MCMCpack Goals

- Free, open-source, easy-to-use software for Bayesian inference.
- Provide a development environment for easy implementation of non-standard statistical models.
- Provide a distribution mechanism for other researchers with a consistent user interface and documentation.

MCMCpack Design

- "R-like" user interface for Bayesian tools.
- Model-specific design.
- Most computation done in C++ (using the Scythe Statistical Library).
- coda mcmc objects for posterior sample storage and summarization.
- Modular design and hidden functions to ease implementing additional models.

Why Not WinBUGS, JAGS, OpenBugs, etc?

- The BUGS language is good for many things, including quickly developing models. We see it as a complementary tool to MCMCpack.
However . . .

• . . . as noted in the WinBUGS manual:

  Potential users are reminded to be extremely careful if using this program for serious statistical analysis . . . If there is a problem, WinBUGS might just crash, which is not very good, but it might well carry on and produce answers that are wrong, which is even worse (p. 1).

• The BUGS language can be slow (especially for large problems), and the WinBUGS engine does not work for certain problems.

• The various flavors of BUGS require the user to do some modest programming which increases the opportunities for something to go wrong.
**Example 1: Inference for an Item Response Theory (IRT) Model**

Consider:

\[ y_{ij} \sim \text{Bernoulli}(\pi_{ij}), \quad i = 1, \ldots, I \quad j = 1, \ldots, J \]

with

\[ \pi_{ij} = \Phi(-\alpha_j + \beta_j \theta_i) \]

Typically, \( i \) indexes test-takers (subjects) and \( j \) indexes test items.

Prior distributions for the model parameters are:

\[(\alpha_j, \beta_j)' \sim \mathcal{N}(\alpha_0, A_0^{-1}) \quad j = 1, \ldots, J\]

and

\[ \theta_i \sim \mathcal{N}(0, 1) \quad i = 1, \ldots, I \]

To fit this model to some test data from http://work.psych.uiuc.edu/irt/downloads.asp using MCMCpack one can do the following.

```r
post.irt <- MCMCirt1d(testdata, burnin=5000, mcmc=100000, thin=10, A0=.001, store.item=TRUE, store.ability=TRUE, verbose=5000, theta.constraints=list(SUBJECT368="+", SUBJECT356="-"))
```

To look at some traceplots and marginal density estimates one can use:

```r
plot(post.irt)
```

which produces:
Subject 436 and subject 447 both got 75% of the test items correct. Which subject has higher ability?

The posterior probability that subject 436 has higher ability than subject 447 can be calculated with:

\[
\text{mean}(\text{post.irt[,436]} > \text{post.irt[,447]})
\]

which evaluates to 0.65.

---

We might also be interested in which test items do a good job of discriminating between high and low ability subjects. The \( \beta \) parameters provide this information.

We can calculate the posterior expectation of each of the discrimination parameters with:

\[
\text{beta.post} \leftarrow \text{post.irt[seq(from=452, to=530, by=2)]}
\]

print(sort(colMeans(beta.post)))

which produces:

