ParMA: parallelised Bayesian Model Averaging for Generalised Linear Models

Riccardo (Jack) Lucchetti* and Luca Pedini*

*Università Politecnica delle Marche

version 0.91

Abstract

The package `ParMA` provides Bayesian Model Averaging (BMA) in Generalised Linear Models (GLMs). Automatic parallelisation via MPI is supported.

1 Introduction

The central idea in Bayesian Model Averaging (BMA), is to dispense entirely with the concept of model selection, and focus instead on weighted averaging on the whole space of all possible model specifications of the quantities of interest (coefficients, forecasts etc.).

The basic idea, dating back to the work by Madigan and Raftery (1994), starts from the definition of a quantity of interest β.

$$\mathsf{P}(\beta|y) = \sum_{i=1}^{m} \mathsf{P}(\beta|M_i, y)\,\mathsf{P}(M_i|y) \tag{1}$$

where y represents the data, M_i the *i-th* model from a model set $\mathcal{M} = \{M_1, ..., M_m\}$, which in a Bayesian framework becomes an additional parameter; $\mathsf{P}(\beta|M_i, y)$ identifies the model specific posterior distribution for β and $\mathsf{P}(M_i|y)$ the so-called posterior model probability. Of course, posterior moments can also be defined:

$$\mathsf{E}(\beta|y) = \sum_{i=1}^{m} \mathsf{E}(\beta|y, M_i)\,\mathsf{P}(M_i|y) \tag{2}$$

$$\mathsf{V}(\beta|y) = \sum_{i=1}^{m} \left[\mathsf{V}(\beta|y, M_i) + \mathsf{E}(\beta|y, M_i)^2\right]\mathsf{P}(M_i|y) - \mathsf{E}(\beta|y)^2 \tag{3}$$

with $\mathsf{E}(\beta|M_i, y)$, $\mathsf{V}(\beta|M_i, y)$ as, respectively, the posterior expected value and variance of the parameter in the *i-th* model.

Finally, the posterior model probability is obtained via Bayes' rule as follow:

$$\mathsf{P}(M_i|y) = \frac{p(y|M_i)\mathsf{P}(M_i)}{\sum_{j=1}^{m} p(y|M_i)\mathsf{P}(M_i)} \tag{4}$$

where $p(y|M_i)$ is the marginal data density (also referred to as marginal likelihood) and $\mathsf{P}(M_i)$ the prior probability for M_i.

The technique employed in this package follows ideas by Green (1995, 2003); Lamnisos et al. (2009) and uses "Reversible Jump MCMC" (RJMCMC) scheme, i.e. a MCMC simulation in which both parameters of interest and models are sampled *jointly*, with great benefits with respect to standard MCMCs alternatives both in terms of flexibility.

The rest of the paper is organised as follows: Section 2 lays down the statistical background for GLMs and RJMCMC; Section 3 describes the Bayesian algebraic representation of models; Section 4 deals with parallelisation and convergence; in Section 5 we discuss in detail the package and its main features.

2 Statistical background

2.1 GLMs

As is well known, the GLM is a statistical framework that includes as special cases several models widely used in the statistical and econometric practice. Let $y_1, ..., y_n$ be n observations of a dependent variable from the exponential family with density function $f(y)$:

$$f(y_i) = \exp\left[\frac{y_i\theta_i - b(\theta_i)}{a_i(\phi)} + c(y_i, \phi)\right],$$

where θ_i is the "canonical" (location) parameter, ϕ is a dispersion parameter, and $a(\cdot)$, $b(\cdot)$, $c(\cdot)$ are known functions. It can be shown that $\mathsf{E}(y_i) = \frac{\partial b(\theta_i)}{\partial \theta_i}$ and $\mathsf{V}(y_i) = \frac{\partial^2 b(\theta_i)}{\partial \theta_i^2} a_i(\phi)$

It is assumed that the conditional expectation of y_i given a set of k covariates x_i, $\mathsf{E}(y_i|x_i) = \mu_i$ is a continuous transformation of a linear combination:

$$l(\mu_i) = \eta_i = x_i^T \beta \tag{5}$$

where $l(\cdot)$ is known as the *link function*.

Maximum likelihood (ML) estimation of GLMs can be carried out, in a frequentist framework, via Iterative Weighted Least Squares on the transformed variable $z_i = \eta_i + (y_i - \mu_i)\frac{\partial \eta_i}{\partial \mu_i}$, where the weights w_i are defined as:

$$w_i = \left[\frac{\partial^2 b(\eta_i)}{\partial \eta_i^2}\left(\frac{\partial \eta_i}{\partial \mu_i}\right)^2\right]^{-1}$$

In this way the ML estimator of β can be defined as:

$$\hat{\beta} = (X^T W X)^{-1} X^T W z$$

where X is the $n \times k$ matrix of covariates and W is the $n \times n$ diagonal weight matrix with elements w_i.

