

Using the BACC Software for Bayesian Inference

William J. McCausland (`mccaus@bayes.econ.umn.edu`) *

Department of Economics, University of Minnesota

Abstract. The BACC software provides its users with tools for Bayesian Analysis, Computation and Communications. A new version of the software, described here, implements these tools as extensions to popular mathematical applications such as Gauss or Matlab. The user has available, within the application, special purpose commands for posterior simulation and related tasks, alongside familiar built-in commands for matrix computation, graphics, program flow control, and I/O. Examples illustrate the use of the software within Gauss. Several models are currently available, including a normal linear model. Developers can extend the software by specifying new models.

1. Introduction

The BACC software provides its users with tools for Bayesian Analysis, Computation and Communications. The software supports several model specifications, which consist of statistical descriptions of the unknown (e.g. parameters) and known (e.g. data) quantities of the model, and Monte Carlo algorithms to simulate ¹ draws from the prior and posterior distributions of the unknown quantities.

The user can do several things with the BACC software.

- Create a model instance, by selecting a model specification and providing any known quantities of the model, such as data and fixed hyper-parameters.
- Simulate draws from the prior and posterior distributions of the unknown quantities.
- Estimate expectations of prior and posterior functions of interest and their standard errors.

* Support for the development of BACC was provided by NSF grant SBR-9731037 to John Geweke at the University of Minnesota. The author would like to thank Prof. Geweke for supporting this work and for helpful suggestions on this paper. Thanks also to Hülya Eraslan and John Stevens for high quality research assistance.

¹ Here and elsewhere, simulating draws of a target random vector means drawing pseudo-random vectors, the sample mean of any function of whose theoretical counterparts converges almost surely to the mean of the function of the target random vector, whenever the latter mean exists.

- Generate estimates of the marginal likelihood and estimate their standard errors.
- Determine the robustness of expectations of posterior functions of interest to changes to the prior specification.

BACC software has been available in various forms for several years. Before the current version, the software was stand-alone. It featured command line instructions, control files, and standard file structures. The current version implements the same tools, and some new ones, as extensions to popular mathematical applications such as Gauss. The BACC software is now a dynamically linked library, which can be loaded into the application while it is running. Loading it enables special purpose BACC commands. As with the application's built-in commands, the user issues the new commands at the application's command prompt.

The new version has two main advantages. First, the user no longer needs to learn how to use specialized non-commercial software. To learn the new commands, the user consults a manual, describing the new commands in the same way that the manual of the application itself describes its built-in commands. Second, the new BACC commands can be used alongside built-in commands for matrix computation, graphics, program control and I/O. Examples in this paper demonstrate the utility of this second advantage.

Since the tools are still implemented in a compiled language, these advantages come at no cost in computational speed. Posterior simulation is still many times faster than it is with code written in an application's interpreted language.

The paper is organized as follows. Section 2 describes in more detail what a model specification is, gives a detailed description of one of the currently available model specifications, and mentions the others. Section 3 describes two different economic models for aggregate production, and illustrates the features of the software by demonstrating various inferential procedures. Section 4 discusses how model developers can extend the software by adding new model specifications.

2. Model Specifications Supported

The BACC software currently supports five model specifications. We describe here in detail one of them, a normal linear model with Gibbs posterior simulation. Examples throughout the paper demonstrate the BACC system using this particular model specification.

Each model specification includes a prior distribution and a data distribution. The prior distribution is the marginal distribution of a model's unobserved quantities. The data density is the conditional distribution of observed endogenous quantities given unknown quantities and observed exogenous quantities. Inference using a model is supported by two Monte Carlo algorithms. One simulates from the prior distribution; the other, from the posterior distribution, the conditional distribution of unobserved quantities given observed quantities.

2.1. A NORMAL LINEAR MODEL

The normal linear model described here is a variant of the Seemingly Unrelated Regressions (SUR) model. It allows cross-equation covariate coefficient equality restrictions. Special cases include the familiar single equation regression model, and the SUR model (block diagonal covariate matrix, with T row blocks).

