UNIVERSITÀ POLITECNICA DELLE MARCHE
Facoltà di Economia G.Fuà

DIPARTIMENTO DI SCIENZE ECONOMICHE E SOCIALI (DISES)
Ph.D. degree in Economics - XXXI cycle

Doctoral Thesis

A Multi-parallel BMA approach for Generalized Linear Models in Gretl

Author:
Luca Pedini

Supervisor:
Prof. R. Lucchetti

Abstract

The impossibility of recognizing the Data Generating Process leads to potential uncertainties in model specification that are often ignored in common selection methodologies: the model averaging approach is a promising alternative which directly deals with this issue by simply considering a quantity of interest across the model space, with the obvious advantage of avoiding possible miss-specifications or wrong conclusions induced by the choice of a single "best" model.

From a practical viewpoint, model averaging belongs to both the Frequentist and the Bayesian framework, but in the present dissertation, I will follow the second one due to its major flexibility and potentiality: in particular, a Bayesian Model Averaging (BMA) scheme based on Markov Chain Monte Carlo (MCMC) simulations, which jointly samples parameters and models, is proposed for Generalized Linear Models. The interest in Generalized Linear Models finds its motivation in Microeconometrics where, for instance, binary choice models are in common use and where the application of such techniques is still an unexplored field.

A software implementation in Gretl, via a package of functions, is then provided, having care of addressing new computational challenges, mainly the parallelization of the process via MPI (Message Passing Interface): parallelization is somehow a typical routine in standard Monte Carlo experiments, where, due to the independence of the sampling procedure, the maximal benefit in terms of CPU time gain could be achieved; however the same is not so straightforward in MCMC experiments. I will show how a simple application of parallelization in MCMCs is still useful, both in terms of a better exploration of model and parameter space and in terms of time savings. Finally, the afore-mentioned Gretl package opens the possibility of an automated procedure ready for use for the common user too, with the implicit recommendation of "reading carefully the package leaflet".

The economic implications of model averaging are explored in a Treatment Evaluation problem with Propensity Score matching: model averaging is commonly used in forecasting problems with linear models, where the driving-idea of producing an estimate which balances the ones of each specification (properly weighted) leads to more robustness with respect to "guessing" any single one. In Propensity Score evaluation the choice of which variables should be included is often ignored, and the consequent specification could be determinant in the final treatment effect estimation; a similar argument can be, therefore, applied: could model averaging be profitable in the Propensity Score definition instead of guessing which variable should be included?

I will investigate, as empirical illustration, the economic effect of tax rebates on consumption, using as case study the 2014 Italian income tax rebate, which introduced an increase in individual monthly salary of 80€ to employees. A dataset from

the “Survey on Household Income and Wealth” (SHIW) held by the Bank of Italy is built and three different techniques concerning model averaging are compared: a first one, which uses the model averaged posterior mean of the parameter of interest to build the Propensity Score, and then performs matching and treatment evaluation in the Frequentist manner; a second one averages the Frequentist treatment effects across the different models recognized by the BMA procedure, using as weight the posterior model distributions; and finally, a fully Bayesian procedure in which each sampled parameter by the BMA procedure defines a propensity score used to derive a model specific treatment effect, which is then aggregated according to the model probabilities. Matching is performed via pairwise nearest neighborhood under different caliper and different data order: the last two BMA methodologies balance the estimates between different specifications, leading to a treatment evaluation less affected by the discretion in the choice of which variables include in the Propensity Score definition; the first proposed BMA technique, instead, does not always guarantee this condition. Moreover, it manifests a higher variability across different matching set-ups, as opposed to the fully Bayesian method which shows the highest robustness taking into account an additional source of uncertainty: the propensity score distribution.

Acknowledgments

I believe that words are imperfect means for perfect purposes, they can convey wonderful ideas, but in the end they are not complete since they are affected by interpretation, any word has a certain degree of vagueness.

However, words in their uncertainty, can express emotions: a good writer (not my case) would keep the reader stuck in his story; a good storyteller would charm people with his speeches; a good philosopher would make people wonder about what he has said.

The word is an art which, in its incompleteness, is a channel for that secret chamber inside each one, which hides different thoughts, different memories, different sensations: those aspects which are fragments of a whole puzzle, in which every individual has its own dimension.

For this reason, I would like to use, here, some words to thank those people who think have contributed to my life so far, and especially to these last three years (only three?!) of PhD.

I would start by thanking my family, for all the support provided so far, even though we have come across bad and good moments, I can always count on you, my dear family.

I would thank, then, my "big and little" brother Federico: despite the distance and the different routes we have decided to follow, it is like we are the same little boys we used to be.

My thanks also goes to all the people, friends, that I have met during my university studies and have decided to be by my side till now, thank you Silvano and Elisabetta.

I thank my colleagues, and in particular Andrea, Diego, Gloria, Raffaele, Sabrina, Thomas and Visar for every single discussion, fun moment that we had. Special mentions to Emanuele, Chiara and Francesco: I would like to thank Emanuele for the time we worked in the same "office" and for every single chat we had.

As regards Chiara and Francesco, probably this whole section would not be enough to express my sincere gratitude since their contribute is far beyond what common words can say: they have accompanied me in every step, they have "put up with" my craziness and together we have shared a lot of experiences. It is quite rare to find such good companions, and I sincerely hope that we will travel together for a little bit longer.

Last, but not the least, I would like to thank my supervisor and mentor Jack: if I had not attended his Econometrics class seven years ago, I would have had another job now, probably much more boring. Thanks to him, I have appreciated the craftsmanship, which lies inside the world of Econometrics, and thanks to him I have known the Gretl project. He is truly an inspiration to me.

Thanks to you all,
Luca

Contents

Abstract **3**

Acknowledgments **5**

1 Literature review **9**

1.1 Introduction . . . 9
1.2 Selection . . . 10
1.2.1 Selection based on Information . . . 12
1.2.2 Selection based on Testing . . . 14
1.2.3 Model Confidence Set . . . 16
1.2.4 Shrinkage regression and LASSO type estimators . . . 17
1.3 Model Averaging . . . 19
1.3.1 Frequentist Model Averaging . . . 20
1.3.2 Bayesian Model Averaging . . . 26
1.3.3 BMA in practice . . . 36

2 RJMCMC and Parallelization in Gretl **39**

2.1 Statistical background: GLMs . . . 40
2.2 BMA in GLMs: a brief overview . . . 43
2.3 Reversible Jump Markov Chain Monte Carlo . . . 44
2.3.1 An automated RJMCMC for GLM . . . 46
2.3.2 Prior choices and how to approximate μ and V . . . 47
2.3.3 The RJMCMC "in a nutshell" . . . 48
2.4 Parallel MCMCs . . . 49
2.4.1 Convergence in parallel . . . 51
2.5 The package . . . 54
2.5.1 The public function . . . 54
2.5.2 Private functions: the model ID set-up . . . 56
2.5.3 Private functions: the RJMCMC . . . 57
2.5.4 Private functions: the model dynamics . . . 57
2.5.5 MPI implementation . . . 58
2.5.6 A brief analysis of currently available RJMCMC and BMA packages . . . 60
2.6 Empirical example . . . 62
2.6.1 The importance of being parallelized . . . 66
2.6.2 Summary . . . 67

3 A BMA analysis of Propensity Score Matching 69
3.1 Introduction 69
3.2 Propensity Score matching and uncertainty 71
3.2.1 A quick overview on Propensity Score 71
3.2.2 The Bayesian Treatment evaluation 72
3.2.3 Model Uncertainty issues 74
3.3 Empirical analysis 76
3.3.1 The 80€ tax credit and some economic background 77
3.3.2 Dataset and a first analysis 79
3.3.3 A Bayesian Model Averaging counter-analysis 83
3.3.4 More checks 87
3.3.5 An additional experiment 91
3.3.6 Summary 93

Conclusion 95

Bibliography 97

Appendices 109

A Balancing properties 111

B RJMCMC output for the miss-specification example 115

C Parallelization in the tax rebate example 117

D RJMCMC algorithm - the package 119

Chapter 1

Literature review

1.1 Introduction

Economists have inherited from physical sciences the myth that scientific inference is objective, and free of personal prejudice. This is utter nonsense. All human knowledge is human belief; more accurately human opinion. What often happens in physical sciences is that there is a high degree of conformity of opinion. When this occurs, the opinion held by most as asserted to be an objective fact, and those who doubt it are labeled as "nuts". [...] The false idol of objectivity has done great damage to economic science.

E. Leamer, 1983

By using these words Edward Leamer cast doubt on most empirical works of the time: Economics was apparently following the right route, but in the end, it was just wandering into the fog: as is often the case, practitioners had in mind a representation of the reality, which, in theory, should have helped to simplify the understanding of the world, but in practice was advocated as an absolute truth.

This misleading behavior was mainly due to two different tendencies: a theory-driven one where empirical results were secondary and on the other hand, just the opposite, a data-driven approach. The former could easily lead to a dogmatic and aseptic scenario where theory is totally disconnected from reality; whereas the latter is what Leamer would address as Sherlock Holmes inference (Leamer, 1974, 1983), where the detective weaves together all pieces of the evidence into a plausible story, what is inferred without data is considered biased and finally the "culprit" confess! The consequence is the impoverishment of the Science, with even worst effects when this is translated into action.

It is quite clear how both approaches are not antithetical but should coexist and a fair compromise is the equilibrium between what can be inferred from data, what data really say and what the scientist has in his mind. Science is a public process, where ideas are continuously judged; nothing, in general, is eternal just because is human thought[1]: if exceptions are initially regarded with suspicion, when many appear, a shift in knowledge (Kuhn, 1962; Lakatos, 1976) occurs which reshapes previous thoughts. The main ingredient and, at the same time, the main output of the whole process is the notion of *model*. A model is a simplification of the

[1]Leamer (1974).

world, a caricature that cannot say the whole story in every detail, but can help in understanding the main drivers of a phenomenon thanks to its simplicity, in a similar fashion as a geographical map leads to a destination even though it does not represent the real space[2].

This line of reasoning becomes particularly interesting in Economics, where there is no room for a correct experimental design like in physical sciences: the object of interest is the human behavior whose determinants are variable, numerous and interdependent. A model in Economics still has a *raison d'être*, but it should be handled much more carefully, because of its complex nature: we have just partial information and the Data Generating Process (DGP) is even more mysterious than in other cases. When we hear about the bad consequences of econometric models we should ask whether the motivation lies in this miscalculation: to quote Hendry (1980), for many years economists have been proud of the discovery of some philosopher's stone, but history teaches that transmuting metal into gold is a taboo.

Starting from this premise the correct way of facing the problem is not surely the fear of acting, but it should be the fear of not acting properly: in order to let Economics gain a "general" and "resonated" consensus, we should consider our "experimental" design accurately, the context, the analytical tools, their motivations and assumptions as expressed by Angrist and Pischke (2010).

What I will present in this Chapter is a brief overview of the main techniques of model building, in particular: Section 1 is concerned with the *model selection* approach, where, given a set of rival specifications the interest is in finding the best one according to some criteria; in Section 2, I will consider a totally different view: *model averaging*. The idea behind is that choosing a single model implies many uncertainties (Chatfield, 1995), so it could be a reasonable alternative calculating our interest quantities over a set of specifications, weighted according to the probability of each model to be "true".

A final remark is necessary: model selection will be mainly analyzed from a Frequentist perspective, whereas model averaging from a Frequentist and Bayesian one; this choice can be motivated by the fact that Bayesian selection is extremely close, in the methodologies used, to the model averaging one, so I will postpone any comments in that Section.

1.2 Selection

The model building process could be thought, in its simplest form, as a reply to three main questions:

- Which kind of relationship exists between the variables?
- Which variables are important?
- How to make proper inference once answered the previous ones?

Using mathematical notation, we can represent a model as:

$$y = f(X) + \varepsilon$$

[2]See Magnus and Vasnev (2007).

where y is the dependent variable, X the matrix of independent variables, ε the error term, and $f()$ a generic function, such as a linear one (linear model).

Our set of questions could be reformulated in: how to choose f, which elements of X are relevant and how we can use the whole expression to make, for example, forecast or further analysis.

Unfortunately, the first topic will remain mostly unexplored in this work, not because of its lack of importance, but because some route choices have to be made; hence we will assume the functional form $f()$ as known and fixed for each specification: sometimes, actually quite often, this is synonymous with *nesting*, which actually leads to the second question, variable selection.

In this case, two key notions seem to be *parsimony* and *encompassing*. Let us suppose to choose between a general and a restricted model, i.e. obtainable omitting variables from the general one, regardless of the choice we incur in the so-called *bias-variance trade-off*: the more variables we use, the less we are prone to omit relevant information but at the potential cost of less efficiency (bigger variance); conversely if we use few regressors we may face the opposite problem, biases[3].

The concept of parsimony tries to balance the two; a useful analogy could be the Occam's razor principle: focusing only on the relevant information and shaving every useless repetition.

Encompassing is the tool used to reach this equilibrium: it is said of a model to encompass another one if it coveys the same amount of information, with the implicit consequence that a small specification encompassing a bigger one is a more appropriate choice (parsimony); the problem is how to establish this condition and following both the literature and the common routines applied in practice, the two major guidelines are surely Information measures and testing.

What is implicitly assumed throughout the thesis is the adherence to the so-called $\mathcal{M}$-*closed* framework: in the Bayesian literature (Bernardo and Smith, 1994), model comparison problems can be distinguished into three different categories, the $\mathcal{M}$-closed, the $\mathcal{M}$-complete and the $\mathcal{M}$-open settings. The $\mathcal{M}$-closed one assumes that the DGP is included in the model space, i.e. the set of models of interest, and it represents the predominant assumption in both model selection and model averaging works. However, the underlying idea is quite restrictive, since DGP can be too complex or totally unknown, so more general perspectives, i.e. the $\mathcal{M}$-complete and the $\mathcal{M}$-open settings, could be necessary. In particular, the $\mathcal{M}$-complete framework assumes that the DGP is known, but is too complex to be implemented and so it is excluded from the model space. In this way every specification is seen as an approximation which tries to balance accuracy and simplicity. The $\mathcal{M}$-open setting is even more extreme, as it postulates either the totally ignorance of the DGP or the knowledge of it, but the impossibility of obtaining any consistent approximation[4].

Despite the fact that this distinction has been enlarged to embrace the Frequentist

[3]This is actually much more complicated: as explained in De Luca et al. (2018), adding more variables does not always mean reducing the bias of the parameter of interest, since this is only valid when the general model *coincides* with the DGP. When restricted and general models are both *underspecified*, i.e. smaller that the real DGP, adding more variables does not mean necessarily improving the omitting variable bias. Conversely, bigger specifications do not always lead to less efficient estimates: in homoskedastic linear models, restricted regression coefficients will be more efficient than the general counterparts, but the condition is no more guaranteed outside this context.

[4]In practice, the $\mathcal{M}$-complete framework is often included in the $\mathcal{M}$-open one.

perspective, the $\mathcal{M}$-complete and $\mathcal{M}$-open assumptions are extremely challenging, even though some recent works have started to be concerned with them (Hoeting et al., 1999; Clyde and Iversen, 2013; Lu and Su, 2015; Zhang et al., 2016; Ando and Li, 2017); as a consequence any discussion related to these settings will be avoided and the $\mathcal{M}$-closed framework is assumed unless otherwise specified.

Moreover, the general set-up assumes less regressors than observations (despite the brief overview in subsection 1.2.4), so big data analysis is not of primarily concern, but will be of course covered in future developments.

1.2.1 Selection based on Information

The idea behind the Information-based approach starts with the so-called Kullback-Leibler (KL) information measure (Kullback and Leibler, 1951): given an unknown density function f and an approximation $g(|\theta)$ which depends on some parameters θ, the KL measure is defined as[5]:

$$KL(f,g) = \int f(x)[\log f(x) - \log g(x|\theta)]dx = E_f[\log f(x)] - E_f[\log g(x|\theta)] \qquad (1.1)$$

Equation (1.1) represents the discrepancy between the theoretical distribution and its approximation: the impossibility of knowing f (hence its expected value $E_f[\log f(x)]$) and the parameter θ, which needs to be estimated, prevents us to employ (1.1) directly. This turns out to be a minor issue, because the element $E_f[\log f(x)]$ simplifies when comparing different models based on the same distribution, as in the following example

$$\begin{aligned} KL(f,g_i) - KL(f,g_j) &= E_f[\log f(x)] - E_f[\log g_i(x|\theta)] - E_f[\log f(x)] + E_f[\log g_j(x|\theta)] \\ &= E_f[\log g_j(x|\theta)] - E_f[\log g_i(x|\theta)] \end{aligned}$$

where g_i and g_j correspond to two different model specifications.