The above was adapted by Gamerman (1997) to a Bayesian set-up by means of a MCMC scheme: assuming to be interested in the posterior distribution of β, $f(\beta|y) \propto f(y|\beta)f(\beta)$, where $f(y|\beta)$ designates the likelihood function and $f(\beta)$ the prior, which in this case is equal to $\beta \sim N(m_0, V_0)$, then a suitable sampling scheme is the following,

1. Set as initialisation $\beta^{(0)}$;
2. At the *i-th* iteration, draw $\beta^{(i)}$ from the proposal density $q(\beta|\beta^{(i-1)}) = N(m^{(i)}, V^{(i)})$, where:

$$V^{(i)} = (V_0^{-1} + X^T W(\beta^{(i-1)})X)^{-1} \tag{6}$$
$$m^{(i)} = V^{(i)}(V_0^{-1} m_0 + X^T W(\beta^{(i-1)}) z(\beta^{(i-1)})); \tag{7}$$

where $z(\beta^{(i-1)})$ and $W(\beta^{(i-1)})$ defines respectively the transformed variable z and the weight matrix W computes with $\beta^{(i-1)}$;

3. Accept the new draw with probability $\alpha(\beta^{(i-1)}, \beta^{(i)})$, defined via a standard Metropolis-Hastings scheme as:

$$\alpha(\beta^{(i-1)}, \beta^{(i)}) = \min\left[\frac{f(\beta^{(i)}|y)q(\beta^{(i-1)}|\beta^{(i)})}{f(\beta^{(i-1)}|y)q(\beta^{(i)}|\beta^{(i-1)})}; 1\right]$$

where $q(\beta^{(i)}|\beta^{(i-1)})$ is a normal density evaluated at $\beta^{(i)}$ with mean and variance, respectively, computed with eqs. (6) and (7).

2.2 The RJMCMC framework

One of main advantages of RJMCMC is that it makes it possible to sample a vector of parameters β, whose size can be different from one iteration to the next: this feature is especially valuable in a context like ours, where for example the Markov Chain may jump from a fairly general model with many covariates to a restricted one with few, or vice versa. Consider two MCMC iterations, i and j: the indices i and j implicitly refer to the corresponding model specification, so β_i and β_j should be understood as shorthand for β_{M_i} and β_{M_j}, respectively. Therefore, the dimensions of the two vectors may be different.

The sampling is accomplished by introducing a differentiable function $(\beta_j, u_j) = g(\beta_i, u_i)$ which maps the current β_i, of dimension k_i, into a different space of dimension k_j, which corresponds to β_j.

In Green (2003); Hastie and Green (2012) and, especially in Lamnisos et al. (2009, 2013) we find a particularly suitable function $g(\cdot)$ for GLMs[1]: assume that the parameter β_i has posterior mean μ_i and variance V_i, then the transformation function from (β_i, M_i) to $(\beta_j, M_j) = g(\beta_i, M_i)$ could be

$$\beta_j = g(\beta_i, M_i, u_i) = \mu_j + B_j \upsilon \tag{8}$$

where B is the Cholesky decomposition of the correspondent covariance matrix, μ_j and V_j the posterior mean and variance of β_j and υ is defined as:

$$\upsilon = \begin{cases} [RB_i^{-1}(\beta_i - \mu_i)]^{k_j} & \text{if } k_j < k_i \\ RB_i^{-1}(\beta_i - \mu_i) & \text{if } k_j = k_i \\ R\begin{pmatrix} B_i^{-1}(\beta_i - \mu_i) \\ u \end{pmatrix} & \text{if } k_j > k_i \end{cases}$$

with k as the number of variables, R a random permutation matrix; the notation $[...]^{k_j}$ indicates the first k_j elements of the vector and finally u, a $k_j - k_i$ vector of random numbers with density $f(\cdot)$, in general a standard Normal or a Student t.

The probability of accepting the move is:

$$\rho = \min\left[\frac{P(\beta_j, M_j|y)q(M_i|M_j)}{P(\beta_i, M_i|y)q(M_j|M_i)} \cdot \frac{|B_j|}{|B_i|} \cdot G; 1\right] \tag{9}$$

with $q(M_j|M_i)$ as the model transitional kernel, where we implicitly assume independence from the sampling of β, and:

$$G = \begin{cases} f(u) & \text{if } k_j < k_i \\ 1 & \text{if } k_j = k_i \\ f(u)^{-1} & \text{if } k_j > k_i \end{cases}$$

If the kernel is independent from the parameters, the model movements are determined independently from those of the parameters. Therefore, we can select the new model M_j first, and its corresponding parameter vector β_j next, via the function $g(\cdot)$. The permutation matrix R makes the movements to lower dimensional models stochastic and actually plays no role in the acceptance ratio.

