Computational Economics **23:** 201–218, 2004.

Using the BACC Software for Bayesian Inference

WILLIAM J. McCAUSLAND
Département de sciences économiques, Université de Montréal, CIREQ and CIRANO, CP 6128, Succursale Centre-ville Montréal, QC H3C-3J7 Canada;
E-mail: william.j.mccausland@umontreal.ca

(Accepted 29 January 2003)

Abstract. The BACC software provides its users with tools for Bayesian Analysis, Computation and Communications. The current version of the software, described here, implements these tools as extensions to popular mathematical applications such as Matlab, S-PLUS, R, and Gauss, running under Windows, Linux or Unix. From the user's perspective, there is a seamless integration of special-purpose BACC commands for posterior simulation and related tasks with powerful built-in commands for matrix computation, graphics, program flow control, and I/O. Examples demonstrate the use of the software within Matlab. Nineteen models are currently available, and many others are planned. BACC is designed to be extendible, not only by the developers of BACC, but also by others who wish to implement their own models and thus make them available to BACC users. While model development requires programming in C, several design features of BACC facilitate this development. BACC is freely available at http://www2.cirano.qc.ca/~bacc/.

Key words: Bayesian inference, econometric software, seemingly unrelated regressions

1. Introduction

The BACC software provides its users with tools for Bayesian Analysis, Computation and Communications. The current version of the software, described here, implements these tools as extensions to popular mathematical applications such as Matlab, S-PLUS, R, and Gauss, running under Windows, Linux or Unix. The choice of application and operating system, and the fact that R (http://www.r-project.org/) and BACC are both free, make BACC widely accessible.

A central concept behind the BACC software is that of a *model*, a full statistical description of unknown (e.g., parameters) and known (e.g., data) quantities.

A *model specification* consists of a model and Monte Carlo samplers to simulate[1] draws from the prior and posterior distributions of the unknown quantities. The implementation (in the compiled programming language C) of a model specification is invisible to the user, who uses generic commands `priorsim` and `postsim` for prior and posterior simulations. BACC currently implements nineteen different model specifications.

A *model instance* consists of a model, and values of the known quantities of that model, supplied by the user.

Several BACC commands are for working with models and model instances. With them, the user can do the following tasks.

- Create a model instance, by selecting one of the supported models and providing any known quantities of the model, such as data and fixed hyperparameters. Section 3.2 gives examples.
- Simulate draws from the prior and posterior distributions of the unknown quantities. Section 3.3 gives examples.
- Access prior and posterior draws, and evaluations of the log prior density and log likelihood at all posterior draws. Sections 3.3 through 3.9 give examples, and demonstrate their application for Bayesian analysis and communications.
- Generate estimates of the marginal likelihood and estimate their numerical standard errors. The marginal likelihood is a value summarizing the out-of-sample prediction record of a model for a particular data set. Sections 3.8 and 3.9 demonstrate the use of the marginal likelihood for Bayesian model comparison and Bayesian model averaging.

Other BACC commands play supporting roles. The following tasks do not involve model instances, and are not even exclusively Bayesian, but they are very useful tools complementing the previous commands, and are difficult to find in mathematical software applications.

- Estimate population means and their standard errors, for samples that may be weighted and may feature serial dependance. Since posterior draws are typically autocorrelated, and since results for alternative priors can be obtained by reweighting posterior draws, this is very useful in the Bayesian context. Section 3.7 gives an example.
- Estimate densities using a kernel smoother, for samples that may be weighted. Section 3.5 gives an example.

These and many other commands not demonstrated here are documented in the user manuals, available at the BACC website.

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 BACC software is now a dynamically linked library, which can be loaded into the application while it is running, thereby enabling special purpose BACC commands. As with the application's built-in commands, the user issues the new commands at the application's command prompt.

Most of the BACC code, written in the compiled programming language C, is independent of the target mathematical application and operating system. Code specific to a mathematical application is kept to a minimum, and serves to format data according to the conventions of the application. This design principle has allowed the BACC developers to implement versions of BACC for several mathematical applications, running under different operating systems. BACC for

the Ox programming language, described in Doornik (1999), may be available in the future.

The current version has two main advantages. First, the user no longer needs to learn how to use special-purpose software. To learn the new commands, the user consults a manual, describing the new commands in the same way that the manual of the appliction 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 low-level language, these advantages come at no cost in computational speed. Posterior simulation is often many times faster than it is with code written in an application's interpreted language.