We should, thus, restrict the attention to $E_f[\log g(x|\theta)]$. In this case, we cannot directly compute this quantity because the expectation operator depends on f. However we could overcome this obstacle, by simply moving to $E_y[E_f(\log g(x|\theta))]$, where y denotes the data.

It can be proved, following Akaike (1998), that the log-likelihood of the selected model $g(x|\theta)$ evaluated in its maximum $\hat{\theta}_{ML}$, i.e. $\log p(y|\hat{\theta}_{ML}) = l(\hat{\theta}_{ML})$ plus a correction term asymptotically equal to the number of regressors k, is an unbiased estimator of this quantity:

$$E_y[E_f(\log g(x|\theta))] = l(\hat{\theta}_{ML}) - k$$

this leads ultimately to the celebrated *Akaike's Information Criterion*:

$$AIC = -2l(\hat{\theta}_{ML}) + 2k \qquad (1.2)$$

Given a set of concurrent models, the AIC identifies the loss given by the KL divergence, so hypothetically the specification which minimizes it, is the most adherent to $f(x)$. From Akaike's discovery, several aspects of the Information approach

[5]Notice that f does not depend on parameters. According to the original framework: "parameters are human concepts" (Kullback and Leibler, 1951).

were questioned: the *implicit* assumptions that among the set of models the correct one exists or the asymptotic nature of the results. The former was dealt by Takeuchi (1976) with the derivation of a possible generalization, often named nowadays as *Takeuchi's Information Criterion* (TIC, in brief). The second issue, instead, was partly addressed by Sugiura (1978); Hurvich and Tsai (1989) who derived a corrected version of AIC (AIC_c) when the sample is "relatively" small in relation to the regressor number[6]. Both of them were characterized by the adjustment of the penalized term $2k$, which conducts to a even more general representation of Information Criteria:

$$IC = -2l(\hat{\theta}_{ML}) + p(n,k) \tag{1.3}$$

where $p(n,k)$ represents a penalty function depending on the sample size n and the number of regressors k. However, only Akaike's IC and the two aforementioned extensions (TIC, AIC_c), following Burnham and Anderson (2003), are Information Criteria per se, since they directly derive from the "Information theory"; a wide and quite acknowledged category composed by the Bayesian IC (BIC, Schwarz (1978)) or the Hannan-Quinn IC (Hannan and Quinn, 1979), despite the fact that they can be obtained from (1.3), convey a different information. Consider as an example the BIC:

$$BIC = -2l(\hat{\theta}_{ML}) + k \log n$$

In Burnham and Anderson (2003), the BIC is categorized as a *dimension consistent* Information Criterion, which implies a balance between the loss given by the KL measure and the number of regressors, assuming that the fewer variables we use the better our approximation is. In other words if we assume a set of different specifications, a model which is favorite by the AIC, could be easily outperformed by another one according to the BIC if the difference in the KL is small and has fewer variables.

From a historical point of view, moreover, the BIC arises in Bayesian Statistics as a crude approximation of the posterior model probability, once a non-informative prior is assumed. We will deepen these definitions later, for the moment it is enough to state that by posterior model probability $P(M_i|y)$, where y identifies the data and M_i a specific model in a model space $\mathcal{M}$, we mean the probability of the selected specification to be the true one, after our initial idea provided by the prior is updated through the data. In particular, following Raftery (1995):

$$P(M_i|y) \approx \frac{\exp\left(-\frac{1}{2}\Delta BIC_i\right)}{\sum_{j=1}^{M} \exp\left(-\frac{1}{2}\Delta BIC_j\right)}$$

where $\Delta BIC_i = BIC_i - BIC^*$, with BIC_i as the BIC of the *i-th* model (M_i) and BIC^* as the minimum BIC obtainable in the set $\mathcal{M}$.

It is quite clear how, at least from this perspective, the Bayesian Information Criterion has little to do with the KL measure in its original formulation, but actually, an artificial relationship can be built thanks to the general formula (1.3).

[6]The thumb rule is $n/k < 40$ where n and k are respectively the number of observations and of regressors.

1.2.2 Selection based on Testing

The second, and probably most known, approach for undertaking model selection is *testing*, i.e. verifying if a parameter is significantly different from zero through a simple *t-test* or *F-test*[7].

Despite its simplicity, many drawbacks can be found. The main one, in order of importance, is inferential: as noted in many works such as Chatfield (1995); Magnus and Durbin (1999) when a specification is selected, the inference is conditioned on that one, a fact that is quite often ignored. It can be shown that the conditioning is likely to imply biases in the estimator (if the selected model is not "true" or at least it is not a good approximation of the DGP), hence all derived conclusions could be misleading.

Another theme that happens to be crucial is the reuse of the same data set: virtuous practice implies the division of the sample into three different sets, a training set for estimation, a validation and testing one for verifying the fit. When the same data are repeatedly exploited for the same scope such as iteratively testing some hypothesis, i.e. *data mining*, spurious relationships may appear. Using an example illustrated by White (2000), let us suppose to send to a large number of individuals a free copy of a stock market newsletter: the group, in particular, is split into two on the basis of the type of the news, on the one hand an upward prediction, on the other a downward one. The next week only the group who receives the correct prediction will have a new copy, and the sub-division is then repeated. After several months there can still be a rather large group who have received perfect predictions, and who might pay for such "good forecasts", but in fact, this is merely a coincidence!

Actually, the data mining problem deserves a closer look: especially in recent years, a renewed interest in the theme has been brought into attention thanks to the so-called *machine learning* techniques. Following these procedures, it is possible to build algorithms which, given a *big* dataset, can recollect the "correct" pattern among variables (a model) adjusting iteratively their output to the desired level of goodness of fit. It could appear a bit confusing considering data mining positively given the previous argument, but "carving" information in the same dataset could not be so negative if some conditions are met, such as high dimensionality of the sample.

A similar thesis is shown in our context of testing by White (1990): for a given set of models and a battery of tests, the more the sample size grows (toward infinity) and the smaller test sizes are employed, the more likely tests will select the correct specification. Thus type I and type II errors fall asymptotically to zero, and only the true specification survives[8].

Therefore, stepwise procedures, i.e. testing iteratively if a variable is significant or not, could make sense provided asymptotic theory could apply and some further regularities are met: two common views are specific-to-general and general-to-specific.

Specific-to-general starts with an essential specification, quite often the null model with the only constant as regressor, and then adds a variable at each step: if the addition improves the fit according to a goodness measure or simply through

[7]As a matter of fact there are also alternatives such as the more general *Wald test*, or the other Likelihood based tests, such as the LM or LR tests.

[8]It is worth noting, however, that the bias issue illustrated at the beginning of the section still persists and could be quite troublesome.

a comparison test (F or W) between the initial model and the new one, then the new variable is retained and the procedure is repeated. General-to-specific works in the opposite direction: a wide model (GUM, Generalized Unrestricted Model) is simplified till every remaining variable is rejected to be discarded.

PC-GETS and Autometrics

General-to-specific has known a very interesting development towards automatization in recent years: in case of linear or Augmented Distributed Lag (ADL) models it is possible to build a stepwise procedure with some interesting features known as PC-GEtS[9](Hendry and Krolzig, 1999).

The differences with respect to the standard procedure mainly lie in[10]: *a*) the validation of some desirable property on the residuals of each model (*congruence*[11]), which, if not verified, prevents the deletion of the variable until the condition is restored; *b*) the dynamics of the deletions.

The removal of a regressor is analyzed through a *multi-path search*: every irrelevant variable defines a deletion path in which the stepwise regression is separately held together with the congruence analysis and where each initial model has the related insignificant regressor omitted by default. With k irrelevant variables, multi-path search produces k final models that could be different, in which case the final choice is driven by the minimum BIC.

Deletions are performed via simple *t-test* or *F-test* in case of blocks of variables; moreover, in order to enhance the whole procedure, two common extensions are: *pre-search*, which runs initial significance tests with particularly high size to remove highly insignificant variables; and *post-search* (Hendry and Krolzig, 2004) which, in turn, implies the repetition of the multi-path search or a simplified version when different final specifications are obtained before advocating the use of BIC.

Automated GEtS possesses some interesting and potential pros such as consistency in model selection if all assumptions are met (Campos et al., 2003) and a possible extension if there are more variables than observations (Doornik, 2009b), but several cons too, which can be split into computational ones and statistical ones. In the first category we find the useless repetitions of the same models during multi-path search and the continuous congruence testing. Both of them are, in fact, dealt in an evolution of GEtS, *Autometrics* (Doornik, 2009a). Autometrics is based on the same principle with the not negligible difference of implementing a *tree structure search*, where from the initial GUM several branches, corresponding to models with an insignificant variable less, are produced via testing until the whole model space is analyzed, having care of skipping or pruning already encountered specifications: if a deletion test is rejected and a model could not be furthermore simplified it is named as "terminal". Only terminals are tested for congruence: if this condition is not met, the model is discarded and we backtrack to its first parent which respects the criterion. All these models are then collected and the procedure repeated until only relevant variables are held or the most simplified congruent model is achieved.

[9]PC-Gets is actually the name of the package which implements this procedure in the software Ox.

[10]See Granger and Hendry (2005) for further details.

[11]It is a said of a model to be congruent if its residual are homoskedastic (both conditionally and not), not autocorrelated and normal.

The statistical drawbacks, excluding the common stepwise issues[12], concern primarily the so-called cost of inference and cost of search, which, respectively, identify the problem of detecting relevant variables and of retaining insignificant ones. As a matter of fact, when a variable of interest has a near-zero value coefficient, the stepwise procedure struggles to retain it (Castle et al., 2013), while in the case of cost of search, unless we imply a strict significant test size α, we will always have an expected number of adventitious variables equal to $\bar{k} = \alpha k$ where k is the number of irrelevant covariates.

Finally, another aspect should be considered, the lack of flexibility: despite Autometrics seems to work even in more general context than linear models, this opportunity does not seem fully explored; moreover when the congruence conditions are not met, it is not really clear how to proceed: it is possible to lower the tolerance (the size of the tests) but the properties of the whole procedure are not guaranteed anymore. For these reasons some alternatives are worth of interest.

1.2.3 Model Confidence Set

A quite different approach to selection, even though it is often categorized as a selection one, is the *model confidence set* (MCS) by Hansen et al. (2011). A very elementary introduction could be the following: building a set of models that contains the best one with a certain probability according to some criteria, in a similar way like the usual confidence interval works for estimators.

As a matter of fact, MCS aims to construct a set of plausible models with similar explicative power, so in general its definition becomes highly dependent on the data availability: with less informative data it is likely to have a wide model set (sometimes correspondent to the whole model space), whereas in the opposite case a small one[13].

The formal description starts with a set of concurrent models $\mathcal{M}_0$, whose elements are checked via an *equivalence test* $\delta_{\mathcal{M}}$ for their equal "goodness". When $\delta_{\mathcal{M}}$ is rejected, the equivalence is not verified, so an *elimination rule* $e_{\mathcal{M}}$ deletes the bad performance models; the procedure is repeated until the first acceptance[14] occurs. The individual specifications inside $\mathcal{M}_0$ are evaluated in terms of a loss function L_{M_i}[15], where M_i identifies the *i-th* element of $\mathcal{M}_0$. The losses are then mutually compared using $d(M_i, M_j) = L_{M_i} - L_{M_j}$ which is the main ingredient for the equivalence test; it is possible to state the null hypothesis as:

$$H_0 : E(d(M_i, M_j)) = 0 \quad \forall\, i, j \in \mathcal{M}_0$$

where $E()$ identifies the expected value operator. The whole algorithm is summarized as follows:

1. Define the set $\mathcal{M}_0$ and calculate $d(M_i, M_j) = L_{M_i} - L_{M_j}$ for each element;

[12]The afore-mentioned biases and small sample problems.

[13]This feature opens the possibility of *model uncertainty* (Chatfield, 1995) as we will see more accurately in the next section.

[14]The authors underline the similarity of the MCS procedure with the trace-test for selecting the rank of matrix such as Anderson (1984); Johansen (1988). Notice, moreover, that the problem of sequential testing, in this case, is avoided because the procedure is halted when the null is accepted.

[15]As an example if y is the dependent variable and $\hat{y}$ its forecast, the loss is any function of their difference.

2. Test H_0 as above, via the equivalence test $\delta_{\mathcal{M}}$ with size α;

3. If the null hypothesis is accepted, stop the precedure and define $\mathcal{M}_0 = \mathcal{M}^*_{1-\alpha}$ where $\mathcal{M}^*_{1-\alpha}$ is the Model Confidence Set for $1-\alpha$; otherwise use the elimination rule to reduce $\mathcal{M}_0$ and repeat from Step 2.

As for the elimination rule we can assume many different formulations, as long as models contributing the most to $E(d(M_i, M_j))$ i.e. the worst ones, are erased firstly; what should be verified is the so-called *coherency* between equivalence test and elimination rule. The coherency condition guarantees that the probability of our final model set asymptotically tends to the selected confidence $1 - \alpha$ and from a practical point of view, it assures to avoid anomalies which can emerge from absurd combinations of $\delta_{\mathcal{M}}$ and $e_{\mathcal{M}}$.

Finally, in Hansen et al. (2011) we found a disclosure of some common examples of equivalence tests and elimination rules too: some interesting ones can be mutated from the common quadratic-form test or t-test, whereas another category, based on linear regression models, is based on a likelihood framework which links MCS to the Information Criteria.

Except for the quadratic-form equivalence test which has a χ^2 distribution, the other ones do not have a standard one, so bootstrap methods are required with a slight modification to the original framework[16].

1.2.4 Shrinkage regression and LASSO type estimators

The availability of bigger and bigger datasets, which turns to be a major theme in recent years, poses new questions and challenges to variable selection problems, in particular how to handle situations of few significant variables among a huge amount of not significant ones (*sparsity*), where there are potentially more regressors than observations.

Under this second scenario traditional least squares methods become problematic; consider as an example a linear model $y = X\beta + \varepsilon$ where y is the dependent variable, X a matrix of regressors and ε the error term; then the solution of the optimization problem

$$\min\,(y - X\beta)^T(y - X\beta)$$

is not unique anymore[17].

But even assuming less variables than observations in a big dataset with many regressors makes common selection routines unfeasible: quite rarely coefficients are set to zero during the estimation process so testing is strongly required, but as we have seen, spurious relationships may be included and biases can heavily undermine each conclusion. Information Criteria are equally troublesome, since all models need to be investigated, leading to a computational complexity which grows exponentially in the number of regressors (with k variables, 2^k models are available).

[16]It should be noted how de facto the quadratic-form has little application in practice compared to the correspondent t-test version: this is mainly due to the estimation of the variance matrix which could become easily not feasible.

[17]In this subsection I will focus on linear models, but the same argument can be easily extended to more complex ones, such as Generalized Linear Models.

A plausible solution could be imposing some *regularization* inside the estimation procedure, which, loosely speaking, corresponds to imposing a constraint on the coefficients in the optimization problem of least squares:

$$\min\,(y - X\beta)^T(y - X\beta) \quad \text{subject to} \quad f(\beta) \leq t \tag{1.4}$$

with $f()$ as a constraint function and t a given threshold.

Equation (1.4) identifies the *shrinkage regression* framework, where depending on the choice of the constraint function, different estimators with different properties are obtained. A common choice in variable selection problem seems to be the l_1 norm $\|\|_1$, which leads to the celebrated LASSO estimator by Tibshirani (1996):

$$\min\,(y - X\beta)^T(y - X\beta) \quad \text{subject to} \quad \|\beta\|_1 \leq t \tag{1.5}$$

Notice that the peculiar shape of the constraint imposes sparsity in the coefficients, so not only some parameters are shrunk exactly to zero, but also assuming the number of significant regressors *always* smaller than the number of observations, implies the uniqueness of LASSO estimator even if there are more regressors than observations. Finally the choice of the threshold t is fundamental as well because it determines the magnitude of the shrinking effect: some common methods used for choosing this value range from cross-validation to the minimization of goodness of fit measures.