[1] Another interesting approach for only binary data is provided by Holmes and Held (2006).

2.3 Prior choices and other technicalities

The definition of priors proposed here follows common practice in the BMA literature:

$$\beta_i | M_i \sim N(\mu_{0,i}, V_{0,i})$$

that is the prior of β on model M_i, with respectively, prior mean $\mu_{0,i}$ and prior variance $V_{0,i}$.

Some clarifications, however, are needed: in general, the parameter for the constant term has a separated (improper) prior distribution, following the argument put forward by Fernandez et al. (2001) for linear models. The constant is to be included always, and in practice this is accompanied by centring all other regressors, so as to make them orthogonal to the constant. In linear models, if α denotes the intercept parameter, we assume $\mathsf{P}(\alpha) \propto 1$.

The same argument, however, cannot be fully applied to other GLMs due to the non-linearity of the link function. A solution is proposed in Lamnisos et al. (2009), who follows the suggestion by Brown et al. (1998); Sha et al. (2004) of a standard Normal distribution with large variance of the following form:

$$\alpha \sim N(0, h) \rightarrow \begin{pmatrix} \alpha \\ \beta_i \end{pmatrix} \sim N\left(\begin{bmatrix} 0 \\ \mu_{0,i} \end{bmatrix}, \begin{bmatrix} h & \mathbf{0}^T \\ \mathbf{0} & V_{0,i} \end{bmatrix} \right) \tag{10}$$

0where h is set to a large number[2] (Lamnisos et al., 2009) and $\mathbf{0}$ is suitably sized vector of zeros. Using this second possibility implicitly assumes that the constant is always present in every specification, and all the other regressors have to be demeaned as in the original framework.

Several alternatives exist for the prior covariance matrix $V_{0,i}$: two common ones are the (a) ridge prior cI, with $c > 0$ or (b) the *Zellner-g* prior, i.e. $g(X_i^T X_i)^{-1}$ with $g > 0$. The ridge prior does not allow for prior correlation among regressors as the Zellner-g prior does, and tends to produce a more evident shrinkage effect, i.e. more parsimonious models are preferred, even though it is heavily affected by the measurement scale of the variables; for this reason when such prior is used the *a priori* standardisation of the regressors is almost mandatory. Lamnisos et al. (2009, 2013) provide examples of Bayesian model selection procedure with Probit models using ridge priors.

As for the Zellner-g alternative, a well-known modification for non-linear GLMs is $gn(X_i^T X_i)^{-1}$, where n is the number of observations: this reflects more directly how the covariance should be derived from the Unit Information Prior covariance matrix by Kass and Wasserman (1995), which is generally the most common choice.

For the model prior, we use the Binomial distribution:

$$\mathsf{P}(M_i) = \prod_{j=1}^{k} \pi_j^{\delta_{ij}} (1 - \pi_j)^{1 - \delta_{ij}} \tag{11}$$

where k is the total number of covariates considered, $0 \leq \pi_j \leq 1$ is the prior probability that the *j-th* variable is significant and δ_{ij} is an indicator of the variable inclusion[3].

2.4 The RJMCMC sampler "in a nutshell"

The basic MCMC scheme is summarised in Lamnisos et al. (2013):

1. Set the initial β_i, relative to model M_i (normally, the full specification);
2. Propose a new model M_j from a transitional kernel $q(M_j | M_i)$ and compute its β_j as in (8);

[2] Usually, $h = 100$.

[3] The choice $\pi_j = 0.5$ leads to the uniform distribution; in the limiting case $\pi_j = 1$ variable j is always included in every model.

3. Accept the move with probability (9), otherwise stay in (β_i, M_i);

4. Repeat from 2 till convergence.

The above scheme, however, may be modified to deal with a potential problem that arises when dominant specifications appear: in this case, the probability of jumping from model M_i to a different model M_j may be small. In this case, it is advisable to re-sample the parameter vector β anyway, to avoid undesirable consequences for the posterior distribution.

Therefore, a resampling step (the so-called *within move*) is introduced when a new couple (β_j, M_j) is rejected; in this case, a new β_i, which corresponds exactly to an iteration of Gamerman's MCMC, is sampled. The corresponding μ_i and V_i for the new sampled parameter may be updated with eqs. (6) and (7) obtained in the resampling step.