The number of observations of each variable is T . There are m vectors of endogenous observed variables: y_1, \dots, y_m . Each vector is $T \times 1$. Also, there are m corresponding matrices of exogenous observed variables: Z_1, \dots, Z_m . Each matrix is $T \times k$. The unobserved quantities are a $k \times 1$ covariate coefficient parameter β , and an $m \times m$ precision (inverse of variance) parameter H . The data distribution is described by

$$y = Z\beta + \epsilon$$

$$\epsilon|Z \sim N(0, H^{-1} \otimes I_T)$$

where

$$y \equiv \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} \quad Z \equiv \begin{bmatrix} Z_1 \\ \vdots \\ Z_m \end{bmatrix} \quad \epsilon \equiv \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_m \end{bmatrix}$$

In the prior distribution of β and H , the two parameters are independent. The marginal distribution of β is normal, with mean $\underline{\beta}$ and precision \underline{H} . The marginal distribution of H is Wishart, with mean $\underline{\nu}\underline{S}^{-1}$ and degrees of freedom $\underline{\nu}$.

$$\beta \sim N(\underline{\beta}, \underline{H}_\beta^{-1}) \quad H \sim W(\underline{S}^{-1}, \underline{\nu})$$

The Monte Carlo algorithm for the prior generates independent draws from the prior distribution. The algorithm for the posterior is a two-block Gibbs sampler, based on the following conditional posterior distributions.

$$\beta|H, y, Z \sim N(\bar{\beta}, \bar{H}_\beta^{-1}) \quad H|\beta, y, Z \sim W(\bar{S}^{-1}, \bar{\nu})$$

Table I. Other BACC Models.

Data Distribution	Prior Distribution
Poisson	gamma
uniform	Pareto
i.i.d. finite state	Dirichlet
non-stationary first-order-Markov finite-state	Dirichlet

where

$$\bar{H}_\beta \equiv \underline{H}_\beta + Z'(H \otimes I_T)Z$$

$$\bar{\beta} \equiv \bar{H}_\beta^{-1}[\underline{H}_\beta \underline{\beta} + Z'(H \otimes I_T)y]$$

$$\bar{S} \equiv \underline{S} + [s_{ij}] \quad s_{ij} \equiv u'_i u_i \equiv (y_i - Z_i \beta)'(y_i - Z_i \beta)$$

$$\bar{\nu} \equiv \underline{\nu} + T$$

2.2. OTHER MODELS

The four other currently available model specifications feature simple data distributions with fully conjugate priors, analytic expressions for the normalized posterior distribution, and independence Monte Carlo algorithms for both prior and posterior distributions. The data distributions and the conjugate priors are given in Table I.

3. Features of the BACC System

Use of the BACC software is illustrated using its Gauss version. Special Gauss commands implementing the BACC system appear in capital letters, to distinguish them from built-in Gauss commands. Gauss command names are case-insensitive.

3.1. TWO INSTANCES OF THE NORMAL LINEAR MODEL

We consider two particular instances of the normal linear model. The observed data, taken from Berndt and Wood (1975) are 25 observations (1947 through 1971) of aggregate production data. Table II lists the

Table II. Names of Berndt and Wood Variables.

Variable	Description	Filename
Y	Quantity of Value Added Input	QV
p	Price Index of Value Added Input	PV
K	Quantity of Capital Services	QK
r	Rental Price Index for Capital Services	PK
L	Quantity of Labor Input	QL
w	Price Index for Labor Input	PL

variable names, their descriptions, and the filenames in which the data appear.

The first model instance is based on the Constant Elasticity of Substitution (CES) production function:

$$Y = A \left[\delta K^{(\sigma-1)/\sigma} + (1 - \delta) L^{(\sigma-1)/\sigma} \right]^{\sigma/(\sigma-1)}$$

