




ParMA: Parallelized Bayesian Model Averaging for Generalized Linear Models

Riccardo (Jack) Lucchetti 
Università Politecnica delle Marche

Luca Pedini 
Università Politecnica delle Marche

Abstract

This paper describes the `gretl` function package **ParMA**, which provides Bayesian model averaging (BMA) in generalized linear models. In order to overcome the lack of analytical specification for many of the models covered, the package features an implementation of the reversible jump Markov chain Monte Carlo technique, following the original idea by Green (1995), as a flexible tool to model several specifications. Particular attention is devoted to computational aspects such as the automatization of the model building procedure and the parallelization of the sampling scheme.

Keywords: BMA, GLM, RJMCMC, parallelization, `gretl`.

1. Introduction

The issue of model uncertainty has become a prominent topic in modern Statistics and Econometrics; in order to deal with the matter, two diverging strands of literature have developed over the past fifty years: model selection techniques and, more recently, model averaging techniques.

In the former approach, first a statistical model is chosen among the possible alternatives and then one or more quantities of interests are computed; in doing so, however, *model uncertainty* (Chatfield 1995), i.e., the impossibility of recognizing whether the selected model is really true is implicitly ignored. Therefore, inference is performed without taking into proper account the conditioning on the selected model.

With model averaging, instead, the quantities of interest are averaged over the whole model space, i.e., the set containing every specifications, with weights reflecting the adherence of the correspondent model to the data. In particular, Bayesian model averaging (BMA), thanks to its flexibility and great potential in terms of interpretability and inferential analysis, has

become the leading approach¹: the basic idea, dating back to the work by [Madigan and Raftery \(1994\)](#), starts from the definition of a quantity of interest β . In principle, β could be any quantity pertaining to a parametric statistical model: a parameter, a hypothesis test, a forecast. However, in the brief exposition that follows, we will assume for simplicity that it is a model parameter. The posterior distribution of β is

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

where y represents the data, M_i the i -th model from a model set $\mathcal{M} = \{M_1, \dots, M_m\}$, which in a Bayesian framework becomes an additional parameter; $\mathbf{P}(\beta | M_i, y)$ identifies the model specific posterior distribution for β and $\mathbf{P}(M_i | y)$ the so-called posterior model probability. In other words, Equation 1 defines a mixture distribution, where in order to account for model uncertainty, the model specific posterior of β is averaged across the set of models, with weight equal to the posterior model probability.

Two related quantities are the posterior model averaging expected value and the posterior model averaging variance given by:

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

$$\mathbf{VAR}(\beta | y) = \sum_{i=1}^m [\mathbf{VAR}(\beta | y, M_i) + \mathbf{E}(\beta | y, M_i)^2] \mathbf{P}(M_i | y) - \mathbf{E}(\beta | y)^2 \quad (3)$$

with $\mathbf{E}(\beta | M_i, y)$, $\mathbf{VAR}(\beta | M_i, y)$ 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:

$$\mathbf{P}(M_i | y) = \frac{p(y | M_i) \mathbf{P}(M_i)}{\sum_{j=1}^m p(y | M_j) \mathbf{P}(M_j)}$$

where $p(y | M_i)$ is the marginal data density (also referred to as marginal likelihood) and $\mathbf{P}(M_i)$ the prior probability for M_i . The classical application of BMA computes Equations 2 and 3 directly, often ignoring Equation 1: when closed formula for both posterior moments and posterior model probabilities are available, both summations can be easily derived with the only computational burden given by the cardinality of the model set: in a typical nesting scenario, where M_i is a conditional model with up to k possible covariates, the model space cardinality equals 2^k .

In the literature, two main mechanics have been provided to circumvent the problem: the first one, known as *Occam's window* ([Madigan and Raftery 1994](#)) aims to reduce the model space by removing too much complex and redundant specifications in terms of their posterior model probability. Actually, the true benefit derives only in the computation of moments for a small set of models: in order to perform the procedure properly, the *a priori* computation of model posterior for every specifications is necessary, a not negligible effort. This leads to the second possibility devised by [Madigan, York, and Allard \(1995\)](#): Markov chain Monte Carlo model composition (MC³). If analytical posterior moments and model posteriors are

¹For a comprehensive review of the topic, see [Steel \(2020\)](#).

available, a standard Markov chain Monte Carlo (MCMC) sampler, such as the Metropolis-Hastings (Hastings 1970) scheme, can be adopted to sample models from the model space: in this way posterior model probabilities (which are actually the target distribution of the simulation) are obtained avoiding the computation of the normalizing constant and Equations 2 and 3 are computed accordingly on the sampled models only.

However, analytical formulations exist only for a handful of cases, notably the linear one, while non-linear alternatives such as binary or count models pose several questions on how to apply model averaging efficiently with as few approximations as possible. The issue is still open, and several proposals have been put forward and implemented in software.

For linear models, for instance, several R packages (R Core Team 2022) exist: **BMA** by Raftery, Hoeting, Painter, Volinsky, and Yeung (2022) exploits both Occam's window or MCMC with the Bayesian information criterion (BIC) approximation for model posteriors; **BMS** by Zeugner and Feldkircher (2015) which, in turn, performs model averaging by MCMC on the model space with great flexibility in the choice of priors and model proposal kernels; **BAS** by Clyde, Ghosh, Littman, Li, and Van de Bergh (2022) which introduces Bayesian adaptive sampling (BAS) as an alternative to MCMCs; **mombf** by Rossell (2022), that allows to perform Bayesian model selection or averaging with the so-called non-local priors (Johnson and Rossell 2010, 2012); **BayesVarSel** by Garcia-Donato and Forte (2018), which is mainly concerned in the computation of Bayes factors providing several tools for model selection and averaging with different prior choices.²

In the `gretl` eco-system, BMA for linear models is implemented in the package **BMA** by Błażejowski and Kwiatkowski (2015), which makes use of the MCMC design with the addition of jointness measures; Błażejowski and Kwiatkowski (2018) introduces, instead, the **BACE** package, which runs Bayesian average of classical estimates (Sala-I-Martin, Doppelhofer, and Miller 2004) for linear and time series (augmented distributed lags) models.

Generalized linear models (GLMs) do not generally yield closed formula for posterior moments and model probabilities, so BMA is problematic. It is not a mere coincidence that model averaging in these kind of models has been considered much less often, despite the wide applicability of GLMs in social sciences. Among the R packages mentioned above, GLMs can be handled by **BMA**, which considers maximum-likelihood (ML) estimators for posterior moments and BIC approximation for model posteriors; **BAS**, which uses again ML estimators and Laplace approximation for model posteriors; and **mombf**, which considers only the computation of Bayes factor in probit models using again approximations around the ML estimators.

In this paper, we introduce a `gretl` package for performing BMA on the GLM: following the original idea by Green (1995, 2003) and in particular the subsequent work by Lamnisos, Griffin, and Steel (2009) in probit models, we use a reversible jump Markov chain Monte Carlo (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, as the same sampling scheme is adopted for different types of GLMs, and in terms of interpretability, since Equation 1 can be readily computed via parameter sampling and the quantities in Equation 2 and 3 follow directly from the sample counterparts.

²For a comprehensive comparison between R packages see Amini and Parmeter (2011).

Compared to the three R packages listed above, the advantage of our approach is that the RJMCMC scheme does away with much of the need for approximations. Of course, RJMCMC has its drawbacks too: the sampling scheme proposed here is rather demanding in terms of CPU time, but we tackle down the issue by introducing parallelization.

As for available RJMCMC software, the original Fortran program **AutoRJ** by Green (2003) or its C version by Hastie and Green (2012) only deals with some common model exploration problems. Other official releases were not directly available until the appearance of the R package **rjmc** by Gelling, Schofield, and Barker (2019); this package exploits the modified framework by Barker and Link (2013) to provide some model selection interest quantities (e.g., Bayes factor) and reaches a tremendous flexibility in terms of model specification allowing even more customizable cases than standard GLMs. However, the package requires the *a priori* availability of posterior sampled parameters for each model involved, obtainable by some extra code or software, as the package does not provide such individual sampling scheme. Our package, on the other hand, circumvents this problem by sampling parameters and models together, which also provides model averaging quantities directly.

The R package **spikeSlabGAM** by Scheipl (2011) covers a range of techniques known as stochastic search variable selection (SVSS, see George and McCulloch 1993): as we will see later, SVSS is the main alternative to RJMCMC for the joint sampling of models and parameters, and in **spikeSlabGAM** the main focus is on variable selection for (spatial) generalized additive mixed regression models. The probabilities of inclusion for each regressor are computed also taking into account functions of the original covariates, such as interactions or polynomials; in order to mitigate the computational burden, SVSS also provides the option of parallelizing the chain. The main differences with the package we are describing here are:

- Our package is more heavily oriented towards the model averaging philosophy instead of variable selection, although of course **ParMA** can be also used for variable selection and **spikeSlabGAM** can be used for model averaging.
- **ParMA** targets simple generalized linear models instead of generalized additive models and offers a larger selection of link functions.
- **spikeSlabGAM** is focused on spike and slab priors, while the RJMCMC technique used in **ParMA** is closer to the original MCMC approach.
- Last but not least, **spikeSlabGAM** is an R package, while **ParMA** is a gretl package. Although R and gretl both belong to the free software ecosystem and R packages are relatively easy to use from within the gretl interface, the two communities are relatively distinct, so the two packages cater for different audiences.

The rest of the paper is organized 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 parallelization and convergence; in Section 5 we discuss in detail the package and its main features; Section 6 provides three empirical applications of the package on different GLMs; finally, Section 7 concludes.

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, \dots, 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 $\mathbf{E}(y_i) = \frac{\partial b(\theta_i)}{\partial \theta_i}$ and $\mathbf{VAR}(y_i) = \frac{\partial^2 b(\theta_i)}{\partial \theta_i^2} a_i(\phi)$

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

$$l(\mu_i) = \eta_i = x_i^\top \beta$$

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

ML estimation of GLMs can be carried out, in a frequentist framework, via iterative weighted least squares (IWLS) 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:

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

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

$$V^{(i)} = (V_0^{-1} + X^\top W(\beta^{(i-1)})X)^{-1} \tag{4}$$

$$m^{(i)} = V^{(i)}(V_0^{-1}m_0 + X^\top W(\beta^{(i-1)})z(\beta^{(i-1)})); \tag{5}$$

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:

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

where $q(\beta^{(i)} | \beta^{(i-1)})$ is a normal density evaluated at $\beta^{(i)}$ with mean and variance, respectively, computed with Equations 4 and 5.

2.2. BMA in GLMs

As already stated in Section 1, the main difficulty in the Bayesian treatment of GLMs is analytical intractability; in the BMA context, this problem is made even worse because model posteriors also have to be specified.

Common routines applied in practice involve the use of the standard BMA paradigm with the Occam's window or MCMCs but at the cost of several approximations which ranges from ML estimators and the corresponding variance as proxies for posterior moments, to the use of Laplace or BIC approximations for posterior model probabilities. Nevertheless, all of these shortcuts heavily rely on some regularities conditions which may not be met in practice.