In short:

1. Set the initial β_i related to the model M_i, in general the full specification;

2. Propose a new model M_j from a transitional kernel $q(M_j|M_i)$ and compute its β_j as in (8);

3. Accept the move with probability (9), otherwise *propose a resampling of β_i in M_i following a single iteration of Gamerman procedure.*

4. Repeat from 2, till convergence.

3 Algebraic representation of models

The common Bayesian analysis of a variable selection scenario considers a model M_i as an additional parameter of interest; clearly an algebraic representation for such an object is called for.

A straightforward representation uses binary vectors: given k potential regressors, a specific model is a $k \times 1$ vector, where each element corresponds to one regressor, and is 1 when that variable is included in the model and 0 otherwise. Clearly, each model can also be represented by an integer, by taking each entry of that vector as a binary digit. For example, in a model with 4 potential covariates, x_1, x_2, x_3, x_4, the full model is,

$$M_i = \{x_1, x_2, x_3, x_4\} \rightarrow \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \rightarrow 15 \text{ (hex 0f)}$$

whereas a different model M_j, where x_2 is omitted would be

$$M_j = \{x_1, x_3, x_4\} \rightarrow \begin{bmatrix} 1 & 0 & 1 & 1 \end{bmatrix} \rightarrow 11 \text{ (hex 0b)}$$

Storing information for each model, such as the posterior mean μ and covariance matrix V, could in principle be accomplished by defining an array of suitable memory structures, indexed by model id. This, however, creates a problem when the set of possible covariates exceeds 32, since the number of possible models exceeds 2^{32} and handling structures of that size becomes technically problematic.

In `ParMA`, we circumvented the issue by exploiting the fact that, although the model space can be potentially very large, only a small subset is going to be actually visited by the MCMC iterations, and we only need to store it as an element of an associative array (known in gretl as a "bundle"), using the hexadecimal representation of the model id as the key.

The hexadecimal representation is preferred to the decimal representation, because gretl lacks an integer type, and therefore when the number of models is very large, numerical accuracy can be an issue. Moreover, storing models in a bundle has the advantage that once the information on a model is stored, this does not need to be recomputed each time the related model appears, leading to considerable time saving. This method rests on the

possibility of storing the bundle in RAM, but this should not be a problem since in BMA applications as long as the number of regressors is the one commonly found in real-world applications.

4 Parallelisation in MCMCs

Apparently, there seems to be little room for parallelisation in MCMCs, where the Markov property is used to set up a sequential process. In fact, parallelisation is possible, but special attention is required: splitting a MCMC across several cores could lead to failure if convergence to the stationary state is not reached by each single MCMC thread and the burn-in time required is large if compared to the total amount of iterations (Amdahl, 1967; Rosenthal, 2000). A plausible guideline is provided by Gelman and Rubin (1992); Brooks and Gelman (1998), who introduce some indices to monitor the convergence rate of multiple chains.

When these requirements are met, parallelisation can still bring about large computational efficiency gains, although the benefits in terms of CPU time may not scale linearly with the number of cores or networked computers, as in the standard independent Monte Carlo.

Notice, moreover, that splitting a MCMC in several ones in parallel can improve the exploration of the parameter space too: when the target distribution is multimodal, a single chain may get stuck in local maximum points; running the same MCMC in parallel, possibly with different starting points, may help overcome the problem.

4.1 Convergence in parallel

As already pointed out, the idea of running the same MCMC on different cores, splitting the total number of iterations and combining the single contribution as if it was the sampling result of a unique MCMC requires two main conditions, namely the convergence of the chains and a small burn-in time. In fact, these two requirements are closely linked, as a small burn-in size, in general, is appropriate where convergence is fast, so what is required is a measure of divergence *between* chains.

A rigorous solution to this problem is provided by Gelman and Rubin (1992): the authors analyse the scenario of a *univariate* parameter β simulated n times in c parallel chains (or cores). Given an unbiased estimator $\bar{\beta}$ of $\mathsf{E}(\beta)$, the between (*intra core*) variance B and the within (inside the same core) variance W defined as,

$$B = \frac{n}{c-1}\sum_{i=1}^{c}(\bar{\beta}_i - \bar{\beta})^2 \tag{12}$$

$$W = \frac{1}{c(n-1)}\sum_{i=1}^{c}\sum_{j=1}^{n}(\beta_{ji} - \bar{\beta}_i)^2 \tag{13}$$

where β_{ij} is the sampled parameter at iteration j in core i; $\bar{\beta}_i = \frac{1}{n}\sum_{j=1}^{n}\beta_{ji}$ and $\bar{\beta} = \frac{1}{c}\sum_{i=1}^{c}\bar{\beta}_i$; the Gelman-Rubin convergence measure is given by:

$$\hat{R} = \frac{\hat{V}}{W}; \tag{14}$$

where

$$\hat{V} = \frac{n-1}{n}W + \left(\frac{c+1}{c}\right)\frac{B}{n}$$

Clearly, the closer equation (14) is to 1, the more similar the chain are in terms of β, so

convergence is deemed to be achieved when $R \simeq 1$.[4]. The extension to a multivariate set-up is given in Brooks and Gelman (1998), where a generalisation of $\hat{R}$ is proposed given β as a $k \times 1$ parameter vector: define the matrices

$$\mathbf{B} = \frac{n}{c-1} \sum_{j=1}^{c} (\bar{\beta}_j - \bar{\beta})(\bar{\beta}_j - \bar{\beta})^T$$

$$\mathbf{W} = \frac{1}{c(n-1)} \sum_{j=1}^{c} \sum_{i=1}^{n} (\beta_{ij} - \bar{\beta}_j)(\beta_{ij} - \bar{\beta}_j)^T$$

as multivariate versions of (12) and (13); then the new convergence statistics is:

$$\tilde{R} = \frac{n-1}{n} + \frac{c+1}{c}\lambda \tag{15}$$

where λ is the maximum eigenvalue of $\mathbf{W}^{-1}\mathbf{B}/n$. Again, convergence is reached when (15) is close to 1; a commonly used threshold is $\tilde{R} \leq 1.2$.

As a matter of fact, virtuous practice should impose to back the Brooks and Gelman statistics up with additional diagnostics: the previous ones, as already pointed out in Brooks and Gelman (1998), consider the heterogeneity of each parallel chain as a whole, but actually initial samples may diverge quite remarkably especially if insufficient burn-in time is provided. For this purpose, the convergence should be also analysed graphically, by visualising the sequence of the sampled parameters as well as the running mean plot for both the parallel chains and the resulting single one. Moreover, Geweke (1992) and Heidelberger and Welch (1983) propose two distinct diagnostics for testing convergence: the former, is a robust univariate test on the mean difference between a parameter sample coming from the starting 0.1 replications (sub-sample A, with $n_A = 0.1n$ replications where n is the total number of iterations) of the chain and another sample given by the ending 0.5 replications (sub-sample B, with $n_B = 0.5n$ replications). For a generic parameter β, the test is simply given by

$$Z = \frac{\bar{\beta}_A - \bar{\beta}_B}{\sqrt{(S(0)_A/n_A) + (S(0)_B/n_B)}} \sim N(0,1)$$

where $\bar{\beta}$ is the sample mean, the subscripts identify the the related sub-samples, and finally $S(0)$ is the spectral density at 0 of the related sample parameters.

In the Heidelberg-Welch diagnostic, the following quantity is defined:

$$B_n = \frac{(\sum_{i=1}^{nt} \beta_i - \lfloor nt \rfloor \bar{\beta})}{\sqrt{nS(0)}}, \quad 0 \leq t \leq 1 \tag{16}$$

where again, β is a univariate parameter sampled n times and $S(0)$ the spectral density at 0. Equation 16 is asymptotically distributed as a Brownian bridge and the Cramer-von Mises statistic can be used to test the stationarity of the univariate parameter sequence. In case of rejection, the first α items in the chain are discarded and the test is re-computed. This process is iterated for $\alpha = 0.1, 0.2, ...$ until the test is accepted, or α reaches 0.5 and the test still fails. In the latter case, we can conclude that stationarity is not achieved.

5 Using the package

The `ParMA` package provides a main function, called `bma_glm`, and three auxiliary functions, `bma_printout`, `marginal_graph` and `mcmc_checks` briefly described in Section 5.3. The `bma_glm` function performs the numerical computation, and it is illustrated in the next

[4] In their paper Gelman and Rubin (1992) propose other different indices either build as modification of (14) or on different quantities such as quantiles.

section, with a special attention to the settings used to parallelise the algorithm effectively. The other functions are used for pretty-printing the main results and plotting the posterior distribution for the model parameters, and is described in Section 5.3.

Parallelisation is implemented by using the Message Passing Interface (MPI) specification via the `mpi` block construct. Users, however, should be aware that CPU time is not a straightforward function of the number of processors used. Finally, although each unit of MPI parallelisation, known as *process*, can actually employ one or more threads, in gretl each process employs a single thread of the machine as a default option and in `ParMA` this is always the case. For this reason, even if we are working with MPI processes, the term "thread" has to be intended as a synonym of process.

5.1 The main function

The public function which performs the BMA procedure is `bma_glm`, and its signature is:

```
function bundle bma_glm( series y, list X, string glm_type,
                         int ndraw, int burn, bundle params )
```

The function returns a "bundle", which is the term used in gretl for an associative array, holding the results.

