February 5, 2004

A Short Introduction to WinBUGS

Ciprian Crainiceanu

WinBUGS = **B**ayesian analysis software **U**sing **G**ibbs **S**ampling for **W**indows. It can be downloaded for free from

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

Do not forget to download the key for unrestricted use. It is on the same page. The current version is WinBUGS 1.4. This course uses heavily the WinBUGS manual, which is recommended together with the large number of examples.

1 Why Bayesian analysis using MCMC is good for you

1. Has a very solid decision-theoretical framework
2. It is intuitive:
    a. Combines the prior distribution (prior beliefs and/or experience) with the likelihood (experiment) to obtain the posterior distribution (accumulated information).
    b. In this context knowledge is equivalent to distribution and knowledge precision can be quantified by the precision parameter.
3. **It is exact.** Asymptotic approximations are not used. The "plug–in" principle is avoided.
4. Newly developed numerical methods (MCMC) make computations tractable for *practically all models*.
5. Focus shifts from model estimation to model correctness.
6. **WinBUGS**

2 MCMC methods

a. Suppose that we have a model for which we know the full joint distribution of all quantities.

b. We want to sample values of the unknown parameters from their conditional (posterior) given the data (observed values)

c. Gibbs sampling's basic idea: successively sample from the conditional distribution of each variables given all the other.

d. Metropolis-within-Gibbs algorithm: appropriate for difficult full conditionals and does not necessarily generate a new value at each iteration

e. Under broad conditions this process eventually provides samples from the **joint posterior distribution** of the unknown variables.

MCMC simulations contain an **adaptive or burn-in** period which needs to be discarded. The actual **correlated samples from the target distribution** are used for inference using simple summary statistics such as **posterior mean**, **posterior quantiles**, etc.

3 WinBUGS program structure

Every WinBUGS program has three main parts: the actual model, the data and the initial values. All these can be contained in the same or in separate files. The files can be text (.txt) and/or WinBUGS specific files (.odc).

I personally favor the (.odc) file for the excellent **fold** capability. This allows one to write code, data or initial values between two arrows and then collapse the entire folder by clicking on one of the arrows. To get an idea, open a new file in WinBUGS (Menu->File->New) and create a new folder using the following steps (Menu->Tools->Create Fold). You can now write between the two arrows and then collapse the folder. This is especially good when multiple models, data sets, text etc. are stored in the same file.

The standard WinBUGS program stored in one file looks like

```
model

#This is a comment
    {                                #begin model
    ...
    }                                #end model

#Data here in one of the two formats
```

```
#Initial parameter values here
```

Writing in the BUGS file follows the same rules as in, say, Notepad or Word. We will present a few examples to illustrate WinBUGS programming.

Of course, separate files could also be used for model, data and initial values. This is actually required if one wants to use the scripting capability of WinBUGS described in section 6.

Why do you think that folding is helpful?

4 Examples

Generally, real data have a very complex structure which is explained by complex models. Bayesian model specification (and WinBUGS) takes advantage very often of simple conditional relationship to build such complex models.

We discuss now three examples for the implementation of statistical models in WinBUGS.

4.1 The line example

Consider a set of 5 observed (x, Y) pairs $(1, 1)$, $(2, 3)$, $(3, 3)$, $(4, 3)$, $(5, 5)$. We want to fit the following model in WinBUGS

$$\begin{cases} Y_i & \sim & \text{NORMAL}(\mu_i, \tau) \\ \mu_i & = & \alpha + \beta(x_i - \bar{x}) \end{cases} , \tag{1}$$

where $\bar{x}$ is the mean of x's. Here τ is the precision of the normal distribution $\tau = \sigma^{-2}$.

To complete the specification of the model α, β and τ are given proper but minimally informative priors

$$\begin{cases} \alpha,\ \beta & \sim & \text{NORMAL}(0, 10^{-6}) \\ \tau & \sim & GAMMA(10^{-3}, 10^{-3}) \end{cases} , \tag{2}$$

where the mean and variance of GAMMA(a, b) distribution are $a/b = 1$ and $a/b^2 = 10^3$ respectively.

The BUGS language allows a concise expression of the model. Open a new file in WinBUGS and write the following code

```
model
    {for (i in 1:N)
        {Y[i]~dnorm(mu[i],tau)
         mu[i]<-alpha+beta*(x[i]-mean(x[]))}
```

```
alpha~dnorm(0,1.0E-6)
beta~dnorm(0,1.0E-6)
tau~dgamma(1.0E-3,1.0E-3)
sigma<-1/sqrt(tau)
}
```

Note that the BUGS code follows literally the model specification. Open the **Model** menu and open **Specification**. Highlight the keyword model at the beginning of the model description by double-clicking on it. Execute check model from the specification tool. "model is syntactically correct" should appear in the status line at the bottom left corner of the screen.