For a generalization of the standard paradigm, there are several possibilities: [Chen, Huang, Ibrahim, and Kim \(2008\)](#), for instance, use conjugate priors ([Chen and Ibrahim 2003](#)) and define model probabilities in a very appealing way via the Bayes factor by using two MCMCs: one from the posterior distribution and one from the prior distribution of the parameters under the unrestricted model. In this way, model averaging quantities can also be derived with the notable advantage of the conjugate prior set-up.

The previously mentioned SSVS technique by [George and McCulloch \(1993\)](#) is another alternative: parameters and models are sampled jointly using the Gibbs sampler via data augmentation. Parameters are sampled from their posterior distributions conditioned on the model, which in turn are used to sample the model from the posterior model probability conditioned on the parameters; parameters are always drawn from the full model, avoiding any transdimensional transformation and in case a variable is likely to be absent, its related sampled parameter is set near zero. Nevertheless, the definition of the conditional posterior model distribution is far from being simple, and depends heavily on the link function of the GLMs.³

The route chosen in this paper follows the RJMCMC framework by [Green \(1995\)](#) instead. RJMCMC can be thought of as a modification of the original MCMC framework for model exploration where, instead on focusing only on models, the parameter β_i and the related model M_i are *jointly* sampled using transdimensional transformations: at each step a specification is proposed and the corresponding parameters are attached not using another sampling scheme, but simply transforming the ones of the previous step via an *ad-hoc* function. The problem of potentially different number of parameters between models is taken into account through the so-called *matching* variable, which acts as a substitute parameter in order to balance the overall dimension. Moreover, [Green \(2003\)](#); [Hastie and Green \(2012\)](#) extend the general framework to consider model selection and automated procedure; whereas [Holmes and Held](#)

³[Frühwirth-Schnatter and Wagner \(2006\)](#); [Frühwirth-Schnatter and Frühwirth \(2007, 2010\)](#); [Frühwirth-Schnatter and Wagner \(2010\)](#) are some examples of SVSS in GLMs.

(2006); Fouskakis, Ntzoufras, and Draper (2009); Lamnisis *et al.* (2009) propose some notable contribution in GLMs.

Clearly, SVSS and RJMCMC are close in spirit: for example, both procedures compute the model averaging posterior quantities as the sample equivalents of the simulated parameters. However, they have different pros and cons: the Gibbs sampling technique makes SVSS faster, but less flexible. In RJMCMC, the same sampling scheme can be applied to different kinds of GLMs and the definition of proposal distributions for the MCMC turns out to be simpler, especially in the framework proposed here.

2.3. 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 this way, it is possible to connect drawings with different size, which is something usually not called for in classic MCMC applications. The variable u_i is the so-called *matching variable*: it is assumed to be a random variable (a standard normal or a Student t), whose purpose is guaranteeing that the total dimension of the space given by (β_i, u_i) is equal to the dimension of (β_j, u_j) . As a consequence, the variable u_i contains the potential parameters which are deleted or added in order to have the size of β_i match that of β_j .

For example, suppose that β_i is a vector of 2 elements (β_{i1}, β_{i2}) , and that we want to jump to a 3-element vector $\beta_j = (\beta_{j1}, \beta_{j2}, \beta_{j3})$ via $\beta_j = g(\beta_i)$. Since the size of β_j is greater than the size of β_i , we need to add to β_i a variable u whose dimension is $3 - 2 = 1$. Therefore, $\beta_j = g(\beta_{i1}, \beta_{i2}, u_{i1})$.

The general framework starts from the definition of the target distribution, $P(\beta_i, M_i | y)$:

$$P(\beta_i, M_i | y) \propto p(y | M_i, \beta_i) P(\beta | M_i) P(M_i)$$

where $p(y | M_i, \beta_i)$ is the likelihood of model M_i ; $P(\beta | M_i)$ the prior on the parameter conditioned on the model, and finally $P(M_i)$ the prior distribution of the specification.

The RJMCMC algorithm samples the couple (β_j, M_j) , given the current state (β_i, M_i) firstly proposing a model movement from M_i to M_j with probability given by the kernel $q(M_j | M_i)$, and then deterministically computing $(\beta_j, u_j) = g(\beta_i, u_i)$. The required matching variables are generated accordingly, and the overall acceptance probability of the movement is⁴:

$$\rho = \min \left[\frac{P(\beta_j, M_j | y) f(u_j) q(M_i | M_j)}{P(\beta_i, M_i | y) f(u_i) q(M_j | M_i)} \cdot \left| \frac{\partial g(\beta_i, M_i; u_i)}{\partial(\beta_i, M_i; u_i)} \right|; 1 \right] \quad (6)$$

⁴In Godsill (2001, 2003) some additional considerations and remarks regarding RJMCMCs are provided.

where $f(\cdot)$ denotes density functions, $\frac{\partial g(\beta_i, M_i; u_i)}{\partial(\beta_i, M_i; u_i)}$ the Jacobian of the transformation $g(\cdot)$ which is requested in order to take into account the change of measurement from (β_i, u_i) to (β_j, u_j) . Great attention should be devoted to the choice of correct proposal kernels and transformation functions: as pointed out by Green (2003), this choice is crucial and the closer they are to the real posteriors, the more efficient the chain is⁵. Unfortunately, this choice is often troublesome, but a fair compromise is provided in Section 2.4, following the idea of a plausible automated method for RJMCMCs.

The move from model M_i to model M_j can be accomplished in many ways; here, we consider two: in the simple procedure by Madigan *et al.* (1995), the difference between M_i and M_j is given by addition or deletion of one variable. Alternatively, in the same spirit to a proposal by Lamnisos *et al.* (2009), at each iteration the number of variables to change p is drawn from a binomial distribution, with parameters k' and ω , as in

$$\binom{k'}{p} \omega^p (1 - \omega)^{k' - p} \quad (7)$$

and then the *direction* of the move (add, swap or delete) is chosen. Finally, the final proposed model is chosen uniformly from the subset of models accessible given the selected move.

2.4. An automated RJMCMC sampler

In Green (2003); Hastie and Green (2012) and, especially in Lamnisos *et al.* (2009); Lamnisos, Griffin, and Steel (2013) we find a particularly suitable function $g(\cdot)$ for GLMs⁶: 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

$$\beta_j = g(\beta_i, M_i, u_i) = \mu_j + B_j v \quad (8)$$

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 v is defined as:

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

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 of a standard normal or a Student's t distribution. Notice that the parameter β is treated as a multivariate normal, which in a first step gets standardized and then corrected via the mean and covariance matrix of the new model. The normal choice could be objectionable, but Green (2003) shows how this proposal is a good compromise between efficiency of the chain and simplicity.

⁵Some contributions in this direction are Brooks, Giudici, and Roberts (2003) and Barker and Link (2013): the former extends the framework considering efficient proposal distribution, whereas the latter transforms the RJMCMC into a Gibbs sampling with potential tremendous advantages in computational terms.

⁶Another interesting approach for only binary data is provided by Holmes and Held (2006).

The probability in Equation 6 becomes:

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

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.

Notice that, in the special case where the posterior parameter distributions $P(\beta_i | M_i, y)$ are normal, Equation 9 reduces to the ordinary MCMC acceptance rate:

$$\rho = \min \left[\frac{\mathbf{P}(M_j | y) q(M_i | M_j)}{\mathbf{P}(M_i | y) q(M_j | M_i)}; 1 \right]$$

on account of the fact that $\mathbf{P}(\beta_i, M_i | y) = \mathbf{P}(\beta_i | M_i, y) \mathbf{P}(M_i | y)$.

2.5. Prior choices and other technicalities

The sample scheme defined here provides a general solution for model building problems, but its actual effectiveness depends on the regularity of the data, the computation of μ_i and V_i , and, of course, the choice of the prior distributions.

The latter element always plays an important role, but in our context this is made even more crucial since the acceptance probability from Equation 9 depends on these via the joint posterior $\mathbf{P}(\beta_i, M_i | y)$ and the Jacobian term. It could easily happen that a particular prior choice for a particular GLM may performs rather poorly in a GLM of a different kind.

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 [Fernández, Ley, and Steel \(2001a\)](#) for linear models. The constant is to be included always, and in practice this is accompanied by centering all other regressors, so as to make them orthogonal to the constant. In linear models, if α denotes the intercept parameter, we assume $\mathbf{P}(\alpha) \propto 1$. The same argument, however, cannot be fully applied to other GLMs due to the nonlinearity of the link function. A solution is proposed in [Lamnisos *et al.* \(2009\)](#), who follows the suggestion

by Brown, Vannucci, and Fearn (1998); Sha *et al.* (2004) of a standard normal distribution with large variance of the following form:

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

where h is set to a large number⁷ (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., $g(X_i^\top X_i)^{-1}$ with $g > 0$.⁸ 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* standardization 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 $gn(X_i^\top X_i)^{-1}$, where n is the number of observations: this reflects more directly how the covariance should be derived from the unit information prior (UIP) covariance matrix by Kass and Wasserman (1995), which is generally the most common choice.⁹

Notice that in case of linear models the additional parameter related to variance of the error term, σ^2 , is assumed to have a diffuse prior, $P(\sigma^2) \propto \frac{1}{\sigma^2}$.

For the model prior, we use the binomial distribution:

$$P(M_i) = \prod_{j=1}^k \pi_j^{\delta_{ij}} (1 - \pi_j)^{1 - \delta_{ij}} \quad (10)$$

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.¹⁰ Turning the discussion on the posterior mean μ_i and covariance matrix V_i of the parameters of interest, these can be determined in various ways ranging from Laplace method to more advanced techniques: Green (2003) suggested to run previous MCMCs on each model to detect correct estimates; Hastie and Green (2012) introduces, instead, the use of mixture distributions. In Lamnisos *et al.* (2009, 2013) a wide variety of methods is exposed: from Laplace approximation, to Gamerman (1997) IWLS or more complex solutions such as the efficient proposal to maximize acceptance rate by Brooks *et al.* (2003). In the end, the IWLS seems to be an

⁷Usually, $h = 100$.

⁸Possibly, a hyper-prior can be adopted for c and/or g , but that of course increases the computational cost, so in practice, these parameters are kept fixed: the ridge prior c is commonly chosen via grid-search or cross-validation approaches (Lamnisos, Griffin, and Steel 2012), whereas for g , some proposed values are $g = n$ or $g = k^2$, with k as the total number of covariates.

⁹Some specific values for g are $g = 1$ which directly reflects the UIP covariance matrix; $g = 4$ as suggested by Fouskakis *et al.* (2009) for logistic regressions; $g = 9.87/k$ by Hanson, Branscum, and Johnson (2014). Another important reference for g -prior choices is Gelman, Jakulin, Pittau, and Su (2008).

¹⁰The 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, on the contrary, when $\pi_j = 0$ it is never included.

optimal compromise between efficiency and computational complexity, so we will exploit this choice for our framework, with some modifications explained below.

2.6. The RJMCMC sampler in a nutshell

The basic MCMC scheme is summarized 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 Equation 8.
3. Accept the move with probability from Equation 9, otherwise stay in (β_i, M_i) .
4. Repeat from 2 till convergence.

Notice that in our case the posterior mean μ_i and covariance matrix V_i are obtained via Equations 5 and 4 via a single iteration on the ML estimator. In the case of linear models we can substitute this approximation with its analytical counterpart.