Assuming zero profits and competitive markets, with prices p , w , and r prevailing for Y , L , and K , the factor demand equations are

$$\log(Y/K) = \beta_1 + \beta_3 \log(r/p)$$

$$\log(Y/L) = \beta_2 + \beta_3 \log(w/p)$$

where

$$\beta_1 \equiv (1 - \sigma) \log A - \sigma \log \delta$$

$$\beta_2 \equiv (1 - \sigma) \log A - \sigma \log(1 - \delta)$$

$$\beta_3 \equiv \sigma$$

Adding a residual $\epsilon_{i,t}$ for both equations i and each observation t yields the CES model instance:

$$\begin{bmatrix} \log(Y_1/K_1) \\ \vdots \\ \log(Y_{25}/K_{25}) \\ \log(Y_1/L_1) \\ \vdots \\ \log(Y_{25}/L_{25}) \end{bmatrix} = \begin{bmatrix} 1 & 0 & \log(r_1/p_1) \\ \vdots & \vdots & \vdots \\ 1 & 0 & \log(r_{25}/p_{25}) \\ 0 & 1 & \log(w_1/p_1) \\ \vdots & \vdots & \vdots \\ 0 & 1 & \log(w_{25}/p_{25}) \end{bmatrix} \cdot \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} \epsilon_{1,1} \\ \vdots \\ \epsilon_{1,25} \\ \epsilon_{2,1} \\ \vdots \\ \epsilon_{2,25} \end{bmatrix}$$

The second model instance is based on the Generalized Leontieff (GL) cost function:

$$C = Y [d_{KK}r + d_{KL}\sqrt{rw} + d_{LL}w + d_{LK}\sqrt{wr}]$$

where $d_{LK} = d_{KL}$.

This cost function yields the following factor demand equations for K and L .

$$K/Y = \beta_1 + \beta_3\sqrt{w/r}$$

$$L/Y = \beta_2 + \beta_3\sqrt{r/w}$$

where $\beta_1 = d_{KK}$, $\beta_2 = d_{LL}$, and $\beta_3 = d_{LK} = d_{KL}$.

Adding a residual $\epsilon_{i,t}$ for both equations i and each observation t yields the GL model instance:

$$\begin{bmatrix} K_1/Y_1 \\ \vdots \\ K_{25}/Y_{25} \\ L_1/Y_1 \\ \vdots \\ L_{25}/Y_{25} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \sqrt{w_1/r_1} \\ \vdots & \vdots & \vdots \\ 1 & 0 & \sqrt{w_{25}/r_{25}} \\ 0 & 1 & \sqrt{w_1/p_1} \\ \vdots & \vdots & \vdots \\ 0 & 1 & \sqrt{w_{25}/p_{25}} \end{bmatrix} \cdot \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} \epsilon_{1,1} \\ \vdots \\ \epsilon_{1,25} \\ \epsilon_{2,1} \\ \vdots \\ \epsilon_{2,25} \end{bmatrix}$$

In both CES and GL model instances,

$$\begin{bmatrix} \epsilon_{1t} \\ \epsilon_{2t} \end{bmatrix} | r_t, w_t, p_t \sim \text{i.i.d. } N(0, H^{-1})$$

3.2. INITIALIZING THE SOFTWARE

The following Gauss code loads the BACC library and the Gauss graphics library, and initializes the BACC system.

```
/* Load BACC and Gauss graphics library. */
library
  /NFS/icarus/d3/home/bacc/BACC/Gauss/libBACC,
  pgraph;

/* Initialize BACC software. */
INITBACC;
```

3.3. CREATING INSTANCES OF MODELS

The user creates an instance of the normal linear model by specifying the known quantities $\underline{\beta}$, \underline{H}_β , $\underline{\nu}$, \underline{S} , y , and Z . For the CES instance, these quantities are stored in the Gauss matrices `BETA_ces`, `H_ces`, `NU_ces`, `S_ces`, `y_ces`, and `Z_ces`.