The set of models currently available to the user is limited, and some users want to become model developers and implement their own models for BACC. In doing so, developers take advantage of BACC design features facilitating model development. Once they have implemented a new model, developers can apply their models as BACC users, and can make their models available to other BACC users.

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 the BACC software from a user's perspective. It demonstrates the implementation of several Bayesian inferential procedures. Section 4 describes the BACC software from a model developer's perspective. It discusses the development of new model specifications for BACC.

2. Model Specifications

The BACC software currently supports nineteen model specifications. We describe here in detail one of them, a normal linear model with independent Monte Carlo simulation from the prior distribution, and posterior simulation using Gibbs sampling. 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 draws of unobserved quantities 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 ($m = 1$), 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: $y_1, \ldots, y_m$. Each vector is $T \times 1$. Also, there are m corresponding matrices of exogenous observed variables: $Z_1, \ldots, Z_m$. 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

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

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

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 $\underline{\nu}$.

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

The Monte Carlo sampler for the prior generates independent draws from the prior distribution. The posterior sampler is an example of a Gibbs sampler. It generates draws of unobserved quantities that are first-order Markov, rather than independent. The transition from state (β, H) to the next state (β', H') is governed by the following law. The value β' is drawn from the distribution $\beta | H, y, Z$, and the value H' is drawn from $H | \beta', y, Z$. The sampler is made relatively easy by the fact that these conditional posterior distributions are familiar distributions, easy to draw from:

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

where

$$\begin{aligned}
&\overline{H}_\beta \equiv \underline{H}_\beta + Z'(H \otimes I_T)Z, \\
&\overline{\beta} \equiv \overline{H}_\beta^{-1}[\underline{H}_\beta \underline{\beta} + Z'(H \otimes I_T)y], \\
&\overline{S} \equiv \underline{S} + S, \quad S_{ij} \equiv u_i' u_i \equiv (y_i - Z_i\beta)'(y_i - Z_i\beta), \\
&\overline{\nu} \equiv \underline{\nu} + T.
\end{aligned}$$

Table I. Names of Berndt and Wood variables.

Variable	Description
Y	Quantity of value added input
p	Price index of value added input
K	Quantity of capital services
r	Rental price index for capital services
L	Quantity of labor input
w	Price index for labor input

2.2. OTHER MODELS

At the time of writing, nineteen other models are available to the user. Eleven are variants of the univariate linear regression model. Residuals may be normal, finite mixtures of normals, or Student t. Dependent variables may be directly observed, or limited in any one of three different ways, including by discretization (e.g., Probit) or by censoring (e.g., Tobit).

There are two univariate regression models with autoregressive residuals. One is a base model, and the second features hidden finite Markov states and state dependent means, as in Hamilton (1989).

There are three finite state models: one where states are i.i.d., one with states that are stationary first-order Markov, and one with states that are non-stationary first-order Markov, where the distribution of the initial state need not be the invariant distribution. The remaining models are the Poisson-gamma model, and the uniform-Pareto model. Plans are to expand the number of available models. A priority is time series models.

3. BACC from the User's Perspective

To get started, the user simply downloads the BACC software and documentation and follows the installation instructions.

Use of the BACC software is illustrated using its Matlab version. Special BACC commands are distinguished from built-in Matlab commands by the comment '`% BACC`'.

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 I lists the variable names, and their descriptions.

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

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

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,

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

3.2. CREATING INSTANCES OF MODELS

The user creates an instance of the normal linear model by specifying the known quantities $\underline{\beta}$, $\underline{H}_\beta$, $\underline{\nu}$, $\underline{S}$, y, and Z. For the CES instance, these quantities are stored in the Matlab matrices `betaUnderbar_ces`, `HUnderbar_ces`, `nuUnderbar_ces`, `SUnderbar_ces`, `y_ces`, and `Z_ces`.

