Applied Bayesian Modeling
A brief R2WinBUGS tutorial

Christopher Hare
University of Georgia
chare@uga.edu
Supplementary R and WinBUGS code:
http://www.christopherdhare.com

ICPSR Summer Program 2014

Contents

1 Bayesian modeling using WinBUGS

WinBUGS is a powerful (and free!) program to perform Bayesian analysis. It also provides a stand-alone GUI (graphical user interface) that can be more user-friendly and also allows for the real-time monitoring of the chains. WinBUGS (Version 1.4.3) can be downloaded here: `http://www.mrc-bsu.cam.ac.uk/software/bugs/the-bugs-project-winbugs/`. Steps for installing WinBUGS:

1. Download and install `WinBUGS14.exe`. I recommend downloading and installing in your user's profile directory (e.g., `/Documents/`) simply to avoid annoying authentication error messages.
2. Download and install the patch for Version 1.4.3. You can do this simply by opening the file from within WinBUGS and following the instructions at the top.
3. Download and install the free key for unrestricted use, again by simply by opening the file from within WinBUGS and following the instructions at the top.

2 What is R2WinBUGS?

`R2WinBUGS` is a package for the `R` statistical platform that allows you to run WinBUGS without leaving `R`. Why might we want to do this? There are a few reasons:

1. We really like `R` (who doesn't?) and want to stay in this world.
2. It can be a pain to get data from Stata, SPSS, .csv, etc. files into WinBUGS format. It is easier to do this within `R`.
3. It can also be a pain to get the chain results out of WinBUGS into `R` to run convergence diagnostics, summarize the results, and create graphs.
4. Staying within `R` allows us to create script files so that we can easily replicate the analysis, including running convergence diagnostics and organizing the results.

Installing the `R2WinBUGS` package is easy: simply select **Install Packages** from the **Packages** menu in `R` and select the `R2WinBUGS` package. You will then need to load the package with the command:
`library(R2WinBUGS)`.
More information about the `R2WinBUGS` package can be found from these sources:

- `http://cran.r-project.org/web/packages/R2WinBUGS/index.html/`
- `http://www.jstatsoft.org/v12/i03/`

3 An example using the Angell data

To demonstrate use of the `R2WinBUGS` package, let's run a simple linear regression model using the `Angell` data (`http://socserv.socsci.mcmaster.ca/jfox/Books/Applied-Regression-2E/datasets/Angell.pdf`) available in the `car` package. First load the libraries and load and attach the `Angell` dataset:

```
# Load R libraries "R2WinBUGS" "coda" and "car"
library(R2WinBUGS)
library(coda)
library(car)
```

```
# Load data
data(Angell)
attach(Angell)
```

We next take a brief look at the data:

```
Angell[1:5,]
          moral hetero mobility region
Rochester  19.0   20.6     15.0      E
Syracuse   17.0   15.6     20.2      E
Worcester  16.4   22.1     13.6      E
Erie       16.2   14.0     14.8      E
Milwaukee  15.8   17.4     17.6     MW

summary(Angell)
     moral           hetero         mobility     region
 Min.   : 4.20   Min.   :10.60   Min.   :12.10   E : 9
 1st Qu.: 8.70   1st Qu.:16.90   1st Qu.:19.45   MW:14
 Median :11.10   Median :23.70   Median :25.90   S :14
 Mean   :11.20   Mean   :31.37   Mean   :27.60   W : 6
 3rd Qu.:13.95   3rd Qu.:39.00   3rd Qu.:34.80
 Max.   :19.00   Max.   :84.50   Max.   :49.80
```

We want to estimate a simple linear regression model in which `moral` is regressed onto `hetero` and `mobility`. This model can be estimated within the familiar frequentist framework as below:

```
res <- lm(moral ~ hetero + mobility)
summary(res)
Call:
lm(formula = moral ~ hetero + mobility)

Residuals:
   Min     1Q Median     3Q    Max
-5.071 -1.194 -0.206  1.738  4.195

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 19.94076    1.19265  16.720  < 2e-16 ***
hetero      -0.10856    0.01699  -6.389 1.34e-07 ***
mobility    -0.19331    0.03543  -5.456 2.74e-06 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 2.243 on 40 degrees of freedom
Multiple R-squared:  0.6244,    Adjusted R-squared:  0.6056
F-statistic: 33.25 on 2 and 40 DF,  p-value: 3.126e-09
```