The following code creates the CES and GL model instances. The BACC command `MINST` stands for Model INSTance.

```

/* Load data and construct data matrices for
   both model instances */
load Y[] = "QV"; load K[] = "QK"; load L[] = "QL";
load p[] = "PV"; load r[] = "PK"; load w[] = "PL";
one = ones(25,1); zero = zeros(25,1)};
y_ces = log(Y./K) | log(Y./L);
Z_ces = (one ~ zero ~ log(r./p)) |
        (zero ~ one ~ log(w./p));
y_gl = K./Y | L./Y;
Z_gl = (one ~ zero ~ sqrt(w./r)) |
        (zero ~ one ~ sqrt(r./w));

/* Create prior parameter matrices for both
   model instances */
BETA_ces = {0.5, -0.5, 1.0};
Sigma_ces =
  {1.0225 0.9775 0, 0.9775 1.0225 0, 0 0 0.25};
H_ces = inv( Sigma_ces );
S_ces = {0.0001 0, 0 0.0001};
NU_ces = 3;

BETA_gl = {2.0, 2.0, 0.5};
Sigma_gl = {4.0 0 0, 0 4.0 0, 0 0 0.25};
H_gl = inv( Sigma_gl );
S_gl = {0.0001 0.0, 0.0 0.0001};
NU_gl = 3;

/* Create both model instances */
knowns_ces =
  {"BETA_ces" "H_ces" "NU_ces" "S_ces" "Z_ces" "y_ces"};
knowns_gl =
  {"BETA_gl" "H_gl" "NU_gl" "S_gl" "Z_gl" "y_gl"};
unknowns = {"beta" "H"};
MINST("ces", "nlm", unknowns, knowns_ces );

```

```
MINST("gl", "nlm", unknowns, knowns_gl );
```

3.4. SIMULATING FROM THE PRIOR AND POSTERIOR

The following code simulates 1000 draws each from the prior and posterior joint distributions of β and H for the CES model instance. First, it simulates an initial 100 draws from the posterior. It plots two time series (fig. 1) of evaluations of the log joint density of $\beta, H, y|X$, to help assess convergence. The first covers all draws, including the outlying first one, drawn from the prior. The second time series omits the first draw, so as to display the others at an appropriate scale. Then it filters out the first 10 posterior draws and simulates another 910. Then it simulates 1000 draws from the prior.

```
/* Simulate 100 posterior samples. */
POSTSIM( "ces", 100, 1 );

/* Get log prior and log likelihood evaluations. */
logPrior_ces = GETLOGPRIOR( "ces" );
logLike_ces = GETLOGLIKE( "ces" );

/* Calculate log joint evaluations. */
logJoint_ces = logPrior_ces + logLike_ces;
/* Plot two time series of log joint evaluations. */
window(1,2,0);
setwind(1);
xy( seqa(1,1,100), logJoint_ces );
nextwind;
xy( seqa(2,1,99), logJoint_ces[2:100] );
endwind;

/* Keep only posterior samples 11 through 100. */
POSTFILTER( "ces", seqa(11,1,90) );
/* Simulate 910 more. */
POSTSIM( "ces", 910, 1 );
/* Simulate 1000 prior samples. */
PRIORSIM( "ces", 1000,1 );
```

Figure 1. Evaluations of the Log Joint Density at Initial Samples

3.5. ' PLOTTING A HISTOGRAM FOR A POSTERIOR QUANTITY

The following code extracts vectors of posterior draws from internal BACC data structures. It plots a histogram (fig. 2) for the posterior distribution of the elasticity of substitution $\sigma = \beta_3$.

```

/* Get prior and posterior simulated draws of
   each element of beta. */
beta1p_ces = GETUNKNOWNVECTOR("ces","betaprior",1);
beta2p_ces = GETUNKNOWNVECTOR("ces","betaprior",2);
beta3p_ces = GETUNKNOWNVECTOR("ces","betaprior",3);
beta1_ces = GETUNKNOWNVECTOR("ces","beta",1);
beta2_ces = GETUNKNOWNVECTOR("ces","beta",2);
beta3_ces = GETUNKNOWNVECTOR("ces","beta",3);

/* Plot posterior histogram of beta3, the
   elasticity of substitution. */
histp(beta3_ces,25);

```

3.6. USING KERNEL SMOOTHING TO ESTIMATE DENSITIES

The following code uses kernel smoothing to estimate the prior and posterior densities of the elasticity of substitution $\sigma = \beta_3$. It plots (fig. 3) the two densities on the same graph, to allow a visual comparison of the two densities. The SMOOTH command performs the kernel smoothing.

```

/* Arrange SMOOTH to estimate both densities only
   on the interval [0.5,1.5]. */
krange = "absolute"; range_a1 = 0.5; range_a2 = 1.5;

```

Figure 2. Posterior Histogram of the Elasticity of Substitution σ

```
’, ’
```

Figure 3. Estimated Prior and Posterior Densities for σ

```
’, ’
```

```
/* Estimate both prior and posterior densities */
logWeight = zeros(1000,1);
{x,yprior} = SMOOTH( logWeight, beta3p_ces );
{x,ypost} = SMOOTH( logWeight, beta3_ces );

/* Plot prior and posterior densities. */
xy( x, ypost ~ yprior );
```

3.7. USING SCATTERPLOTS TO VISUALIZE BIVARIATE DISTRIBUTIONS

The following code creates a scatterplot (fig. 4) of the joint posterior distribution of the capital weight δ and the elasticity of substitution σ .