<table>
<thead>
<tr>
<th>beta.ITEM18</th>
<th>beta.ITEM28</th>
<th>beta.ITEM16</th>
<th>beta.ITEM37</th>
<th>beta.ITEM19</th>
<th>beta.ITEM32</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.033332789</td>
<td>0.073744210</td>
<td>0.084909720</td>
<td>0.217837674</td>
<td>0.276429999</td>
<td>0.315174596</td>
</tr>
<tr>
<td>beta.ITEM36</td>
<td>beta.ITEM20</td>
<td>beta.ITEM40</td>
<td>beta.ITEM21</td>
<td>beta.ITEM1</td>
<td>beta.ITEM34</td>
</tr>
<tr>
<td>0.331350234</td>
<td>0.411436261</td>
<td>0.418098884</td>
<td>0.419030443</td>
<td>0.439532906</td>
<td>0.460565349</td>
</tr>
<tr>
<td>beta.ITEM10</td>
<td>beta.ITEM24</td>
<td>beta.ITEM14</td>
<td>beta.ITEM29</td>
<td>beta.ITEM35</td>
<td>beta.ITEM31</td>
</tr>
<tr>
<td>0.462356437</td>
<td>0.472680964</td>
<td>0.481691248</td>
<td>0.482167728</td>
<td>0.514685111</td>
<td>0.564987099</td>
</tr>
<tr>
<td>beta.ITEM39</td>
<td>beta.ITEM36</td>
<td>beta.ITEM33</td>
<td>beta.ITEM3</td>
<td>beta.ITEM35</td>
<td>beta.ITEM31</td>
</tr>
<tr>
<td>0.601600337</td>
<td>0.623693132</td>
<td>0.627042930</td>
<td>0.634003442</td>
<td>0.641052025</td>
<td>0.672391363</td>
</tr>
<tr>
<td>beta.ITEM29</td>
<td>beta.ITEM2</td>
<td>beta.ITEM14</td>
<td>beta.ITEM30</td>
<td>beta.ITEM12</td>
<td>beta.ITEM13</td>
</tr>
<tr>
<td>0.689007474</td>
<td>0.706561889</td>
<td>0.721453000</td>
<td>0.742149494</td>
<td>0.779142766</td>
<td>1.039535127</td>
</tr>
<tr>
<td>beta.ITEM9</td>
<td>beta.ITEM26</td>
<td>beta.ITEM24</td>
<td>beta.ITEM15</td>
<td>beta.ITEM12</td>
<td>beta.ITEM15</td>
</tr>
<tr>
<td>0.851232006</td>
<td>0.901178559</td>
<td>0.902609937</td>
<td>0.969587117</td>
<td>1.039535127</td>
<td>1.041296658</td>
</tr>
<tr>
<td>beta.ITEM12</td>
<td>beta.ITEM26</td>
<td>beta.ITEM24</td>
<td>beta.ITEM15</td>
<td>1.099500195</td>
<td>1.217643770</td>
</tr>
<tr>
<td>1.282921879</td>
<td>1.377932523</td>
<td>1.377932523</td>
<td>1.377932523</td>
<td>1.377932523</td>
<td>1.377932523</td>
</tr>
</tbody>
</table>
We can also calculate the posterior expected ranks of the discrimination parameters:

```r
beta.post.ranks <- matrix(NA, nrow(beta.post), ncol(beta.post))
colnames(beta.post.ranks) <- colnames(beta.post)
for (i in 1:nrow(beta.post)) {
  beta.post.ranks[i,] <- rank(beta.post[i,])
}
print(sort(colMeans(beta.post.ranks)))
```

Example 2: Calculating Bayes Factors for Model Comparison

Suppose that the observed data \( y \) could have been generated under one of two models \( M_1 \) and \( M_2 \).

A natural thing to ask from the Bayesian perspective is, “what is the posterior probability that \( M_1 \) is true (assuming either \( M_1 \) or \( M_2 \) is true)?”

How can we calculate this? Well, using Bayes theorem we can write:

\[
Pr(M_k|y) = \frac{p(y|M_k) Pr(M_k)}{p(y|M_1) Pr(M_1) + p(y|M_2) Pr(M_2)}, \quad k = 1, 2
\]

The key quantities here are \( p(y|M_1) \) and \( p(y|M_2) \) which are called marginal likelihoods or integrated likelihoods.

It is instructive to look at the posterior odds in favor of one model (say \( M_1 \)):

\[
\frac{Pr(M_1|y)}{Pr(M_2|y)} = \frac{p(y|M_1)}{p(y|M_2)} \times \frac{Pr(M_1)}{Pr(M_2)}
\]

What this means is that if we want to move from the prior odds in favor of \( M_1 \) to the posterior odds in favor of \( M_1 \) we simply multiply the prior odds by:

\[
B_{12} = \frac{p(y|M_1)}{p(y|M_2)}
\]

which is called the Bayes factor for \( M_1 \) relative to \( M_2 \).

In words,

\[
\text{posterior odds} = \text{Bayes factor} \times \text{prior odds}
\]
Since the posterior odds equal the Bayes factor when the models are equally likely
a priori, the Bayes factor is a measure of how much support is available in the
data for one model relative to another.