Data can be loaded in two formats. A combination of the two formats can also be used. The data can be represented using S-plus object notation

```
list(x=c(1,2,3,4,5), Y=c(1,3,3,3,5), N=5)
```

or as

```
list(N=5)
x[]     Y[]
1       1
2       3
3       3
4       3
5       5
END
```

To load the data using format 1, highlight the key word list at the start of data description and then execute the *load data* command from the specification tool. "data loaded" should appear in the status line. To load the data using format 2, highlight the key word list and execute *load data*, then highlight the x at the start of the rectangular array and again execute *load data*.

The second format is especially nice when data is stored in large rectangular arrays like is the case with data in Excel or Access data bases.

Warning: Do not use Copy/Paste directly from Excel because WinBUGS does not understand the spacing. Do Copy/Paste twice going from Excel->Notepad->WinBUGS.

Warning: Do not forget the keyword END at the end of the rectangular array. This is specific for WinBUGS 1.4 and was not used in the previous versions

The MCMC sampler must be given initial values that are consistent with every implicit or explicit constraint

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

Highlight the word list and then execute the *load inits* command from the Specification tool. "initial values loaded: model initialized" should appear in the status line.

WinBUGS is now ready to generate samples. From the **Model** menu open the **Update** tool, and from the **Inference** menu open the **Samples** tool.

Type the name of each node to be monitored in the *node* box and click on *set* after each name. You can adjust the number of simulations in the **Update**. Click on the *update* button of the **Update** tool and wait until the simulation is over. The time needed for simulation is displayed in the left corner of the window.

To look at the output of all parameters at once type "*" in the node box of the **Samples** tool. This tool offers summary statistics, traces, kernel density plots, autocorrelation functions, and running quantile estimates. Clicking on *coda* provides all the simulated values in two files. This values can be used for more sophisticated analyses.

Assessing the convergence of chains can be done using CODA. However, the most practical way of assessing convergence towards the target distribution is to run multiple chains with various initial values and visually inspect the histories of these chains.

4.2 Stacks: robust and ridge regression

Birkes and Dodge (1993) apply different regression models to the stack-loss data of Brownlee (1965). This features 21 daily responses of stack loss (y), the amount of ammonia escaping, with covariates being air flow (x_1), temperature (x_2) and acid concentration x_3. Data looks like this

Day	Stack loss	air flow	temp	acid
1	42	80	27	89
2	37	80	27	88
...	...	...	...	...
21	15	70	20	91

We first assume a linear regression with a variety of error structures

$$\begin{cases} \mu_i & = & \beta_0 + \beta_1 z_{1i} + \beta_2 z_{2i} + \beta_3 z_{3i} \\ y_i & \sim & \text{NORMAL}(\mu_i, \tau) \\ y_i & \sim & \text{DOUBLEEXP}(\mu_i, \tau) \\ y_i & \sim & \text{LOGISTIC}(\mu_i, \tau) \\ y_i & \sim & t(\mu_i, \tau, d) \end{cases} , \tag{3}$$

where

$$z_{ij} = \frac{x_{ij} - \bar{x}_{.j}}{sd(x_{.j})}$$

are the covariates standardized to have zero mean and unit variance.

Maximum likelihood estimates for the double exponential (Laplace) distribution minimize the sum of absolute deviations. The other options are alternative heavy tailed distributions. We use a t with four degrees of freedom, but with more data it would be possible to allow this parameter to be unknown.

Initially we allow β_1, β_2 and β_3 to have independent "noninformative" priors. We also consider "ridge regression", intended to avoid the instability due to correlated covariates. This is equivalent to assuming the regression coefficients of the standardized covariates to be exchangeable and

$$\beta_j \sim \text{NORMAL}(0, \phi), \quad j = 1, 2, 3.$$

We can apply ridge regression to each of the possible error distributions. The following model incorporates all models described here

```
model{
    #Standardize x's
    for (j in 1:p)
        {for (i in 1:n)
            {z[i,j]<-(x[i,j]-mean(x[,]))/sd(x[,j])}}


    #Likelihood definition
    d<-4        #degrees of freedom for t
    for (i in 1:N){
    #One and only one of the next four code lines is not
    #commented
    #   Y[i]~dnorm(mu[i],tau);
    #   Y[i]~ddexp(mu[i],tau);
        Y[i]~dlogis(mu[i],tau);
    #   Y[i]~dt(mu[i],tau,d);
        mu[i]<-beta0+beta[1]*z[i,1]+beta[2]*z[i,2]+beta[3]*z[i,3]}

    #Priors
    beta0~dnorm(0,1.0E-6)
    for (j in 1:p){
    #   beta[j]~dnorm(0,1.0E-6)     #independent coefficients
        beta[j]~dnorm(0,phi)}       #exchangeable coefficients

    tau~dgamma(1.0E-3,1.0E-3)
    phi~dgamma(1.0E-3,1.0E-3)

    #standard deviation of error distribution
    #sigma<-sqrt(1/tau)             #normal errors
```

```
#sigma<-sqrt(2)/tau              #double exponential errors
 sigma<-sqrt(pow(PI,2)/3)/tau   #logistic errors
#sigma<-sqrt(d/(tau*(d-2)))     #t errors on d d.f.
}
```