Figure 4. Scatterplot of σ versus δ

```

'/* Calculate delta and sigma */
delta = 1 / (exp(( beta1_ces - beta2_ces)./beta3_ces)+1);
sigma = beta3_ces;

/* Plot scatter-plot of (delta,sigma) pairs. */
_plctrl = {-1};
xy(delta,sigma);

```

3.8. INVESTIGATING INEQUALITY RESTRICTIONS

Suppose we are interested in a model restricting the elasticity of substitution σ to be less than unity. The following code calculates the Bayes factor in favor of a model in which the prior for β is truncated to the region $\{\beta \in \mathbb{R}^3 : \beta_3 < 1\}$ against the unconstrained model. The Bayes factor in favor of the restricted model is 2.0. It also calculates the posterior mean and standard deviation of β_3 ; first conditional on the restricted model, and then conditional on the unrestricted model. In both cases, the mean is 0.911, the standard deviation is 0.029, and the standard error of the mean estimate is 0.0011. The `EXPECT1` command calculates all of these values.

```

/* Functions of interest are indicator functions. */
Iprior = (beta3p_ces .< 1);
Ipost = (beta3_ces .< 1);

/* Calculate prior and posterior probabilities of
beta3 < 1, and bayes factor. */
{priorP,std,nse,rne} = EXPECT1(logWeight,Iprior);
{postP,std,nse,rne} = EXPECT1(logWeight,Ipost);

```

```

Bayesfactor = postP/priorP;

/* Reweight samples according to inequality
   restriction beta3 < 1, and calculate mean,
   standard deviation, and numerical standard
   errors. */
logRweight = logWeight + (.not Ipost) * -300;
{restrMean,std,nse,rne} =
  EXPECT1(logRweight,beta3_ces);

/* In unrestricted model, calculate mean, standard
   deviation and numerical standard errors. */
{unrestrMean,std,nse,rne} =
  EXPECT1(logWeight,beta3_ces);

```

3.9. COMPARING THE CES AND GL AGGREGATE PRODUCTION MODELS

The following code first simulates 1010 posterior draws for the GL model instance and filters out the first 10. Then it calculates estimates of the marginal likelihood of both the CES and GL models. Then it computes the Bayes factor and posterior odds ratio for the CES model against the GL model, assuming that both models are equally likely *a-priori*, or equivalently, that the prior odds ratio is unity. Then it calculates estimates of the posterior mean and standard deviation of the elasticity of substitution between capital and labor at the geometric mean of w_t/r_t , conditional on the compound model in which the CES and GL models are equally probable *a-priori*. The result is a posterior probability weighted average of the posterior mean conditional on the CES model and the posterior mean conditional on the GL model. The computed results are summarized in Table III.

```

/* Simulate 1010 posterior samples for the GL model
   instance and delete the first 10. */
POSTSIM( "gl", 1010, 1 );
POSTFILTER( "gl", seqa(11,1,1000) );

/* Get posterior samples of each element of beta. */
beta1_gl = GETUNKNOWNVECTOR( "gl", "beta", 1 );
beta2_gl = GETUNKNOWNVECTOR( "gl", "beta", 2 );
beta3_gl = GETUNKNOWNVECTOR( "gl", "beta", 3 );

/* Calculate the log marginal likelihoods and

```

Table III. Comparing the CES and GL Production Models.

