

Bayesian Statistics, 22S:138

Lab 6, Nov. 19, 2004

Using WinBUGS for Model Comparison, Part 1

Using the Deviance Information Criterion to compare link functions in a generalized linear model

The file "beetles.bug" in the "Handouts" section on the course web page contains WinBUGS code and data for the "Beetles" problem from volume 2 of the classic BUGS examples. I changed the code slightly so that it would run under WinBUGS.

The problem is a dose-response study in which each of eight different groups of approximately 60 beetles were exposed to different concentrations of an insecticide for 5 hours. The response variable was how many beetles were killed. The question of interest was the relationship between concentration of insecticide and proportion killed.

A binomial likelihood is appropriate for the data from each dose level. Let n_i denote the number of beetles exposed to dose i , $i = 1, \dots, 8$ and r_i denote the number killed. If the observations on each group are conditionally independent given the model parameters, the likelihood for all the data is proportional to:

$$\prod_{i=1}^8 p_i^{r_i} (1 - p_i)^{n_i - r_i}$$

This likelihood would take on its maximum possible value if there were 8 free parameters and each p_i were equal to $\frac{r_i}{n_i}$. The log of this maximum possible likelihood is the "saturated log likelihood."

Frequentists would compare models for these data by looking at the "deviance," which is 2 * (saturated log likelihood - maximized log likelihood under the model of interest).

We wish to compare the fit of three different 2-parameter models to these data. Each is of the form

$$\text{transformed}(p_i) = \alpha^* + \beta * (\text{dose}_i - \bar{\text{dose}})$$

The difference is in which link function is used to transform the p_i 's to values on the whole real line.

We will use the Deviance Information Criterion (DIC), a measure of comparative model fit that is built into WinBUGS and can be accessed from the Inference menu. Here is the section of the WinBUGS manual that discusses the DIC. Further explanation is on pp. 180-182 of your textbook.

The DIC Tool dialog box is used to evaluate the Deviance Information Criterion (DIC; Spiegelhalter et al., 2002) and related statistics - these can be used to assess model complexity and compare different models. Most of the examples packaged with WinBUGS contain an example of their usage.

It is important to note that DIC assumes the posterior mean to be a good estimate of the stochastic parameters. If this is not so, say because of extreme skewness or even bimodality, then DIC may not be appropriate. There are also circumstances, such as with mixture models, in which WinBUGS will not permit the calculation of DIC and so the menu option is greyed out. Please see the WinBUGS 1.4 web-page for current restrictions:

<http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml>

set: starts calculating DIC and related statistics - the user should ensure that convergence has been achieved before pressing set as all subsequent iterations will be used in the calculation.

clear: if a DIC calculation has been started (via set) this will clear it from memory, so that it may be restarted later.

DIC: displays the calculated statistics, as described below; please see Spiegelhalter et al. (2002) for full details; the section Tricks: Advanced Use of the BUGS Language also contains some comments on the use of DIC.

The DIC button generates the following statistics:

Dbar: this is the posterior mean of the deviance, which is exactly the same as if the node 'deviance' had been monitored (see here). This deviance is defined as $-2 * \log(\text{likelihood})$: 'likelihood' is defined as $p(y - \theta)$, where y comprises all stochastic nodes given values (i.e. data), and θ comprises the stochastic parents of y - 'stochastic parents' are the stochastic nodes upon which the distribution of y depends, when collapsing over all logical relationships.

Dhat: this is a point estimate of the deviance ($-2 * \log(\text{likelihood})$) obtained by substituting in the posterior means θ_{bar} of θ : thus $\text{Dhat} = -2 * \log(p(y - \theta_{\text{bar}}))$.

pD: this is 'the effective number of parameters', and is given by $\text{pD} = \text{Dbar} - \text{Dhat}$. Thus pD is the posterior mean of the deviance minus the deviance of the posterior means.

DIC: this is the 'Deviance Information Criterion', and is given by $\text{DIC} = \text{Dbar} + \text{pD} = \text{Dhat} + 2 * \text{pD}$. The model with the smallest DIC is estimated to be the model that would best predict a replicate dataset of the same structure as that currently observed.

We will go through the following steps for each of the three candidate link functions:

1. Un-comment the line in the code with the desired link function. Comment out the lines defining the other link function.
2. Check model, load data, compile, load inits.
3. Set monitors on
 - alpha

- beta
- r.hat
- deviance
- sumllik and D (if you are using the second version of the model)

- Run 1000 updates
- check history plots for convergence
- Open the DIC tool from the Inference menu and "set"
- Run 1000 additional updates
- Get the stats for all monitored parameters, using iterations 1001-2000
- Click "DIC" on the DIC tool to get the DIC
- Label both the stats output and the DIC output as to which link function it represent

Note that there are only two free parameters in each model, so pD should be a sampling-based estimate of the integer 2.

Which of the three link functions appears to fit the data best?