The following code creates the CES and GL model instances. The BACC command `minst` stands for Model INSTance, and it is used to pass the information required to construct the CES and GL model instances. BACC creates an internal representation of the model instance, and returns an identifier (e.g., `mi_ces` and `mi_gl` below) to the user.

```
% Load raw data on prices and quantities.

Y = load('QV'); p = load('PV');
K = load('QK'); r = load('PK');
L = load('QL'); w = load('PL');

% Prepare data matrices.

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 log(r./w)];

% Prepare prior parameters.

betaUnderbar_ces = [0.3; -0.3; 1.0];
HUnderbar_ces = inv([1.0 0.6 0; 0.6 1.0 0; 0 0 0.25]);
SUnderbar_ces = diag([0.0001,0.0001]);
nuUnderbar_ces = 3;

betaUnderbar_gl = [2.0; 2.0; 0.5];
HUnderbar_gl = inv(diag([4.0,4.0,0.25]));
```

```
SUnderbar_gl = diag([0.0001,0.0001]);
nuUnderbar_gl = 3;

% Create model instances for CES and GL cases.

mi_ces = minst('nlm','beta','H', ...
  betaUnderbar_ces, HUnderbar_ces, ...
  nuUnderbar_ces, SUnderbar_ces, Z_ces, y_ces);    % BACC
mi_gl = minst('nlm','beta','H', ...
  betaUnderbar_gl, HUnderbar_gl, ...
  nuUnderbar_gl, SUnderbar_gl, Z_gl,y_gl);         % BACC
```

3.3. SIMULATING FROM THE PRIOR AND POSTERIOR

The following code simulates 10000 draws from the joint prior distribution of β and H for the CES model instance, using the BAC command `priorsim`. Then it simulates 10010 draws from their joint posterior distribution using `postsim`. The `priorsim` and `postsim` commands call build in simulation routines for the appropriate model, according to the model instance identifier (e.g., `mi_ces` below) provided by the user. The first 10 draws are deleted using the `postfilter` command, since these draws may be overly sensitive to the initial distribution of the Markov Chain used for posterior simulation. The `extract` command returns simulation matrices and related information, stored in internal data structures, to the Matlab structure `sim_ces`. Two of the fields of this structure are `logPrior` and `logLike`, vectors of length 10000 of log prior and log likelihood evaluations for the 10000 posterior draws. The vector of evaluations of the log joint distribution is their sum, and the code plots a time series of these values, to assess convergence of the Markov chain to the posterior distribution.

The plot (Figure 1) shows little evidence of sensitivity to the initial distribution.

```
% Simulate 10000 prior samples.
priorsim( mi_ces, 10000,1 );         % BACC

% Simulate 10010 posterior draws.
postsim( mi_ces, 10010, 1 );         % BACC

% Keep only posterior samples 11 through 10010.
postfilter( mi_ces, 11:10010 );      % BACC

% Extract simulation matrices.
sim_ces = extract( mi_ces );         % BACC
```

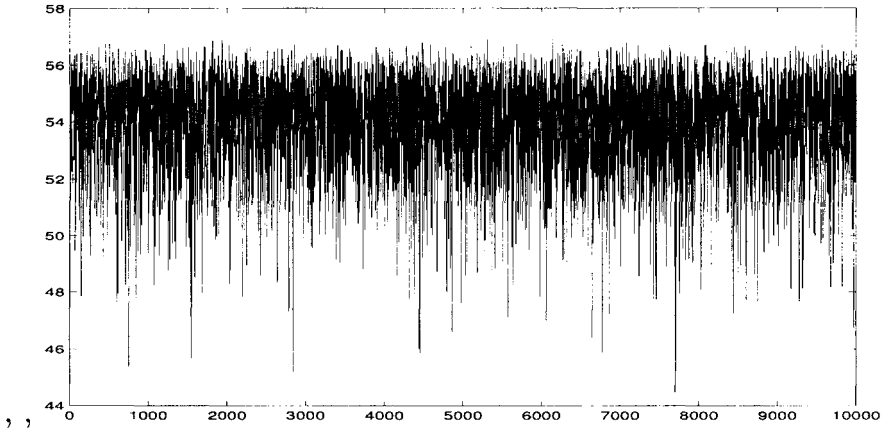

Figure 1. Evaluations of the log joint density at the 10000 posterior draws.

```
% Calculate log joint evaluations.
logJoint_ces = sim_ces.logPrior + sim_ces.logLike;

% Plot time series of log joint evaluations.
plot( logJoint_ces );
```