It appears that LASSO is the solution to all variable selection problems, but there are many aspects that merit a closer look: hypothetically, we would like to have a procedure which classifies all zero coefficients with probability tending to one and whose asymptotic distribution for non-zero coefficients is the same as if only those significant variables are included[18]. If this is the case, the obtained estimator is defined as *oracle efficient*, but LASSO does not guarantee this condition; in fact, zero coefficients may have a probability mass at zero, but this probability is not tending to one and the non-zero coefficient estimates have an asymptotic bias.

An oracle consistent estimator could be easily obtained from LASSO by adjusting the shrinkage induced in the constraint function: Zou (2006) shows that weighing coefficients inside the penalty function by the inverse of the absolute value of their OLS estimate, produces a LASSO estimator with the oracle property (the so-called *adaptive* LASSO).

Some final remarks seem to be necessary since oracle property and adaptive LASSO have limitations: oracle property induces a problem of efficiency in the final estimator, and even if an equivalent condition for finite sample can be established, when data are not (much) sparse, the robustness of the procedure is affected. For these reasons second generation LASSO estimators (often named as debiased or desparsified LASSO[19]) have been introduced: these kind of methods simply estimate standard LASSO coefficients in a first step, and then correct for their bias in the second one; it could be proven that debiased estimators are more efficient, robust and finally are easily usable for inference too[20].

[18]Hence obtainable using simple least squares.

[19]Van de Geer et al. (2014).

[20]Remember that LASSO requires simulated methods for inference or post-model-selection ones (Belloni et al., 2011).

1.3 Model Averaging

Model building, as we have seen so far, is often concerned with the selection of an optimal model in order to explain data at best: the main issue is that, in practice, we do not know exactly the Data Generating Process (DGP), so we cannot have the certainty to choose the correct specification (*model uncertainty*); and if we still want to select one, we are prone to make inference conditional on this selection, often incurring in biases and, sometimes, in even worst consequences[21].

Possible solutions are the aforementioned Model Confidence Set approach or shrinkage methods but in the model selection literature, an appealing alternative, which takes into account both previous problems, is known as *model averaging*[22]. The idea of model averaging is quite simple: considering the estimates across many specifications, instead of focusing only on a single one and building the related interest quantity (e.g. a simple estimator or a forecast) as an averaged measure over the model space, i.e. the set containing models.

In mathematical terms:

$$\begin{gathered} f(\beta)_{AV} = \sum_{i=1}^{M} f(\beta_i)\omega_i \\ 0 \leq \omega_i \leq 1 \\ \sum_{i=1}^{M} \omega_i = 1 \end{gathered} \tag{1.6}$$

where β is the parameter of interest, $f()$ a generic functional form, β_i and ω_i are respectively the parameter and the weight attached to the *i-th model*, where we assume that our model space $\mathcal{M}$ contains M different specifications.

The advantages of considering more than one model allow precisely to not ignore information that sometimes (actually quite often) the selection of a unique model excludes and that can be useful.

For this reason model averaging in Economics has quickly enjoyed popularity in recent years: in economic growth literature, for example, model averaging techniques have led to a much more complete and balanced inference about the growth determinants, with particular reference to the role of geography, integration and institutions, which were only partially analyzed in previous works. Some example of such model averaging literature are Brock and Durlauf (2001); Fernandez et al. (2001b); Sala-i Martin et al. (2004); Eicher and Newiak (2013); Lenkoski et al. (2014); Durlauf (2018).

Koop et al. (1997), instead, exploits the use of Bayesian Model Averaging in autoregressive (fractional) integrated moving average (ARIMA and ARFIMA) models for evaluating the real GNP; Cogley and Sargent (2005) consider the use of model averaging for analyzing inflation; Strachan and Van Dijk (2013) explore Bayesian Model Averaging methods with dynamic stochastic general equilibrium (DSGE) models, whereas George et al. (2008); Koop (2017) propose application of model averaging

[21]Chatfield (1995) is a wonderful reference point for understanding the pros of model uncertainty.

[22]As a matter of fact model averaging can be considered as a shrinkage method too, at least in a broad sense. This point will be clear soon.

to Vector Autoregression Models (VAR) [23]. In Microeconometric literature some examples are Van den Broeck et al. (1994) which deals with uncertainty in production modeling via stochastic frontier models or Sickles (2005) which examines firm inefficiency through Bayesian Model Averaging.

What remains to be explained is how to deal with model averaging; in particular, two main perspectives are available: on the one hand, the Bayesian viewpoint and on the other the Frequentist one. The former focuses on the posterior distribution of the parameter, so our $f(\beta)$ is a density or a probability distribution; the latter, instead, simply replaces $f(\beta)$ with the (Frequentist) parameter estimator, let us say $\hat{\beta}$. The weight ω_i represents in both cases a measure of the probability of a model to be the appropriate one; in the Bayesian case we will use the so-called posterior model distribution, whereas in the Frequentist one some "Information Criteria" based measure.

Finally it is worth mentioning how the so-called *hybrid* methods have reached great consensus recently too. By hybrid methods we generally mean model averaging techniques which combine aspects from the Frequentist and Bayesian perspective in order to achieve a grater flexibility: the major examples are Bayesian Average of Classical Estimates (BACE) by Sala-i Martin et al. (2004), Weighted-Averaged Least Squares (WALS) by Magnus and Durbin (1999); Magnus et al. (2010) and Bayesian Averaging of Maximum Likelihood Estimates (BAMLE) by Moral-Benito (2012).

I will analyze only the strictly Bayesian and the strictly Frequentist methodologies throughout this work, postponing the analysis of hybrid methods for future developments: I will start from the Frequentist Model Averaging (FMA) framework, where the main issues are how to deal with weights[24] and how to make inference with the new averaged quantities (multimodel inference, Burnham and Anderson (2003)). I will then cover the Bayesian framework, having care of analyzing the main ingredients of the recipe even though some details will be left for the next Chapter, where a suitable scheme for Generalized Linear Model is provided[25].

1.3.1 Frequentist Model Averaging

Despite the fact that some applications of model averaging in a Frequentist sense can be dated back to Box and Jenkins (1970), a correct and rigorous dissertation of this method should be considered as a reply to some (computational) difficulties of its Bayesian counterpart. Since throughout this work I will favor the Bayesian perspective, I will try to sketch a brief description of FMA before the analysis of Bayesian Model Averaging: in doing so I will lose the chance of drawing any useful parallelism but the benefit will be a major adherence to the structure of this work.

In the FMA framework we are interested in weighing a Frequentist estimator $\hat{\beta}$ or some function of it across several specifications; just as an example consider the linear model:

[23] This is only a short overview of some applications of model averaging in Economics. For a more detailed analysis see Steel (2018)

[24] Remember that the quantity of interest is simply the estimate of the parameter, which can be obtained straightforwardly in Frequentist case.

[25] More detailed overviews on specific model averaging techniques include Clyde and George (2004); Claeskens and Hjort (2008); Wang et al. (2009); Moral-Benito (2015); Magnus and De Luca (2016); Steel (2018).

$$y = X\beta + \varepsilon$$

where y is a $n \times 1$ vector of response variable, X is the $n \times k$ matrix of explanatory variables and ε the usual disturbance term.

Using the same notation of the previous section the Frequentist correspondent to equation (1.6) is:

$$\hat{\beta}_{AV} = \sum_{i=1}^{M} \omega_i \hat{\beta}_{OLS,i}, \qquad \sum_{i=1}^{M} \omega_i = 1$$

where $\hat{\beta}_{OLS,i}$ is simply the OLS estimator of the correspondent model, where we set $\hat{\beta}_{i,j} = 0$ if the *j-th* regressor is absent from the model i.

The definition of the weight ω_i becomes crucial, since all the properties of the averaged estimator are a consequence of this choice: there is not an unique solution and the possibility to make proper inference becomes unclear due to the complexity involved in the calculation of standard errors or asymptotic distributions (Burnham and Anderson, 2003, 2004).

Some common choices are presented in the following subsubsections.

FMA with IC weights

One of the first method implied to calculate weights was due to Buckland et al. (1997): in particular recalling the definition of Information Criteria given in (1.3) and assuming the same notations as before,

$$IC = -2l(\hat{\theta}_{ML}) + p(n, k)$$

after some rearrangements and by taking the exponential operator we obtain a measure of the "penalized" likelihood:

$$\exp(-\frac{1}{2}IC) = L(\hat{\theta}_{ML}) \exp(-\frac{1}{2}p(n, k)) \tag{1.7}$$

where L is the likelihood of the model. Normalizing (1.7) over the whole model space allows us to treat it as a probability, but also as a weight of the *i-th* model:

$$\omega_i = \frac{exp(-\frac{1}{2}IC_i)}{\sum_{j=1}^{M} exp(-\frac{1}{2}IC_j)} \tag{1.8}$$

where to a higher ω_i corresponds a high probability of the model to be plausible.

For computational easiness Burnham and Anderson (2004) proposes a slightly different version[26]:

$$\omega_i = \frac{exp(-\frac{1}{2}\Delta_i)}{\sum_{j=1}^{M} exp(-\frac{1}{2}\Delta_j)} \tag{1.9}$$

with:

$$\Delta_i = IC_i - IC_{min}$$

[26]Very similar to the previous result of Raftery (1995).

where IC_{min} is the minimum of the M different IC values.

As for the choice of the Information Criterion, Akaike's Information Criterion (AIC) and Bayesian Information Criterion (BIC), with the same difference in usage as the one previously sketched in subsection 1.2.1, are the main options.

It is worth noting that a quite different alternative is provided in Hjort and Claeskens (2003) with the *Focussed Information Criterion* (FIC): while in IC model averaging the weight calculation does not consider the scope of the averaging procedure, mainly because the commonly applied IC are built for only model comparison purposes, the FIC and its derived weights are meant to take into account the "use" of the model. In other words it can easily happen that the weights chosen to be optimal for inference, cannot be equally good for forecasting or analysis of variance: in order to fulfill this task the author introduce an objective function of the parameter of interest, which embodies this "scope"[27], and define a related Information Criterion, the FIC, as asymptotic approximation of the Mean Squared Error of this function. A curious fact that deserves more attention and could affect the effectiveness of this measure is the dimensionality: it turns out that the structure of the chosen function could lead to have a multidimensional FIC (vector) for each model; in these cases its interpretation could be quite challenging and a lot depends on the context of the analysis.

FMA with Mallow's C based weight

Information Criteria weights derived from AIC or BIC provide a compromise between computational easiness and the benefits of averaging[28]; however, little is said about efficiency, i.e. the possibility of building weights so as a corresponding measure of risk is minimized[29], hence a potential most efficient estimator among the averaged ones.

A first attempt in this direction is the afore-mentioned FIC by Hjort and Claeskens (2003) which, depending on the function of interest chosen, produces a more efficient model averaging estimator than the standard IC-based ones; however, in the context of homoskedastic linear models, Hansen (2007) shows how it is possible to reach the most efficient model averaging estimator by simply choosing the weights so as the Mallows'C criterion is minimized: it is proved that Mallow's C is asymptotically equivalent to the average squared error $L(\omega) = (X\beta - X\hat{\beta}_{AV}(\omega))^T(X\beta - X\hat{\beta}_{AV}(\omega))$, where $\hat{\beta}_{AV}(\omega)$ is the model averaging estimator considered as a function of the weight ω, so the weight ω which minimizes it, defines the most efficient estimator.

Following the author, the standard linear model scenario is slightly changed: let $(y_i, \mathbf{x}_i), i = 1..n$ be a random sample with $\mathbf{x}_i = (x_{i1}, x_{i2}, ...)$ the vector of variables;

[27]A useful example is the function used for forecasting in linear model, defined as: $X\beta$ where X is the matrix of regressors, and β the parameters.

[28]Burnham and Anderson (2004) show the gains of Akaike's and BIC weights in terms of MSE compared to standard procedures.

[29]The typical example of such a risk measure is the Mean Squared Error (MSE).

the regression set assumes infinite covariates[30] so,

$$y_i = \sum_{j=1}^{\infty} \beta_j x_{ij} + \varepsilon_i \;\rightarrow\; y_i = \mathbf{x}_i^T \beta + \varepsilon_i \tag{1.10}$$

where $\varepsilon \sim N(0, \sigma^2)$ and $\sum_{j=1}^{\infty} \beta_j x_{ij}$ converges in mean-square.

Consider now $m = 1...M$ models where each one contains the first k_m elements[31] of $\mathbf{x}_i$ with $0 \leq k_1 < k_2 < ...$ and $k_m \leq n$. The *m-th* model is:

$$y_i = \sum_{j=1}^{k_m} \beta_j x_{ij} + e_i, \;\rightarrow\; y_i = \mathbf{x}_{i,m}^T \beta_m + e_i \tag{1.11}$$

where $e_i = \sum_{k_m+1}^{\infty} \beta_j x_{ij} + \varepsilon_i$ is the new error term.

In matrix notation:

$$\mathbf{y} = X_m \beta_m + \mathbf{e} \tag{1.12}$$

with:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{pmatrix}; \; X_m = \begin{pmatrix} \mathbf{x}_{1,m}^T \\ \mathbf{x}_{2,m}^T \\ . \\ . \\ . \\ \mathbf{x}_{n,m}^T \end{pmatrix}; \; \beta_m = \begin{pmatrix} \beta_1 \\ \beta_2 \\ . \\ . \\ . \\ \beta_{k_m} \end{pmatrix}; \; \mathbf{e} = \begin{pmatrix} e_1 \\ e_2 \\ . \\ . \\ . \\ e_n \end{pmatrix}$$

The typical OLS estimates are:

$$\hat{\beta}_m = (X_m^T X_m)^{-1} X_m^T \mathbf{y}$$

Therefore, the model average estimator over the set M is defined as:

$$\hat{\beta}_{AV} = \sum_{m=1}^{M} \omega_m \tilde{\beta}_m \tag{1.13}$$

where $\tilde{\beta}_m$ is $\hat{\beta}_m$ filled with zeros where regressors are excluded in the *m-th* model (to make the summation conformable)[32].

A closely related quantity is the fitted value $\hat{\mathbf{y}}_{AV}$:

$$\hat{\mathbf{y}}_{AV} = P_{X_m}(\omega)\mathbf{y}, \quad P_{X_m}(\omega) = \sum_{m=1}^{M} \omega_m P_m \tag{1.14}$$

[30]With a finite set the results are unchanged, even though it is requested in general an high number.

[31]Notice that each k_m has to be defined, Hansen (2014) suggests each group differs from the other by 4 variables, i.e. $k_m = 4m$, for further details see the related work.

[32]The impact of each β_m on the whole averaged estimator depends heavily on the order of the variables: in other words as each specification is based on the previous one plus some new variables, the first regressors are weighted in each model whereas the last one only in the final.

P_m is the projection matrix[33] $X_m(X_m^T X_m)^{-1} X_m^T$.

Hansen's result follows from the definition of Mallow's C based on the averaged estimator:

$$C(\omega) = \hat{\mathbf{e}}^T \hat{\mathbf{e}} + 2\sigma^2 K(\omega) \tag{1.15}$$

where $\hat{e} = y - \hat{y}_{FMA}$ and $K(\omega) = \sum_{m=1}^{M} \omega_m k_m$.

Taking the expectation and calling the average squared error $L(\omega) = (X\beta - X\hat{\beta}_{AV}(\omega))^T (X\beta - X\hat{\beta}_{AV}(\omega))$ we obtain:

$$E(C(\omega)) = E(L(\omega)) + n\sigma^2$$

where the following result is exploited:

$$E(\hat{\mathbf{e}}^T \hat{\mathbf{e}}) = E(L(\omega)) + n\sigma^2 - 2E(\varepsilon^T P_{X_m}(\omega)\varepsilon) = E(L(\omega)) + n\sigma^2 - 2\sigma^2 K(\omega)$$

Mallow's C is an unbiased estimate of the expected squared error plus a constant; moreover (and above all) it is possible to rewrite (1.15) as:

$$C(\omega) = \underline{\omega}^T \underline{e}^T \underline{e}\ \underline{\omega} + 2\sigma^2 K^T \underline{\omega} \tag{1.16}$$

where $\underline{\omega}$ is the $M \times 1$ vector of weights; $\underline{e}$ is a $n \times M$ matrix which collects the M residuals of each model in (1.11); K a $M \times 1$ vector of the number of parameters.