The function arguments are defined as follows:

- `series y`: the dependent variable;
- `list X`: the list of covariates;
- `string glm_type`: type of model; at present, the recognised options are:
  - `"linear"` for linear models;
  - `"probit"` for binary models;
  - `"logit"` for binary models;
  - `"cloglog"` for binary models;
  - `"poisson"` for count models.
- `int ndraw`: total number of MCMC iterations;
- `int burn`: burn-in iterations (per thread);
- `bundle params`: a bundle for extra optional settings (described below). This argument can be omitted, in which case default choices will be used.

A constant term, if absent, will be automatically added to the covariates list `X`.[5] As for the number of MCMC iterations, note that the `ndraw` setting refers to the *total* number of drawings, that will be automatically split across cores. On the contrary, the burn-in parameter `burn` is kept fixed for each parallel chain. For example, if `ndraw` and `burn` were 10000 and 1000, respectively, using 4 parallel threads will cause each thread to perform 3500 Monte Carlo iterations.

The elements of the output bundle are:

- `sampled_coeff`: matrix array containing the sampled coefficients β;
- `sampled_binmodel`: matrix array holding the sampled models in binary notation;
- `sampled_var`: matrix array holding the covariance matrices for the β coefficients;
- `sampled_mean`: matrix containing the means of sampled β coefficients (one column per thread);

[5] For linear models, the intercept may be excluded by giving it a diffuse prior set-up; see Subsection 2.3.

- `sampled_pip`: matrix containing the *posterior inclusion probability* (pip) for each variable (one column per thread);
- `sampled_modelid`: matrix array containing the summary of the sampled model along with the number of times they have appeared on the related thread simulation;
- `best_models`: bundle containing the best models in terms of model posteriors;
- `GB`: matrix (vector) containing as first entry the multivariate Gelman and Brooks statistics as in (15); the remaining elements are the univariate convergence statistics (14) for each parameter;
- `opt_for_print`: bundle containing additional information to be passed into the printing function (`bma_printout`);
- `elapsed_time` is the elapsed CPU time in seconds;
- `nrep_x_thread`, `burnin`, `thinning`: scalars; the numbers of replications per threads, the burn-in iterations, the thinning interval, respectively.

All the matrix arrays in the output bundle contain as many elements as threads. Therefore, if necessary to join them up into a single matrix, the standard gretl function `flatten` function can be used.

The posterior inclusion probability is defined as the frequency of a covariate being retained through the MCMC iterations after the burn-in.

5.2 Additional options

The arguments listed in the previous subsection determine the main aspect of the RJMCMC procedure used to implement BMA. However, the behaviour of the function can be tuned more finely by passing a bundle with additional options. The keys recognised at the moment are:

- `focus`, list: a list of covariates that have to be always kept in every proposed specification; this list may be a subset of `X`, or contain extra variables. *Default*: a void list;
- `pm`, matrix: prior mean for β. In general, `pm` should be a $k \times 1$ matrix, k being the number of *total* regressors minus the constant, ordered with `focus` first and then `X`. However, a "shorthand" option is allowed: if `pm` is a 1×1 matrix, then the same prior mean will be used for for all covariates. *Default*: 0;
- `pv`, string: choice of prior covariance matrix for β. Three options are available: `ridge` for the ridge prior, `Zellner` for the Zellner prior and `custom` for a user-defined matrix. See below for the scaling factor. *Default*: `params.pv = "Zellner"`;
- `pv_scaling`, matrix: its meaning depends on the `pv` option:
  - if `params.pv = "ridge"`, then `pv_scaling` is interpreted as the scalar shrinkage coefficient c and the prior variance is cI;
  - if `params.pv = "Zellner"` then `pv_scaling` is interpreted as the scalar shrinkage coefficient g and the prior variance is $g(X_i^T X_i)^{-1}$;
  - if `params.pv = "custom"` then `custom pv_scaling` must be a $k \times k$ covariance matrix provided by the user.

  Default: the number of observations n.

- `phi`, matrix: individual prior variable inclusion probabilities π_j as per (11). In general, `phi` should be a $k \times 1$ matrix, but a "shorthand" option is allowed like for `pm` (see above). *Default*: 0.5 (uniform prior);

- `start`, matrix: the binary representation of the initial model from which the Markov chain starts. Each column should contain binary entries as explained in Section 3, and have as many rows as the number of variables that are allowed to be included/excluded during the RJMCMC (no constant and no focus variables). The matrix should have as many columns as threads. If, however, `start` is an 1×1 matrix, then the same setting is understood to be applied to the entire matrix, so if `start` equals 0 or 1, each chain will start from the null or full model, respectively. *Default*: 1.