4.3 Pathogen counts: Poisson regression

Craniceanu *et al.*, 2003 present a Bayesian model for the ICR survey of national waterborne pathogen concentrations. The data was obtained by conducting an 18 month long survey that included 350 major water users. The basic data set for this analysis is the number of pathogens counted.

Consider a given water sample of volume V taken from a natural water body at a given time. Denote by C the unobserved pathogen concentration. One expects to have on average $N = C \times V$ pathogens in the volume of water sampled. However, the counting process of microscopic organisms is imperfect and the expected number of pathogens actually counted is only a fraction, R, of the total number N. R is called the recovery rate (for simplicity it will be considered constant and known). Thus the expected number of organisms counted is $R \times V \times C$.

The quantity of concern is C which is unobserved while the outcome of the counting process is the number of pathogens observed.

A variety of factors could have a causal relationship with pathogen concentrations such as water quality (turbidity, temperature), basin characteristics (land usage, standardized flow or residence time, type of water body), seasonality, and spatial and temporal correlations.

Bayesian model

$$\left\{\begin{array}{lcl} Y_{ij} & \sim & \text{POISSON}(\lambda_{ij}) \\ \lambda_{ij} & = & V_{ij}R_{ij}C_{ij} \\ \log(C_{ij}) & = & X_{ij}^T\beta + t_{ij} \\ t_{ij}|s_i, \sigma_t & \sim & \text{NORMAL}(s_i, \sigma_t^2) \\ s_i|r_{k(i)}, \sigma_s & \sim & \text{NORMAL}(r_{k(i)}, \sigma_s^2) \\ r_k|\mu, \sigma_r & \sim & \text{NORMAL}(\mu, \sigma_r^2) \end{array}\right. , \tag{4}$$

Here Y_{ij} is the observed pathogen count for site i and month j. The site number i runs from 1 to M, where M is the number of sites considered. The month index j runs from 1 to N_i, where generally $N_i = 18$, but sometimes $N_i < 18$. Conditional on their means, the random time-site effects t_{ij}, site effects s_i and regional effects r_k are independent and normally distributed.

In order to write the WinBUGS program I used an indexing procedure that can be useful in many applications. First, I stacked the vectors of observations for each site and formed vectors of responses and covariates. A one-to-one correspondence between the couple (i, j) and the observation number k as follows

site(i)	month	obs. # at site i(j)	obs. # (k)
1	1	1	1
1	2	2	2
1	4	3	3
...	...	...	...
1	18	17	17
2	2	1	18
2	3	2	19
...	...	...	...

Indexing can be very important, especially in models with hierarchies of random effects.

```
model {

#The original program contains a model for random recovery rates
#but for simplicity of presentation we consider a constant
#recovery rate equal to the mean empirical recovery rate

#Count part of the model

    for (k in 1:N)
        {Y[k]~dpois(lambda[k])
         lambda[k]<-V[k]*RY[k]*C[k]
         log(C[k])<-b[1]*tu[k]+b[2]*te[k]+b[3]*tcoli[k]+b[4]*bas1[k]+
                    b[5]*bas2[k]+b[6]*bas3[k]+b[7]*ph[k]+
                    b[8]*logpop[site[k]]+b[9]*logart[site[k]]+
                    b[10]*lvs[site[k]]
        t[k]~dnorm(s[site[k]], taut)
        RY[k]<-0.11
    }

#Definition of site effects
    for (i in 1:M)
        {s[i]~dnorm(r[reg[i]],taus)}

#Definition of regional effects
    for (j in 1:P)
        {r[j]~dnorm(mu,taur)}

#Priors on the coefficients of covariates
    for (i in 1:ncovariates)
        {b[i]~dnorm(0.0,1.0E-6)}
```

```
#Prior on the overall mean
    mu~dnorm(0.0,1.0E-6)

#Prior on the precision parameters
    taut~ dgamma(0.001,0.001)
    taus~ dgamma(0.001,0.001)
    taur~ dgamma(0.001,0.001)

#Deterministic transformations. Standard deviations
    sigmat<-1/sqrt(taut)
    sigmas<-1/sqrt(taus)
    sigmar<-1/sqrt(taur)
    }
```