Equation (1.16) displays how the Mallow's C is a quadratic function, so that its minimization should not be problematic; in particular if we choose the minimizing vector $\underline{\omega}^* = \text{argmin } C(\underline{\omega})$, under the constraint $\sum_{m=1}^{M} \omega_m = 1$, as long as the following requirements are met:

a) $(y_i, \mathbf{x}_i)$ are i.i.d.;

b) a homoskedastic linear framework is used;

c) nesting;

d) the weight set should be restricted to a finite one of the form $\omega_i \in \{0, \frac{1}{c}, \frac{2}{c}, ..., 1\}$ for some constant c;

then, $\underline{\omega}^*$ asymptotically minimizes the squared error based on averaged estimators with respect to any other choices of $\underline{\omega} \in W$, where W is the set of weights consistent with the above constraint:

$$\frac{L(\underline{\omega}^*)}{\inf_W\ L(\underline{\omega})} \xrightarrow{p} 1$$

Conditions *a*)-*d*) have been the object of many analysis in further studies, since if on the one hand, they guarantee the final result, on the other hand, they appear a bit binding: condition *a*) is extremely difficult to be relaxed as the applicability of asymptotic theorems depends heavily on this one; *b*) can be split into two different cases: departure from linear models and heteroskedasticity. The former is

[33] Notice that $P_{X_m}(\omega)$ is symmetric but, in general, not idempotent. However it has other interesting properties, see Hansen (2007).

quite challenging too, because Mallow's C is closely linked to linear specifications, so embracing a wider perspective means finding an equivalent measure with similar properties for non-linear cases, a task still unexplored. Heteroskedasticity, instead, has been addressed directly by Hansen and Racine (2012) which proposes, following quite the same procedure as before, a robust average least squares estimator with jackknife weights (leave-one-out cross validation); and by Liu et al. (2016) with a Feasible Generalized Least Squared approach. Finally assumptions *c*) and *d*) are both modified in Wan et al. (2010) achieving similar conclusions.

Multimodel inference

Frequentist Model Averaging is primarily used in forecasting, so little attention is devoted to inference aspects: what is requested to the averaged estimator is *consistency*, a property which is easily satisfied both with IC weights and Mallow's C ones.

However, overcoming this limitation is fundamental to expand the applicability of Frequentist Model Averaging: a first attempt in this direction is the standard error estimation by Buckland et al. (1997).

Assuming that the parameter vector of interest β is common to all specifications[34] and the weights are known (not random), we can calculate the variance of (1.6) as:

$$V(\hat{\beta}_{AV}) = \sum_{i=1}^{M} \omega_i^2 V(\hat{\beta}_i) + \sum_{i=1}^{M} \sum_{j\neq i} \omega_i \omega_j Cov(\hat{\beta}_i, \hat{\beta}_j)$$

which can be rewritten as:

$$V(\hat{\beta}_{AV}) = \sum_{i=1}^{M} \omega_i^2 V(\hat{\beta}_i) + \sum_{i=1}^{M} \sum_{j\neq i} \omega_i \omega_j \rho_{i,j} \sqrt{V(\hat{\beta}_i) V(\hat{\beta}_j)} \tag{1.17}$$

where $\rho_{i,j}$ is the correlation between parameter i and j. The principal problem in (1.17) is exactly $\rho_{i,j}$: in the extreme cases of perfect correlation (less probable) or uncorrelation, we obtain simplified versions of (1.17), which respectively are

$$V(\hat{\beta}_{AV}) = \left[\sum_{i=1}^{M} \omega_i [V(\hat{\beta}_i)]^{\frac{1}{2}} \right]^2 \tag{1.18}$$

and

$$V(\hat{\beta}_{AV}) = \sum_{i=1}^{M} \omega_i^2 V(\hat{\beta}_i)$$

It is obvious that choosing one of the two is quite challenging, since it is rare to have such particular situations, so what is suggested is to consider the above formulas as simple starting points. Intermediate cases ($0 < \rho_{i,j} < 1$) are surely more interesting but require a different approach to the problem, such as simulated inference (bootstrap).

[34]This should also imply the case in which one or more parameters are excluded from the model, i.e. their coefficients are zero.

Two elements, however, are worth noting: in the original framework the variance of each parameter is substituted by their MSE, so for instance (1.18) becomes:

$$V(\hat{\beta}_{AV}) = \left[\sum_{i=1}^{M} \omega_i [V(\hat{\beta}_i) + b_i^2]^{\frac{1}{2}}\right]^2$$

where $b_i = \beta_i - \beta$ is the misspecification bias which arises in estimating β in the *i-th* model and it is calculated by using $\hat{b_i} = \hat{\beta}_i - \hat{\beta}_{AV}$.

The second element is just a refinement by Burnham and Anderson (2003): if we do not want to rely on simulated inference, the previous variance equations seem to perform badly when nesting is assumed, so an alternative version of (1.17) is provided as a possible solution[35]:

$$V(\hat{\beta}_{AV}) = \sum_{i=1}^{M} \omega_i [V(\hat{\beta}_i + b_i^2)]$$

which can also be extended into the original framework[36].

Turning the discussion to the asymptotic distribution of (1.6), it is appealing to apply the result that a linear combination of Normal variables is Normal too; nevertheless this is not always the case since a lot depends on the assumptions of randomness of the weights: Hjort and Claeskens (2003) show how, depending on the type of weights and on the framework, not standard distribution could arise and the Normal approximation becomes not so advisable. A bias corrected procedure, together with a plausible general framework for asymptotic distribution, is then derived following a Focussed Information Criterion driven perspective: this could appear a huge advance, but as a matter of fact the same problems encountered in the derivation of FIC (multidimensionality) still persist.

1.3.2 Bayesian Model Averaging

From a historical perspective Bayesian Model Averaging (Madigan and Raftery, 1994; Raftery, 1996; Hoeting et al., 1999) was the first attempt to consider the model averaging idea in a coherent and consistent framework, and, thanks to the flexibility of the Bayesian set-up, it surely has the advantage to lead to a clearer and more immediate inference about model comparisons or variable inclusions, an aspects that is not directly available in FMA. Accordingly, model averaging weights implied in FMA, are not always imputable as model probabilities, e.g. consider FIC weights which derives from an optimization procedure based on a specific chosen function[37].

It is a well known fact that in Bayesian statistics we consider the parameter of interest, let us say β, as a random variable, so talking of its distribution should not surprise; in particular we refer to its posterior distribution as $p(\beta|y)$, where y identifies the data. In math terms:

[35] Even in this case however we are using an approximation!

[36] An apparently different approach is provided in Magnus et al. (2010).

[37] AIC and BIC weights, instead, can be thought as model probabilities: Burnham and Anderson (2003), in particular, show how both ones can be represented as approximations of posterior model probabilities (see next sections) under particular conditions.

$$P(\beta|y) \propto p(y|\beta)P(\beta)$$

where $p(y|\beta)$ is the likelihood of the parameter[38], and $P(\beta)$ its prior distribution, that is our initial idea about the parameter of interest.

The correspondent model averaged quantity is thus:

$$P(\beta|y) = \sum_{i=1}^{M} P(\beta|M_i, y)P(M_i|y) \tag{1.19}$$

where M_i represents the *i-th* model drawn from a model space $\mathcal{M} = (M_0, M_1, ..)$, $P(\beta|M_i, y)$ is the posterior distribution of the parameter in model M_i and $P(M_i|y)$ the posterior model distribution. Notice, how in the Bayesian framework a model M_i is seen as a "parameter" too, in particular, it is often labeled as a categorical variable which embodies the link between dependent and independent variables.

Model averaging expected value and variance are:

$$E(\beta|y) = \sum_{i=1}^{M} E(\beta|y, M_i)P(M_i|y) \tag{1.20}$$

$$V(\beta|y) = \sum_{i=1}^{M} \big[V(\beta|y, M_i) + E(\beta|y, M_i)^2\big] P(M_i|y) - E(\beta|y)^2 \tag{1.21}$$

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

In general, we are more interested in these two last expressions, but if we can find a way to easily compute the posterior distribution too, these are obtained straightforwardly from it and the additional gain could be potentially high.

We will see in the next part, that an effective method of doing this could be *sampling*, that is obtaining a sample of parameter drawings whose distribution and moments (obtained easily by their sample counterparts) should reflect the posterior. In model averaging, however, some additional difficulties which can undermine canonical sampling schemes for the parameter, are encountered, as a consequence, a more general framework needs to be defined.

Finally the model posterior $P(M_i|y)$ is the weight attached to each element in the above equations and defines how much each specification is plausible, but, unfortunately, its computation is pretty challenging.

From this brief introduction, it could appear how BMA is extremely complex to be performed especially in comparison to its Frequentist alternative and, as a matter of fact, it is. However the potential gain in terms of higher information provided and in terms of applicability are surely more substantial than the costs.

In the following subsection, I will draw reader's attention to the two main elements of (1.19) taken each one individually; I will provide only afterwards a more complete and scrupulous disclosure on how model averaging is performed in practice.

[38]What in the previous sections was named $L()$.

Posterior parameter distribution: the linear case

Posterior parameter distribution can be obtained using either analytical formulas or sampling. In the first case a common example is the linear model:

$$y = X\beta + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2 I)$$

where y is a $n \times 1$ dependent variable, X a $n \times k$ matrix of k covariates, ε the error term and I the identity matrix.

In order to obtain the posterior $P(\beta, \sigma^2|y) \propto p(y|\beta, \sigma^2)P(\beta, \sigma^2)$, the canonical way to proceed is assuming a Normal-Inverse Gamma conjugate prior[39]:

$$P(\beta, \sigma^2) = P(\beta|\sigma^2)P(\sigma^2) = N(\mu_0, \sigma^2 V_0)IG(a_0, b_0)$$

where $\beta|\sigma^2 \sim N(\mu_0, \sigma^2 V_0)$, $\sigma^2 \sim IG(a_0, b_0)$, with $\mu_0, \sigma^2, a_0, b_0$ as the related parameters. It is a common convention to write in compact form this whole distribution as $NIG(\mu_0, V_0; a_0, b_0)$, the Normal-Inverse Gamma.

The likelihood function is defined as:

$$p(y|\beta, \sigma^2) = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{n}{2}} \exp\left[-\frac{1}{2}\frac{(y - X\beta)^T(y - X\beta)}{\sigma^2}\right]$$

where n identifies the number of observations.

The posterior, after some rearrangements can be seen as:

$$p(\beta, \sigma^2|y) \propto \left(\frac{1}{\sigma^2}\right)^{\frac{k}{2}} \exp\left[-\frac{1}{2\sigma^2}(\beta - \mu)^T V^{-1}(\beta - \mu)\right] \times \left(\frac{1}{\sigma^2}\right)^{\frac{a}{2}+1} \exp\left[-\frac{1}{2\sigma^2}b\right]$$

with:

$$\begin{aligned} \mu &= [V_0^{-1} + X^T X]^{-1}(X^T y + V_0^{-1}\mu_0) \\ V &= (X^T X + V_0^{-1})^{-1} \\ a &= a_0 + \frac{n}{2} \\ b &= b_0 + \frac{1}{2}[\mu_0^T V_0^{-1}\mu_0 + y^T y - \mu^T V^{-1}\mu] \end{aligned}$$

That is a $NIG(\mu, V; a, b)$.

In order to have the posterior on a particular model M_i, assuming nesting, all we have to do is just selecting the related variables in X and adapting priors and likelihood.

A sampling approach from the NIG posterior is equally simple: in Bayesian framework it is quite common to simulate a target distribution, in our case the posterior, especially if not of a standard form: among the various procedures ranging from standard Monte Carlo experiments to Importance Sampling, the most known for its flexibility is the Markov Chain Monte Carlo simulation. In a MCMC experiment a stochastic process governed by the Markov property is constructed such that, after

[39] A conjugate prior, once combined with the likelihood, yields a posterior falling in the same distribution family. A natural conjugate prior has the additional property that it has the same functional form as the likelihood function.

some burn-in iterations, it converges to the desired distribution. In the linear case we can use the *Gibbs MCMC*, i.e. starting from an initial value for σ^2, sample β from a multivariate Normal with mean μ and covariance matrix $\sigma^2 V$. Then sample σ^2 from a $IG(a, b)$, where the parameter b is currently updated with β. The process is then iterated, and the resulting sampled $\{\beta; \sigma^2\}$ represents the joint distribution, while each component individually provides the marginal posterior distribution.

The prior for model parameters

Prior distributions represent the information available before analyzing the data and that it could be meaningful to include as part of the posterior; despite from a theoretical point of view any plausible distribution can be considered, in practice some common choices are made in order to make computation much easier. The previous conjugate prior is an example, but for what concerns the present work whose first aim is BMA, the solution proposed by Fernandez et al. (2001a) is particularly appealing and so it will be briefly described[40].

Let us rewrite the linear model framework as:

$$y = \iota\alpha + X\beta + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2 I)$$

where we expressly exclude the constant term ι (a vector of ones) and its parameter α from the set of regressors X. This choice is easily motivated by the assumption of the constant term always included in every specification: when a parameter is common to all models, Fernandez et al. (2001a) show how an improper prior[41] can be placed on that parameter with no significant consequences to the general framework and with potential great interpretative and computational benefits.

As a consequence, they assume not only a common α, but also a common σ^2 with prior distributions:

$$P(\alpha) \propto 1$$
$$P(\sigma^2) \propto \sigma^{-2}$$

Furthermore, a diffuse improper prior on the constant term often leads to assume its independence with respect to the other regressors, a condition easily achievable by simply centering all the covariates ($\iota^T X = 0$).

Model specific parameters, β_i, instead, requires only proper priors[42], and in our case:

$$P(\beta_i | M_i, \sigma^2) \sim N(0, \sigma^2 g (X_i^T X_i)^{-1})$$

where X_i identifies the matrix of covariates included in model M_i, and $g > 0$ a parameter. Such peculiar structure is known as *Zellner's g-prior* (Zellner, 1986),

[40]As a matter of fact Fernandez et al. (2001a) propose a disclosure of different priors, but what will be presented here is probably the most used in BMA analysis.

[41]Improper priors are not actually probability distributions, as they do not integrate to unity over its domain; nevertheless they can be equally useful because they express uninformative conditions.

[42]The posterior model distribution will be not determined in case of diffuse priors of model specific parameters (Fernandez et al., 2001a; Steel, 2018).

where the covariance matrix "reproduces" the one of a standard OLS estimator. A common alternative could be the so-called *ridge prior* covariance, cI, with I as the identity matrix and $c > 0$ as a shrinkage parameter: it implies the *a priori* independence of coefficients and it tends to reduce the impact of the likelihood; however for the moment I will draw reader's attention to the Zellner's prior following Fernandez et al. (2001a).

Under this peculiar set-up, β_i becomes the main parameter of interest and its posterior distribution is given by a Student t, with respectively posterior mean and variance as:

$$E(\beta_i|M_i, y) = \frac{g}{1+g}\hat{\beta}_{OLS,i}$$
$$V(\beta_i|M_i, y) = \frac{(y-\bar{y}\iota)^T(y-\bar{y}\iota)}{n-3}\frac{g}{1+g}\left(1-\frac{g}{1+g}R_i^2\right)(X_i^T X_i)^{-1}$$

where $\hat{\beta}_{OLS,i}$ is the OLS estimator for model M_i, $\bar{y}$ is the mean of the dependent variable, n the number of observations and R_i^2 the OLS R-squared for model M_i.

It is quite clear how the choice of the scalar g is determinant since it works as a regularization parameter: a high value of g is associated with a higher impact of the likelihood function in the posterior, whereas a small value favors the prior distribution impact.

Two common approaches used for defining g are to choose a fixed value which reflects some desirable properties or to place an *hyperprior*. The first one is often more convenient, even though its real contribution should be analyzed with care, because some fixed choices of g can lead to counterintuitive results in model comparison contexts (Liang et al., 2008), so it is highly recommended to verify the impact of different fixed choices before choosing the final value. Some common values are:

- $g = n$, with n as the number of observations, which represents the so-called Unit Information Prior (UIP) by Kass and Wasserman (1995);
- $g = k^2$ with k as the number of regressors in the full model, as suggested by the Risk Inflation Criterion (RIC) by Foster and George (1994);
- $g = n/k_i$ with k_i as the number of covariates in model M_i;
- $g = \sqrt{n}$;
- $g = max(k^2, n)$, the so-called benchmark prior by Fernandez et al. (2001a).