3.4. PLOTTING A HISTOGRAM FOR A POSTERIOR QUANTITY

The following code creates vectors of prior and posterior draws for each element of the unknown vector β. The fields `betaPrior` and `beta` of the structure `sim_ces` are three dimensional arrays, containing the prior and posterior simulation matrices for β. The first two dimensions give the row and column of β. The third dimension is the simulation dimension. Each value of the third index gives a different draw of β. The built-in Matlab command `squeeze` is used to create vectors of length 10000 from $1 \times 1 \times 10000$ arrays. Each vector is representative of either the prior or posterior distributions of one of the elements of β. The vector `b3_ces`, for example, is the vector of the 10000 posterior draws of β_3, the elasticity of substitution. The Matlab command `hist` plots a histogram (Figure 2) for the posterior distribution of this quantity.

```
% Create vectors of prior and posterior draws for
% each element of beta.

b1_cesPrior = squeeze(sim_ces.betaPrior(1,1,:));
b2_cesPrior = squeeze(sim_ces.betaPrior(2,1,:));
b3_cesPrior = squeeze(sim_ces.betaPrior(3,1,:));
b1_ces = squeeze(sim_ces.beta(1,1,:));
```

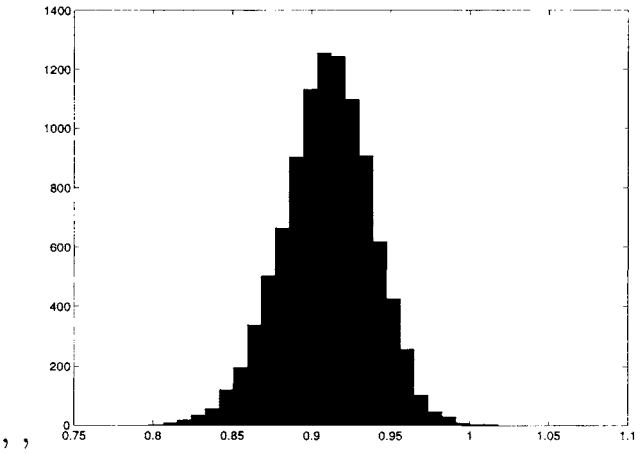

Figure 2. Posterior histogram of the elasticity of substitution σ.

```
b2_ces = squeeze(sim_ces.beta(2,1,:));
b3_ces = squeeze(sim_ces.beta(3,1,:));

% Plot histogram of the elasticity of substitution.
hist(b3_ces,25);
```

3.5. USING KERNEL SMOOTHING TO ESTIMATE DENSITIES

The following code uses kernel smoothing to estimate the prior and posterior densities of the elasticity of substituting $\sigma = \beta_3$. The `weightedSmooth` command performs the kernel smoothing. Unlike most kernel smoothers available in mathematical software, it handles weighted samples, which is useful in a Bayesian context.

The code plots (Figure 3) the two densities on the same graph, to allow a visual comparison of the two densities. The BACC command `weightedSmooth` performs the kernel smoothing.

```
% Estimate and plot prior and posterior densities
% on the interval [0.0,2.0].

% Specify common domain for the plots.

KRANGE = 'absolute';
RANGE_A1 = 0.0; RANGE_A2 = 2.0;

% Estimate prior and posterior densities using
% the BACC kernel smoother weightedSmooth.
```

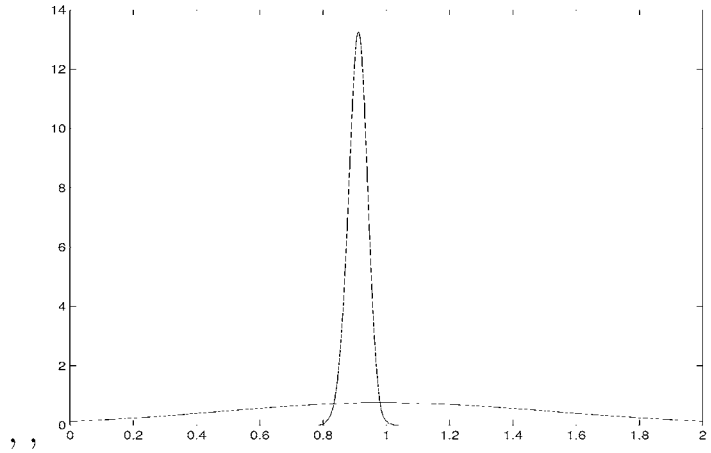

Figure 3. Estimated prior and posterior densities for σ.

```
logWeights = zeros(10000,1);
[x yprior] = ...
  weightedSmooth( logWeights, b3_cesPrior);  % BACC

logWeightPost = sim_ces.logWeightPrior;
[x ypost] = ...
  weightedSmooth( logWeights, b3_ces);       % BACC

% Plot both on same graph.

plot(x,[yprior,ypost]);
```

