Running WinBUGS from within R

1 Batch Mode

Although WinBUGS includes methods to analyze the output of the Markov chains like summary statistics and kernel density estimates, it is often desirable to be able to save the output and read it into R for further posterior analysis or use R first to manipulate the original data before reading into WinBUGS. Another potential use is to run the same WinBUGS model with different choices of prior parameters to check the sensitivity of the posterior to prior assumptions. For these purposes, the R package R2WinBUGS makes use of the batch mode facility and provides the tools to call WinBUGS directly after data manipulation in R. And it is possible to directly import the WinBUGS output to R, for further processing, graphical displays, convergence diagnostics with CODA or posterior predictive simulations.
A detailed explanation of R2WinBUGS can be found in
Sturtz, S., Ligges, U., Gelman, A. (2005): R2WinBUGS: A Package for Running WinBUGS from R. Journal of Statistical Software 12(3), 1-16.

2 Installation of R2WinBUGS

R2WinBUGS is available from CRAN (Comprehensive R Archive Network), i.e.
`http://CRAN.R-Project.org`
or one of its mirrors. R2WinBGS can be installed by typing `install.packages("R2WinBUGS")` at the `R` prompt. Remember to load the package with `library("R2WinBUGS")`.

3 Setting up files and vectors and running

We will consider the introductory binomial example used in the introduction to WinBUGS. We first need to set up the model, data, and initial value files. Then, the main function `bugs()` takes the data and initial values, automatically writes a WinBUGS script, compiles and runs the model in WinBUGS and saves the output for easy access within `R`.

1. Write the BUGS model in an ASCII file (or if you have a model file in WinBUGS .odc format, open it in WinBUGS and save it as plain text), e.g. `betabin.txt`, containing:

   ```
   model
   {
   theta ~ dbeta(9.2,13.8)
   x ~dbin(theta,n)
   }
   ```

   and store it in your working directory.

2. Go into `R`. Change to the working directory where you want to store your input/output files.
3. Prepare data and inputs for the `bugs` function.
   The data can either be a named list with names corresponding to variable names in the `model.file` or a vector or list of the names of the data objects used by the model. In the example above:

   ```
       > data = list x =15,n=20)
       > inits = list(theta=0.8)
   ```

   If data have already been written in a file called '`data.txt`' to the working directory, it is possible to specify `data="data.txt"`. Note that input for WinBUGS must not exceed a certain number of digits. Moreover, it needs an `E` instead of *e* in scientific notation. Scientific notation is particularly desirable because of the limitation in the number of digits. The `R` function

   ```
     bugs.data(data, dir = getwd(), digits = 5, data.file = "data.txt")
   ```

   writes a data file for WinBUGS to read into the working directory and automatically adjusts the format. The default `digits=5` reformats for instance the number 123456.789 to 1.23457E+05. `data` can be a named list of the data for the WinBUGS model or a vector or list of the names of the data objects. This function returns the name of `data.file`.

   Similarly,

   ```
    bugs.inits(inits, n.chains, digits,
                   inits.files = paste("inits", 1:n.chains, ".txt", sep = ""))
   ```

   prepares the inital value files.
   `inits` is a list with `n.chains` elements where each element is itself a list of starting values for each chain of the WinBUGS model. Quite often, it is desirable to define a function that generates starting values, so that different chains can be automatically initialized at different points, e.g. if you want to run the above model 5 times with different starting values:

   ```
     inits=function(){list(theta=runif(5))}
   ```

   If `inits()` is not specified, `bugs()` will just use the initial values generated by WinBUGS, but WinBUGS can crash when reasonable initial values are not specified.
4. Specify the parameters that need to be monitored and its simulated values saved, e.g. here

   ```
   parameters=c("theta")
   ```
5. Run the WinBUGS model from within R by calling `bugs()`:

```
> bb.sim <- bugs(data,inits,parameters.to.save=parameters, model.file="betabin.txt",
                 n.chains=5, n.iter=2000,n.burnin=1000,working.directory=getwd())
```

Note that if no `working.directory` is specified, WinBUGS in- and output fiies will be stored in a temporary working directory.