An interesting trick used in WinBUGS is called nested indexing. We used nested indexing twice in the program, in the definition of time-site and site random effects. For example, for time-site effects we introduced an additional column vector called `site[]`, where `site[k]` is the site corresponding to the k-th observation. Therefore the BUGS code

```
 for (k in 1:N)
        {t[k]~dnorm(s[site[k]], taut)}
```

specifies that the time-site effect t_k is centered around the site effect s_i corresponding to observation k. Similarly

```
    for (i in 1:M)
        {s[i]~dnorm(r[reg[i]],taus)}
```

specifies that the time-site effect s_i is centered around the site effect r_g corresponding to site i. Here `reg[i]` represents the index of the region corresponding to the i-th site effect.

Data. Data needed for this program are

a. `Y[]`: $N \times 1$ dimensional vector of pathogen counts

b. `V[]`: $N \times 1$ vector of volumes of water analyzed

c. `tu[]`, `te[]`, `tcoli[]`, `bas1[]`, `bas2[]`, `bas3[]`, `ph[]`: $N \times 1$ dimensional vectors of covariates

d. `site[]`: $N \times 1$ dimensional vector of site indexes

e. `logpop[]`, `logart[]`, `lvs[]`: $M \times 1$ dimensional vectors of covariates

f. `reg[]`: $M \times 1$ dimensional vector of region indexes

g. `N`, `M`, and `P`: numbers of observations, sites and regions respectively.

Note that data in a., b., c., d. can be entered in rectangular format using an $N \times 10$ matrix. Data in e. and f. can be entered in rectangular format using an $M \times 4$ matrix. The three constants from point g. can be entered in a list.

Initial values. Initial values should be provided for the following random variables

a. `t[]`: the $N \times 1$ dimensional vector of time-site random effects.

b. `s[]`: the $M \times 1$ dimensional vector of site random effects.

c. `r[]`: the $P \times 1$ dimensional vector of regional random effects.

d. `b[]`: the vector of parameters of covariates

e. `mu`: the overall mean

f. `taut`, `taus`, `taur`: preciosions of time-site, site and regional random effects respectively.

5 Mixing and Reparameterization

Mixing is the property of the MCMC to move rapidly throughout the support of the posterior distribution of parameters given the data. Finding a good candidate model can be a very computationally intensive process of exploring many models. Therefore, reducing computational time for MCMC simulations could be critical.

The within-chain correlation for MCMC samples determines the required number of MCMC simulations to achieve a target accuracy. Strong posterior correlation among parameters was found to be the main reason for high within-chain correlation for MCMC sampling (Gilks and Roberts 1996). We explore methods of reducing these posterior correlations through simple techniques that can easily be implemented for a wide range of models.

There exist non-standard sampling algorithms, random and adaptive direction sampling, simulated tempering, Metropolis-coupled MCMC, and data augmentation, to reduce within-chain correlation without reducing posterior correlation among parameters (Gilks and Roberts 1996; Robert and Casella, 1999). However their net improvement in mixing for complex models is not clearly understood and their applicability limited by lack of software. For standard MCMC simulations WinBugs is the available tool.

We will use the Effective Sample Size (ESS) as a measure of mixing. ESS for a parameter is the sample size of an independent sample giving the same estimation

accuracy as the correlated MCMC sample. If K is the total number of correlated MCMC simulations and ρ_k is the lag k autocorrelation for one parameter, then

$$\text{ESS} = \frac{K}{1 + 2\sum_{k=1}^{\infty} \rho_k}. \tag{5}$$

ESS can be estimated using the sample autocorrelation function (ACF).

Improving mixing of MCMC simulation is model specific and solutions are often a mixture of art and science. Therefore, we seek simple solutions for the lack of efficiency of MCMC that can easily be implemented for a variety of models and situations.

A critical feature of the ICR data set is the large number of zero counts ($\geq$ 90%) which carry relatively little information about the parameters. For example, 500 counts that are all zero are consistent with a mean of 10^{-4} as well as 10^{-10}. However, for assessing health risk, a mean of 10^{-4} has rather different implications than a mean of 10^{-10}, so distinguishing between them is a concern. This lack of information appears also to cause poor mixing. To investigate this problem consider a test example.