3.6. USING SCATTERPLOTS TO VISUALIZE BIVARIATE DISTRIBUTIONS

The following code creates scatterplots (Figure 4) of the joint prior and posterior distributions of the capital weight δ and the elasticity of substitution σ.

```
% Display bivariate distribution of delta and sigma.

% Calculate vectors of prior draws of delta and sigma.

deltaPrior = 1 ./ (1+exp( (b1_cesPrior - b2_cesPrior) ...
                  ./ b3_cesPrior));
sigmaPrior = b3_cesPrior;
```

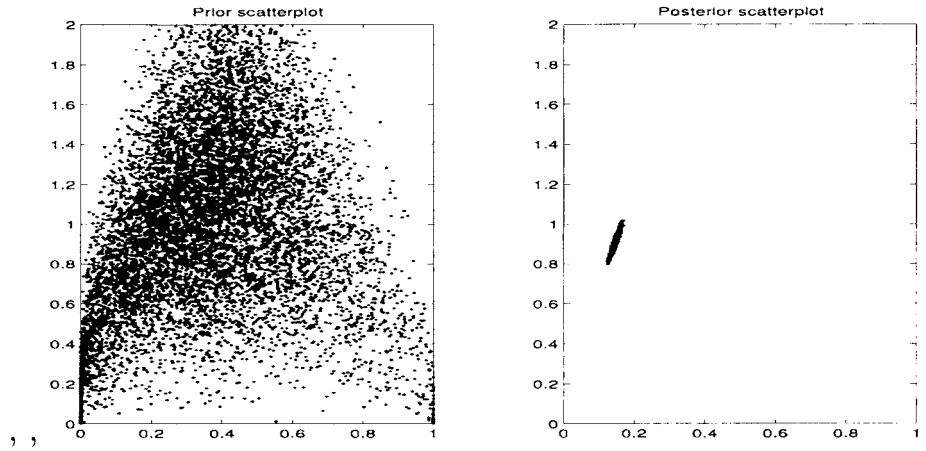

Figure 4. Prior and posterior scatterplots of σ versus δ.

```
% Calculate posterior samples of delta and sigma.

delta = 1 ./ (1+exp( (b1_ces - b2_ces) ./ b3_ces));
sigma = b3_ces;

% Display two scatterplots of (delta, sigma) pairs.

subplot(1,2,1);
plot(deltaPrior,sigmaPrior,'.')
axis([0.0 1.0 0.0 2.0]);
title('Prior scatterplot');

subplot(1,2,2);
plot(delta,sigma,'.');
axis([0.0 1.0 0.0 2.0]);
title('Posterior scatterplot');
```

3.7. 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 very close to 2.0, since the prior probability of the restriction is 0.5 and the posterior probability is very close to 1. 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 estimates of the mean are 0.910, the

estimates of the standard deviation are 0.028, and the standard error of the mean estimate is 0.0003. The BACC command `expect1` calculates all of these values. It uses time series methods for calculating standard errors for the mean of a possibly weighted sample.

```
% Functions of interest are indicator functions.
Iprior = double(b3_cesPrior < 1);
Ipost = double(b3_ces < 1);

% Calculate prior and posterior probabilities of
% beta3 < 1, and bayes factor using BACC command
% expect1.

logWeights = zeros(10000,1);
[priorP,stddev,nse,rne] = ...
 expect1(logWeights, Iprior)         % BACC
[postP,stddev,nse,rne] = ...
 expect1(logWeights, Ipost)          % BACC
BayesFactor = postP/priorP

% Reweight samples according to inequality
% restriction beta3 < 1, and calculate mean,
% standard deviation, numerical standard errors,
% and relative numerical efficiency.

logReweights = logWeights;
logReweights(~Ipost) = -500;
[restrMean,stddev,nse,rne] = ...
 expect1(logReweights, b3_ces)        % BACC

% In unrestricted model, calculate mean, standard
% deviation and numerical standard errors.

[unrestrMean,stddev,nse,rne] = ...
 expect1(logWeights, b3_ces)          % BACC
```

3.8. COMPARING THE CES AND GL AGGREGATE PRODUCTION MODELS

The following code first simulates 10010 posterior draws for the GL model instance using the BACC command `postsim` and filters out the first 10, using the BACC command `postfilter`. Then it calculates estimates of the marginal likelihood of both the CES and GL models, using the BACC command `mlike`.