Placing an *hyperprior* on g, instead, would mean building a hierarchical Bayesian model: in other words, g is considered as a random variable with a given prior distribution defined hyperprior, which in turn depends on some parameter (*hyperparameters*) chosen by the researcher. In this way the estimation proceeds as a chain from the hyperprior to the prior to the final posterior; the use of hyper-priors is very common in Bayesian Statistics or Econometrics as it allows to avoid the choice of a fixed value parameter which can be sometimes quite arbitrary, enhancing the flexibility and the precision of the method. The choice of hyperpriors and hyperparameters, moreover, is not so strict as the one requested for the typical prior and its parameter.

The major drawback of using hyperpriors is the complexity involved, especially from a computational point of view[43].

Despite the use of a fixed value g or a hyperprior on g, what should be the main concern, especially in a BMA framework, is that such a choice, so as the more general choice for a prior, is fundamental and heavily determinant in the definition of posterior distributions for both parameters and model too (as we will se in the next subsection): in general, there is not such a universal choice which will work in the same manner for every kind of data; furthermore a lot depends also on the methodology used since a particular prior can work extremely well under a particular design, but poorly performs under a different one. Unfortunately, robustness to prior choices is a "quite" difficult objective to reach.

Posterior model distribution: the model prior

Let us start by the formal definition of the posterior model distribution through Bayes' rule:

$$P(M_i|y) = \frac{p(y|M_i)P(M_i)}{\sum_{j=1}^{M} p(y|M_i)P(M_i)} \tag{1.22}$$

where $p(y|M_i)$ is the marginal data density[44] and $P(M_i)$ the prior distribution of model i. Before examining these two elements in detail, it is worth noting that the posterior model probability is also the main ingredient in Bayesian model selection, where the aim is to find the specification which maximizes it[45]. As a consequence, the methods used in this section for computing the model posterior in model averaging, can be extended without loss of generality to model selection too.

The prior distribution $P(M_i)$ is chosen by the "model builder" according to its belief: the general procedure is to assume that models are equally probable (uniform distribution, $P(M_i) = \frac{1}{|\mathcal{M}|}$ where $|\mathcal{M}|$ identifies the cardinality of the set $\mathcal{M}$) when there is little awareness or information about the distribution (diffuse prior); otherwise a very common alternative is the Binomial one:

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

where given k total variables, $0 \leq \pi_j \leq 1$ is the prior probability that the *j-th* variable is significant and δ_{ij} is an indicator of the variable inclusion in the model[46]. When a common π is assumed, the Binomial prior can be expressed as:

$$P(M_i) = \pi^{k_i}(1-\pi)^{k-k_i}$$

with k_i as the number of regressors in M_i. Notice that $\pi < 0.5$ will imply that parsimonious models are more probable a priori, whereas $\pi > 0.5$ will favor larger

[43]Some applications of hyperpriors in BMA literature are Zellner and Siow (1980); Liang et al. (2008).

[44]Sometimes named as marginal likelihood.

[45]Sometimes, instead of using the model posterior, the marginal data density is used for model comparison: the ratio between two different marginal data density is known as Bayes Factor.

[46]Notice again that if $\pi_j = 0.5$ we fall in the case of uniform distribution; and if $\pi_j = 1$ variable j is always included (in every model).

models. If we further assume a common inclusion probability π distributed as a Beta with parameters a, b we end up with the Binomial-Beta[47]:

$$P(M_i) = \frac{B(k_i - 1 + a, k - k_i + b)}{B(a, b)}$$

with $B()$ the Beta distribution.

According to these distributional forms, it is supposed that the inclusion of one variable is independent of one another: this is often not true in practice due to the correlation. A plausible alternative is the so-called *dilution* prior, where a correction term is added to include this effect; however, the possible benefits from a practical point of view are still under analysis[48].

Posterior mode distribution: the marginal data density

It should be quite obvious how the prior model distribution is not a concern from a computational point of view; what, instead, is much more troublesome to compute is the marginal data density. In mathematical terms:

$$p(y|M_i) = \int p(y|\beta_i, M_i) P(\beta_i|M_i) d\beta_i \tag{1.23}$$

with $p(y|\beta_i, M_i)$ as the likelihood in M_i and $P(\beta_i|M_i)$ the prior on the correspondent parameter β_i.

The integral involved does not often have a closed expression: a remarkable example of an explicit solution is the linear model, where assuming the previous Normal-Inverse Gamma set-up leads to:

$$y|M_i \sim t\left(X_i\mu_0, \frac{b_0}{a_0}(I + X_i V_0 X_i^T)\right)$$

$$p(y|M_i) = \frac{b_0^{a_0}\Gamma(a_0+\frac{n}{2})}{(2\pi)^{\frac{n}{2}}\Gamma(a_0)}|I + X_i V_0 X_i^T|^{-\frac{1}{2}}[b_0 + (y - X_i\mu_0)^T(I + X_i V_0 X_i^T)^{-1}(y - X_i\mu_0)]^{a_0+\frac{n}{2}}$$

with X_i as the regressor matrix in model M_i, $t()$ as a Student t distribution and $\Gamma()$ as the Gamma function. Recalling Fernandez et al. (2001a), the marginal data density under a Zellner-g prior for β and an improper one for α and σ^2 becomes:

$$p(y|M_i) \propto \left(\frac{1}{1+g}\right)^{\frac{k_i}{2}} \left[(y - \bar{y}\iota)^T(y - \bar{y}\iota)\left(1 - \frac{g}{1+g}R_i^2\right)\right]^{\frac{n-1}{2}}$$

where I assume the same notation as the previous part.

From these expressions it is possible to verify the influence of the parameter prior on the marginal data density too: the Zellner-g prior set-up, in particular, allows to extend the previous analysis on the choice of the scalar g in model comparison frameworks.

In the case of parameter posterior, high values of g will imply the prevalence of the likelihood function over the prior; in model averaging or selection problems, instead, they will also lead to a strong preference for parsimonious models. However,

[47]The parameters a and b are in general defined *ad hoc*.

[48]For a detailed example in an economic scenario see Durlauf et al. (2008).

allowing $g \to \infty$ will produce the so-called *Lindley paradox*, that is the preference for the null model (model with only the constant term) over any others. A correct evaluation of priors, therefore, becomes again a crucial and fundamental aspect to consider.

Apart from linear models closed formula solutions for marginal data density are extremely rare, so we have to rely on approximations, a common distinction is between *a*) analytical approximations; *b*) numerical approximations (sampling).

I will turn attention on the first category for the rest of the subsection mainly because they are extremely common and used in practice; as for the second one what can be said is that it will produce more precise estimates, but it requires a full Markov Chain Monte Carlo for each model to be computed (see Chib (1995)), hence it would be very time consuming if an additional sampling were performed on the model space as well.

The most known analytical approximation is based on the so-called *Laplace's method*: assuming $P(\beta|y, M_i) \propto P(y|\beta, M_i)P(\beta|M_i)$ peaked in its maximum[49] β^* the Laplace method builds a Taylor expansion up to the second order of the log-posterior around β^* and after exponentiating obtains a normal distribution with mean β^* and variance equal to $\Sigma^* = (-H(\beta^*))^{-1}$, with $H()$ as the Hessian of the log-posterior.

Integrating this new expression leads to:

$$p(y|M_i) \approx (2\pi)^{k/2}|\Sigma^*|^{\frac{1}{2}}p(y|\beta^*, M_i)P(\beta^*|M_i) \tag{1.24}$$

where k is the dimension of β. As the sample size grows to infinity the approximation improves and the error term is negligible[50].

In practice, however, (1.24) is somehow modified, in particular a very used variation is defined by substituting β^* with the Maximum-Likelihood (ML) estimator and Σ^* with the information matrix. The accuracy is lower, especially if the prior on the parameter vector has some informative power, but the calculation is much more easier[51].

Before concluding a useful remark is necessary: following Raftery (1996) it is possible to compute the posterior model probability as

$$P(M_i|y) = \frac{\frac{P(M_i)}{P(M_0)}B_{i,0}}{\sum_{j=1}^{M}\frac{P(M_j)}{P(M_0)}B_{j,0}} \tag{1.25}$$

where $B_{i,0} = \frac{P(y|M_i)}{P(y|M_0)}$ is the Bayes Factor between the *i-th* model and a reference one (M_0), in general, the model with the only constant. As a matter of fact, (1.25) is only apparently a remedy to our problems, as the computation of Bayes Factor implies the one of marginal data density.

It could be proven that a similar approximation to the Laplace's one for the whole Bayes factor is available, and if additional conditions are met, notably a normal prior

[49]In general this corresponds to the case where the likelihood is peaked in its maximum $\hat{\beta}$ too.

[50]The thumb rule is the following: in order to have a good approximation the sample size must be at least bigger than $5k$; the optimal result is achieved for values bigger than $20k$.

[51]See Kass and Vaidyanathan (1992); Kass and Raftery (1995); Bollen et al. (2012) for the general framework.

on the parameters, the resulting formula is quite simple[52].

A rough, but sometimes used approximation for the Bayes Factor is provided by the difference $S_{i,0}$ between the Schwarz Criteria (BIC) of the two models:

$$S_{i,0} = \log p(y|\beta_i, M_i) - \log p(y|\beta_0, M_0) - \frac{1}{2}(k_i - k_0)\log(n)$$

where $\log p(y|\beta_i, M_i)$ is the log-likelihood of M_i, k_i the number of parameters in the correspondent model and n as the number of observations. Following Kass and Raftery (1995):

$$\frac{\log(B_{i,0}) - S_{i,0}}{\log(B_{i,0})} \rightarrow 0$$

Notice how this calculation neglects prior specifications, and implicitly assumes $\exp(S_{i,0}) \rightarrow B_{i,0}$, but as a matter of fact, the violation of this last condition is more common than one could expect[53].

At this point, it is quite clear that we can follow different paths to determine $P(M_i|y)$: using (1.22) with the Laplace approximation for the marginal likelihood or (1.25) with the Bayes Factor. Regardless of the choice, we incur in a computational issue, i.e. both denominators have calculations over the whole model space which can be easily huge, hence some refinements are requested.

Posterior model probability: a further look

Computing directly model posteriors via (1.22) could be extremely demanding from a computational point of view, due to the impact of the normalizing factor i.e. the denominator. A plausible alternative with the not negligible advantage of avoiding this computation, could be building a Markov Chain Monte Carlo simulation which moves inside the model space and allows for the specification of $P(M_i|y)$ as the number of transitions in M_i, normalized for the total amount of iterations.

The idea is mutated from the Metropolis-Hastings scheme (Hastings, 1970): given a model space $\mathcal{M}$ and a starting specification M_i, we sample M_j from a transitional kernel $q(M_j|M_i)$, which represents the probability of the movement from M_i to M_j and accept this one with probability $\rho(M_i, M_j)$ defined as:

$$\rho(M_i, M_j) = \min\left\{1, \frac{P(M_j|y)q(M_i|M_j)}{P(M_i|y)q(M_j|M_i)}\right\}, \quad M_i \rightarrow M_j \tag{1.26}$$

where $P(M_j|y)$ is the target distribution computable as before.

Following this method we exploit the convergence property of the Markov Chain, which allows us to identify important specifications with a higher precision than common approximation methods.

Notice, moreover, that $\frac{P(M_j|y)q(M_i|M_j)}{P(M_i|y)q(M_j|M_i)}$ can be rewritten as:

$$\frac{P(M_j|y)q(M_i|M_j)}{P(M_i|y)q(M_j|M_i)} = \frac{p(y|M_j)P(M_j)q(M_i|M_j)}{p(y|M_i)P(M_i)q(M_j|M_i)}$$

where we exploit the fact that the normalizing constant simplifies.

[52]See Raftery (1996) for the full explanation.

[53]Kass and Wasserman (1995) is the main reference for this problem and how to handle it.

What remains to be defined is $q(M_j|M_i)$, which identifies the proposal distribution of how models are sampled in the chain from the current M_i. It is built according to the definition of the *neighborhood* of the current model: by neighborhood we generally mean models which directly derives from the current one through addition, deletion or switching of variables. The original framework, known as *Markov Chain Monte Carlo Model Composition* (MC^3) due to Madigan et al. (1995), assumes a symmetric and uniformly distributed kernel: starting from model M_i we move to a model either with a variable more or with a variable less with the same probability[54], in particular if k is the total number of regressors, $q(M_j|M_i) = \frac{1}{k}, \quad \forall i, j$.

As an example consider the case of four regressors x_1, x_2, x_3, x_4 and the current specification as $M_i = (x_1, x_2, x_3)$: following the scheme we can end up in $M_j = \{(x_1, x_2); (x_1, x_3); (x_2, x_3); (x_1, x_2, x_3, x_4)\}$ each one with probability 0.25. The symmetry in the kernel also implies that the return movement $q(M_j|M_i)$ has the same probability.

This leads to:

$$\rho(M_i, M_j) = \min\left\{1, \frac{P(M_j|y)}{P(M_i|y)}\right\}, \quad M_i \to M_j$$

Unfortunately, despite its simplicity, the original MC^3 pattern has several drawbacks: firstly it includes only *local* moves, i.e. only one variable per time is changed, as opposed to *global* ones where many variables are changed; secondly, some specifications are not considered at all through the iterations; finally from a computational point of view as k grows or there is a high correlation between regressors, the convergence can be very slow.

Many solutions have been proposed in the literature, notably the introduction of different kernels $q()$ (Hans et al., 2007; Bottolo and Richardson, 2010), or even totally new sampling methods (Clyde et al., 2011; Hastie, 2005; Lamnisos et al., 2013): we will briefly cover the *Shotgun Stochastic Search* (Hans et al., 2007) for the first case and *adaptive sampling* by Lamnisos et al. (2013) for the second one for their theoretical and computational contribution.

In Shotgun Stochastic Search the neighborhood of a generic model M_i is split into three subsets according to additions moves ($\mathcal{M}_i^+$), where all specifications with a variable more are considered; deletions moves ($\mathcal{M}_i^-$) with all models with a variable less; and replacement moves ($\mathcal{M}_i^0$) with models with the same dimension obtained by swapping a variable[55].

From each one a specific model (proposal) is sampled using a probability proportional to a score function defined as $S() = p(y|M)P(M)$ normalized within the set, and finally the new model is sampled among these three according to the updated score for the three proposals. The computational burden of the operation is dramatically reduced by the means of parallel computing.

[54]Notice that the "switch" case, i.e. the substitution of a variable with another one without changing dimensionality, is not considered.

[55]It is actually possible to consider bigger movements, that is changing more than one single variable per time.

It can be shown how the standard acceptance probability could be rewritten as:

$$\rho(M_i, M_j) = \min\left\{1, \frac{\sum_{s \in nbd(M_i)} p(y|M_s)P(M_s)}{\sum_{s \in nbd(M_j)} p(y|M_s)P(M_s)}\right\}, \quad M_i \to M_j$$

where $nbd(M_i)$ indicates exactly the set of models $(\mathcal{M}_i^+ \cup \mathcal{M}_i^- \cup \mathcal{M}_i^0)$ [56]. SSS leads to a great boost in convergence time as the proposal identifies, through the score function, more probable models with a higher frequency.

Turning the discussion to the second case, i.e. when the whole sampling scheme is modified, an appealing tool whose importance has grown incredibly in recent year, is the *adaptive MCMC* simulation.

Intuitively, an adaptive MCMC tunes some parameters of the chain automatically by inserting additional information at each or at some iteration. This can be thought of as a sort of learning mechanism which enhances the quickness of the convergence and the mixing capacities. The additional information could come from the past history (or some statistics based on the past history) of the chain, or in more advanced studies from simultaneously launching the simulation via parallel programming. However in doing this the Markov property is broken, so the convergence of the MCMC to an invariant distribution is not guaranteed anymore. In Atchadé and Rosenthal (2005); Roberts and Rosenthal (2007, 2009) some criteria are shown to reestablish similar properties, in particular it is requested a diminishing importance of the learning as the number of iterations grows.

Lamnisos et al. (2013) is an example of adaptive methods for model search: in particular starting from model movements which include addition, deletion and switching as equally probable, the number of the changing variable from a model to another is the object of the adaption. It is assumed that this quantity is generated from a Binomial distribution with probability of success equal to ζ and number of trials $K - 1$ where K is the maximum amount of variables that can be changed. In general, K is chosen *ad hoc*, and only the parameter ζ is updated inside the Markov Chain for model dynamics: small values of ζ lead to few changes (local movements), high values imply instead the contrary (global movements). It can be proven that adaption leads to a better exploration of the model space avoiding the controversial choice of the number of changing variables; a quicker and a more efficient convergence may be established as well.