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 Equations 5 and 4 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 Equation 8.
3. Accept the move with probability from Equation 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,

$$M_i = \{x_1, x_2, x_3, x_4\} \rightarrow [1 \ 1 \ 1 \ 1] \rightarrow 15 \text{ (hex 0f)}$$

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

$$M_j = \{x_1, x_3, x_4\} \rightarrow [1 \ 0 \ 1 \ 1] \rightarrow 11 \text{ (hex 0b)}$$

Storing information for each model, such as the posterior mean μ and covariance matrix V (see Section 2.4), 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 as long as the number of regressors is the one commonly found in real-world applications.

4. Parallelization in MCMCs

Parallelization in simple Monte Carlo experiments is a well established practice in computational statistics: instead of computing the whole simulation in a single core of the processor or in a single machine, splitting simulations across cores or several networked computers and aggregating the result accordingly leads to a massive CPU time gain. The key requirements is *independence* in sample drawings.

Apparently, there seems to be little room for parallelization in MCMCs, where the Markov property is used to set up a sequential process. In fact, parallelization 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 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, parallelization 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.

This kind of parallelization in MCMCs, often labeled as “vertical” or “embarrassing parallelizable” is the one we chose for our software package. In the next Sections, technical details will be explained more fully.¹¹

4.1. Convergence in parallel

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 analyze the scenario of a *univariate* parameter β simulated n times in c parallel chains (or cores) starting from overdispersed points. Given an unbiased estimator $\bar{\beta}$ of $E(\beta)$, the between (*intra core*) variance B and the within (inside the same core) variance W defined as,

$$B = \frac{n}{c-1} \sum_{i=1}^c (\bar{\beta}_i - \bar{\beta})^2 \quad (11)$$

$$W = \frac{1}{c(n-1)} \sum_{i=1}^c \sum_{j=1}^n (\beta_{ji} - \bar{\beta}_i)^2 \quad (12)$$

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:

$$\hat{R} = \frac{\hat{V}}{W} \quad (13)$$

where

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

Clearly, the closer Equation 13 is to 1, the more similar the chain are in terms of β , so convergence is deemed to be achieved when $R \simeq 1$.¹² The extension to a multivariate set-up is given in Brooks and Gelman (1998), where a generalization of \hat{R} is proposed given β as a $k \times 1$ parameter vector: define the matrices

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

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

¹¹It is worth noting that other approaches for parallel computing in MCMC exist: notably, the so-called “horizontal” MCMCs. With this technique, a single chain is run, but some of the inner components are parallelized. In general, this approach may provide more effective gains than its vertical counterpart, but requires more stringent conditions from a theoretical point of view.

¹²In their paper Gelman and Rubin (1992) propose other different indices either build as modification of Equation 13 or on different quantities such as quantiles.

as multivariate versions of Equation 11 and 12; then the new convergence statistics is:

$$\tilde{R} = \frac{n-1}{n} + \frac{c+1}{c}\lambda \quad (14)$$

where λ is the maximum eigenvalue of $\mathbf{W}^{-1}\mathbf{B}/n$. Again, convergence is reached when Equation 14 is close to 1; a commonly used threshold is $\tilde{R} \leq 1.2$.

In practical situations, the Brooks and Gelman statistics should be backed up by some 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 analyzed graphically, by visualizing the sequence of the sampled parameters as well as the running mean plot for both the parallel chains and the resulting single one.

In addition, Geweke (1992) and Heidelberger and Welch (1983) propose two distinct diagnostic tests for assessing convergence: the former is a robust univariate test on the mean difference between a parameter sample coming from the starting $0.1n$ replications (sub-sample A) of the chain and another sample given by the ending $0.5n$ replications (sub-sample B), n being the number of MCMC iterations. 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 related sub-sample, and finally $S(0)$ is the spectral density at 0 (long-run variance) of the parameters.

In Heidelberger and Welch's test, the following quantity is defined:

$$B_n = \frac{(\sum_{i=1}^{nt} \beta_i - \lfloor nt \rfloor \bar{\beta})}{\sqrt{nS(0)}}, \quad 0 \leq t \leq 1 \quad (15)$$

where again, β is a univariate parameter sampled n times and $S(0)$ the spectral density at 0. Equation 15 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 $\alpha = 0.1, 0.2, \dots$ 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. The package

The methods outlined in the previous Sections are implemented as a `gretl` function package called **ParMA** (parallelized model averaging). **ParMA** is available on the `gretl` function packages archive at http://gretl.sourceforge.net/current_fnfiles/ParMA.zip. For instructions on how to access the `gretl` package repository the primary reference is Cottrell and Lucchetti (2022), but a brief account can also be found in Lucchetti and Pignini (2017).

The command

```
? pkg install ParMA.zip
```

will install the package from the official online repository. Alternatively, if the package has already been downloaded and is present as a local file, the following alternative can be used:


```
? pkg install /path/to/local/ParMA.zip --local
```

where of course `/path/to/local` has to be substituted with the appropriate file path. Once this is done, the package is loaded using the `include` command:

```
? include ParMA.gfn
```

The **ParMA** package provides a main function, called `bma_glm`, and three auxiliary functions, `bma_printout`, `marginal_graph` and `mcmc_checks`. The `bma_glm` function performs the numerical computation, and it is illustrated in the next Section, with a special attention to the settings used to parallelize the algorithm effectively. The other functions are used, respectively, for pretty-printing the main results, plotting the posterior distribution for the model parameters as well as additional diagnostic analysis, and are described in Section 5.3.

Parallelization is implemented by using the Message Passing Interface (MPI) specification, which has been used in `gretl` since 2014 alongside other technology, such as OpenMP. As Gropp, Lusk, Doss, and Skjellum (1996) say, “MPI [...] is a specification for a standard library for message passing that was defined by the MPI Forum, a broadly based group of parallel computer vendors, library writers, and applications specialists.”. MPI is an extremely effective parallelization architecture, and has been previously used in `gretl` in the **johansensmall** package (Schreiber and Jensen 2021) for bootstrap computation of p values for the Johansen cointegration test.

The ability of `gretl` scripts to use MPI depends on a series of prerequisites: the preliminary installation of a suitable MPI package is surely the major one¹³, although a lot depends on the operating system too. For Windows and Mac OS X users, for example, running a `gretl` snapshot version is enough to assure the correct working of MPI commands, once the MPI package is correctly installed. The software, in fact, will automatically detect the binaries needed for parallelization. In Linux platforms, instead, it is required to build `gretl` from the git source enabling the parallelization option.¹⁴

However, the technicalities of the MPI implementation are totally transparent to the user of the **ParMA** package, and the only step that is needed to enable parallelization (of course, on suitable hardware) is to set the `mpi` scalar in the options bundle to a number greater than 1 (see Section 5.2). Notice that even though each unit of MPI parallelization, known as *process*, can in principle 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.

Users, finally, should be aware that CPU time is not a straightforward function of the number of processors used. Section 6.2 provides an illustration of the impact of parallelization in a real-life example.

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)
```

¹³For Windows the MS-MPI toolkit; whereas for Linux or Mac OS X the choice is Open MPI or MPICH.

¹⁴For further details we refer to Cottrell and Lucchetti (2022, 2021).

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 recognized 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`.¹⁵ 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 threads, after the burn-in stage. On the contrary, the burn-in parameter `burn` is kept fixed for each parallel chain. Therefore, each thread will execute a number of draws d that is equal to

$$d = \text{burn} + \frac{\text{ndraw}}{c}$$

where c is the number of threads used. For example, if `ndraw` and `burn` were 10000 and 1000, respectively, using 4 parallel processes will cause each thread to perform 3500 Monte Carlo iterations ($3500 = 1000 + 10000/4$).

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).
- `sampled_pip`: Matrix containing the *posterior inclusion probability* (PIP) for each variable (one column per thread).

¹⁵For linear models, the intercept may be excluded by giving it a diffuse prior set-up; see Section 2.5.

- `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`: A bundle containing the best specifications in terms of model posteriors. The number of models is determined by the option `params.threshold` (see next section).
- `GB`: Matrix (vector) containing as first entry the multivariate Gelman and Brooks statistics as in Equation 14; the remaining elements are the univariate convergence statistics from Equation 13 for each parameter.
- `opt_for_print`: Bundle containing additional information to be passed into the printing function (`bma_printout`).
- `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` 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 Section determine the main aspect of the RJMCMC procedure used to implement BMA. However, the behavior of the function can be tuned more finely by passing a bundle with additional options. The keys recognized 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 excluding 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 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 $g(X_i^\top X_i)^{-1}$.
 - 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 Equation 10. 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 a 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.

Note that it is the user's responsibility to ensure that the starting points are suitably overdispersed. However, this point may, in empirical applications, be less crucial than it seems: the proposal kernel of the chain is primarily concerned with models, so even starting from a single model specification a thorough exploration of the model space should be guaranteed when many covariates are present. *Default:* 1.

- **kernel**, scalar: Choice for the kernel (see discussion at the end of Section 2.3); 0 gives the simpler choice *à la* Madigan *et al.* (1995). 1 is used for the more sophisticated alternative. *Default:* 0.
- **change_regr**, scalar: Only used if **kernel** is 1. The integer k' in Equation 7.
- **prob_regr**, scalar: Only used if **kernel** is 1. The probability ω in Equation 7.
- **resamp**, Boolean: Enables the resampling step (within move), when a new parameter and model proposal is rejected. *Default:* **resamp** = 0.
- **center**, scalar: Prior regularization of the covariates. 0: no modification, 1: centering, 2: standardization. *Default:* 1.
- **mpi**, integer: Number of threads used to parallelize the Markov chains. *Default:* if MPI is enabled in **gretl**, the number of available physical cores, as reported by the **ncores** key in the **\$sysinfo** bundle; otherwise 1.
- **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 best models in the related output bundle and it primarily applies to the printout of the procedure. Only models whose posterior probability exceeds **threshold** will be stored and printed. *Default:* 0.1.

5.3. Auxiliary functions

The package provides three auxiliary function for further processing: **bma_printout** and **marginal_graph** are used for displaying the output, while **mcmc_checks** 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.¹⁶ 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.

¹⁶For the list of recognized formats, see the reference to the `gretl` command `plot`.

- `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, Flegal, and Jones (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.

The option `Geweke` computes Geweke’s convergence diagnostic (Geweke 1992) as per Section 4.1. By default, the statistic is computed for each parameter and the p value, using a standardized 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`.¹⁷ The `Heidelberg` option, instead, computes the Heidelberg and Welch convergence diagnostic (again, see Section 4.1): 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.

Further details will be provided in the empirical illustration section.

6. Empirical illustrations

In this section, we will provide three empirical applications in order to illustrate the **ParMA** package: a well-known BMA example from the economic growth literature (Fernández, Ley, and Steel 2001b), an example of Poisson regression from Cameron and Trivedi (2013) and finally an experiment in binary models using the Mroz (1987) dataset.

In the former our aim is to show how the ability of the RJMCMC algorithm to replicate the benchmark results in a linear model framework: linear models, as already pointed out, present the advantage of analytical tractability, so common routines can adopt a straightforward MCMC method on the model space. RJMCMC, on the other hand, can also be applied to non-linear models due to its greater flexibility. Of course, given its greater computational cost, we are not advocating the use of RJMCMC for linear models, but we are providing this example here as a robustness check.

In the latter example, instead, we will directly test the performance of our code on a count data model, fully exploiting the parallelization of the process: great attention will be given to

¹⁷When the test is performed on the whole chain, by aggregating of the parallel ones, the proportion of the samples to use is computed from the initial and ending parts of the single chains and then aggregated.

the impact of parallelization in CPU time and the convergence of the chains using the Brooks and Gelman statistics as a diagnostic check (see Section 4.1).

The final example will provide a detailed analysis on the Brooks and Gelman multivariate statistic and its behavior for several number of drawings. Additional diagnostic analyses are provided too.

6.1. Linear regression example

The article by [Fernández *et al.* \(2001b\)](#) is considered a milestone in the Econometrics literature for what concerns BMA applications: the underlying idea was to refine, by introducing model uncertainty via BMA, the previous analysis by [Sala-I-Martin \(1997\)](#) on the economic growth of 140 countries using a set of heterogeneous covariates ranging from economic development indicators to socio-cultural features. In particular, the original authors considered a smaller dataset of $n = 72$ countries with $k = 41$ regressors, and defined a BMA experiment using MCMC technique.

The original set-up was as follows:

- For the intercept, a diffuse prior was adopted.
- For all other parameters, the normal prior was used, with mean zero and *Zellner-g* covariance matrix (see Section 2.5), with $g = k^2$.
- For the models, a uniform prior was used.
- Finally, the MCMC chain length was set to 2000000, with an extra 1000000 iterations as burn-in.

In the following, we are going to replicate the same design using RJMCMC with just two slightly modifications: in order to save time we parallelize the algorithm to 4 cores and we run 200000 drawings after a burn-in of 50000.

The gretl script is reported below