Question to consider: Would it make sense to use the deviance to compare different models if informative priors were being used to incorporate information from other datasets?

```

model
{
  for (i in 1:N) {
    r[i] ~ dbin(p[i], n[i]);

# alternative link functions (try one at a time):
    logit(p[i]) <- alpha.star + beta*(x[i]-mean(x[]));
    # probit(p[i]) <- alpha.star + beta*(x[i]-mean(x[]));
    # cloglog(p[i]) <- alpha.star + beta*(x[i] - mean(x[]));

    r.hat[i] <- p[i]*n[i]; # fitted values
  }
  alpha.star ~ dnorm(0.0, 1.0E-3);
  beta ~ dnorm(0.0, 1.0E-3);
  alpha <- alpha.star - beta*mean(x[]);
}

```

```

data
list(x = c(1.6907, 1.7242, 1.7552, 1.7842, 1.8113, 1.8369, 1.8610, 1.8839),
     n = c(59, 60, 62, 56, 63, 59, 62, 60),
     r = c(6, 13, 18, 28, 52, 53, 61, 60), N = 8
)

# Here is an version of the model that lets you see how WinBUGS is calculating
# the log likelihood ;

model
{
  # constants used in computing normalizing constants of log likelihood

  for (i in 1:N) {
    lfactn[i] <- logfact(n[i])
    lfactr[i] <- logfact(r[i])
    lfactmminr[i] <- logfact(n[i] - r[i])
  }

  # saturated log-likelihoods ;
  for (i in 1:(N-1)) {
    llike.sat[i] <- lfactn[i] - lfactr[i] - lfactmminr[i] +
      r[i]*log(r[i]/n[i]) + (n[i]-r[i])*log(1-r[i]/n[i]);
  }
  llike.sat[N] <- lfactn[N] - lfactr[N] - lfactmminr[N] + 0 ;
  # assign this one separately to avoid error due to ;
  # trying to evaluate log(0) ;

  for (i in 1:N) {
    r[i] ~ dbin(p[i], n[i]);

# alternative link functions (try one at a time):
    # logit(p[i]) <- alpha.star + beta*(x[i]-mean(x[]));
    # probit(p[i]) <- alpha.star + beta*(x[i]-mean(x[]));
    cloglog(p[i]) <- alpha.star + beta*(x[i] - mean(x[]));

# log likelihood for sample i ;
    llike[i] <- lfactn[i] - lfactr[i] - lfactmminr[i] +
      r[i]*log(p[i]) + (n[i]-r[i])*log(1-p[i]);

    r.hat[i] <- p[i]*n[i]; # fitted values
  }
}

```

```

}
alpha.star ~ dnorm(0.0, 1.0E-3);
beta ~ dnorm(0.0, 1.0E-3);
alpha <- alpha.star - beta*mean(x[]);

sumllik <- sum(llike[]) # -2 times this is what WinBUGS calls the deviance
D <- 2 * (sum(llike.sat[]) - sumllik); # frequentist deviance
}

```

Example of computing a discrepancy measure using posterior predictive distribution: Newcomb's speed of light data

This example is taken from the Gelman, Carlin, Stern and Rubin textbook. The code is on the course web page under Handouts as newcomb.bug.

```

model
{
for(i in 1:N) {
y[i] ~ dnorm(mu, tau)
ypred[i] ~ dnorm(mu,tau) # predicted values
# (draws from posterior predictive dist'n)
}
mu ~ dnorm(0, .00001)
tau ~ dgamma(.0001, .0001)
sigmasq <- 1 / tau
s <- 1
smallest <- ranked(ypred[], s) # smallest predicted value
smallthan <- step(y[6] - smallest) # 1 if smallest
# predicted value is smaller than smallest obs value
}

```

```

data
list(y = c(28, 26, 33, 24, 34, -44, 27, 16, 40, -2, 29, 22, 24, 21, 25,
30, 23, 29, 31, 19, 24, 20, 36, 32, 36, 28, 25, 21, 28, 29,37, 25, 28,
26, 30, 32, 36, 26, 30, 22, 36, 23,27, 27, 28, 27, 31, 27, 26, 33, 26,
32, 32, 24, 39, 28, 24, 25, 32, 25, 29, 27, 28, 29, 16, 28), N=66)

```

```

inits
list(mu=20, tau=1 )

```

Checking for an outlier in the squirrel data

This example is taken from the Berry book. It concerns a study performed at the University

of Minnesota in which researchers studied squirrels to determine the relationship between the amount of rainfall in the preceding 3 days and the osmolarity (concentration) of the squirrels' urine. (As Dave Barry would say, "I am not making this up"!).

The code for this example is on the course web page under Handouts as squirrel3.bug.

```

model
{
x.bar <- mean(x[]) ;
for(i in 1:N){
Y[i] ~ dnorm(mu[i], tau)
Ypred[i] ~ dnorm(mu[i], tau) # predicted values
aresid[i] <- abs(Y[i] - mu[i]) # actual residuals
arespred[i] <- abs(Ypred[i] - mu[i]) # predicted residuals
mu[i] <- alpha + beta * (x[i] - x.bar)
}
s <-8
largest <- ranked(arespred[], s)
largthan <- step(largest - aresid[3])
sigma <- 1/sqrt(tau)
alpha ~ dnorm(0, 1.0E-6)
beta ~ dnorm(0, 1.0E-6)
tau ~ dgamma(1.0E-3, 1.0E-3)
}

```

```

list(Y = c(1479, 1500, 516, 1815, 1118, 1702, 1348, 1337), x = c(0.45,
0.0, 0.0, 0.24, 0.12, 0.0, 0.35, 0.44),
N = 8)

```

```

list(alpha = 0, beta = 0, tau = 1)

```

Note: Also, try the some diagnostic plots.