1.3.3 BMA in practice

So far we have analyzed the individual components of (1.19) separately, however it seems appropriate to move towards a more complete and wider perspective, trying to consider how *de facto* BMA is used: it is said that *"in theory there is no difference between theory and practice, but in practice there is"* and nothing is more valid in this context than this statement.

If we are lucky enough to have analytical formulas it could be reasonable to use directly (1.19-1.21): since the mixture distribution involved by the model averaging

[56] It is a common assumptions to reduce the cardinality of this whole set by erasing the elements with the lowest score up to a threshold arbitrarily chosen.

posterior distributions is often complex to be handled, only the posterior mean and variance are computed. The major problem is related to the cardinality of the model space: remember that with k variables excluding the constant term, there are 2^k potential models to analyze, an amount that can grow extremely fast with the addition of new covariates.

The most known solution is probably the *Occam's window* by Madigan and Raftery (1994) which simply reduces the models to be considered exploiting two simple principles:

1. if some models are significantly worst than the best one according to the posterior model distribution than these are removed. The threshold for significance is in general chosen arbitrarily;

2. exclude complex models if simpler (nested) ones receive major support from data ($P(M_i|y)$).

Occam's window can greatly reduce the elements of the summation over the model space, however there are still some drawbacks: in order to perform the procedure the whole model set must be mapped at least once, hence we will save time in computing $E(\beta|y)$ and $V(\beta|y)$ for the sole selected set, but $P(M_i|y)$ will be calculated for every model[57].

This leads to the second approach for BMA, MCMC simulations for a numerical approximation of $P(M_i|y)$, without the necessity of the normalizing factor. The classical choice is the MC^3 framework with symmetric and uniform movements, which can be easily generalized with different model proposals as we have seen in subsection 1.3.2. If analytical formulas for the marginal data density are available no problems should emerge, otherwise a very common solution is the already mentioned Laplace's approximation. Model averaged posterior moments are then easily obtained, provided model specific posterior moments have a closed formula.

As a matter of fact, the main difficulty that could be faced adopting this procedure is the lack of analytical formulas for moments too: approximations can be used, but their applicability could be questioned, especially, when they are applied for even the marginal data density.

It could be appealing to sample parameters simultaneously with the sampling of models, in this way, we could not only obtain model averaging posterior distribution for parameters, but also solve the previous issue concerning moments by using their sample counterpart.

Unfortunately, trying to accommodate the standard MCMC design to these new requirements is quite troublesome and actually arises many questions regarding how to proceed: sampling parameters at each accepted model during iterations is meaningful as long as we do know ex-ante their posterior, but when this is not possible, every time we change model specification, the chain on the parameter in that particular model is broken.

We could try to run a whole MCMC on the parameter after a first one on the model space is concluded, so that to sample exactly many parameters as many time each model appears, avoiding every problem during the first MCMC. But is it truly optimal? Not really, first of all because we treat the two MCMCs separately.

[57]And the related marginal data density is the big computational problem!

For this reason the *classical* BMA approach, i.e. the one depicted in this part, is particularly suitable for linear models, where analytical expression are available, but becomes unfeasible in more general contexts such as the one considered in this work i.e. Generalized Linear Models. Next chapter will introduce the modern BMA framework, where we can actually simultaneously sample model and parameters, overcoming this huge limitation.

Chapter 2

Reversible Jump MCMC and Parallelization in Gretl

In the previous Chapter we have seen a brief and hopefully comprehensive disclosure of the main model building strategies: model selection on the one hand and model averaging on the other.

The impossibility of recognizing the true Data Generating Process leads to potential uncertainties in model selection, inducing potential wrong conclusions; model averaging, instead, deals with this issue directly, guaranteeing more robust and more accurate estimations. Some applications of model averaging in Economics were previously outlined, but what seems really interesting is the few attention for Microeconometrics, so the main focus of this Chapter will be providing a software implementation via Gretl of Bayesian Model Averaging (BMA) for Generalized Linear Models (GLMs), where, for example, binary choice models reflect economic agent behavior in terms of taking a particular action or not.

The choice of a Bayesian framework could be motivated by its great flexibility and potentiality: despite the Frequentist counterpart, BMA leads *directly* to posterior model probabilities together with the probability of inclusion for each singular regressor, making inference immediate; statistical properties such as consistency in the model choices[1] (Fernandez et al., 2001a) or estimates which minimize the Mean Squared Errors (Raftery and Zheng, 2003) are easily verified too and, finally, the applicability in finite-sample contexts is not negligible as well.

However, the choice of BMA is not painless as it could appear either: the choice for prior distributions is often crucial and in case of GLMs analytical formulas are generally not available neither for the posterior model distribution nor for the posterior parameter distribution. The *classical* BMA methodology, as we have seen previously, is tailored for linear models (which of course are a special case of GLM), but unless we are ready to use lots of approximations or to apply two-step procedures for posterior parameter distributions, it is quite inadequate in a more general context like this one.

The *modern* BMA approach is the answer to the problem: the idea is to build a MCMC which samples simultaneously parameters of interest and the related models,

[1]Model choice consistency refers to the fact that if the data have been generated by a specific model inside the model space, as the sample size grows, the posterior distribution of that model should converge to unity.

in doing so not only it is possible to derive model averaging parameter posteriors, but also the new Markov Chain target distribution allows for some computational gains in terms of not approximating model posteriors.

Different methods are available, but the Reversible Jump Markov Chain Monte Carlo (RJMCMC) by Green (1995) seems to be the more adherent to the scope of this work due to its greater flexibility, so the software implementation will be entirely based on this technique. RJMCMC can be considered as a modification of the standard MCMC framework sketched in the previous Chapter, where, instead of focusing only on models, the joint posterior of parameters and models is the target distribution of interest: in this way the different dimensionality of parameters is taken into account leading to a sample scheme for parameters too.

The focus of this Chapter, however, is not much on the methodology applied, which is already known and has already been used in the context of model selection (Lamnisos et al., 2009), but on the software implementation of this one in a BMA context for GLMs: in doing so, several computational aspects which were often ignored and commonly not considered in previous available software packages are now analyzed, first of all the parallelization of the MCMC thanks to the new MPI (Message Passing Interface) support provided by Gretl. The idea of parallel MCMCs seems a bit counterintuitive mainly because parallelization works extremely well in standard Monte Carlo experiment where statistical independence between drawings can be established, but is actually feasible even with sequential processes. Not only under some circumstances we can gain a massive boost in CPU time, but we can enhance the exploration of the model/parameter space too.

In particular, the here-proposed Gretl package will deal with binary and count models[2] (offering different choices for the link function) with flexibility in the choice of parameter priors, model movement kernel, and finally the already mentioned possibility of parallelization. Therefore, a quick solution for model building in common microeconometric problems is proposed, making it simple, even for the common user, to perform such a complex procedure.

The Chapter is organized as follows: Section 2 sketches the statistical background of GLMs (very briefly) and of RJMCMCs, with great attention to the original idea of a plausible "automated" choice of MCMC (Green, 2003); Section 3 describes the concept of running multiple MCMCs in parallel and the related convergence issues. I will devote Section 4 to describing the peculiarities of this Gretl package, since different computational aspects will be analyzed. A simple empirical illustration is also offered whose scope is to underline the impact of model averaging compared to a standard estimation strategy and the gains due to parallelization.

2.1 Statistical background: GLMs

Let $y_1, ..., y_n$ be n observations of a dependent variable from the exponential family with density function $f(y)$:

[2]Linear model are non analyzed because common MCMCs routines can be easily applied with greater efficiency than the RJMCMC (due to the availability of closed formulas). Moreover BMA packages for linear models are available in almost any statistical-econometric software.

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

where θ_i and ϕ are respectively a location and a scale parameter, $a()$, $b()$, $c()$ are known functions.

It can be shown that:

$$\begin{aligned} E(y_i) &= \frac{\partial b(\theta_i)}{\partial \theta_i} \\ V(y_i) &= \frac{\partial^2 b(\theta_i)}{\partial \theta_i^2} a_i(\phi) \end{aligned}$$

The design of Generalized Linear Models assumes that, given a set of k regressors, the mean $E(y_i) = \mu_i$ is related to these covariates by the following expression:

$$l(E(y_i)) = \eta_i = x_i^T \beta \tag{2.1}$$

where $l()$ is known as *link function*, and $\eta_i = x_i^T\beta$ is the usual linear predictor. According to the formulation of the link l, we can model data from the exponential family, which ranges, for instance, from the canonical Gaussian distribution to Binomial or Poisson ones[3]: in the Gretl package here proposed, in particular, the link functions included are

Distribution y_i	Model	Link function
Bernoulli	Binary choice	Probit
		Logit
		Cloglog
Poisson	Count	Log

In order to fully understand which problems arises in a Bayesian framework, let us assume, as an example, that we are dealing with binary data $y_i \sim Be(\mu_i)$ and suppose a logit link function:

$$x_i^T\beta = \log\left(\frac{\mu_i}{1-\mu_i}\right) \to \mu_i = \frac{\exp(x_i^T\beta)}{1+\exp(x_i^T\beta)}$$

It can be shown quite easily that the likelihood function is given by:

$$p(y|\beta) = \prod_{i=1}^{n}\left[\left(\frac{\exp(x_i^T\beta)}{1+\exp(x_i^T\beta)}\right)^{y_i}\left(1-\frac{\exp(x_i^T\beta)}{1+\exp(x_i^T\beta)}\right)^{1-y_i}\right]$$

and if the prior on β[4] is normal $N(\mu_0, V_0)$, the resulting posterior has no more a closed expression (and implicitly the resulting moments too):

[3]The linear model is a particular case of GLM which derives from the identity link function, however in the rest of the work I will refer the term GLMs to non-linear cases such as Binomial or Poisson models.

[4]In the binary framework the only parameter of interest is β.

$$p(\beta|y) \propto \prod_{i=1}^{n}\left[\left(\frac{\exp(x_i^T\beta)}{1+\exp(x_i^T\beta)}\right)^{y_i}\left(1-\frac{\exp(x_i^T\beta)}{1+\exp(x_i^T\beta)}\right)^{1-y_i}\right]\times\exp\left(-\frac{1}{2}(\beta-\mu_0)^T V_0^{-1}(\beta-\mu_0)\right)$$

The same argument can be extend for other kinds of models and link functions[5], so it could be appealing to find a different path, i.e. sampling, with the chance of a general framework that could be used for any or almost many link functions[6].

A powerful idea is provided by Gamerman (1997) which exploits Fisher's result on maximum-likelihood estimation of GLMs, in particular we can obtain an equivalent Frequentist estimator using Iterative Weighted Least Squares on the transformed variable $z = \eta + (y-\mu)\frac{\partial\eta}{\partial\mu}$, where η, y, μ are the matrix correspondents of η_i, y_i, μ_i; in particular:

$$\hat{\beta} = (X^T W X)^{-1} X^T W z \tag{2.2}$$

where the weight matrix W is defined as diagonal with elements:

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

The value of z is the one updated step by step via $\hat{\eta} = X^T\hat{\beta}$.

Gamerman (1997) transposes this procedure in a Bayesian viewpoint, starting from a normal prior on β with mean m_0 and variance-covariance matrix V_0, i.e. $\beta \sim N(m_0, V_0)$, proposes the following sampling scheme mutated from Metropolis-Hastings MCMC:

1. Set as initialization β_0;
2. At the *i-th* iteration, sample $\beta^{(i)}$ from the proposal $q(\beta|\beta^{(i-1)}) = N(m^{(i)}, V^{(i)})$, where:

$$V^{(i)} = (V_0^{-1} + X^T W^{i-1} X)^{-1} \tag{2.3}$$
$$m^{(i)} = V^i(V_0^{-1} m_0 + X^T W^{i-1} z^{i-1}) \tag{2.4}$$

   where for the computation of z^{i-1}, W^{i-1} we use $\beta^{(i-1)}$;
3. Accept the new parameter with probability $\alpha(\beta^{(i-1)}, \beta^{(i)})$ and set $\beta^{(i)} = \beta^{(i)}$, otherwise $\beta^{(i)} = \beta^{(i-1)}$. The accept-reject probability is defined as in standard Metropolis-Hastings scheme as:

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

   where $f(\beta^{(i)}|y) \propto f(y|\beta^{(i)})f(\beta^{(i)})$, that is the product between likelihood function and prior; $q(\beta^{(i)}|\beta^{(i-1)})$ is a normal density evaluated at $\beta^{(i)}$ with mean and

[5]Binary/multinomial data with probit or logit links, or the count model with the logarithm link are all examples of not analytical posteriors.

[6]Different sampling schemes can be proposed, but it is quite difficult to find a sufficient general framework which can embody different GLMs: of course, there is a lack in efficiency, but the flexibility is of primary interest in our case.

variance, respectively, equal to eqs. (2.3) and (2.4); the same argument is valid for $q(\beta^{(i-1)}|\beta^{(i)})$ with $\beta^{(i)}$ used to compute the mean and variance of the normal, evaluated this time in $\beta^{(i-1)}$.

2.2 BMA in GLMs: a brief overview

As we have already seen, BMA is primarily concerned with the computation of the mixture distribution:

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

and the related moments:

$$\begin{aligned} E(\beta|y) &= \sum_{i=1}^{M} E(\beta|y, M_i)P(M_i|y) \\ V(\beta|y) &= \sum_{i=1}^{M} \big[V(\beta|y, M_i) + E(\beta|y, M_i)^2\big] P(M_i|y) - E(\beta|y)^2 \end{aligned}$$

where it is assumed the same notation of Chapter 1.

However, unlike linear models, no analytical solution are provided under standard prior set-up[7] for neither moments nor model posteriors, so the canonical BMA application based on the directly exploitation of the above formulas, maybe with the Occam's window help or the MCMC framework, seems to be a bit restrictive. Common routines, quite often applied in practice involve the use of maximum-likelihood estimators and the corresponding variance as proxies for posterior moments, and the use of Laplace or BIC approximations for posterior model distributions. Nevertheless, all of these shortcuts heavily rely on some regularities conditions which are often not met in practice, so their contribution may be questionable[8] and incomplete.

A generalization of the standard paradigm is required, and two alternative contributions in this direction are the Reversible Jump Markov Chain Monte Carlo (RJMCMC) by Green (1995) and the Stochastic Search Variable Selection (SSVS) by George and McCulloch (1993). RJMCMC can be literally considered as a modification of the MCMC framework for model exploration where, instead on focusing only on models, parameters and models are jointly sampled: at each step a specification is proposed and the related parameters are attached not using another sampling scheme, but simply transforming the ones of the previous step via an *ad-hoc* function. The potential different dimensionality between parameters is taken into account thanks to the so-called *matching* variable, which acts as a substitute parameter in order to balance the overall dimension[9]. In Green (2003); Green and Hastie (2009);

[7]The Normal distribution is an example.

[8]As an example consider the article by Amini and Parmeter (2011), who analyzes different BMA packages for linear models in R, comparing their different outputs.

[9]Further details can be found in the next section.

Hastie and Green (2012) the general framework is extended to consider model selection and automated procedure; whereas some notable contribution in GLMs are Holmes and Held (2006); Fouskakis et al. (2009); Lamnisos et al. (2009).

SSVS, instead, tackles the problem of simultaneous sampling by using the Gibbs MCMC via data augmentation: parameters are sampled from their posterior distribution 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 probability in the conditional model posterior and consequently the sampled parameter, are set near zero. Nevertheless, the definition of the conditional posterior model distribution is far from being simple, and depends heavily on the model set-up (linear models, binary ones, etc.[10]).

The computational efficiency is surely higher for SVSS, but it lacks of flexibility: in RJMCMC the same sampling scheme can be applied for different kind of GLMs, a feature precluded in SVSS, and for this reason RJMCMC is the framework here adopted[11]. Notice, moreover, how both procedures lead to the model averaging posterior distributions and moments by simply handling accordingly the sampled values.

Before concluding it is intriguing to discuss a third possibility which comes up besides RJMCMC and SSVS: the framework illustrated by Chen et al. (2008). Assuming parameter conjugate priors as stated in Chen and Ibrahim (2003) results in posterior model probabilities computable drawing drawing only two MCMC samples: one from the posterior distribution and one from the prior distribution of the parameters under the unrestricted model. In this way common model comparison measures (Bayes factor) can be easily derived, and at the same time, model averaging quantities too, with the not negligible advantage of the conjugate prior set-up.