Quantity	CES	GL	Compound
Prior probability	0.5	0.5	1.0
Posterior probability	0.003	0.997	1.0
Elasticity of Substitution:			
mean	0.911	0.519	0.520
std	0.029	0.130	0.132

```

log Jacobian. */
p=0.5;
{logML_ces,MLNSE_ces } = MLIKE( "ces", p );
{logML_gl,MLNSE_gl } = MLIKE( "gl", p );
logJacobian = sumc(y_ces);

/* Calculate the Bayes factor in favor of
the CES over the GL model, and the posterior
model probabilities. */
BayesFactor =
exp( logML_ces - logML_gl - logJacobian );
PriorOddsRatio = 1;
PosteriorOddsRatio = PriorOddsRatio * BayesFactor;
PosteriorProb_ces = PosteriorOddsRatio /
(1+PosteriorOddsRatio);
PosteriorProb_gl = 1 - PosteriorProb_ces;

/* Calculate the weighted posterior mean and standard
deviation of the elasticity of substitution at the
geometric mean of the wage to rental rate ratio. */
elastSubst_ces = beta3_ces;
w_over_r = exp( meanc( log(w./r) ) );
r_over_w = 1/w_over_r;
elastSubst_gl =
0.5 * beta3_gl * sqrt( w_over_r ) ./
(beta1_gl + beta3_gl * sqrt(w_over_r))
+ 0.5 * beta3_gl * sqrt( r_over_w ) ./
(beta2_gl + beta3_gl * sqrt(r_over_w));

{mean_ces,std_ces,nse,rne} =
EXPECT1(logWeight,elastSubst_ces);
{mean_gl,std_gl,nse,rne} =

```

```

    EXPECT1(logWeight, elastSubst_gl);
    mean_comp = PosteriorProb_ces * mean_ces
                + PosteriorProb_gl * mean_gl;
    std_comp =
    sqrt( PosteriorProb_ces * meanc(elastSubst_ces.^2) +
          PosteriorProb_gl * meanc(elastSubst_gl.^2) -

```

4. Developing Model Specifications

Currently, the task of developing a model specification involves writing code in C defining the prior and data distributions and the Monte Carlo algorithms to simulate from the prior and posterior distributions. Subsection 4.1 describes this task in more detail. Work is in progress that will enable a new and easier way to create model specifications. This is the subject of subsection 4.2.

4.1. CURRENT METHOD FOR DEVELOPING MODEL SPECIFICATIONS

Currently, a model developer specifies a model by writing and providing routines to perform model-specific computations. The routines fall under four categories: unknown quantity simulation routines, density evaluation routines, auxiliary quantity computation routines and miscellaneous routines.

The simulation routines implement Monte Carlo algorithms for simulating from the prior and posterior distributions. For both algorithms, the developer supplies routines to draw from an initial distribution, to draw from candidate proposal kernels and to evaluate the corresponding Hastings ratios. There may be one proposal kernel and a degenerate (indentially equal to unity) Hastings ratio (independence Monte Carlo), one proposal kernel and a non-degenerate Hastings ratio (Hastings–Metropolis algorithm), several proposal kernels with degenerate Hastings ratios (Gibbs algorithm) or several proposal kernels with at least one non-degenerate Hastings ratio (Hastings–Metropolis–Green algorithm, also known as Metropolis within Gibbs).

The density evaluation routines evaluate the prior and the likelihood at given values of the unknown quantities. The availability of prior and likelihood values for each simulated posterior draw allows marginal likelihood estimation. It also allows the user to compare an alternate model with a base model and do inference conditional on the alternate model, using only the simulated posterior draws from the base model.

The model developer may also supply routines to calculate auxiliary quantities, those that are the result of intermediate calculations. Auxiliary quantities may be sufficient statistics, which need to be calculated only once, or convenient intermediate quantities that facilitate calculations. An example of the latter is the vector of residuals $u \equiv y - Z\beta$ in the normal linear model. This quantity is used both in the evaluation of the likelihood and in the simulation of H from its conditional (on β) posterior distribution. Providing a routine to calculate u makes its value available for both uses, and ensures that it is only calculated once for each new draw of β .

Miscellaneous routines are used for such things as checking user input for correctness and performing parameter transformations.

The BACC software includes core routines that perform tasks that are common to all model specifications. The core routines co-ordinate the simulation of unknown quantities and the evaluation of the prior and likelihood. It updates auxiliary quantities when, and only when necessary, and maintains prior and posterior simulation matrices.