- `kernel`, scalar: choice for the kernel (see discussion at the end of Subsection 2.2); 0 gives the simpler choice *à la* Madigan et al. (1995). 1 is used for the more sophisticated alternative used in Lamnisos et al. (2009). Default: 0.

- `change_regr`, scalar: only used if `kernel` is 1. The number of potential variable to change;

- `prob_regr`, scalar: only used if `kernel` is 1. The probability which determines how many variables to change;

- `resamp`, Boolean: enables the resampling step (within move), when a new parameter and model proposal is rejected. *Default*: `resamp = 0`;

- `center`, scalar: prior regularisation of the covariates. 0: no modification, 1: centring, 2: standardisation. *Default*: 1.

- `mpi`, integer: number of threads used to parallelise the Markov chains. *Default*: the number of available physical cores, as reported by the `ncores` key in the `$sysinfo` bundle.

- `thinning`, scalar: the so-called "thinning interval", for discarding draws during the execution of the MCMC. For example, if the thinning interval is 2, then every third draw is retained; if it's 0, all draws are kept. *Default*: 0;

- `seed`, integer: random number generator seed. *Default*: none;

- `display`, Boolean: prints directly the output via `bma_printout`. *Default*: 1.

- `threshold`, scalar: defines the number of best models in the related output bundle and it primarily applies to the printout of the procedure. Sets a threshold for printing out "significant" modes. Only models whose posterior probability exceeds `threshold` will be stored and printed. *Default*: 0.1.

5.3 Auxiliary functions

The package provides three auxiliary functions for further processing: `bma_printout` and `marginal_graph` are used for displaying the output, while `diagn_check` offers a selection of diagnostic procedures.

The `bma_printout` function takes as its only input the bundle created by the main function `bma_glm` and prints it out. Note that the printout is enabled by default when you run `bma_glm` but can be switched off by using the `display` extra option (see Section 5.2). The `bma_printout` can be useful if you decide to estimate the model quietly, store it away and print out the results at a later time.

The public function `marginal_graph` is used to plot the marginal posterior distribution of the chosen parameter β; the syntax is the following:

```
function void marginal_graph( bundle b, list X, string save )
```

where `b` is the output bundle from `bma_glm`; `X` the list of variables for which the plot is wanted; `save` is a string used for saving the plot to a file. If `save` is an empty string or is omitted, the plot will be displayed on screen; otherwise, the plot will be saved under file name specified by `save`. Note that the format will be dictated by the file extension.[6] This function computes the kernel density estimate using Gaussian kernels as well as histograms of the sampled parameter β for the variables contained in `X`: notice that the results are obtained conditioning the sample upon the inclusion of the variable in the specifications.

Finally, the function `mcmc_checks` provides additional diagnostics: the main function `bma_glm` provides directly only the Brooks and Gelman statistics for convergence, which of course are of interest in case of parallel computing, but often need to be supported by additional measures. The function signature is

```
function bundle mcmc_checks(bundle b, string what, bundle opt )
```

where, again, `b` is the output bundle from `bma_glm`; `what` indicates which diagnostic procedure is desired: at present, the available choices are `"plot", "ESS", "Geweke", "Heidelberg"`. See later in this Section for a description.

The `opt` argument may contain a bundle for additional options, such as the `opt.chain` option, i.e., a Boolean flag which takes the value 1 for computing the desired statistic on each parallel chain separately.

The `"plot"` diagnostic option produces a plot of the sequence of sampled parameters for the variables contained in `opt.plotlist`, along with their autocorrelation function plot (ACF) and their running mean plot. Note that although visual inspection can be a useful tool to detect stationarity, convergence and autocorrelation of the parameters, it should be noted that, in RJMCMC experiments of this kind, autocorrelation tends to be "artificially" high. This is due to the fact that the primary target of the sampling scheme are models, and their parameters follow in a second step; in this way it is quite common for a parameter to remain stuck on a single value for many iterations. This can happen in two cases: first, when the related variable is included in the model but the parameter is not changing because model moves are rejected (this is common when few models dominate the posterior probability); the second one, instead, when the variable is excluded and its parameter is thus set implicitly to 0. While in the first case enabling thinning or within moves is sufficient, in the second case an alternative solution may be inspecting the same plots, conditional on parameter inclusion. The available options for the `"plot"` command are:

- `opt.condition`, Boolean flag: produce a conditional plot if 1;
- `opt.lag`, lag order for ACF plot;
- `opt.display`, display plot on screen if 1;
- `opt.namesave`: if `opt.display` is 0, this is a string which identifies the file name and format for the plot file; by default it is saved in the current working directory using the variable name and the `pdf` extension.

Note that in order to produce the plots, the gretl `multiplot` package (Schreiber and Tarassow, 2020) is required.