2.3 Reversible Jump Markov Chain Monte Carlo

The Reversible Jump Markov Chain Monte Carlo (Green, 1995, 2003), allows to sample the parameter of interest β, accounting for its potential different dimension iteration per iteration: this corresponds, for instance, to the case in which across simulations we are investigating different models for the parameter, such as from a general one with many covariates to some more specific ones with few.

This 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 different drawings with different dimensionality, an aspect that is ignored in classical 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 scope is nothing but guaranteeing that the total dimension of the space given by (β_i, u_i) is equal to the dimension of (β_j, u_j). This is the most important assumption that must hold during the MCMC,

[10]Frühwirth-Schnatter and Wagner (2006); Frühwirth-Schnatter and Frühwirth (2007, 2010); Frühwirth-Schnatter and Wagner (2010) are some examples.

[11]RJMCMC can be easily applied to not nested cases too, whereas, for SSVS, nesting is a fundamental requirements.

as a consequence, the variable u_i contains the potential parameters which are deleted or added in order to match the dimensionality of β_j and vice versa.

For example suppose that β_i is a vector of elements (β_{i1}, β_{i2}), and we want to obtain a $\beta_j = (\beta_{j1}, \beta_{j2}, \beta_{j3})$ as $\beta_j = g(\beta_i)$; since the dimension of β_j is greater than the dimension of β_i, we need to add to β_i a variable u whose dimension is simply the difference between the two. Therefore $\beta_j = g(\beta_{i1}, \beta_{i2}, u_{i1})$.

In our context we can identify the indexes "i" or "j" by the correspondent model specification of each β, hence β_i refers to β_{M_i}.

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 is[12]:

$$\rho = \min\left[\frac{P(\beta_j, M_j|y)f(u_j)q(M_i|M_j)}{P(\beta_{M_i}, M_i|y)f(u_i)q(M_j|M_i)}\left|\frac{\partial g(\beta_i, M_i; u_i)}{\partial(\beta_i, M_i; u_i)}\right|; 1\right] \tag{2.5}$$

where $f()$ denotes density functions, $\frac{\partial g(\beta_i, M_i; u_i)}{\partial(\beta_i, M_i; u_i)}$ the Jacobian of the transformation $g()$ which is requested in order to take into account the change of measurement from (β_i, u_i) to β_j, u_j .

As a matter of fact the framework can be further generalized: the function $g()$ can be substituted by a proposal kernel for the parameter β,

$$\rho = \min\left[\frac{P(\beta_j, M_j|y)q(M_i|M_j)q(\beta_i, u_i|\beta_j, u_j)}{P(\beta_{M_i}, M_i|y)q(M_j|M_i)q(\beta_j, u_j|\beta_i, u_i)}; 1\right]$$

where the proposal $q(\beta_j, u_j, M_j|\beta_i, u_i, M_i) = q(M_j|M_i)q(\beta_j, u_j|\beta_i, u_i)$ allows to specify separately the model movement $q(M_j|M_i)$ from the parameter proposal $q(\beta_j, u_j|\beta_i, u_i)f(u_i)$, where we need to account for the generation of the random variable u_i[13].

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

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

[13]Notice how the previous scheme is actually a particular case of this one: the variable u is generated from a standard Normal ad then deterministically via $g()$ the new parameter is generated. In this case $q(\beta_j, u_j, M_j|\beta_i, u_i, M_i)$ reduces to $q(M_j|M_i)f(u_i)$, but the Jacobian term needs to be added. Moreover, it is implicitly assumed that both schemes, the one which uses the function $g()$ and the one which uses a proposal $q(\beta_j, u_j|beta_i, u_i)$ can be combined too.

the closer they are to the real posteriors, the more efficient the chain is[14]. Quite unfortunately this choice is often troublesome, but a fair compromise is provided in the next Subsection, following the idea of a plausible automated method for RJMCMCs.

2.3.1 An automated RJMCMC for GLM

In Green (2003); Green and Hastie (2009) and, especially in Lamnisos et al. (2009, 2013) we find a particularly suitable function $g()$ for GLMs[15]: 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[16]

$$\beta_j = g(\beta_i, M_i, u_i) = \mu_j + B_j \upsilon \tag{2.6}$$

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

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

with k as the number of variables, R a random permutation matrix; the notation $[...]^{k_j}$ indicates the first k_j elements of the vector and finally u, a $k_j - k_i$ vector of random numbers with density $f()$, in general a standard normal. Notice that the parameter β is actually treated as a multivariate normal, which in a first step is standardized and then the mean and covariance matrix of the new model are attached to it. The normal choice could be questionable, but Green (2003) shows how this proposal is a good compromise between efficiency of the chain and simplicity.

The probability (2.5) becomes:

$$\rho = \min\left[\frac{P(\beta_j, M_j|y)q(M_i|M_j)}{P(\beta_i, M_i|y)q(M_j|M_i)}\frac{|B_j|}{|B_i|}G; 1\right] \tag{2.7}$$

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

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

The independence of the kernel from the parameter allows us to separately determine the model movements from those of the parameters, in particular we can

[14]Some contributions in this direction are Brooks et al. (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.

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

[16]It turns out that this kind of transformation is particularly suitable for GLMs with a unique parameter vector to estimate, such as the ones for binary or counting models: in linear models, on the contrary, the parameters of interest are β and σ, each one with its particularity. For this reason this particular function is not recommended and additional modifications are required.

select firstly the new model M_j, and then its related β_j via the function $g()$. The permutation matrix R makes the movements to lower dimensional models stochastic and actually plays no role in the acceptance ratio.

Notice that if the posterior parameter distributions $P(\beta_i|M_i, y)$ are actually Normal, (2.7) reduces to the common MCMC acceptance rate:

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

via the fact that $P(\beta_i, M_i|y) = P(\beta_i|M_i, y)P(M_i|y)$.

2.3.2 Prior choices and how to approximate μ and V

The sample scheme here defined represents a general solution for model building problems, but, actually, its effectiveness depends a lot on the regularity of the data, the quality of the approximations μ_i and V_i, and, of course, the prior choices.

Obviously, this final element plays an important role which is exacerbated in this context by the fact that the acceptance probability (2.7) depends heavily on these via the joint posterior $P(\beta_i, M_i|y)$ and the Jacobian term. It could easily happen for example, that a particular prior choice for a peculiar GLM, performs poorly when we use a different GLM.

The definition of priors which is here proposed follows the common routines in BMA literature, that is:

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

represents 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 connected to the constant term of the regressor matrix has a separated prior distribution, an improper one, according to the argument of Fernandez et al. (2001a) for linear models. This comes from the fact that the the constant is assumed to be always included, and, in practice, this is accompanied by the demeaning of all other regressors in order to establish independence between constant and other covariates. The same argumentation, unfortunately, cannot be fully applied to GLM due to the impact of the link function.

A solution could be either considering the constant as another regressor, so preventing any special treatment or the compromise of allowing for a diffuse constant prior with a Normal distribution of the following form:

$$\alpha \sim N(0, h) \to \begin{pmatrix} \alpha \\ \beta_i \end{pmatrix} \sim N\left(\begin{bmatrix} 0 \\ \mu_{0,i} \end{bmatrix}, \begin{bmatrix} h & \underline{0}^T \\ \underline{0} & V_{0,i} \end{bmatrix}\right) \tag{2.8}$$

where α is the parameter of the constant term, h its variance, which in general is set to 100 (Lamnisos et al., 2009)[17]; $\underline{0}$ is a vector full of zeros to match the different dimensionality between the constant and the other covariates. Using this second possibility implicitly assumes that the constant is always present in every

[17]In general this choice is not so determinant.

specification, and all the other regressors have to be demeaned as in the original framework.

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

As for the Zellner-g alternative, a well-known modification for GLMs is $gn(X_i^T X_i)^{-1}$, where n is the number of observations: this reflects more directly how this covariance should be considered as a function of the Unit Information Prior covariance matrix by Kass and Wasserman (1995), which is generally the most common choice. The parameter c and g can be fixed or a hyperprior can be placed: the second choice is surely more correct, but leads to a heavier computational burden. It is not a mere coincidence that, in practice, the fixed value approach is favored: the ridge prior c is commonly chosen via grid-search or cross-validation approaches (Lamnisos et al., 2012); whereas for g, some proposed values are $g = 1$, which is an easy and common solution often adopted in practice; $g = 4$ as suggested by Fouskakis et al. (2009) for logistic regressions; $g = 9.87/k$, with k as the total number of covariates, by Hanson et al. (2014)[18].

As for the model prior we assume the Binomial distribution:

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

where given k total variables, $0 \leq \pi_j \leq 1$ is the prior probability that the *j-th* variable is significant and δ_{ij} is an indicator of the variable inclusion[19].

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; Green and Hastie (2009) introduces, instead, the use of mixture distributions. An appealing alternative, which is also the one proposed in this Chapter can be found in Lamnisos et al. (2009, 2013), who show how the moments estimates[20] obtained via Gamerman MCMC on GLMs are a good solution; moreover relying on a single iteration initialized on the frequentist estimator seems to provide good results too.

2.3.3 The RJMCMC "in a nutshell"

The dynamics of the whole Markov Chain is summarized in Lamnisos et al. (2013):

[18] Another important reference for g-prior choices is Gelman et al. (2008)

[19] Notice again that if $\pi_j = 0.5$ we fall in the case of uniform distribution; and if $\pi_j = 1$ variable j is always included (in every model).

[20] eqs. (2.3) and (2.4).

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 (2.6);
3. Accept the move with probability (2.7), otherwise stay in (β_i, M_i);
4. Repeat from 2, till convergence.

However this scheme, when top specifications appear, produces the storage of the same estimate of the parameter, with potential consequences for its posterior distribution, therefore, a resampling (within move) may be introduced when a new couple (β_j, M_j) is rejected; in other words, when we fail to move to a new model, given the current state (β_i, M_i), a new β_i, which corresponds exactly to an iteration of Gamerman's MCMC, is proposed. The correspondent μ_i and V_i for the new sampled parameter may be updated with eqs. (2.3) and (2.4) obtained in the resampling step.

In sum:

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 (2.6);
3. Accept the move with probability (2.7), otherwise *propose a resampling of β_i in M_i following a single iteration of Gamerman procedure.*
4. Repeat from 2, till convergence.

2.4 Parallel MCMCs

Parallel computing applied in Monte Carlo simulations exploits two different ideas:

- running the same process over different cores or networked computers, simultaneously, and then aggregating each result with the aim of obtaining as many replications as possible;
- splitting a "long" process into smaller units, that can be separately computed by different machines (*"divide et impera"* philosophy).

The gains are double, because we highly speed up procedures that generally would take long time to be fully performed, moreover, we can try to build *ad hoc* schemes, which can overcome some limitations, such as the impossibility of analyzing huge dataset or huge model spaces.

A simple application of parallelization in a Monte Carlo experiment could be the simulation of a known random variable x: suppose to be interested in calculating the mean of this variable, we can either simulate in a unique run N values x_i and then compute the sample mean $\bar{X}$,

$$\bar{X} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

or exploit parallel processing by splitting the N iterations across c cores, independently. In each one, the sample mean of the related population is obtained and then each result is appropriately aggregated to have the final estimate,

$$\bar{X}_i = \frac{1}{N_i}\sum_{j=1}^{N_i} x_j \to \bar{X} = \frac{1}{c}\sum_{i=1}^{c} \bar{X}_i$$

where $N_i = N/c$, i.e. the fraction of simulations per core.

Therefore, when iterations are independent, the only problems derive from physical failures of the single machines involved, which lead to the impossibility of obtaining a part of the parallelized process[21].

However this whole line of reasoning concerns standard Monte Carlo simulations, is it possible that all the previous arguments are equally valid for Markov Chain Monte Carlo too?

The answer is positive, but, of course additional attention is requested.

Firstly MCMCs are sequential processes with a necessary burn-in time, so the *divide et impera* philosophy should be analyzed with much care, as the benefit of multiple chains over the benefit of a single one is not so straightforward.

At a first glance, the impossibility of shortening (splitting) the burn-in could undermine the possible time gain but, as a matter of fact, the issue seems to be ill-posed. Replicating the same MCMCs in parallel as chunks of a bigger one leads to a speed-up in sampling, as long as the convergence rate is fast and the proportion of burn-in period is *small* compared to the total amount of iterations (Amdahl, 1967). When this is not the case, a single long chain can be more suitable.

A plausible guideline is provided by Gelman and Rubin (1992); Brooks and Gelman (1998), who introduce some indexes which can help to monitor the convergence rate of each chain, and the possible advantages of using additional machines: I will investigate these ones in the next subsection.

Notice, however, that when we are only interested in obtaining replications of the same process for a better exploration of the parameter space, with little attention paid in CPU time, the use of simultaneous MCMCs is extremely useful: as an example consider the problem of multimodal distribution. A single chain can halt in local maximum points, preventing a full exploration of the parameter space unless a sufficient high number of iteration is provided; in these cases running the same MCMC algorithm in parallel can solve the problem, possibly with different starting points.

An interesting extension of the these two applications is proposed by Scott et al. (2016), where a dataset is split into different non-overlapping subsets, each one dispatched for the execution of an identical MCMC. Finally synthesis measures are individually applied and combined according to a consensus scale, such as a variability measure, which summarizes the importance of the relative restricted dataset.

[21]This issue is deeply analyzed in Rosenthal (2000). Some common solutions involve queuing the work to reliable computers or simply parallelizing in different cores of a single device.

Possible further applications of parallelization to MCMCs are the so-called "horizontal" chains, as opposed to the previous ones which are often labeled as "vertical"[22], where only some components of a single long chain are computed in parallel in order to gain a boost in computational time. The conditional independence is the requested assumption so that the Markov Property is not violated: for instance, assume to have three parameters $\theta_1, \theta_2, \theta_3$, if θ_1 and θ_2 are conditionally independent given θ_3, in symbols $\theta_1 \perp \theta_2 | \theta_3$, we can compute separately MCMCs for θ_1 and θ_2, once obtained the values of θ_3.

The conditional independence condition is extremely difficult to support in practice, hence the use of horizontal MCMCs have often been "resized" to the *ex-ante* parallel calculation of some quantities that will be used in a single MCMC e.g. the *a priori* computation of each marginal data density on the whole model space. Alternatively, the so-called *pre-fetching* by Brockwell (2006), where in a canonical Metropolis-Hastings MCMC, the task of sampling and accept the move are separated: a binary tree is built where each node represents a sampled parameter and its "sons" are the correspondent parameters in case of acceptation and in case of rejection, without having care of which one is correct. In order to ascertain which crossroads to take, the accept-reject ratios of the MCMC are computed in parallel after the sampling step; the motivation lies in the fact that sampling is in general not computational expensive, whereas judging the move is.

2.4.1 Convergence in parallel

Vertical MCMCs may appear less effective and rawer than horizontal ones, but their applicability is easier and actually can be extremely beneficial: the idea of independently running the same MCMC on different cores exploits the fact that if convergence is reached, similar results should be achieved by each chain, and aggregating is comparable to having run a single long chain in its stationary distribution. This allows to apply parallelization for both the applications previously encountered: splitting a long procedure to save time and encouraging a better exploration of the parameter space. We will see very soon as the software implementation of the RJMCMC employs both the vertical and the horizontal strategy, with particular attention for the first one, whose presence is somehow dominant with respect to the second, as a consequence of this, an in-depth analysis of the convergence and in particular of how to monitor the convergence in parallel is necessary.

A first analysis of the problem is considered by Fosdick (1959) where multiple chains are run till the sample average of the parameters becomes approximately equal in each MCMC, which corresponds to a very simple idea of convergence. However many other works seem to propose naive methodologies or, in other words, the topic was restricted to "qualitative" analysis, such as visual comparison of density functions: a famous example is Gelfand and Smith (1990). The most known studies which tried to build some quantitative and more rigorous measure, are due to Gelman and Rubin (1992) and its extension due to Brooks and Gelman (1998).