```
? open growth_BMA.gdt --quiet
? include ParMA.gfn
? list X = 2..42
? modeltype = "linear"
? n_iter = 200000
? burn_in = 50000
```

As for the bundle of additional options, we will use the package defaults, with only these modifications:

```
? bundle param
? param.pv_scaling = nelem(X)^2
? param.mpi = 4
? param.seed = 1234567
```

The function call is:

```
? b = bma_glm(GDP_growth, X, modeltype, n_iter, burn_in, param)
```

Execution produces the following output:

```
-----  
Bayesian Model Averaging with Generalized Linear Model  
-----
```

```
Type of specification: Linear model  
Model Prior: P(M) ~ Uniform  
Model dynamics: MCMCMC - add/delete (1)var  
Resampling allowed: No  
MPI - threads: 4  
Number of iterations/burn-in/thinning: 200000/50000/0  
-----
```

Overall sampling statistics

	mean	std.err.	pip	c. mean	c. se	BGstat
AbsoluteLat	0.00000	0.00003	0.04748	0.00000	0.00015	1.00052
SpanishColony	0.00019	0.00155	0.05661	0.00341	0.00562	1.00144
FrenchColony	0.00018	0.00114	0.04911	0.00368	0.00367	1.00080
BritishColony	-0.00006	0.00063	0.03489	-0.00183	0.00283	1.00119
War	-0.00035	0.00137	0.09076	-0.00390	0.00263	1.00257
LatinAmerica	-0.00197	0.00432	0.22889	-0.00859	0.00495	1.00099
SubSahara	-0.01233	0.00839	0.77239	-0.01597	0.00575	1.00057
OutwarOrient	-0.00006	0.00053	0.03265	-0.00174	0.00235	1.00031
Area	-0.00000	0.00000	0.03189	-0.00000	0.00000	1.00081
PrScEnrollment	0.00399	0.00917	0.20025	0.01993	0.01012	1.00101
LifeExpectancy	0.00082	0.00035	0.92036	0.00089	0.00026	1.00265
GDP60	-0.01604	0.00322	0.99740	-0.01608	0.00311	1.00308
Fr_Mining	0.01905	0.02352	0.46204	0.04124	0.01681	1.00475
DegreeCapital	0.00125	0.00146	0.47280	0.00264	0.00091	1.00354
YrsOpenEco	0.00679	0.00792	0.47642	0.01426	0.00501	1.00674
Age	-0.00000	0.00002	0.09003	-0.00005	0.00003	1.00010
Fr_Buddhist	0.00255	0.00579	0.20154	0.01266	0.00621	1.00181
Fr_Catholic	-0.00043	0.00296	0.12793	-0.00333	0.00767	1.00193
Fr_Confucian	0.05587	0.01432	0.99035	0.05641	0.01328	1.00101
EthnoLingFract	0.00036	0.00187	0.06123	0.00586	0.00499	1.00169
Fr_Hindu	-0.00314	0.01101	0.11767	-0.02673	0.02000	1.00091
Fr_Jewish	-0.00019	0.00275	0.03385	-0.00558	0.01389	1.00147
Fr_Muslim	0.00816	0.00757	0.61024	0.01337	0.00494	1.00088
PrimaryExport	-0.00095	0.00349	0.09760	-0.00977	0.00620	1.00232
Fr_Protestants	-0.00572	0.00716	0.44944	-0.01273	0.00498	1.00056
RuleofLaw	0.00757	0.00839	0.50472	0.01500	0.00528	1.00279
Growth_pop	0.00511	0.04626	0.03307	0.15466	0.20395	1.00097
WorkPop	-0.00023	0.00216	0.04553	-0.00507	0.00885	1.00064
SizeLabForce	0.00000	0.00000	0.07225	0.00000	0.00000	1.00064
HighEnrollment	-0.00146	0.01007	0.04551	-0.03199	0.03535	1.00018
PublEduptct	0.00081	0.02593	0.03827	0.02113	0.13092	1.00214

Revolution	-0.00003	0.00094	0.02749	-0.00106	0.00555	1.00063
PoliticalRights	-0.00014	0.00055	0.08822	-0.00159	0.00109	1.00257
CivillLiberty	-0.00028	0.00087	0.12527	-0.00223	0.00129	1.00025
Fr_EnglishLan	-0.00058	0.00229	0.08743	-0.00667	0.00441	1.00336
Fr_ForeignLan	0.00022	0.00127	0.06424	0.00348	0.00374	1.00160
ExchRateDist	-0.00000	0.00002	0.07699	-0.00005	0.00004	1.00140
EquipInv	0.16046	0.06805	0.92494	0.17348	0.05243	1.00168
NotEquipInv	0.02421	0.03161	0.42363	0.05716	0.02181	1.00392
SDBlackMkt	-0.00000	0.00000	0.04553	-0.00001	0.00001	1.00064
BlackMkt	-0.00147	0.00344	0.19112	-0.00771	0.00372	1.00281

 Gelman & Brooks multivariate R: 1.025

Best specifications (Posterior > 0.10):

No model posteriors exceed the chosen threshold of 0.10.

The best model detected is 000000046845800c - P(M|y) = 0.006885

Covariates: SubSahara LifeExpectancy GDP60 DegreeCapital Fr_Confucian
 Fr_Muslim Fr_Protestants RuleofLaw EquipInv NotEquipInv

Figure 1 compares the covariate posterior inclusion probabilities obtained via our function and the original ones by [Fernández et al. \(2001b\)](#): as it can be seen, the results are similar even using far fewer iterations¹⁸.

It is interesting to examine the univariate and multivariate versions of the Gelman and Brooks statistic: each statistic is far below the usual 1.1 threshold (note that the usual value in the literature is 1.2). This can be taken to imply that the four parallel Markov chains have converged to the target distribution.

In Figure 2 we plot the marginal posterior densities for the coefficients of GDP60, SubSahara, Fr_Catholic, AbsoluteLat. The plot can be produced through following lines of code:

```
? list_for_marg = GDP60 SubSahara Fr_Catholic AbsoluteLat
? marginal_graph(b, for_marg, "")
```

where `b` is the output bundle from the `bma_glm` function.

6.2. Poisson regression example

Here we perform BMA on the the popular count model example provided by [Cameron and Trivedi \(2013\)](#) on doctor visits: the dataset is provided in the standard `gretl` installation under the name `rac3d.gdt` and provides $n = 5190$ observations from the *Australian Health Survey* 1977-1978. The dependent variable `DVISITS` contains the number of consultations with doctors in the last two weeks; we use the original list of covariates, that is socioeconomic characteristics (`SEX`, `AGE`, `AGESQ`, `INCOME`), health insurance status indicators (`LEVYPLUS`, `FREEPOOR`, `FREEREPA`), health status measures (`ILLNESS`, `ACTDAYS`) and long-term status measures (`HSCORE`, `CHCOND1`, `CHCOND2`).

¹⁸Of course, it is possible to obtain a more precise result by increasing the number of iterations: with 500000 drawings and a burn-in of 100000 the differences between the benchmark and **ParMA** are negligible.

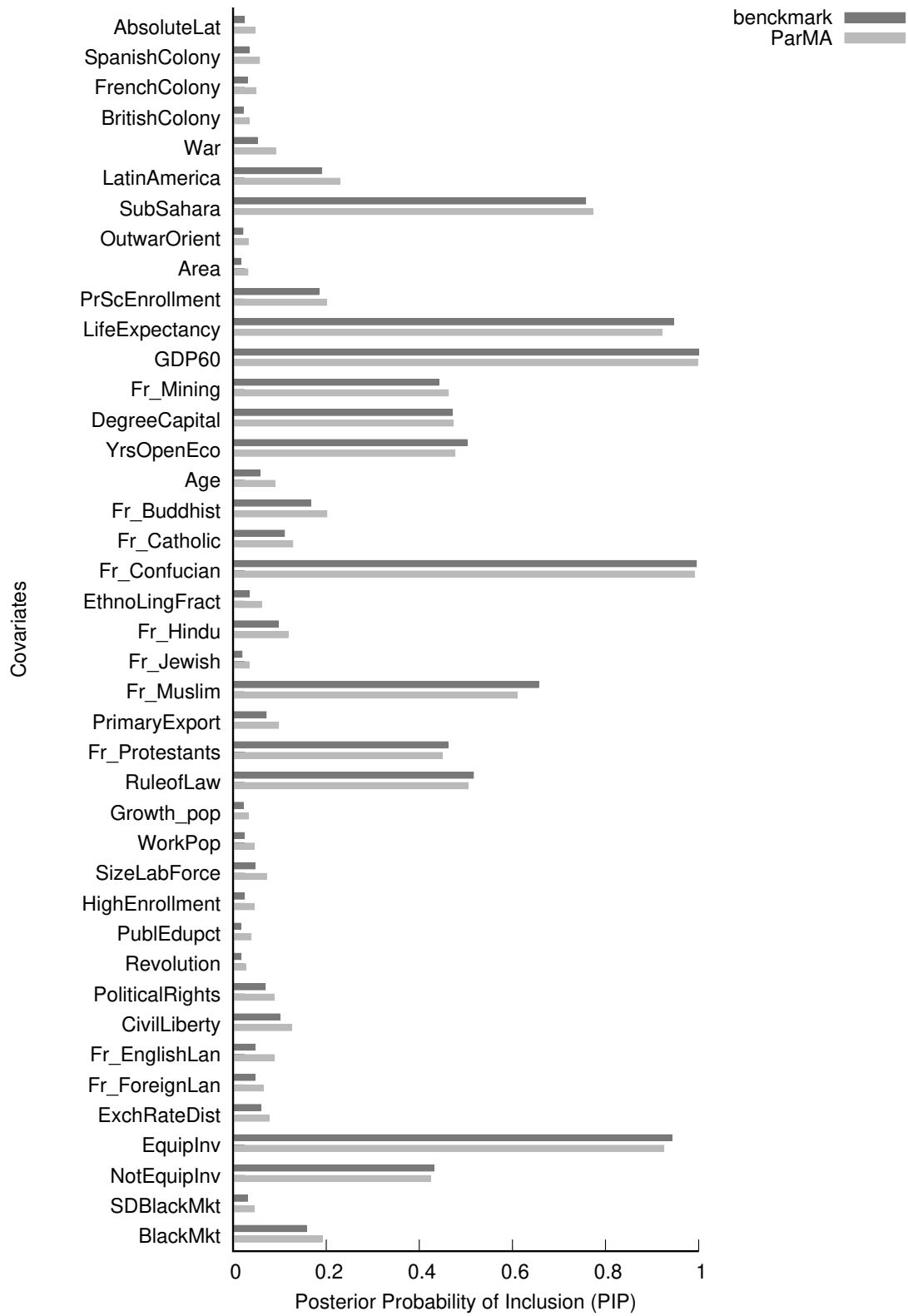


Figure 1: Comparison in posterior probability of inclusion (PIP) between [Fernández *et al.* \(2001b\)](#) and **ParMA**.

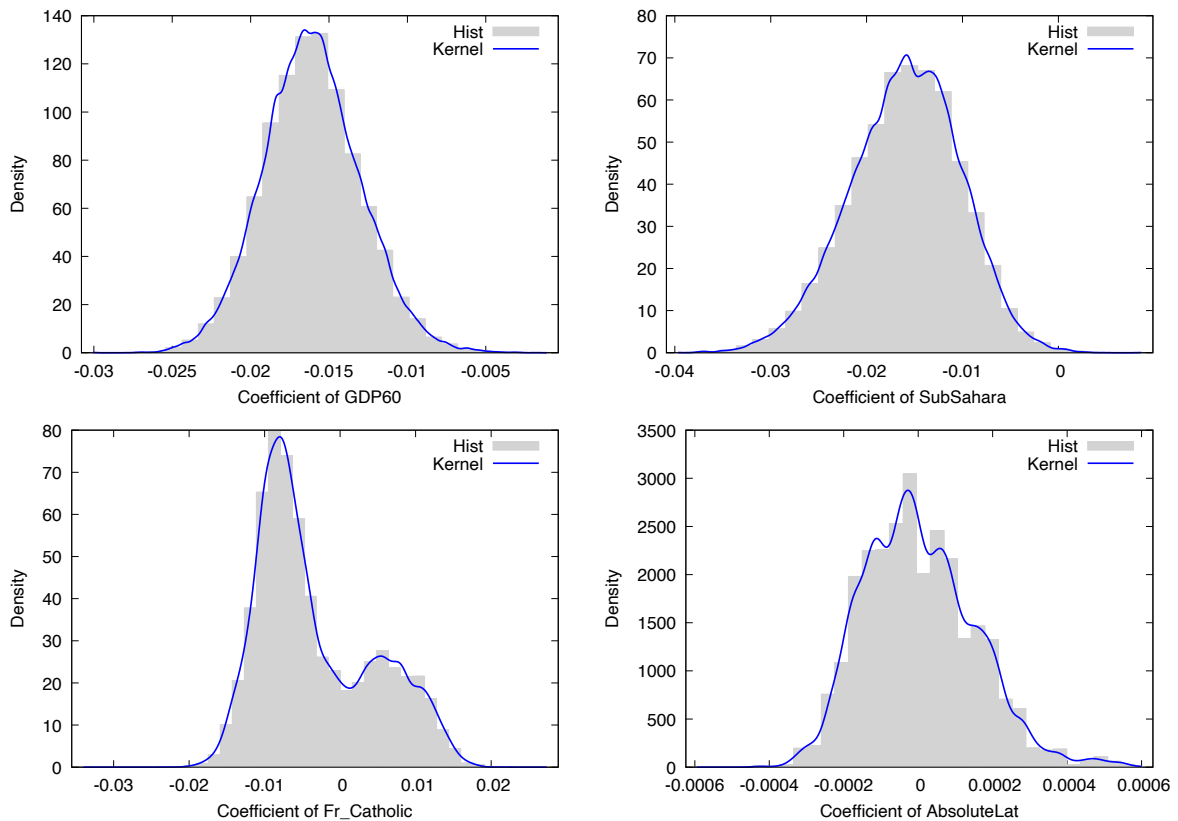


Figure 2: Marginal kernel densities and histograms (conditional on model inclusion) for the posteriors of GDP60, SubSahara, Fr_Catholic, AbsoluteLat parameters.

Our major item of interest in this application is monitoring the performance of parallelization together with the convergence of the chains: to do so, we define a standard set-up, that is, a uniform model prior; a normal prior for the parameters with zero mean and Zellner- g prior ($g = n$); add/delete proposal kernel and 200000 draws after a burn-in of 20000.

We estimate the model with different degrees of parallelization, from 1 to 32 threads. Of course, we keep the burn-in *fixed*, so each parallel chain will be trained over the 20000 burn-in iterations, and the number of desired replications (200000) will be split over the various threads. The experiment is performed on a machine with 20 physical cores: the RJMCMC runs on more than 20 threads use hyper-threading.

The gretl script is as follows:

```
? open rac3d.gdt --quiet
? include ParMA.gfn

? list X = SEX AGE AGESQ INCOME LEVYPLUS FREEPOOR FREEREPA
          ILLNESS ACTDAYS HSCORE CHCOND1 CHCOND2

? modeltype = "poisson"
? bundle param
? param.seed = 1234567
```

	Threads									
	1	2	4	8	12	16	20	21	22	32
Time (sec.)	1334.99	468.49	169.81	96.01	77.60	71.60	67.70	109.02	106.35	103.50
Multivariate R	1.000	1.002	1.013	1.009	1.009	1.012	1.022	1.023	1.021	1.027
<i>Posterior model probabilities for top three models</i>										
Model c5c	0.173	0.182	0.172	0.173	0.180	0.169	0.175	0.177	0.178	0.180
Model c1c	0.135	0.135	0.132	0.133	0.131	0.125	0.132	0.133	0.132	0.131
Model a5c	0.111	0.106	0.116	0.116	0.112	0.114	0.109	0.108	0.110	0.111
<i>Posterior inclusion probabilities</i>										
SEX	0.939	0.943	0.948	0.944	0.947	0.941	0.943	0.941	0.945	0.944
AGE	0.617	0.629	0.608	0.605	0.615	0.592	0.602	0.617	0.609	0.617
AGESQ	0.352	0.342	0.365	0.368	0.358	0.380	0.367	0.351	0.364	0.358
INCOME	0.212	0.208	0.201	0.205	0.199	0.205	0.206	0.208	0.204	0.203
LEVYPLUS	0.088	0.079	0.079	0.084	0.087	0.084	0.089	0.085	0.085	0.083
FREEPOOR	0.601	0.606	0.598	0.600	0.604	0.599	0.603	0.603	0.603	0.606
FREEREPa	0.047	0.045	0.042	0.049	0.045	0.048	0.050	0.048	0.049	0.045
ILLNESS	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
ACTDAYS	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
HSCORE	0.772	0.776	0.774	0.775	0.776	0.770	0.778	0.781	0.785	0.777
CHCOND1	0.041	0.046	0.041	0.039	0.039	0.039	0.043	0.046	0.043	0.043
CHCOND2	0.053	0.053	0.054	0.054	0.053	0.052	0.053	0.057	0.060	0.054

Table 1: Comparison of multi-thread performance of `bma_glm` in Poisson regression. Results which report more than 20 threads use hyperthreading.