Though beyond the scope of this tutorial, it is worth noting that we can create a series of dummy variables from nominal data (like `region`) with the command:

```
region.dummies <- model.matrix(~region)
region.dummies[1:5,]
  (Intercept) regionMW regionS regionW
1           1        0       0       0
2           1        0       0       0
3           1        0       0       0
4           1        0       0       0
5           1        1       0       0
```

How do we implement a Bayesian form of the linear regression model? How about we say that our response variable (`moral`) is normally distributed with mean μ and precision τ. μ is a function of three parameters: α (the intercept term), β_1 (the regression coefficient for the `hetero` variable), and β_2 (the regression coefficient for the `mobility` variable).

We will set diffuse priors for these three parameters as well as the precision term: $\alpha \sim N(0,100)$, $\beta_1 \sim Unif(-1000,1000)$, $\beta_2 \sim Unif(-1000,1000)$, and $tau \sim Gamma(1,0.1)$.

Here's what that implementation looks like in BUGS (available in the file `angell.bug`):

```
model {
for(i in 1:N){
moral[i] ~ dnorm(mu[i], tau)
mu[i] <- alpha + beta1*hetero[i] + beta2*mobility[i]
}
# priors
alpha ~ dnorm(0, .01)
beta1 ~ dunif(-1000,1000)
beta2 ~ dunif(-1000,1000)
tau ~ dgamma(1,0.1)
}
```

We next need to prepare the data to pass to WinBUGS. In addition to defining the three variables (`moral`, `hetero`, and `mobility`), we also need to define `N` (the number of observations) because it is undefined in the BUGS code `N`. Below we create a `list` of the data (which we creatively name `data`) that includes all four components:

```
N <- nrow(Angell)
data <- list(moral=moral, hetero=hetero, mobility=mobility, N=N)
```

We also need to generate initial (or starting) values for the parameters. In `R` we can (and probably should) define a function that generates initial values randomly drawn from some distribution. Below we define a function (`inits()`) that creates starting values for each of the parameters.

```
inits <- function(){
list(alpha=rnorm(1), beta1=rnorm(1), beta2=rnorm(1), tau=runif(1,0,2))
}

inits()
$alpha
```

```
[1] 0.8095699

$alpha2
[1] 1.971686

$beta1
[1] -1.290824

$beta2
[1] -0.2639362

$tau
[1] 0.2916423
```

Finally we need to set the names of the parameters that we want to monitor:

```
parameters <- c("alpha", "beta1", "beta2", "tau")
```

We are now ready to use the `bugs()` function, which calls WinBUGS. We store the results in the object `sims`, and specify the location of the BUGS file, the data, the parameters, the initial values, as well as how many chains we want to run and how long we want to run them. We say we want to run two chains for 25,000 iterations, discarding the first 20,000 values, and thinning the remaining values by 5. Finally, we need to specify the location of the WinBUGS directory (where you installed WinBUGS):

```
sims <- bugs(model.file="c:/Dropbox/Files/PhD/Summer 2014/ICPSR/Bayesian Modeling/angell.bug",
data = data,
parameters = parameters,
inits = inits,
        n.chains = 2,
n.iter = 25000, n.burnin = 20000, n.thin = 5,
bugs.directory = "c:/Users/Christopher/Documents/WinBUGS14/")
```

We want to make sure the results have face validity (we'll get into more formal convergence diagnostics next). We can first plot the summary statistics for the monitored parameters:

```
# Plot results
plot(sims, display.parallel = TRUE)
```

We can also print the summary statistics. We might also want to store the sampled values for each parameter (from each chain) in separate objects. Below we store the chain values for β_1 in the objects `b1` and `b2`. This allows us to plot and compare the sampled values for a given parameter:

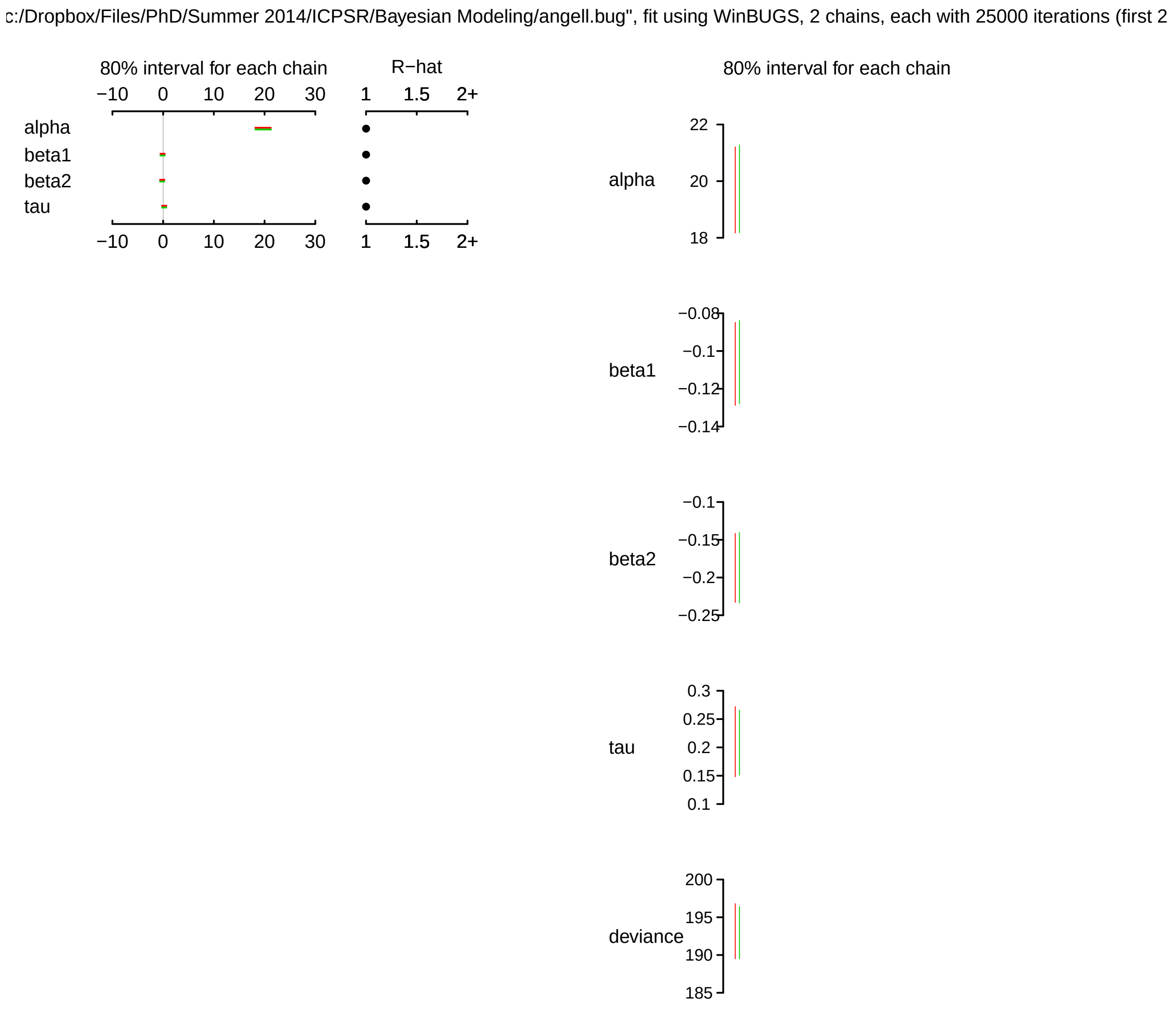
```
# Examine samples
print(sims)
b1 <- sims$sims.array[,,"beta1"][,1]
b2 <- sims$sims.array[,,"beta1"][,2]
#
plot(b1,type="l",col="red")
lines(b2,type="l",col="blue")
```