The model developer also has available a library of matrix routines. These evaluate multivariate densities, draw multivariate random variables, and perform routine matrix computations.

4.2. A PROPOSED METHOD FOR DEVELOPING MODEL SPECIFICATIONS

The current BACC software promotes an appropriate division of labor. The model developer provides only the code specific to the model under development, and takes advantage of code already available for model-independent tasks and matrix computation. However, the burden on the developer to program the required routines is still formidable.

Under development is a program that converts a much simpler representation of a model specification into the required code. The simpler representation of the normal linear model specification is shown below as an example.

It starts with a list of tokens to represent unknown and known quantities. It then gives symbolic expressions for evaluating the prior and likelihood. After this, it provides routines to simulate prior and posterior draws. In this example, the same method is used to generate the initial prior draw, subsequent prior draws, and the initial posterior draw. All these are drawn from the prior distribution of β and H . Two posterior candidate proposal kernels are given for posterior simulation. No Hastings ratios are provided, so they are assumed to be unity. That is, the MCMC algorithm to draw posterior samples is a two-block Gibbs sampler.

A list of auxiliary quantity definitions follows. Each consists of a symbolic expression describing how the auxiliary quantity is calculated.

Knowns: beta_, H_, nu_, S_, Z, y

Unknowns: beta, H

Prior: $N(\text{beta}|\text{beta}_, \text{H}_) * \text{Wi}(\text{H}|\text{S_inv}, \text{nu}_)$

Likelihood: $N_SSE(\text{S}|\text{T}, \text{H})$

Prior Initial:

Prior Kernel:

Posterior Initial:

beta ~ $N(\text{beta}_, \text{H}_)$;

H ~ $\text{Wi}(\text{inv}(\text{S}_), \text{nu}_)$

Posterior Kernel:

beta ~ $N(\text{beta}_{_}, \text{H}_{_})$;

H ~ $\text{Wi}(\text{inv}(\text{S}_{_}), \text{nu}_{_})$

Updates:

T = rows(Y)

M_ZZ = autoMomentMatrix(Z,T)

M_ZY = crossMomentMatrix(Z,Y,T)

H__ = H_ + blockTrace(H,M_ZZ)

H__beta__ = H_ * beta_ + blockTrace(H,M_ZY)

beta__ = solve(H__, H__beta__)

u = y - Z * beta

S = autoMomentMatrix(u,T)

S__ = S_ + S

nu__ = nu_ + T

5. Conclusion

We have described an innovation in software for Bayesian Analysis, Computation and Communications that allows its use within standard mathematical software packages. We have demonstrated the use of the software using an economic example.

The example illustrates the straightforward use of Bayesian software to do inferential tasks whose non-Bayesian counterparts are much more difficult. We have compared non-nested models, estimated the posterior mean and standard deviation of a non-linear function of the parameters of a model, compared a base model with an alternate model

with inequality restrictions imposed, and done inference conditional on this alternate model. Finally, we have estimated the posterior mean and standard deviation of a function of interest that takes into account our uncertainty about two competing models but avoids the pre-testing problem.

The software, manuals, and other materials are available at the web site <http://www.econ.umn.edu/~bacc/>.

References

- Berndt, E.R.: 1991, *The Practice of Econometrics: Classic and Contemporary*. Addison-Wesley, Reading, MA, USA, 1994.
- Berndt, E.R. and Wood, D.O.: 1975, 'Technology, Prices and the Derived Demand for Energy', *Review of Economics and Statistics* **Vol. 57:3**, pp.259–268
- Green, P.J.: 1995, 'Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination.' *Biometrika* **82**, pp.711–732
- Hastings, W.K.: 1970, 'Monte Carlo Sampling Methods Using Markov Chains and Their Applications.' *Biometrika* **57**, pp.97–109
- Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. and Teller, E.: 1953, 'Equation of State Calculations by Fast Computing Machines.' *J. Chem. Phys.* **21**, pp.1087–1092
- Zellner, A.: 1962, 'An Efficient Method of Estimating Seemingly Unrelated Regressions and Tests of Aggregation Bias.' *Journal of the American Statistical Association* **57**, pp.500–509