The marginal likelihood is a value summarizing the out-of-sample prediction record of a model for a particular data set. It is useful for Bayesian model comparison and Bayesian model averaging. Both uses are demonstrated below.

The code then computes the Bayes factor, which is the ratio of the two marginal likelihoods, and the posterior odds ratio (ratio of posterior model probabilities) in favor of the CES model against the GL model, assuming that both models are equally likely a priori. This posterior odds ratio is simply the product of the prior odds ratio, equal to 1 here, and the Bayes factor.

The `mlike` command uses the algorithm described in Gelfand and Dey (1994).

```
% Simulate 10010 posterior samples for the GL model
% instance, delete the first 10, and create vectors
% of posterior draws for the elements of beta.

postsim( mi_gl, 10010, 1 );                % BACC
postfilter( mi_gl, 11:10010 );             % BACC
sim_gl = extract( mi_gl );                 % BACC
b1_gl = squeeze(sim_gl.beta(1,1,:));
b2_gl = squeeze(sim_gl.beta(2,1,:));
b3_gl = squeeze(sim_gl.beta(3,1,:));

% Calculate the log marginal likelihoods and
% log Jacobian.

p=0.5;
[logML_ces,MLNSE_ces] = mlike( mi_ces, p );  % BACC
[logML_gl,MLNSE_gl] = mlike( mi_gl, p );     % BACC
logJacobian = sum(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;
```

3.9. MODEL AVERAGING

The following code calculates estimates of the posterior mean and standard deviation of the elasticity of substitution between labor and capital at the geometric mean of the ratio w_t/r_t of their prices, 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 II. The very low weight on the GL model is due to an incompatibility of data information and prior information: the OLS estimate of β_3 in that model has the 'wrong' sign.

```
% 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 = b3_ces;
w_over_r = exp( mean( log(w./r) ) );
r_over_w = 1./w_over_r;
elastSubst_gl = ...
    0.5 * b3_gl * sqrt( w_over_r ) ./ ...
          (b1_gl + b3_gl * sqrt(w_over_r)) ...
  + 0.5 * b3_gl * sqrt( r_over_w ) ./ ...
          (b2_gl + b3_gl * sqrt(r_over_w));

mean_std_ces = [mean(elastSubst_ces), std(elastSubst_ces)]
mean_std_gl = [mean(elastSubst_gl), std(elastSubst_gl)]

mean_std_comp = [...
 posteriorProb_ces * mean_std_ces(1) + ...
 posteriorProb_gl * mean_std_gl(1), ...
 sqrt( ...
  posteriorProb_ces * mean(elastSubst_ces.^2) + ...
  posteriorProb_gl * mean(elastSubst_gl.^2) - ...
  (posteriorProb_ces * mean_std_ces(1) + ...
   posteriorProb_gl * mean_std_gl(1)).^2 )]
```

4. BACC from the Model Developer's Perspective

Users have a limited number of model specifications from which to choose. BACC is designed to be extendible, however, and not only by the developers of BACC, but also by others who wish to implement their own models.

Table II. Comparing the CES and GL production models.

Quantity	CES	GL	Compound
Prior probability	0.5	0.5	1
Posterior probability	1.000	1.3e–19	1
Elasticity of substitution:			
mean	0.910	0.446	0.910
std	0.028	0.087	0.028

There are two obvious advantages to developing a new model in the BACC framework. First, the developer can apply the model as a user, and take advantage of the convenience of working within a general purpose mathematical application. Second, the developer can make the model available to other BACC users.

There are also other advantages to developing models for BACC. Three important features of BACC facilitate the development of new models:

1. Much of the overhead that is common to all simulation routines has already been written by the BACC developers.
2. The structure of BACC allows model developers to take advantage of the work of other BACC model developers.
3. There is a library of useful matrix routines for developers.