The `ESS` yields the numerical standard errors, the effective sample size for each parameter as well as the multivariate version. The computation follows the batch mean approach by Vats et al. (2019). Notice that effective sample size is inversely related to autocorrelation, so similar arguments to the ones above apply here too. The option available for this command is `opt.batchsize` which, as its name suggests, is used to set the batch size. The default value is the square root of the number of replications.

[6] For the list of recognised formats, see the reference to the gretl command `plot`.

The option `Geweke` computes Geweke's convergence diagnostic (Geweke, 1992). By default the statistic is computed for each parameter and the p-value, using a standardised normal distribution, is provided. It is possible to change the fraction of starting and ending sample to use, which is by default set to 0.1 and 0.5, via `opt.Gewekesample`.[7]

The `Heidelberg` option, instead, computes the Heidelberg and Welch convergence diagnostic: the p-values for each sample proportion are displayed for the parameters.

The `Geweke` and `Heidelberg` options rely on the computation of the *long-run variance* (also known as the spectral density at 0); internally, this is handled via the built-in function `lrvar`, which uses a Bartlett kernel with an adjustable window size.

6 Changelog

0.9 Initial release

0.91 Added the diagnostic function `mcmc_checks`; note that this introduces a dependency on `multiplot`.

References

Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, pages 483–485. Association for Computing Machinery.

Brooks, S. P. and Gelman, A. (1998). General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7(4):434–455.

Brown, P. J., Vannucci, M., and Fearn, T. (1998). Multivariate bayesian variable selection and prediction. *Journal of the Royal Statistical Society B*, 60(3):627–641.

Fernandez, C., Ley, E., and Steel, M. F. (2001). Benchmark priors for bayesian model averaging. *Journal of Econometrics*, 100(2):381–427.

Gamerman, D. (1997). Sampling from the posterior distribution in generalized linear mixed models. *Statistics and Computing*, 7(1):57–68.

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, pages 457–472.

Geweke, J. (1992). Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In *Bayesian Statistics (J.M. Bernardo, J.O. Berger, A.P. Dawid and A.F.M. Smith eds.)*, volume 4, pages 169–193. Clarendon Press, Oxford, UK.

Green, P. J. (1995). Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732.

Green, P. J. (2003). Trans-dimensional markov chain monte carlo. In *Highly Structured Stochastic System*, pages 179–198. Oxford University Press.

Hastie, D. I. and Green, P. J. (2012). Model choice using reversible jump markov chain monte carlo. *Statistica Neerlandica*, 66(3):309–338.

Heidelberger, P. and Welch, P. D. (1983). Simulation run length control in the presence of an initial transient. *Operations Research*, 31(6):1109–1144.

[7] In order to preserve the idea of the test even when it is performed on the whole chain made by the aggregation of the parallel ones, the proportion of the samples to use is built using the initial and ending parts of the single chains and then aggregated.

Holmes, C. C. and Held, L. (2006). Bayesian auxiliary variable models for binary and multinomial regression. *Bayesian Analysis*, 1(1):145–168.

Kass, R. E. and Wasserman, L. (1995). A reference bayesian test for nested hypotheses and its relationship to the schwarz criterion. *Journal of the American Statistical Association*, 90(431):928–934.

Lamnisos, D., Griffin, J. E., and Steel, M. F. (2009). Transdimensional sampling algorithms for bayesian variable selection in classification problems with many more variables than observations. *Journal of Computational and Graphical Statistics*, 18(3):592–612.

Lamnisos, D., Griffin, J. E., and Steel, M. F. (2013). Adaptive monte carlo for bayesian variable selection in regression models. *Journal of Computational and Graphical Statistics*, 22(3):729–748.

Madigan, D. and Raftery, A. E. (1994). Model selection and accounting for model uncertainty in graphical models using occam's window. *Journal of the American Statistical Association*, 89(428):1535–1546.

Madigan, D., York, J., and Allard, D. (1995). Bayesian graphical models for discrete data. *International Statistical Review*, 63(2):215–232.

Rosenthal, J. S. (2000). Parallel computing and monte carlo algorithms. *Far East Journal of Theoretical Statistics*, 4(2):207–236.

Schreiber, S. and Tarassow, A. (2020). *multiplot*. gretl package version 0.2.

Sha, N., Vannucci, M., Tadesse, M. G., Brown, P. J., Dragoni, I., Davies, N., Roberts, T. C., Contestabile, A., Salmon, M., Buckley, C., et al. (2004). Bayesian variable selection in multinomial probit models to identify molecular signatures of disease stage. *Biometrics*, 60(3):812–819.

Vats, D., Flegal, J. M., and Jones, G. L. (2019). Multivariate output analysis for markov chain monte carlo. *Biometrika*, 106(2):321–337.