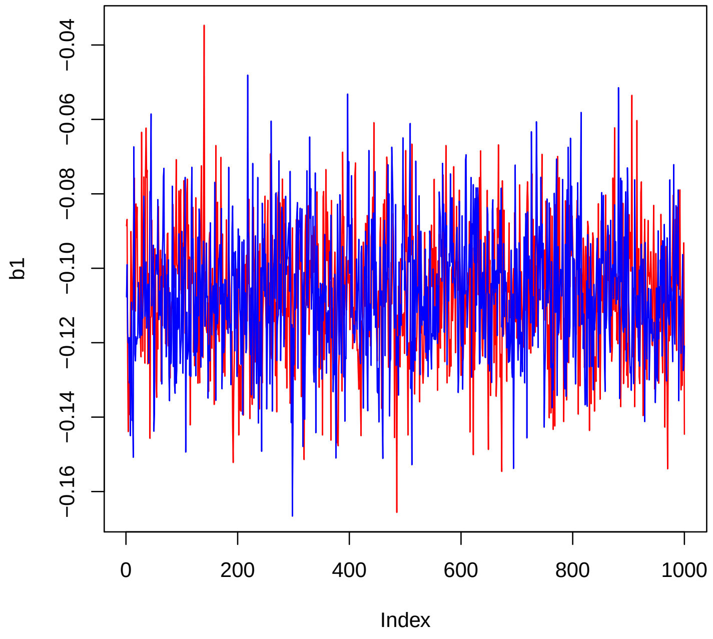
```
#
plot(density(b1),lwd=2,col="red")
lines(density(b2),lwd=2,col="blue")
```

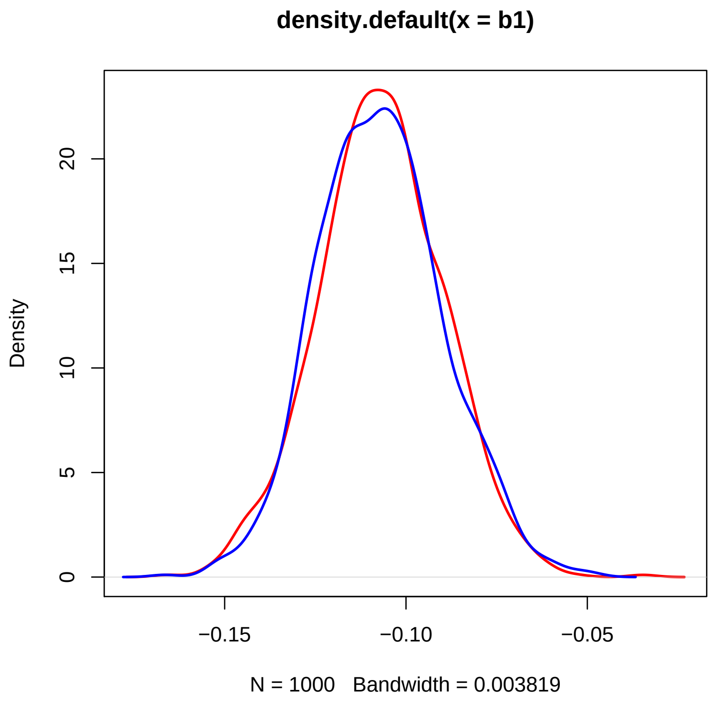
Finally, we can use convergence diagnostic functions available in the `coda` package. Below we run the Geweke diagnostic and the Gelman-Rubin diagnostic:

```
# Convergence diagnostics
A <- as.mcmc.list(sims)

# Geweke diagnostic
geweke.diag(A)
[[1]]

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

   alpha    beta1    beta2 deviance      tau
 -1.2776   1.2711   0.9953   1.5812  -1.2039


[[2]]

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

   alpha    beta1    beta2 deviance      tau
   2.214   -1.591   -2.018   -1.006   -2.412

# Gelman-Rubin diagnostic
gelman.diag(A)
Potential scale reduction factors:

         Point est. Upper C.I.
alpha             1          1
beta1             1          1
beta2             1          1
deviance          1          1
tau               1          1

Multivariate psrf

1
```

4 Plots


c:/Dropbox/Files/PhD/Summer 2014/ICPSR/Bayesian Modeling/angell.bug", fit using WinBUGS, 2 chains, each with 25000 iterations (first 2
80% interval for each chain
R−hat
−10 0 10 20 30
1 1.5 2+
alpha
beta1
beta2
tau
80% interval for each chain
22 20 18
alpha
−0.08 −0.1 −0.12 −0.14
beta1
−0.1 −0.15 −0.2 −0.25
beta2
0.3 0.25 0.2 0.15 0.1
tau
200 195 190 185
deviance


b1
−0.16
−0.14
−0.12
−0.10
−0.08
−0.06
−0.04
0
200
400
600
800
1000
Index


density.default(x = b1)
Density
0
5
10
15
20
−0.15
−0.10
−0.05
N = 1000 Bandwidth = 0.003819