6. A WinBUGS window will pop up and `R` will freeze up. The model will now run in WinBUGS, documented in the Log window within WinBUGS. The following is provided in the Log Window for the beta-binomial example and also stored in a `log.txt` file in the working directory. Note that the script is also created and stored in the file `script.txt` in the working directory.

```
display(log)
check(C:/My Documents/karlsruhe/lecture/WinBUGS/R2W/betabin.txt)
model is syntactically correct
data(C:/My Documents/karlsruhe/lecture/WinBUGS/R2W/data.txt)
data loaded
compile(1)
model compiled
inits(1,C:/My Documents/karlsruhe/lecture/WinBUGS/R2W/inits1.txt)
this chain contains uninitialized variables
gen.inits()
initial values generated, model initialized
thin.updater(1)
update(1000)
set(theta)
set(deviance)
dic.set()
update(1000)
coda(*,C:/My Documents/karlsruhe/lecture/WinBUGS/R2W/coda)
stats(*)

Node statistics
 node  mean  sd  MC error 2.5% median 97.5% start sample
deviance 6.748 2.44 0.0681 3.347 6.387 12.67 1001 1000
theta 0.5601 0.07647 0.002011 0.4117 0.5579 0.7122 1001 1000
dic.stats()

DIC
Dbar = post.mean of -2logL; Dhat = -2LogL at post.mean of stochastic nodes
Dbar Dhat pD DIC
x 6.748 6.302 0.445 7.193
total 6.748 6.302 0.445 7.193
history(*,C:/My Documents/karlsruhe/lecture/WinBUGS/R2W/history.odc)

History
```

```
save(C://My Documents/karlsruhe/lecture/WinBUGS/R2W/log.odc)
save(C:/My Documents/karlsruhe/lecture/WinBUGS/R2W/log.txt)
```

When WinBUGS is finished, the WinBUGS window will close and `R` will work again.

7. If an error message appears, rerun with `debug=TRUE`. This allows you to view the script working and not have WinBUGS close.

   Since the communication of WinBUGS and `R` is based on files, potentially huge files will be saved in the working directory by the `bugs()` call. You might want to delete those files after the desired contents has been imported into `R`.

4 Working with WinBUGS Output

The function `bugs()` returns a complex object of class `bugs`. In order to look at the structure of such an object, type `str(objectname)`. `R2WinBUGS` provides methods corresponding to class `bugs` for the generic functions `print()` and `plot()`. For our example, the commands

```
attach(bb.sim)
print(bb.sim)
plot(density(theta))
```

print a statistical summary of the posterior distribution of the monitored parameters and plot the kernel density estimate obtained from the MCMC samples of the parameter `theta`.

```
Inference for Bugs model at "betabin.txt", fit using WinBUGS,
 1 chains, each with 2000 iterations (first 1000 discarded)
 n.sims = 1000 iterations saved
         mean  sd 2.5% 25% 50% 75% 97.5%
theta     0.6 0.1  0.4 0.5 0.6 0.6   0.7
deviance  6.7 2.4  3.3 4.9 6.4 8.1  12.6

DIC info (using the rule, pD = Dbar-Dhat)
pD = 0.4 and DIC = 7.2
DIC is an estimate of expected predictive error (lower deviance is better).
```

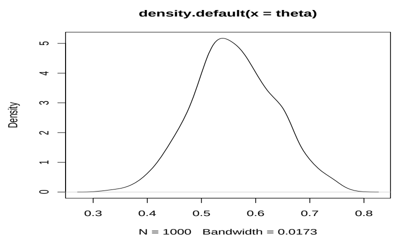

Figure 1: Kernel Density Estimate of `theta`.