Using the `mcmcgg` Package

Jacob Simmering

December 16, 2013

1 Introduction

The `mcmcgg` package includes a set of simple functions for the visualization of output from MCMC chains, specifically those made using OpenBUGS/WinBUGS and imported into R with `coda`. The main feature is that `mcmcgg` provides `ggplot2` graphics over the relatively poor visualizations provided by OpenBUGS/WinBUGS. Additionally, the control over some of the visualizations, specificially with `bgr.gg()`, provides much greater control over the resulting graph quality.

2 A Worked Example

Using OpenBUGS's pump data example (OpenBUGS Examples Vol 1), use OpenBUGS to create the model as defined in the documentation and set two chains using the provided inits. I ran for 1000 iterations but you may choose any number.

2.1 OpenBUGS

After running the MCMC for your desired number of iterations while monitoring the parameters of interest, open the sample window and click the button labeled "coda". This will bring up a set of windows, one per chain and an index. Save each window using the File menu. Note that OpenBUGS defaults to ODC and you will want to change that to txt using the dropdown. I suggest the creative names of `chain1.txt`, `chain2.txt` and `index.txt`.

2.2 Importing into R

Its fairly easy to get the resulting data into R using the `read.coda` function in the `coda` package. Specifically, you will want a command of the form

```
chain1 <- read.coda("path/to/chain1.txt", "path/to/index.txt")
```

and a similar command for chains 2, 3, ...

This results in a CODA object. These objects are similar to a matrix with columns matching to parameters and rows matching to iteration number. You will have one CODA object per chain.

2.3 Using `mcmcgg`

The three major functions of `mcmcgg` are `bgr.gg()`, `history.gg()` and `acf.gg()`. The uses should be clear from the names of each function. Each function expects an argument `x` that is a vector of the chains for the parameter value of interest. This vector can be easily formed using

```
x <- c(abstractParam(indexOfDesiredParam, codaObject1), abstractParam(indexOfDesiredParam, codaObjectN))
```

The resulting vector can be usred directly with the `mcmcgg` functions to produce the plots. The other functions all take as arguments

- `iter`: the number of iterations per chain
- `nchain`: the number of chains in `x`
- `T`: the twice the iteration numbers at which to consider splitting the burn-in and inference samples (`bgr.gg` ONLY)

3 Other Notes

It is possible to combine these functions with functions like `lapply()` or a `for` loop when making graphs for many parameters. It is fairly trivial to do this and may be a future direction for this package but currently, it is required that the user do this by "hand".