It is now possible to calculate Bayes factors and posterior probabilities of models
with MCMCpack.

data(birthwt)
model1 <- MCMCregress(bwt~age+lwt+as.factor(race) + smoke + ht,
data=birthwt,
b0=c(2700, 0, 0, -500, -500, -500, -500),
B0=c(1e-6, .01, .01, 1.6e-5, 1.6e-5, 1.6e-5, 1.6e-5), c0=10, d0=4500000,
marginal.likelihood="Chib95", mcmc=50000)
model2 <- MCMCregress(bwt~age+lwt+as.factor(race) + smoke,
data=birthwt,
b0=c(2700, 0, 0, -500, -500, -500),
B0=c(1e-6, .01, .01, 1.6e-5, 1.6e-5, 1.6e-5, 1.6e-5), c0=10, d0=4500000,
marginal.likelihood="Chib95", mcmc=50000)
model3 <- MCMCregress(bwt~as.factor(race) + smoke + ht,
data=birthwt,
b0=c(2700, -500, -500, -500, -500),
B0=c(1e-6, 1.6e-5, 1.6e-5, 1.6e-5, 1.6e-5), c0=10, d0=4500000,
marginal.likelihood="Chib95", mcmc=50000)

BF <- BayesFactor(model1, model2, model3)
mod.probs <- PostProbMod(BF)

print(BF)
The matrix of Bayes Factors is:

<table>
<thead>
<tr>
<th></th>
<th>model1</th>
<th>model2</th>
<th>model3</th>
</tr>
</thead>
<tbody>
<tr>
<td>model1</td>
<td>1.000</td>
<td>14.08</td>
<td>7.289</td>
</tr>
<tr>
<td>model2</td>
<td>0.071</td>
<td>1.00</td>
<td>0.518</td>
</tr>
<tr>
<td>model3</td>
<td>0.137</td>
<td>1.93</td>
<td>1.000</td>
</tr>
</tbody>
</table>

The matrix of the natural log Bayes Factors is:

<table>
<thead>
<tr>
<th></th>
<th>model1</th>
<th>model2</th>
<th>model3</th>
</tr>
</thead>
<tbody>
<tr>
<td>model1</td>
<td>0.00</td>
<td>2.645</td>
<td>1.986</td>
</tr>
<tr>
<td>model2</td>
<td>-2.64</td>
<td>0.000</td>
<td>-0.658</td>
</tr>
<tr>
<td>model3</td>
<td>-1.99</td>
<td>0.658</td>
<td>0.000</td>
</tr>
</tbody>
</table>

print(mod.probs)

<table>
<thead>
<tr>
<th></th>
<th>model1</th>
<th>model2</th>
<th>model3</th>
</tr>
</thead>
<tbody>
<tr>
<td>model1</td>
<td>0.82768865</td>
<td>0.05878317</td>
<td>0.11354819</td>
</tr>
</tbody>
</table>
Example 3: Single Block Metropolis Sampling for a User-Defined Model

MCMCpack also has some facilities for fitting user-specified models. The `MCMCmetrop1R()` function uses a random walk Metropolis algorithm to sample from a user-defined log-posterior density.

Suppose one is interested in fitting a Bayesian negative binomial regression to the Ornstein data in the car package.

