parallel Bugs workflow

1. Define openbugs model as a string, data as an R list, and inits as a list

```
# Simulated normal dataset
bugs.data = list(x.data=rnorm(1000, 12, 5), N=1000)

# Initial values of parameters. Note that you can use
# R random number functions here
init.list = list(
  chain1=list(mu=runif(1, -50, 50), tau=1),
  chain2=list(mu=-50, tau=10),
  chain3=list(mu=50, tau=.1)
)

# Parameters to collect statistics on
params <- c("mu", "sigma")
```

# Does it help?

1. Parallel time for 100,000 iterations, 3 chains: 60 seconds
2. Sequential time: 115 seconds

```
system.time(bugs(data=bugs.data, inits=init.list,
                 parameters.to.save=params, n.iter=100000,
                 n.chains=3, n.burnin=1000, n.thin=1,
                 model.file="model.txt", debug=FALSE,
                 codaPkg=TRUE, OpenBUGS.pgm=OpenBUGS.exe,
                 working.directory=folder,
                 bugs.seed=as.integer(runif(1, 1, 14))))
   user  system elapsed
   0.17    0.03   60.69
system.time(parallel.bugs('localhost', 3, model.str,
                          bugs.data, 3, init.list, params,
                          OpenBUGS.exe, n.iter=100000))
   user  system elapsed
   0.42    0.00  115.22
```

# References


- Luke Tierney, A. J. Rossini, Na Li and H. Sevcikova (2016). snow: Simple Network of Workstations. R package version 0.4-2. https://CRAN.R-project.org/package=snow
- Jochen Knaus (2015). snowfall: Easier cluster computing (based on snow).. R package version 1.84-6.1. https://CRAN.R-project.org/package=snowfall
- Martyn Plummer, Nicky Best, Kate Cowles and Karen Vines (2006). CODA: Convergence Diagnosis and Output Analysis for MCMC, R News, vol 6, 7-11