4.1. DEVELOPING MODEL SPECIFICATIONS

A model developer specifies a model by writing code for model-specific computations. These computations include simulation routines, density evaluation routines, and routines to update auxiliary quantities. Core BACC routines perform tasks shared by all simulators: co-ordinating tasks, record keeping, and maintaining simulation matrices. The organization of the BACC software permits this division of labor, and facilitates code reuse in the development of models that have elements in common with models that others have already developed.

The simulation routines implement Monte Crlo samplers for simulating from the prior and posterior distributions. For each, the developer writes code to draw from an initial distribution, to draw from various candidate proposal distributions, and to evaluate Hastings ratios for each proposal distribution. The Hastings ratio is a quantity that determines the probability with which a candidate draw is accepted or rejected. The developer can implement several widely used types of Monte Carlo samplers:

– Independence Monte Carlo: one proposal distribution, and a degenerate (indentically equal to unity) Hastings ratio.

- Metropolis-Hastings: one proposal distribution and a non-degenerate Hastings ratio.
- Gibbs: several proposal distributions with degenerate Hastings ratios.
- Metropolis-Hastings-Green: several proposal distributions with at least one non-degenerate Hastings ratio.

The density evaluation routines evaluate the prior and the likelihood at given values of the unknown quantities. The availability of prior and likelihood values at all posterior draws facilitates marginal likelihood estimation, and Bayesian communication.

The model developer may also supply routines to calculate auxiliary quantities, the results of intermediate calculations. An example is the vector of residuals $u \equiv y - Z\beta$ in the normal linear model, a quantity used (indirectly) both in the evaluation of the likelihood and in the simulation of H from its conditional (on β) posterior distribution. The model developer simply provides a routine to calculate u as a function of y, Z and β, and then uses the variable u in other routines, such as the routine to calculate the matrix S (see Section 2.1) of inner products of residual vectors, another auxiliary quantity. S is used in turn in the routines to evaluate the log likelihood function and to draw H from its conditional posterior distribution. The core BACC code maintains variables for auxiliary quantities such as u and S, and updates them when and only when necessary. To do this, it keeps track of relative update times and dependencies between variables (e.g., u depends on Z and β) and relative update times, (e.g., β has changed since the last update of u) in the same way that the Unix `make` utility does.

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, the evaluation of the prior and likelihood, and the efficient updating of auxiliary quantities. It also performs various important record keeping tasks and maintains prior and posterior simulation matrices.

The BACC code also includes a library of matrix routines for evaluating univariate and multivariate densities, drawing univariate and multivariate random variables, and performing routine matrix computations. Developers can use these routines as subroutines for the routines that they provide.

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 and avoids the pre-testing problem.

The software, manuals, and other materials are freely available at the web site http://www2.cirano.qc.ca/~ bacc/.

Acknowledgements

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, and to anonymous referees for helpful comments.

Note

[1] Here and elsewhere, simulating draws of a target random vector from a distribution means generating a pseudo-random finite subsequence of vectors, for an ergodic Markov random process whose stationary distribution is the distribution in question. Whenever the mean of a function of the target random vector exists, the sample mean of the function converges almost surely to it.

References

Berndt, E.R. (1991). *The Practice of Econometrics: Classic and Contemporary.* Addison-Wesley, Reading, MA, U.S.A.

Berndt, E.R. and Wood, D.O. (1975). Technology, prices and the derived demand for energy. *Review of Economics and Statistics*, **57**(3), 259–268.

Doornik, J.A. (1999). *Object-Oriented Matrix Programming Using Ox*, 3rd edn. Timberlake Consultant Press, London and www.nuff.ox.ac.uk/Users/Doornik, Oxford.

Gelfand, A.E. and Dey, D.K. (1994). Bayesian model choice: Asymptotics and exact calculations. *Journal of the Royal Statistical Society B*, **56**, 501–514.

Green, P.J. (1995). Reversible jump Markov Chain Monte Carlo computation and Bayesian model determination. *Biometrika*, **82**, 711–732.

Hamilton, J.D. (1989). A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica*, **57**, 357–384.

Hastings, W.K. (1970). Monte Carlo sampling methods using Markov Chains and their applications. *Biometrika*, **57**, 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**, 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**, 500–509.