5.1 Test example

The following model was used to simulate sets of counts

$$\left.\begin{array}{lcl} Y_i \,|\lambda_i & \sim & \text{Poisson}(\lambda_i) \\ \log(\lambda_i) & = & t_i \\ t_i \,|\mu,\, \sigma_t & \sim & N(\mu, \sigma_t^2) \end{array}\right\} \tag{6}$$

We generated samples of size $n = 200$ with $\sigma = 1$ for $\mu = -2, -1, 0$ and 1 corresponding to unconditional Poisson means for λ_i of 0.22, 0.61, 1.65, 4.48. In this framework the information in the sample is represented by the mean count and the percent of non-zero counts. Table 1 provides a summary of these information measures for 4 simulated samples, one for each combination of parameters.

Since σ is fixed, as μ increases one sees an increase in the sample mean and decrease in the percent of zero counts. We see in Table 1 as μ increases, the samples become more and more informative about μ and σ because the posterior standard deviations of both μ and σ decrease.

MCMC simulations using WinBugs 1.4 are used for Bayesian statistical inference for model (6) for the samples considered. To compare the ESS values one needs to assess their variability due to ACF estimation. For each simulated data set corresponding to a set of parameters we ran 10 independent MCMC simulations and estimated ESS for each run. Table 2 presents the average and the standard errors of estimated ESS for these samples.

In Table 2 as we move from less informative samples to more informative samples, ESS increases dramatically, suggesting that improving mixing is especially important when the individual observations contain little information, as in the ICR data.

Table 1: **Measures of information in the samples**

	$\mu = -2$	$\mu = -1$	$\mu = 0$	$\mu = 1$
Sample mean	0.22	0.48	1.58	3.90
% zero counts	84	67	35	18
SD(μ)	0.43	0.21	0.11	0.08
SD(σ)	0.47	0.24	0.11	0.08
E(μ)	−1.92	−1.07	0.03	0.97
E(σ)	0.66	0.80	0.91	0.91

Notes: Total number of MCMC simulations K=10,000. Entries are the values for three independent samples. SD is the posterior standard deviation and E is the posterior mean.

Table 2: **ESS for posterior simulations of length 10,000**

	$\mu = -2$	$\mu = -1$	$\mu = 0$	$\mu = 1$
$\overline{\text{ESS}}_\mu$	110	221	1569	3140
$\overline{\text{ESS}}_\sigma$	61	179	1391	2229
SE($\overline{\text{ESS}}_\mu$)	6.7	16.1	57.4	110.8
SE($\overline{\text{ESS}}_\sigma$)	4.6	16.0	63.3	79.2

Notes: $\overline{\text{ESS}}_\mu$ and $\overline{\text{ESS}}_\sigma$ are the averages of estimated ESS from 10 independent simulations of the posterior distribution of μ and σ respectively. SE($\overline{\text{ESS}}_\mu$) and SE($\overline{\text{ESS}}_\sigma$) are standard errors of the means $\overline{\text{ESS}}_\mu$ and $\overline{\text{ESS}}_\sigma$.

5.2 Improving Mixing Strategies

We will focus on several simple strategies of improving mixing: centering and orthogonalization of covariates, hierarchical centering of random effects and analytical integration of random effects.

Centering covariates is widely used to make them orthogonal to the intercept. A more complex method is orthogonalization of covariates. If X is the matrix of covariates then we want an invertible matrix A such that $Z = XA$ is orthogonal. Thus, we let A be the inverse of a symmetric square-root of $X'X$. If β is the coefficient

vector in the X parameterization, then $\beta^* = A^{-1}\beta$ is the coefficient vector in the Z parameterization since $X\beta = Z\beta^*$. In the MCMC iterations we use the β^* parameters, but we can also monitor the original parameters $\beta = A\beta^*$ and use them later for inference.

Hierarchical centering is a simple reparameterization technique for models with several layers of random effects. Model (4) is presented with random effects hierarchically centered. An equivalent form of that model without hierarchical centering of random effects is

$$\left.\begin{array}{lcl} Y_{ij}\,|\lambda_{ij} & \sim & \text{Poisson}(\lambda_{ij}) \\ \lambda_{ij} & = & V_{ij}\,R_{ij}\,C_{ij} \\ \log(C_{ij}) & = & X'_{ij}\beta + t_{ij} + s_j + r_{k(j)} + \mu \\ t_{ij} & \sim & N(0,\sigma_t^2) \\ s_j & \sim & N(0,\sigma_s^2) \\ r_{k(j)} & \sim & N(0,\sigma_r^2). \end{array}\right\} \tag{7}$$

In the Bayesian analysis of ANOVA models, hierarchical centering can significantly improve of the posterior correlation structure (Gilks, et al., 1996).

Table 3 presents the ESS for different combinations of reparameterizations for model (4). Because estimates of ESS have their own variability due to ACF estimation, comparing two different ESS may not be very informative when they are close. However ESS values that differ by a factor of 2 are likely to correspond to real differences in mixing rates.