One could do the following:

```r
negbinlogpost <- function(theta, y, X, b0, B0, c0, d0){
  ## log of inverse gamma density
  logginvgamma <- function(sigma2, a, b){
    logf <- a * log(b) - lgamma(a) + -(a+1) * log(sigma2) + -b/sigma2
    return(logf)
  }

  ## log of multivariate normal density
  logdmvnorm <- function(theta, mu, Sigma){
    d <- length(theta)
    logf <- -0.5*d * log(2*pi) - 0.5*log(det(Sigma)) -
    0.5 * t(theta - mu) %*% solve(Sigma) %*% (theta - mu)
    return(logf)
  }

  k <- length(theta)
  beta <- theta[1:(k-1)]
  alpha <- exp(theta[k])
  mu <- exp(X %*% beta)

  ## evaluate log-likelihood at (alpha, beta)
  log.like <- sum(lgamma(y+alpha) - lfactorial(y) - lgamma(alpha) +
    alpha * log(alpha/(alpha+mu)) +
    y * log(mu/(alpha+mu))
  )

  ## evaluate log prior at (alpha, beta)
  ## note Jacobian term necessary b/c of transformation
  log.prior <- logginvgamma(alpha, c0, d0) + theta[k] +
  logdmvnorm(beta, b0, B0)

  return(log.like+log.prior)
}

library(car)
data(Ornstein)
yvec <- Ornstein$interlocks
Xmat <- model.matrix(~sector+nation, data=Ornstein)

post.negbin <- MCMCmetrop1R(negbinlogpost, theta.init=rep(0,14),
  X=Xmat, y=yvec,
  thin=5, mcmc=50000, burnin=1000,
  tune=rep(.7,14),
  verbose=500, logfun=TRUE, optim.maxit=100,
  b0=0, B0=diag(13)*1000, c0=1, d0=1)

summary(post.negbin)

Iterations = 1001:50999
Thinning interval = 2
Number of chains = 1
Sample size per chain = 25000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>SD</th>
<th>Naive SE</th>
<th>Time-series SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>beta.(Intercept)</td>
<td>2.34860</td>
<td>0.1602</td>
<td>0.0010132</td>
<td>0.003964</td>
</tr>
<tr>
<td>beta.sectorBNK</td>
<td>1.70677</td>
<td>0.4014</td>
<td>0.0025385</td>
<td>0.010942</td>
</tr>
<tr>
<td>beta.sectorCON</td>
<td>-0.59210</td>
<td>0.5526</td>
<td>0.0034949</td>
<td>0.015653</td>
</tr>
<tr>
<td>beta.sectorFIN</td>
<td>0.99221</td>
<td>0.2653</td>
<td>0.0016778</td>
<td>0.007198</td>
</tr>
<tr>
<td>beta.sectorHLD</td>
<td>0.30736</td>
<td>0.4251</td>
<td>0.0026884</td>
<td>0.011383</td>
</tr>
<tr>
<td>beta.sectorMAN</td>
<td>0.12265</td>
<td>0.2226</td>
<td>0.0014077</td>
<td>0.005949</td>
</tr>
<tr>
<td>beta.sectorMER</td>
<td>0.23269</td>
<td>0.2728</td>
<td>0.0017253</td>
<td>0.007095</td>
</tr>
<tr>
<td>beta.sectorTRN</td>
<td>0.89300</td>
<td>0.2779</td>
<td>0.0017578</td>
<td>0.007095</td>
</tr>
<tr>
<td>beta.nationOTH</td>
<td>-0.07207</td>
<td>0.2985</td>
<td>0.0018882</td>
<td>0.008870</td>
</tr>
<tr>
<td>beta.nationUK</td>
<td>-0.56004</td>
<td>0.2736</td>
<td>0.0017304</td>
<td>0.008131</td>
</tr>
<tr>
<td>beta.nationUS</td>
<td>-0.84186</td>
<td>0.1511</td>
<td>0.0009559</td>
<td>0.004240</td>
</tr>
</tbody>
</table>

Applied Bayesian Inference in R Using MCMCpack Andrew Martin and Kevin Quinn.
### Future Plans

- add more models (generalized linear mixed models, Dirichlet process mixture models, dynamic linear models, other models suggested by users)
- add more options for prior specifications
- add additional flexible sampling engines (more complicated Metropolis-Hastings methods, slice sampling, etc.)
- improve performance
- add vignettes
- add enhancements to `coda` `mcmc` objects to allow for sampling from posterior and prior predictive distributions

### Acknowledgements

**National Science Foundation**
Program in Methodology, Measurement, and Statistics
(Grants SES-0350646 and SES-0350613)

Washington University, Department of Political Science, and the Weidenbaum Center on the Economy, Government, and Public Policy

[http://wc.wustl.edu](http://wc.wustl.edu)

Harvard University, Department of Government, and the Institute for Quantitative Social Science

[http://www.iq.harvard.edu](http://www.iq.harvard.edu)