```
? param.threshold = 0.08
? cores = seq(1,32)
? loop i = 1..32 --quiet
> n_iter = 200000
> burn_in = 20000
> param.mpi = cores[i]
> b=bma_glm(DVISITS, X, modeltype, n_iter, burn_in, param)
> endloop
```

Table 1 reports the main results of the experiment for selected values of the number of cores¹⁹: for each specific case the elapsed CPU time and the multivariate R statistic are shown, alongside with the posterior model probabilities of the best three specifications and the posterior inclusion probabilities for each covariate.

As can be inferred from the convergence statistics, each parallel chain reached its stationary state; as a further confirmation, the model posteriors of the three best models (which cover about 40% of the total model posterior) and the individual PIPs for the covariates are close to each other across the different scenarios.

A more immediate overview of the fact is provided in Figure 3, where the multivariate convergence statistics are plotted as a function of the number of threads used: at a first glance, each value reported is below 1.03, way below the customary threshold of 1.2. Again, a positive

¹⁹We decided to omit a few columns for the sake of clarity. Results for the omitted columns are available upon request, but are practically identical to what one would get by interpolation.

Model	Covariates
Model c5c	const, SEX, AGE, FREEPOR, ILLNESS, ACTDAYS, HSCORE
Model c1c	const, SEX, AGE, ILLNESS, ACTDAYS, HSCORE
Model a5c	const, SEX, AGESQ, FREEPOR, ILLNESS, ACTDAYS, HSCORE

Table 2: Description of the three best models detected via `bma_glm`. Model labels are the internal hexadecimal representation of the model.

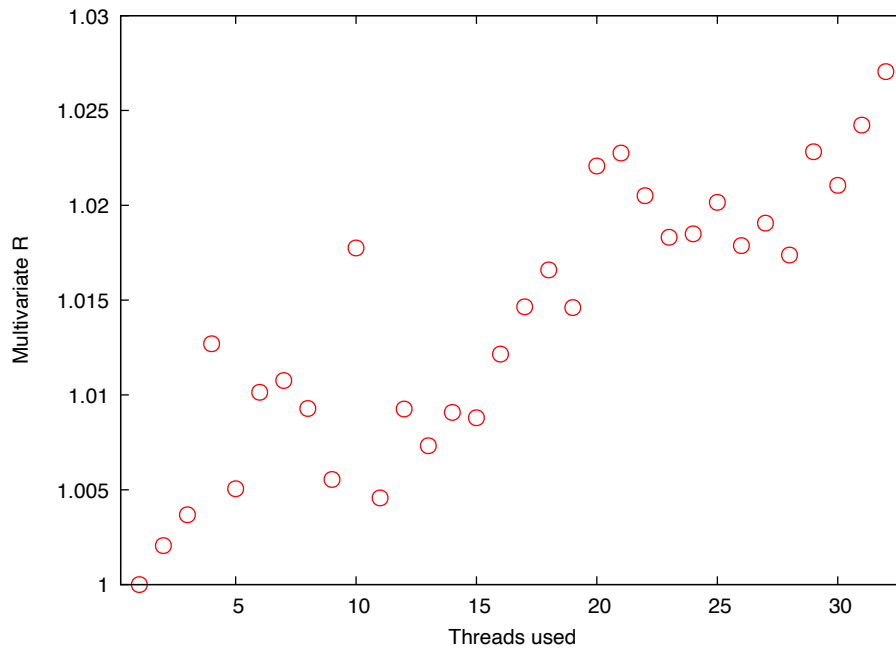


Figure 3: Multivariate R obtained in `bma_glm` for the Poisson example as a function of the used threads.

relationship between convergence measure and number of threads is quite evident: as the number of parallel chains grows, the measure grows as well, because splitting the number of desired drawings over a larger number of threads results in shorter chains. The fact that even with many threads the statistics remains far below 1.20 is due to the fixed burn-in.

CPU times are depicted in Figure 4, where elapsed time in seconds is plotted versus the number of threads. At a first glance, a hyperbolic decay pattern seems to emerge, as is reasonable. However, when the number of threads exceed that of physical cores performance deteriorates: employing more than 20 chains leads to an average execution time above 100 seconds.

6.3. An experiment on convergence

The Brooks and Gelman multivariate statistics is a very useful tool in parallel MCMC experiments to assess whether the Markov chain convergence has taken place. Therefore it may be of interest to study its behavior for a fixed number of threads and an increasing number of drawings.

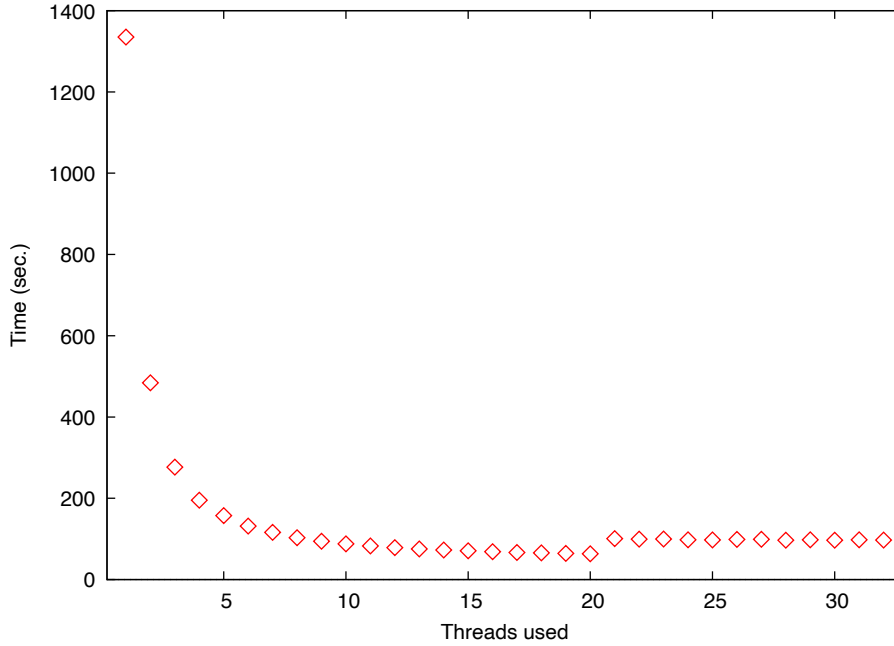


Figure 4: CPU time elapsed by `bma_glm` for the Poisson example as a function of the used threads.

To this end, we will use the famous dataset from [Mroz \(1987\)](#) (natively supplied with `gretl` under the name `mroz87`), and we will set up a probit model of labor market participation for married women, using as covariates the woman’s socio-economic characteristics (`WA`, `WE`, `AX`, `MTR`), as well as her husband’s (`HA`, `HE`, `HW`), together with some indications on the household (`KL6`, `CIT`) and on unemployment in the county of residence (`UN`).

As for the experiment set-up, we will perform BMA on 8 parallel chains, with a number of draws going from 1000 up to a million. The burn-in size will be set to the 10% of the total of drawings. For all other settings, we use the default values (see Sections 5.1 and 5.2).

In the spirit of the original paper by [Brooks and Gelman \(1998\)](#), we will employ different starting points for each chain using the additional option `param.start`:

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
<code>KL6</code>	1	1	1	1	1	1	1	0
<code>WA</code>	0	0	1	0	1	1	0	1
<code>WE</code>	0	0	0	1	0	0	0	0
<code>HA</code>	0	0	0	1	1	1	1	0
<code>HE</code>	1	1	1	0	1	1	0	0
<code>HW</code>	1	0	0	0	0	1	1	0
<code>MTR</code>	0	0	0	0	1	0	1	0
<code>UN</code>	1	1	0	0	1	0	1	1
<code>CIT</code>	0	1	1	0	0	0	1	1
<code>AX</code>	0	0	1	0	0	0	0	0

this matrix identifies the starting model for each thread (column).

The script is the following:

Drawings (total)	Burn-in	R BG	Mod 399	Mod 3b9
1000	100	8.238	0.576	0.183
2500	250	3.673	0.434	0.348
5000	500	1.501	0.495	0.316
7500	750	1.556	0.548	0.271
10000	1000	1.398	0.544	0.283
25000	2500	1.110	0.521	0.297
50000	5000	1.058	0.528	0.286
75000	7500	1.015	0.525	0.301
100000	10000	1.033	0.529	0.284
250000	25000	1.014	0.531	0.289
500000	50000	1.005	0.524	0.292
750000	75000	1.008	0.528	0.288
1000000	100000	1.003	0.526	0.291

Table 3: Comparison of convergence statistics across several scenarios using 8 parallel chains.

```
? open mroz87.gdt --quiet
? include ParMA.gfn
? list X = KL6 WA WE HA HE HW MTR UN CIT AX
? starting = {1, 1, 1, 1, 1, 1, 1, 0;
              0, 0, 1, 0, 1, 1, 0, 1;
              0, 0, 0, 1, 0, 0, 0, 0;
              0, 0, 0, 1, 1, 1, 1, 0;
              1, 1, 1, 0, 1, 1, 0, 0;
              1, 0, 0, 0, 0, 1, 1, 0;
              0, 0, 0, 0, 1, 0, 1, 0;
              1, 1, 0, 0, 1, 0, 1, 1;
              0, 1, 1, 0, 0, 0, 1, 1;
              0, 0, 1, 0, 0, 0, 0, 0}
? modeltype = "probit"
? bundle param = defbundle("start", starting, "mpi", 8, "seed", 271828)
? matrix iters = {1, 2.5, 5, 7.5, 10, 25, 50,
                  75, 100, 250, 500, 750, 1000}*1000
? loop j = 1 .. nelem(iters) --quiet
>   n_iter = iters[j]
>   burn_in = iters[j]*0.1
>   b = bma_glm(LFP, X, modeltype, n_iter, burn_in, param)
> endloop
```

The results are collected in Table 3: the multivariate Brooks and Gelman statistic is reported in column 3; using 8 threads requires at least 25000 iterations to deem that convergence has taken place: when parallel draws are too few²⁰ even after the burn-in phase the chains are quite heterogeneous, as shown by the convergence measure. As further evidence, the fourth and

²⁰Remember that the burn-in is fixed and the total amount of iteration is actually split across the threads.

Model	Covariates
Model 399	const KL6 WA WE HW MTR AX
Model 3b9	const KL6 WA WE HE HW MTR AX

Table 4: Description of the two best models using the binary example from the `mroz87.gdt` dataset.

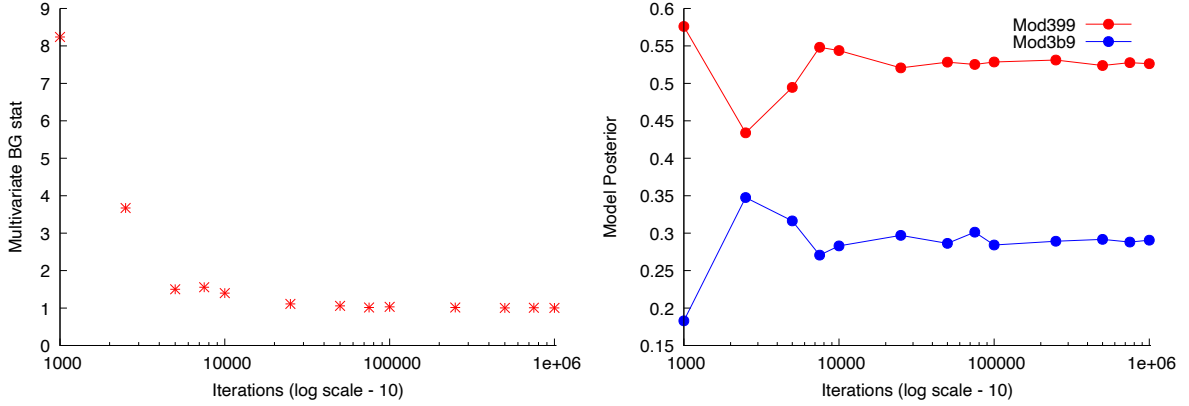


Figure 5: Left panel: Brooks and Gelman multivariate statistics across different iterations. Right panel: posterior model probabilities of the two best models across different iterations.

fifth columns report the posterior probability for two models²¹: again, the numbers stabilize only after the 25000 iterations threshold. These two numerical measures are also plotted for clarity in Figure 5, with convergence statistics in the left-hand pane and the posterior probability for the two best model in the right-hand pane.

As an example of the additional statistics described in Section 4.1, consider the case with 100000 replications, for which the convergence statistic is 1.033. Since this number is quite close to 1, it can be concluded that each parallel chain produces values for β that are similarly distributed. However, it may happen that the initial drawings may be come from a different posterior distribution than the final ones. By considering additional statistics such as the Geweke or Heidelberg tests one may shed some light on this aspect and gain better insight as to the homogeneity of the whole chain. In addition, graphical inspection of the parameter sequences, together with the ACF plot, may be also useful for analyzing the mixing properties of the chain.

In **ParMA** this is accomplished via the auxiliary function `mcmc_checks`: if we call `b` the `bma_glm` output bundle for the 100000 replications,

```
? c = mcmc_checks(b, "plot", _(plotlist = "HW"))
? c = mcmc_checks(b, "ESS")
? c = mcmc_checks(b, "Geweke")
? c = mcmc_checks(b, "Heidelberg")
```

the code above sequentially calls the plot for the HW variable, the effective sample size, Geweke and Heidelberg tests, with no options different from the default ones.²²

²¹Table 4 describes the two specifications.

²²The notation `_(key = object)` is a shorthand for the `defbundle` function.

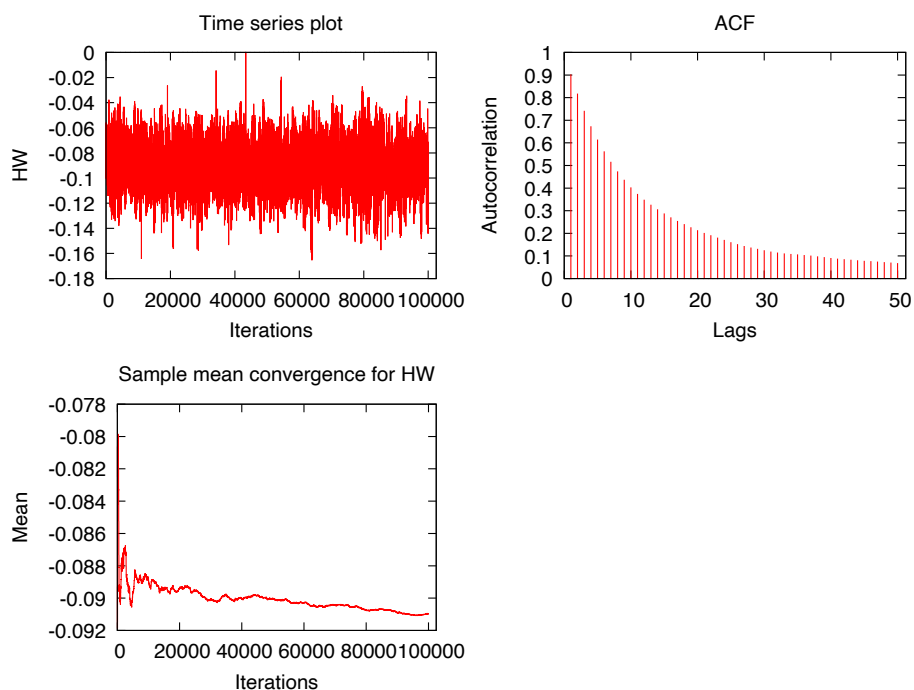


Figure 6: Sequence of sampled parameters for HW (top-left), autocorrelation function plot (top-right) and running mean plot (bottom-left).

The output provided for the `plot` option is reported in Figure 6; the rest of the command output follows:

```
-----Univariate Effective Sample Size-----
```

	ESS	nse
const	2641	0.017952
KL6	2567	0.041130
WA	1989	0.003401
WE	3811	0.010641
HA	12382	0.000496
HE	8405	0.005355
HW	3269	0.006170
MTR	1588	0.472773
UN	11633	0.000638
CIT	8911	0.005600
AX	3255	0.002308

```
-----Multivariate Effective Sample Size-----
```

```
mESS = 5487
```

```
----Geweke Univariate Convergence Statistics----
```

```
|Z| pvalue
```

const	0.2644	0.7915
KL6	2.4173	0.0156
WA	0.8525	0.3939
WE	0.2909	0.7712
HA	1.7999	0.0719
HE	0.9533	0.3405
HW	0.6489	0.5164
MTR	0.5494	0.5827
UN	0.5733	0.5665
CIT	0.5056	0.6131
AX	0.7424	0.4578

----Heidelberg-Welch Convergence Statistics----

	Full	90%	80%	70%	60%	50%
const	0.107	0.166	0.271	0.056	0.288	0.218
KL6	0.000	0.002	0.003	0.003	0.037	0.001
WA	0.000	0.000	0.000	0.000	0.000	0.000
WE	0.243	0.151	0.176	0.385	0.137	0.252
HA	0.221	0.294	0.269	0.198	0.299	0.270
HE	0.048	0.012	0.040	0.010	0.014	0.055
HW	0.000	0.000	0.001	0.001	0.010	0.017
MTR	0.000	0.000	0.001	0.003	0.020	0.021
UN	0.575	0.754	0.739	0.479	0.230	0.110
CIT	0.043	0.060	0.046	0.254	0.407	0.509
AX	0.004	0.002	0.000	0.003	0.012	0.021

By visual inspection of the sequence, the variable considered seems to be stationary, although autocorrelation seems substantial. This is confirmed by the effective sample size statistics (both univariate and multivariate) and this is also partly reflected by the Heidelberg-Welch statistics²³ where the stationarity test is rejected at size 0.05 for the same variables for which the effective sample size is smallest. Geweke's test, conversely, seems to indicate convergence for all coefficients bar one. The conclusion that can be drawn is that convergence is probably achieved, despite a possible autocorrelation problem; this is due to having two models which dominate the posterior model probability.

In order to illustrate the effect of the `resamp` option, we re-run the experiment with the `param.resamp` option set to 1:

```
? n_iter = 100000
? burn_in = 10000
? param.resamp = 1
? b = bma_glm(LFP, X, modeltype, n_iter, burn_in, param)
? c1 = mcmc_checks(b, "plot", _(plotlist = "HW", namesave = "resamp_HW.pdf"))
? c1 = mcmc_checks(b, "ESS")
? c1 = mcmc_checks(b, "Geweke")
? c1 = mcmc_checks(b, "Heidelberg")
```

²³For the Heidelberg-Welch statistics the p values for each sample dimension are reported.

The new output is reported below:

-----Univariate Effective Sample Size-----

	ESS	nse
const	58079	0.003839
KL6	51326	0.009127
WA	31164	0.000865
WE	9542	0.006682
HA	15288	0.000502
HE	8370	0.005366
HW	53513	0.001470
MTR	39160	0.093438
UN	38059	0.000362
CIT	28987	0.002911
AX	59160	0.000548

-----Multivariate Effective Sample Size-----

mESS = 35438

----Geweke Univariate Convergence Statistics----

	Z	pvalue
const	0.9358	0.3494
KL6	0.1061	0.9155
WA	0.1752	0.8609
WE	0.9043	0.3658
HA	0.2506	0.8021
HE	0.1671	0.8673
HW	0.5258	0.5990
MTR	1.1652	0.2439
UN	1.0713	0.2840
CIT	0.6548	0.5126
AX	2.2407	0.0250

----Heidelberg-Welch Convergence Statistics----

	Full	90%	80%	70%	60%	50%
const	0.439	0.667	0.772	0.531	0.546	0.388
KL6	0.260	0.358	0.329	0.408	0.435	0.698
WA	0.403	0.473	0.445	0.733	0.778	0.686
WE	0.818	0.899	0.438	0.167	0.334	0.533
HA	0.697	0.561	0.364	0.575	0.539	0.412
HE	0.755	0.762	0.851	0.583	0.612	0.606
HW	0.519	0.619	0.814	0.865	0.877	0.913
MTR	0.569	0.775	0.542	0.508	0.533	0.468
UN	0.682	0.815	0.690	0.381	0.244	0.215
CIT	0.861	0.925	0.895	0.960	0.908	0.694
AX	0.493	0.781	0.690	0.469	0.395	0.176

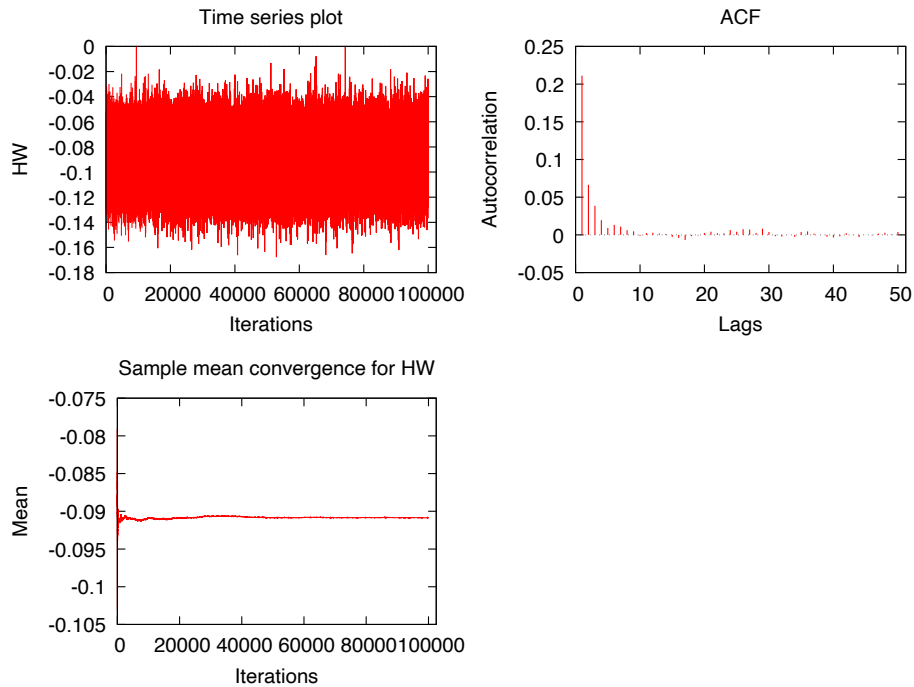


Figure 7: Sequence of sampled parameters for HW (top-left), autocorrelation function plot (top-right) and running mean plot (bottom-left) introducing within-moves.

As can be seen, results are greatly improved under all the metrics considered. Figure 7 reports the sequence plot, the ACF along with the running mean plot; graphical inspection confirms the better autocorrelation properties.

7. Conclusion

The **ParMA** package implements BMA for generalized linear models using the RJMCMC architecture in `gretl`. By doing so, it offers the practitioner a flexible tool for performing Bayesian model averaging on a range of models that are commonly applied in econometrics practice.

The package exploits the MPI architecture for parallelization, and does so in a remarkably simple and transparent way. In other words, the package aims to offer a *user-friendly* solution for parallelizing a CPU-intensive task such as the RJMCMC. Apart from the obvious benefits in terms of CPU time, this choice leads to many advantages in terms of the quality of information one can extract from the MCMC drawings, from detailed data about the posterior densities to the Brooks and Gelman statistic and other additional diagnostic statistics to assess the quality of convergence.

Computational details

The result in this paper are obtained using `gretl` in the 2020d version. Previous versions of the program are not suitable. As for the MPI architecture, Open MPI and MPICH are equally valid choices; for additional details we refer to [Cottrell and Lucchetti \(2021\)](#).

References

- Amdahl GM (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*, pp. 483–485. Association for Computing Machinery. doi:10.1145/1465482.1465560.
- Amini SM, Parmeter CF (2011). “Bayesian Model Averaging in R.” *Journal of Economic and Social Measurement*, **36**(4), 253–287. doi:10.3233/jem-2011-0350.
- Barker RJ, Link WA (2013). “Bayesian Multimodel Inference by RJMCMC: A Gibbs Sampling Approach.” *The American Statistician*, **67**(3), 150–156. doi:10.1080/00031305.2013.791644.
- Błażejowski M, Kwiatkowski J (2015). “Bayesian Model Averaging and Jointness Measures for **gretl**.” *Journal of Statistical Software*, **68**(5), 1–24. doi:10.18637/jss.v068.i05.
- Błażejowski M, Kwiatkowski J (2018). “Bayesian Averaging of Classical Estimates (BACE) for **gretl**.” *Technical report*, Università Politecnica delle Marche (I), Dipartimento di Scienze Economiche e Sociali.
- Brooks SP, Gelman A (1998). “General Methods for Monitoring Convergence of Iterative Simulations.” *Journal of Computational and Graphical Statistics*, **7**(4), 434–455. doi:10.1080/10618600.1998.10474787.
- Brooks SP, Giudici P, Roberts GO (2003). “Efficient Construction of Reversible Jump Markov Chain Monte Carlo Proposal Distributions.” *Journal of the Royal Statistical Society B*, **65**(1), 3–39. doi:10.1111/1467-9868.03711.
- Brown PJ, Vannucci M, Fearn T (1998). “Multivariate Bayesian Variable Selection and Prediction.” *Journal of the Royal Statistical Society B*, **60**(3), 627–641. doi:10.1111/1467-9868.00144.
- Cameron CA, Trivedi PK (2013). *Regression Analysis of Count Data*, volume 53. 2nd edition. Cambridge University Press, Cambridge.
- Chatfield C (1995). “Model Uncertainty, Data Mining and Statistical Inference.” *Journal of the Royal Statistical Society A*, **158**(3), 419–466. doi:10.2307/2983440.
- Chen MH, Huang L, Ibrahim JG, Kim S (2008). “Bayesian Variable Selection and Computation for Generalized Linear Models with Conjugate Priors.” *Bayesian Analysis*, **3**(3), 585–614. doi:10.1214/08-BA323.
- Chen MH, Ibrahim JG (2003). “Conjugate Priors for Generalized Linear Models.” *Statistica Sinica*, **13**(2), 461–476.
- Clyde MA, Ghosh J, Littman ML, Li Y, Van de Bergh D (2022). **BAS: Bayesian Variable Selection and Model Averaging Using Bayesian Adaptive Sampling**. R package version 1.6.2, URL <https://CRAN.R-project.org/package=BAS>.

- Cottrell A, Lucchetti R (2021). *gretl + MPI*. URL <https://sourceforge.net/projects/gretl/files/manual/gretl-mpi.pdf>.
- Cottrell A, Lucchetti R (2022). *gretl User's Guide*. URL <http://gretl.sourceforge.net/gretl-help/gretl-guide.pdf>.
- Fernández C, Ley E, Steel MFJ (2001a). “Benchmark Priors for Bayesian Model Averaging.” *Journal of Econometrics*, **100**(2), 381–427. doi:10.1016/s0304-4076(00)00076-2.
- Fernández C, Ley E, Steel MFJ (2001b). “Model Uncertainty in Cross-Country Growth Regressions.” *Journal of Applied Econometrics*, **16**(5), 563–576. doi:10.1002/jae.623.
- Fouskakis D, Ntzoufras I, Draper D (2009). “Bayesian Variable Selection Using Cost-Adjusted BIC, with Application to Cost-Effective Measurement of Quality of Health Care.” *The Annals of Applied Statistics*, **3**(2), 663–690. doi:10.1214/08-aos207.
- Frühwirth-Schnatter S, Frühwirth R (2007). “Auxiliary Mixture Sampling with Applications to Logistic Models.” *Computational Statistics & Data Analysis*, **51**(7), 3509–3528. doi:10.1016/j.csda.2006.10.006.
- Frühwirth-Schnatter S, Frühwirth R (2010). “Data Augmentation and MCMC for Binary and Multinomial Logit Models.” In *Statistical Modelling and Regression Structures*, pp. 111–132. Springer-Verlag. doi:10.1007/978-3-7908-2413-1_7.
- Frühwirth-Schnatter S, Wagner H (2006). “Auxiliary Mixture Sampling for Parameter-Driven Models of Time Series of Counts with Applications to State Space Modelling.” *Biometrika*, **93**(4), 827–841. doi:10.1093/biomet/93.4.827.
- Frühwirth-Schnatter S, Wagner H (2010). “Stochastic Model Specification Search for Gaussian and Partial Non-Gaussian State Space Models.” *Journal of Econometrics*, **154**(1), 85–100. doi:10.1016/j.jeconom.2009.07.003.
- Gamerman D (1997). “Sampling from the Posterior Distribution in Generalized Linear Mixed Models.” *Statistics and Computing*, **7**(1), 57–68. doi:10.1023/a:1018509429360.
- Garcia-Donato G, Forte A (2018). “Bayesian Testing, Variable Selection and Model Averaging in Linear Models Using R with **BayesVarSel**.” *The R Journal*, **10**(1), 155–174. doi:10.32614/RJ-2018-021.
- Gelling N, Schofield MR, Barker RJ (2019). *rjmcmc: Reversible-Jump MCMC Using Post-Processing*. R package version 0.4.5, URL <https://CRAN.R-project.org/package=rjmcmc>.
- Gelman A, Jakulin A, Pittau MG, Su YS (2008). “A Weakly Informative Default Prior Distribution for Logistic and Other Regression Models.” *The Annals of Applied Statistics*, **2**(4), 1360–1383. doi:10.1214/08-aos191.
- Gelman A, Rubin DB (1992). “Inference from Iterative Simulation Using Multiple Sequences.” *Statistical Science*, **7**(4), 457–472. doi:10.1214/ss/1177011136.
- George EI, McCulloch RE (1993). “Variable Selection via Gibbs Sampling.” *Journal of the American Statistical Association*, **88**(423), 881–889. doi:10.1080/01621459.1993.10476353.

- Geweke J (1992). “Evaluating the Accuracy of Sampling-Based Approaches to the Calculation of Posterior Moments.” In JM Bernardo, JO Berger, AP Dawid, AFM Smith (eds.), *Bayesian Statistics*, volume 4, pp. 169–193. Clarendon Press, Oxford.
- Godsill SJ (2001). “On the Relationship Between Markov Chain Monte Carlo Methods for Model Uncertainty.” *Journal of Computational and Graphical Statistics*, **10**(2), 230–248. doi:10.1198/10618600152627924.
- Godsill SJ (2003). “Proposal Densities and Product-Space Methods.” In *Highly Structured Stochastic System*, pp. 199–202. Oxford University Press.
- Green PJ (1995). “Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination.” *Biometrika*, **82**(4), 711–732. doi:10.1093/biomet/82.4.711.
- Green PJ (2003). “Trans-Dimensional Markov Chain Monte Carlo.” In *Highly Structured Stochastic System*, pp. 179–198. Oxford University Press.
- Gropp W, Lusk E, Doss N, Skjellum A (1996). “A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard.” *Parallel Computing*, **22**(6), 789–828. ISSN 0167-8191. doi:10.1016/0167-8191(96)00024-5.
- Hanson TE, Branscum AJ, Johnson WO (2014). “Informative g -Priors for Logistic Regression.” *Bayesian Analysis*, **9**(3), 597–612. doi:10.1214/14-ba868.
- Hastie DI, Green PJ (2012). “Model Choice Using Reversible Jump Markov Chain Monte Carlo.” *Statistica Neerlandica*, **66**(3), 309–338. doi:10.1111/j.1467-9574.2012.00516.x.
- Hastings KW (1970). “Monte Carlo Sampling Methods Using Markov Chains and Their Applications.” *Biometrika*, **57**(1), 97–109. doi:10.1093/biomet/57.1.97.
- Heidelberger P, Welch PD (1983). “Simulation Run Length Control in the Presence of an Initial Transient.” *Operations Research*, **31**(6), 1109–1144. doi:10.1287/opre.31.6.1109.
- Holmes CC, Held L (2006). “Bayesian Auxiliary Variable Models for Binary and Multinomial Regression.” *Bayesian Analysis*, **1**(1), 145–168. doi:10.1214/06-ba105.
- Johnson VE, Rossell D (2010). “On the Use of Non-Local Prior Densities in Bayesian Hypothesis Tests.” *Journal of the Royal Statistical Society B*, **72**(2), 143–170. doi:10.1111/j.1467-9868.2009.00730.x.
- Johnson VE, Rossell D (2012). “Bayesian Model Selection in High-Dimensional Settings.” *Journal of the American Statistical Association*, **107**(498), 649–660. doi:10.1080/01621459.2012.682536.
- Kass RE, 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. doi:10.2307/2291327.
- Lamnisis D, Griffin JE, Steel MFJ (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. doi:10.1198/jcgs.2009.08027.

- Lamnisos D, Griffin JE, Steel MFJ (2012). “Cross-Validation Prior Choice in Bayesian Probit Regression with Many Covariates.” *Statistics and Computing*, **22**(2), 359–373. doi:10.1007/s11222-011-9228-1.
- Lamnisos D, Griffin JE, Steel MFJ (2013). “Adaptive Monte Carlo for Bayesian Variable Selection in Regression Models.” *Journal of Computational and Graphical Statistics*, **22**(3), 729–748. doi:10.1080/10618600.2012.694756.
- Lucchetti R, Pignini C (2017). “**DPB**: Dynamic Panel Binary Data Models in gretl.” *Journal of Statistical Software*, **79**(8), 1–33. ISSN 1548-7660. doi:10.18637/jss.v079.i08.
- Madigan D, Raftery AE (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. doi:10.1080/01621459.1994.10476894.
- Madigan D, York J, Allard D (1995). “Bayesian Graphical Models for Discrete Data.” *International Statistical Review*, **63**(2), 215–232. doi:10.2307/1403615.
- Mroz TA (1987). “The Sensitivity of an Empirical Model of Married Women’s Hours of Work to Economic and Statistical Assumptions.” *Econometrica*, **55**(4), 765–799. doi:10.2307/1911029.
- Raftery AE, Hoeting J, Painter IS, Volinsky CT, Yeung KY (2022). **BMA**: Bayesian Model Averaging. R package version 3.18.17, URL <https://CRAN.R-project.org/package=BMA>.
- R Core Team (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.
- Rosenthal JS (2000). “Parallel Computing and Monte Carlo Algorithms.” *Far East Journal of Theoretical Statistics*, **4**(2), 207–236.
- Rossell D (2022). **mombf**: Bayesian Model Selection and Averaging for Non-Local and Local Priors. R package version 3.1.3, URL <https://CRAN.R-project.org/package=mombf>.
- Sala-I-Martin X (1997). “I Just Ran Two Million Regressions.” *The American Economic Review*, **87**(2), 178–183. URL <https://www.jstor.org/stable/2950909>.
- Sala-I-Martin X, Doppelhofer G, Miller RI (2004). “Determinants of Long-Term Growth: A Bayesian Averaging of Classical Estimates (BACE) Approach.” *The American Economic Review*, **94**(4), 813–835. doi:10.1257/0002828042002570.
- Scheipl F (2011). “**spikeSlabGAM**: Bayesian Variable Selection, Model Choice and Regularization for Generalized Additive Mixed Models in R.” *Journal of Statistical Software*, **43**(14), 1–24. doi:10.18637/jss.v043.i14.
- Schreiber S, Jensen AN (2021). **johansensmall**: Johansen Cointegration Test in Small Sample. gretl package version 3.3, URL http://gretl.sourceforge.net/current_fnfiles/unzipped/johansensmall.pdf.
- Schreiber S, Tarassow A (2020). *The multiplot Function*. gretl package version 0.2, URL http://gretl.sourceforge.net/current_fnfiles/multiplot.gfn.

- Sha N, Vannucci M, Tadesse MG, Brown PJ, Dragoni I, Davies N, Roberts TC, 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. doi:[10.1111/j.0006-341x.2004.00233.x](https://doi.org/10.1111/j.0006-341x.2004.00233.x).
- Steel MFJ (2020). “Model Averaging and Its Use in Economics.” *Journal of Economic Literature*, **58**(3), 644–719. doi:[10.1257/jel.20191385](https://doi.org/10.1257/jel.20191385).
- Vats D, Flegal JM, Jones GL (2019). “Multivariate Output Analysis for Markov Chain Monte Carlo.” *Biometrika*, **106**(2), 321–337. doi:[10.1093/biomet/asz002](https://doi.org/10.1093/biomet/asz002).
- Zeugner S, Feldkircher M (2015). “Bayesian Model Averaging Employing Fixed and Flexible Priors: The **BMS** package for R.” *Journal of Statistical Software*, **68**(4), 1–37. doi:[10.18637/jss.v068.i04](https://doi.org/10.18637/jss.v068.i04).

Affiliation:

Riccardo (Jack) Lucchetti, Luca Pedini
Department of Economics and Social Sciences (DiSES)
Faculty of Economics
Università Politecnica delle Marche
60121 Ancona, Italy
E-mail: r.lucchetti@univpm.it, l.pedini@staff.univpm.it
URL: <https://www.univpm.it/riccardo.lucchetti>