A first observation is that ESS is much smaller than the MCMC sample size. No clear pattern can be observed for all parameters. However, for the average regional mean μ, Turbidity, Region 8 and Season 1 the combination HC+C+O greatly improves mixing. For each variable in Table 3 we can decide what reparameterization is more suitable.

One can use a max-min strategy that maximizes the smallest ESS over all parameters of interest. In our application the reparameterizations maximizing the smallest ESS are HC+C and HC+C+O. Because the first reparameterization is simpler it was chosen for future analysis.

Consider the following two equivalent models

$$\left.\begin{array}{lcl} Y_{ij}\,|\lambda_{ij} & \sim & \text{Poisson}(\lambda_{ij}) \\ \lambda_{ij} & = & V_{ij}\,R_{ij}\,C_{ij} \\ C_{ij} & = & \exp(X'_{ij}\beta + s_j)\,g_{ij} \\ g_{ij}\,|\alpha & \sim & \text{Gamma}(\alpha,\alpha) \\ s_j\,|r_{k(j)},\,\sigma_s & \sim & N(r_{k(j)},\sigma_s^2) \\ r_k\,|\mu,\,\sigma_r & \sim & N(\mu,\sigma_r^2). \end{array}\right\} \tag{8}$$

Table 3: **Estimated ESS for different model parameterizations**

Variable	Standard	HC	$HC + C$	$C + O$	$HC + C + O$
μ	15	30	200	30	120
Turbidity	85	45	230	560	225
Reservoir	125	65	110	190	110
Region 8	15	75	1160	30	890
Season 1	130	115	290	875	815
σ_r	95	1130	1160	195	1200
σ_s	205	90	110	310	100
σ_t	95	90	55	110	85

Notes: Total number of MCMC simulations K=10,000. HC=Hierarchical centering, C=Centering, O=Orthogonalization

and

$$\left.\begin{array}{lcl} Y_{ij}\,|\lambda_{ij} & \sim & \text{Negative Binomial}\left(\frac{\alpha}{\alpha+\lambda_{ij}}, \alpha\right) \\ \lambda_{ij} & = & V_{ij}\, R_{ij}\, \exp(X'_{ij}\beta + s_j) \\ R_{ij}\,|a,\, b & \sim & \text{Beta}(a, b) \\ s_j\,|r_{k(j)},\, \sigma_s & \sim & N(r_{k(j)}, \sigma_s^2) \\ r_k\,|\mu,\, \sigma_r & \sim & N(\mu, \sigma_r^2). \end{array}\right\} \tag{9}$$

Analytical integration of random effects reduces the dimension of the posterior distribution and can improve ESS. For examples and details see Crainiceanu et al., 2002.

6 Script language in WinBUGS

The scripting language is an alternative to the menu/dialog box interface of WinBUGS. This language can be useful for automating routine analysis. The language works by writing values into fields and clicking on buttons.

A minimum of four files are required:

a. The script file itself. This file has to be in WinBUGS format (.odc)

b. A file containing the BUGS language representation of the model (.txt)

c. A file (or several) containing the data (.txt)

d. A file for each chain containing initial values (.txt).

The shortcut BackBUGS has been set up to run the commands contained in the file script.odc (in the root directory of WinBUGS) when it is double-clicked. A

WinBUGS session may be embedded within any software component that can execute the BackBUGS shortcut.

A list of currently implemented commands in the scripting language can be found in the BUGS manual.

We provide an example of script file that works

```
display('log')                          #Choose display type
check('Test1/pathogen.txt')             #Model check
data('Test1/pathogen_dat.txt')          #Load data
compile(1)                              #Compile model
inits(1, 'Test1/pathogen_in.txt')       #Initialize parameters
gen.inits()                             #Generate initial values
update(4000)                            #4000 burn-in iterations
set(b[])                                #Set b to be monitored
set(mu)
set(taut)
set(taus)
set(taur)
update(10000)                           #Do 10000 iterations
stats(*)                                #Provide stats for all parameters
history(*)                              #Provide chain histories
trace(*)                                #Provide chain traces
density(*)                              #Provide kernel density estimates
autoC(*)                                #Provide autocorrelations
quantiles(*)                            #Provide running quantiles
coda(*,output)                          #Save the two files for coda
save('pathogenLog')                     #Save the output file
```

7 Homework

The ultimate goal of this homework is to construct a WinBUGS program for inference on a nonparametric Bayesian model. A few concepts needed for doing this homework are introduced below.

7.1 Penalized splines as GLMMs

Consider the partially linear model $y_i = W_i^T\gamma + m(x_i) + \epsilon_i$, where ϵ_i are i.i.d. $N(0, \sigma_\epsilon^2)$, ϵ_i is independent of W_i and x_i, W_i is a vector of covariates that enter the model linearly, x_i is another covariate, and $m(\cdot)$ is a smooth function.

We model $m(\cdot)$ using the class of spline functions

$$m(x, \theta) = \beta_0 + \beta_1 x + \ldots + \beta_p x^p + \sum_{k=1}^{K} b_k (x - \kappa_k)_+^p \ ,$$

where $\theta = (\beta_0, \ldots, \beta_p, b_1, \ldots, b_K)^T$ is the vector of regression coefficients, and $\kappa_1 < \kappa_2 < \ldots < \kappa_K$ are fixed knots. Here a_+ is equal to a if $a > 0$ and zero otherwise and $a_+^p = (a_+)^p$. Ruppert, Wand and Carroll (2003) provide a comprehensive framework for semiparametric smoothing using splines, Following Ruppert (2002), we consider a number of knots that is large enough (typically 5 to 20) to ensure the desired flexibility, and κ_k is the sample quantile of x's corresponding to probability $k/(K+1)$, but our results hold for any other choice of knots. To avoid overfitting, we minimize

$$\sum_{i=1}^{n} \left\{ y_i - W_i^T \gamma - m\left(x_i, \theta\right) \right\}^2 + \frac{1}{\lambda} \theta^T D \theta \,, \tag{10}$$

where λ is the smoothing parameter and D is a known positive semi-definite penalty matrix. We focus on matrices D of the form

$$D = \begin{bmatrix} 0_{(p+1)\times(p+1)} & 0_{(p+1)\times K} \\ 0_{K\times(p+1)} & \Sigma^{-1} \end{bmatrix},$$

where Σ is a positive definite matrix and $0_{m\times l}$ is an $m \times l$ matrix of zeros. This matrix D penalizes the coefficients of the spline basis functions $(x - \kappa_k)_+^p$ only. We will use $\Sigma = I_K$.

Let $Y = (y_1, y_2, \ldots, y_n)^T$, W be the matrix with the i-th row equal to W_i^T, X be the matrix with the i-th row $X_i = (1, x_i, \ldots, x_i^p)$, and Z be the matrix with i-th row $Z_i = \left\{ (x_i - \kappa_1)_+^p, \ldots, (x_i - \kappa_K)_+^p \right\}$.

If we divide (10) by the error variance one obtains

$$\frac{1}{\sigma_\epsilon^2} \left\| Y - W\gamma - X\beta - Zb \right\|^2 + \frac{1}{\lambda \sigma_\epsilon^2} b^T \Sigma^{-1} b \,,$$

where $\beta = (\beta_0, \ldots, \beta_p)^T$ and $b = (b_1, \ldots, b_K)^T$. Define $\sigma_b^2 = \lambda \sigma_\epsilon^2$, consider the vectors γ and β as fixed-effects[1] parameters and the vector b as a set of random-effects parameters with $E(b) = 0$ and $\mathrm{cov}(b) = \sigma_b^2 \Sigma$. If $(b^T, \epsilon^T)^T$ is a normal random vector and b and ϵ are independent then one obtains an equivalent model representation of the penalized spline in the form of a linear mixed model (Brumback et al., 1999):

$$Y = W\gamma + X\beta + Zb + \epsilon, \;\; \mathrm{cov} \begin{pmatrix} b \\ \epsilon \end{pmatrix} = \begin{bmatrix} \sigma_b^2 \Sigma & 0 \\ 0 & \sigma_\epsilon^2 I_n \end{bmatrix}. \tag{11}$$

For this model $E(Y) = W\gamma + X\beta$ and $\mathrm{cov}(Y) = \sigma_\epsilon^2 V_\lambda$, where $V_\lambda = I_n + \lambda Z \Sigma Z^T$ and n is the total number of observations. In the LMM described in (11) both W and X

[1] Frequentists consider fixed-effects parameters as unknown constants but, like Bayesians, treat the random-effects as random variables. Bayesians, of course, treat all unknowns as random. However, in the typical Bayesian analysis of a mixed model, the fixed-effects parameters as well as the variance components are given vague, often improper, priors while the random effects are given somewhat informative priors.

correspond to fixed effects. For simplicity of presentation we can redefine $X = [W|X]$, $\beta = (\gamma^T, \beta^T)^T$ and let $p+1$ be the dimension of the new vector β.

Note that model (11) can easily be implemented in WinBUGS. For example the likelihood part of the model can be implemented as

```
for (i in 1:n){
       response[i]~dnorm(m[i],taueps)
       m[i]<-inprod(beta[],X[i,])+inprod(b[],Z[i,])}
```

where `X[,]` is the design matrix for fixed effects (in this case additional covariates and monomials) and `Z[,]` is the design matrix for random effects. These matrices can be obtained outside WinBUGS and then loaded as data into WinBUGS. Matrices can also be manipulated within WinBUGS. The function `inprod(,)` is the inner product of two vectors.

Just as in the case of ridge regression presented in section 4.2 the specification of the LMM is completed by describing the distribution of the b parameters

```
for (k in 1:K){b[k]~dnorm(0,taub)}
```

The model also includes vague priors on all hyperparameters

```
for (l in 1:p+1){beta[l]~dnorm(0,1.0E-6)}
taueps~dgamma(1.0E-3,1.0E-3)
taub~dgamma(1.0E-3,1.0E-3)
```

7.2 The Wage–Union Membership Data

In this subsection the actual homework problem is described. Download from the website the wageunion.txt file containing data on wages/h and union membership for 534 workers. The data were taken from the Statlib website at Carnegie Mellon University `lib.stat.cmu.edu`.

Denote by y_i the union membership of the i-th worker and by x_i his wage/h. We can use a Binomial/Logit model to link y_i and x_i as follows

$$\begin{cases} y_i \mid p_i & \sim & \text{Binomial}(p_i, 1) \\ \text{logit}(p_i) & = & m(x_i) \end{cases} \tag{12}$$

Problem 1. Write WinBUGS programs for the following cases

$$\begin{cases} m(x) & = & \beta_0 + \beta_1 x \\ m(x) & = & \beta_0 + \beta_1 x + \beta_2 x^2 \\ m(x) & = & \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 \end{cases} \tag{13}$$

a. Provide the WinBUGS program for the first case and discuss code changes for the other programs. Shorter programs are encouraged.

b. Plot the data and the posterior means for each of the three mean functions on the same graph.

c. Discuss your results

Problem 2. Suppose now that the logit of $p(x)$ is modeled nonparametrically using a linear ($p = 1$) penalized spline with $K = 20$ knots. Choose the knots at the sample quantiles of x's corresponding to probabilities $1/21, 2/21, \ldots, 20/21$. We use the following model

$$\left\{ \begin{array}{lcl} y_i|x_i & \sim & \text{Binomial}\{p(x_i), 1\} \\ \text{logit}\{p(x_i)\} & = & \beta_0 + \beta_1 x_i + \sum_{k=1}^{K} b_k (x_i - \kappa_k)_+ \\ b_k & \sim & N(0, \sigma_b^2). \end{array} \right. \tag{14}$$

a. Provide the WinBUGS program. The shorter, the better. Hint: use coding ideas presented in Section 7.1.

b. Plot the posterior means and the 95% credible intervals for the mean function.

c. Discuss your results.

d. Discuss mixing properties.

References

Birkes, D. and Dodge , Y. (1993). *Alternative Methods of Regression*. John Wiley and Sons, New York.

Brownlee, K.A. (1965). *Statistical Theory and Methodology in Science and Engineering*. John Wiley and SONS, New York.

Brumback, B., Ruppert, D., & Wand. M.P. (1999). *Comment on "Variable selection and function estimation in additive nonparametric regression using data-based prior"* by Shively, Kohn, and Wood. *J. Amer. Stat. Assoc.*, 94, 794–797.

Crainiceanu, C.M., Ruppert, D., Stedinger, J.R., and Behr, C.T. (2003). *Modeling the U.S. national distribution of waterborne pathogen concentrations with application to Cryptosporidium parvum*. Water Resources Research, 39 (9), 1235–1249.

Crainiceanu, C.M., Ruppert, D., Stedinger, J.R., and Behr, C.T. (2002). *Improving MCMC mixing for a GLMM describing pathogen concentrations in water supplies* in *Case Studies in Bayesian Statistics* vol. VI edited by C. Gatsonis et al., 207–221, Springer Verlag, New York.

Gilks, W.R. and Roberts, G.O., (1996). *Strategies for improving MCMC*. In *Markov Chain Monte Carlo in Practice* (eds W.R. Gilks, S. Richardson and D.J. Spiegelhalter). Chapman and Hall, London, 89-114.

Robert, C.P. and Casella, G. (1999). *Monte Carlo Statistical Methods* Springer, New York.

Ruppert, D. (2002). *Selecting the number of knots for penalized splines.* J. Comp. Statist. & Data Anal., 11, 735–757.

Ruppert, D., Wand, M.P., & Carroll, R.J. (2003). *Semiparametric Regression.* Cambridge, UK: Cambridge University Press.

Spiegelhalter, D., Thomas, A. and Best, N., (2000). *WinBugs Version 1.4 User Manual*.