Gelman and Rubin (1992) assume to have a univariate random variable x simulated n times in c cores, with mean μ and variance σ^2, and an unbiased estimator $\hat{\mu} = \bar{X}$ for μ. If x_{ji} identifies the sampled x in iteration j of the core i, the be-

[22] Or "embarrassingly" parallelisable, but this is not very kind!

tween (intra core) variance B/n and the within (inside the same core) variance W are defined as:

$$B/n = \frac{1}{c-1}\sum_{i=1}^{c}(\bar{X}_i - \bar{X})^2$$
$$W = \frac{1}{c(n-1)}\sum_{i=1}^{c}\sum_{j=1}^{n}(x_{ji} - \bar{X}_i)^2$$

where $\bar{X}_i = \frac{1}{n}\sum_{j=1}^{N} x_{ji}$ and $\bar{X} = \frac{1}{c}\sum_{i=1}^{c}\bar{X}_i$ are respectively the sample mean calculated with the data obtained in the *i-th* core, and the sample mean on the whole data.

The parameter σ^2 can be estimated now with:

$$\hat{\sigma}^2 = \frac{n-1}{n}W + \frac{B}{n}$$

where in general a correction term due to the sampling variability of $\bar{X}$ is added, i.e. $\frac{B}{cn}$[23], so we have finally:

$$\hat{V} = \hat{\sigma}^2 + \frac{B}{cn} \tag{2.9}$$

The Gelman-Rubin convergence measure is then:

$$\hat{R} = \frac{\hat{V}}{W} \tag{2.10}$$

with the following meaning: if $\hat{R}$ is large, more simulation will probably improve the convergence, by contrast, if it is close to 1 the convergence is reached[24].

Equation (2.10) is further improved by taken into account sampling variability in the variance estimates: assuming normality in x_{ij} allows to adjust the statistics by a correction factor, derived from the degrees of freedom of a Student t distribution, obtainable as $df = 2\hat{V}/Var(\hat{V})$. This leads ultimately to:

$$\hat{R}_c = \frac{df}{df-2}\hat{R}$$

or the updated version by Brooks and Gelman (1998):

$$\hat{R}_c = \frac{df+3}{df+1}\hat{R}$$

Notice that the normality assumption is not requested for the standard measure (2.10), even though in the original paper it is implicitly assumed; whereas the corrected version needs it. In order to circumvent the problem Brooks and Gelman (1998) propose additional statistics and refinements: firstly, and maybe surprisingly, the $\hat{R}_c$ performs well even in case of non normality, but instead of computing it at

[23]This is the variance of the sample mean $\bar{X}$, given by $\frac{1}{c}$ times the sample variance of $\bar{X}_i$, B/n.

[24]A quantification of the goodness of $\hat{R}$ often depends on the data, however in the literature a thumb rule is provided: $\hat{R} < 1.2$ is considered as a good proxy for convergence even though sometimes a stricter threshold is placed (1.1) in order to guarantee better estimates.

the end of the chain, the correct approach would be monitoring it throughout the chain, together with $\hat{V}$ and W.

Secondly, additional statistics (more robust to distribution assumptions) can be computed, in particular the so-called *interval-based* $\hat{R}$ and the *empirical central moment* $\hat{R}_s$.

The former is simply obtained calculating the ratio between a quantile range in the empirical distribution of the parameter of the aggregate chain (the one obtained by combining all the single chains), and the mean of the same quantile range on the individual chains:

$$\hat{R}_{int} = \frac{\text{quantile range of the whole chain}}{\text{mean of individual quantile range}}$$

The latter instead is defined as:

$$\hat{R}_s = \frac{\frac{1}{cn-1}\sum_{i=1}^{c}\sum_{j=1}^{n}|x_{ji}-\bar{X}|^s}{\frac{1}{c(n-1)}\sum_{i=1}^{c}\sum_{j=1}^{n}|x_{ji}-\bar{X}_i|^s}$$

Obviously the existence of the "*s-th*" moment is assumed. Notice that the second moment measure is extremely similar to the classical $\hat{R}$ and, as a matter of fact, all central moments statistics tend to be very close to the original one.

So far univariate measures have been analyzed, however an extension in a multivariate set-up is possible: apart from the "easy option" of applying the univariate statistics to each parameter, Brooks and Gelman (1998) provides the multivariate correspondent of $\hat{R}$, in particular given:

$$W = \frac{1}{c(n-1)}\sum_{j=1}^{c}\sum_{i=1}^{n}(x_{ij}-\bar{x}_j)(x_{ij}-\bar{x}_j)^T$$
$$B/n = \frac{1}{c-1}\sum_{j=1}^{c}(\bar{x}_j-\bar{x})(\bar{x}_j-\bar{x})^T$$

which are the multivariate versions of the previous defined quantities, with x_{ij}, $\bar{x}_j$ and $\bar{x}$ as vectors of parameters.

If we define λ as the maximum eigenvalue of $W^{-1}B/n$, the new convergence statistics is:

$$\tilde{R} = \frac{n-1}{n} + \frac{c+1}{c}\lambda \tag{2.11}$$

The interpretation is the same as the canonical case, an index near the unity implies convergence.

It could happen moreover that the matrix W is singular, in this case $\tilde{R}$ would be incalculable, so a possible alternative is to monitor the determinant of both W and B/n. A nice feature of (2.11) is that the index approximates the upper bound of the maximum of the univariate $\hat{R}$ statistics over all the variables. Finally a comment related to plausible alternative is necessary: interval-based indexes are equally feasible in theory, but when the dimensionality is high the computational effort becomes excessive so (2.11) remains a "cheap" and efficient choice.

2.5 The package

Gretl packages are collection of functions (defined by the user), which add new procedures, estimation methods, etc. to the repertory of already implemented ones (built-in functions). In order to use packages, we have to ascertain firstly that they are downloaded and then loaded in our current work session: the first task can be accomplished either via the GUI, or simply invoking in the Gretl console the command `install` followed by the name of the package; the second task, if our aim is to execute the package inside a script or in the command line, implies another built-in command, that is `include` followed, even in this case, by the name of the package file plus its extension (*.gfn* or *.zip*). If a GUI implementation is available for the package too, we can skip the loading part, and simply invoking in Gretl windows its related commands[25].

The package I am going to present is named "`bma_glm`" and performs, as already mentioned, RJMCMC for GLMs introducing parallel computing in MCMC simulations, with the Gelman and Brooks multivariate measure as additional proof of convergence. The contributions of parallelization are substantial since, on the one hand, long simulations can be split into smaller units with a remarkable boost in CPU time and on the other hand, a better exploration of possibly huge parameter and model spaces is encouraged too. Notice, moreover, that parallelization in BMA or selection procedure is not dealt by other competing software packages, making this contribution even more interesting.

The package `bma_glm` includes a main public function with several private ones which handle specific part of the former. In the rest of the section I will briefly describe the core function and the more relevant private ones. A final remark concerns the implementation of MPI (Message Passing Interface): in Gretl, MPI can be invoked via external softwares such as "Open MPI" or "MPICH" and the usual way of compiling a code needs to be adjusted. We will see the main modifications, but any details can be found in the Gretl MPI Guide (Cottrell and Lucchetti, 2017).

2.5.1 The public function

```
function bundle bma_glm(series y            "dep variable",
                        list X              "list of regressors",
                        int type[1:4]       "link function",
                        matrix pr_mean      "prior mean",
                        matrix pr_var       "prior var",
                        int prior_mod[0:1]  "model prior type",
                        matrix Phi          "parameter of prior on beta",
                        bool const_case[0]  "const prior distribution",
                        int kernel          "movement kernel q(M_j|M_i)",
                        int change_var[1]   "numb of variable to change",
                        scalar resamp[0]    "resampling moves",
                        int threads[1]      "number of cores for mpi",
                        int nrep            "iterations of MCMC",
                        int burn            "burn-in sample")
```

[25]For any details we refer to the Gretl Guide (Cottrell and Lucchetti, 2018).

The function bma_glm produces as output a *bundle*, that is a collection of Gretl elements, whose principal ones are are the matrix of sampled parameters (the rows correspond to the regressors, the columns to the iterations), the sampled models, as well as the correspondent of (1.20) and (1.21) provided by the sample moments[26].

The inputs, despite the short description provided inside the quotation marks are here examined in depth:

- y and X are the dependent variable and the list of covariates, wheres the integer type identifies the link function used for GLMs, available until now:
  - type=1 probit link (binary);
  - type=2 logit link (binary);
  - type=3 cloglog link (binary);
  - type=4 log link (Poisson).
- the matrices pr_mean and pr_var identify the prior mean and prior variance of the parameter β, following (2.8): pr_mean is a $k \times 1$ vector where k is the number of covariates in the full model; whereas pr_var is 2×1 vector, whose first entry should be a integer value equal to 1 or 2 identifying the kind of covariance matrix, 1 for a ridge prior cI, 2 for a Zellner one $g(X_i^T X_i)^{-1}$; the second entry instead specify the shrinkage parameter c or g. The user should specify expressly both values, since no default choices are provided inside the function.
- prior_mod and Phi are respectively the typology of prior distribution on models and the related parameter: two alternatives are available, prior_mod=0 is a uniform distribution, i.e. the prior probability of a model M_i is $P(M_i) = \frac{1}{|\mathbb{M}|}$, where the denominator is the cardinality of the whole model set. prior_mod=1 identifies the Binomial prior, where Phi if scalar, is a common probability of relevance for the variables; if in matrix form it allows to attach to each singular regressor the probability of being significant[27]. The possibility of choosing focus regressors, i.e. variables always included in each model, is not expressly provided inside the function, at least at the moment, but an easy alternative is to place high probabilities of inclusion in the entries of Phi representing the focus regressors.
- const_case represents, if 0 a diffuse prior on the constant term of the regressors; if 1 an informative distribution; in this case the constant is treated as another regressor, so its information should be included in pr_mean. When the diffuse prior is used, the other regressors are centered by default if a Zellner prior is used or standardized in case of ridge prior, the overall prior structure on the parameter is the one in (2.8), with $h = 100$. The constant term is always added, so its inclusion in the initial list X is optional as well as in pr_mean.
- kernel can assume three values ranging from 0 to 2:

[26]These are the main components, we can find other ones with no practical interest except for recalling the initial set-up implied.

[27]We allow probability in the range $(0, 1)$. We exclude the extreme values in order to prevent computational issues.

- kernel = 0 standard symmetric and uniform proposal;
- kernel = 1 uniform proposal with addition, deletion and switching;
- kernel = 2 Shotgun Stochastic Search

The additional value `change_var` defines the maximum amount of changing variable in case of `kernel`=1;

- the scalar `resamp`, if different from 0, allows for the resampling scheme as sketched in Section 2.3.1.
- the integer `threads` is simply the number of threads to use for MPI. Additional details will be provided in the dedicated subsection.
- `nrep` and `burn` represent, respectively, the total number of iteration (per core) of the MCMC and the burn-in sample (per core).

Finally, notice that the numbers in square brackets defined in the above function such as `type`[1 : 4], are the minimum and maximum admissible value for the correspondent quantity. When a single entry is provided such as `threads`[1] this has to be intended as the default value.

2.5.2 Private functions: the model ID set-up

Since a nesting framework is analyzed, each model could be represented as a binary vector, whose length is equal to the total number of covariates provided in the full specification and each single element corresponds to a variable: if a regressor is included in the current specification the related entry is 1, otherwise 0. For example consider a four regressor set-up $\{x_1, x_2, x_3, x_4\}$, the model $\{x_1, x_3\}$ can be identified as $\begin{pmatrix}1 & 0 & 1 & 0\end{pmatrix}$.

The binary vector represents, thus, the structure of the model, but has another peculiarity: it can be also seen as a binary number; following the previous example the model containing x_1, x_3 is univocally defined by the base-2 number 1010.

Each binary number can be converted into a decimal one, with the advantage of dealing with a more manageable representation: in the package an array of 2^k bundles is built given k total regressors, each one labeled with an integer ranging from 1 to 2^k. Each bundle, hence, corresponds to a specific model which can be invoked using the name of the array and the correspondent decimal representation as in MOD[decimal id], where MOD is the name chosen for the array. We can actually understand the decimal notation usefulness only considering the fact that it allows for a simple and immediate representation of a model as a bundle; whereas the use of bundles is motivated by the fact that they store model specific values avoiding the need to compute them every time the model appears. The model-bundle strategy here proposed works extremely well when the total number of covariates $k \leq 30$, however it seems to suffer from slowdowns or even potential out-of-memory issues when larger and larger regressor matrices are used. A solution is currently under development.

The private functions `convert(matrix model)`, where the matrix "`model`" is the binary vector representation and `reconv(scalar numb, scalar n)` with "`numb`" as the decimal representation and "`n`" the number of total regressors, are the ones used, as the name suggest, to convert a binary vector into a decimal number and vice versa.

Finally, with the function `fillthebundle(MOD, model, numb, ...)` literally initializes the model-bundle selected via the identifiers `model, numb` in the array MOD with its information such as the related regressors, the prior mean and variance of β adapted to the corrected dimension, and also the initial estimate of (2.4) and (2.3). The bundles are "filled" only the first time they appear as proposal, alongside the MCMC proceeds: in this way every useless repetition is avoided with obvious computational benefits.

2.5.3 Private functions: the RJMCMC

After an initial check-up routine which includes collinearity checks, adjustments in case of diffuse constant and the set-up for the parallelization[28], the entire body of RJMCMC dynamics is executed entirely inside the private function `BMA()`.

The procedure follows exactly the one provided in section 2.3.1: a loop of `nrep` iterations is run, where a proposal new model is chosen via the function `modelmove(¬ )`[29]. The sampled β for the new model is computed through (2.6): the correspondent private function is named `b_born()`. Finally the accept-reject probability is defined as `rho=xmin(1, alpha)`, where `xmin` is a built-in function which computes the minimum of the two quantities provided inside the parenthesis; `alpha` represents the Metropolis-Hastings accept rate[30] calculated with the private function `ratio()`.

When the proposed move is accepted the related model and parameter are stored inside matrices containing the sampled values, provided the burn-in time is over; when the move is rejected a resampling step is available with the option `resamp¬ =1`. This task is accomplished via the `sampling` function, which simply execute a single iteration of Gamerman MCMC; in this case the sub-function `A_R` defines the Metropolis-Hastings ratio.

2.5.4 Private functions: the model dynamics

The method for choosing the new proposal model is defined through the function `modelmove()`, in particular according to the value of the integer `kernel`, one of the following proposal kernels is used:

- `kernel=0` implies the standard symmetric kernel for MC^3 proposed by Hoeting et al. (1999); no additional explanations are required since the definition of a new model is obtained by choosing, from a uniform distribution, one of the entries of the binary vector `model`, i.e. the current specification, and setting the related element to 0, if 1, and vice-versa. Since the resulting proposal kernel is symmetric, the ratio `prop_ratio` defined as $q(M_i|M_j)/q(M_j|M_i)$ in (2.7), is equal to 1.

- `kernel=1` represents the uniform movement kernel with addition, deletion and swapping of at least `change_var` variables. In this case the auxiliary function

[28]All of them are run in the initial part of the public function `bma_glm`.
[29]See the next subsection for further details.
[30]The first element of (2.7).

`Unif_II` is used, where at an initial step the actual number of variables to change is chosen from a uniform distribution; then the move is selected between the available ones[31], using again a uniform distribution. The final proposal model is randomly drawn among the possible specifications obtained from M_i with the selected move and selected number of changing variables, assuming each possible specification as equal probable.

- `kernel=2` corresponds to the Shotgun Stochastic Search by Hans et al. (2007). An *ex-ante* computation of the score for every possible specification is accomplished via MPI: the model space is split into classes according to the number of cores of the machine, then, in parallel, the `score()` function attaches to each model in the class the related $P(\hat{\beta}, M_i|y)$, where $\hat{\beta}$ is simply the Frequentist estimator[32]. In the original framework the score function computes the posterior model distribution, however, since the joint distribution of parameter and model is the new target, in this particular context $P(\hat{\beta}, M_i|y)$ has been chosen as new score[33]. Finally, in the `modelmove` function, the neighborhood of the current model is built using the function `neighbor` by following the theoretical framework of SSS, where the score attached to each model is normalized with the total score of the neighborhood; the proposal is then chosen accordingly.

2.5.5 MPI implementation

MPI implementation in Gretl requires the preliminary installation of a suitable MPI package, such as *Open MPI* or *MPICH*, in addition to some specific platform requirements which can be consulted in the Gretl MPI guide.

Once the parallelization is enabled, this can be executed either directly invoking in a shell prompt the command `gretlmpi` and the selected file to be parallelized, or indirectly, that is inside a Gretl script with the MPI command block `mpi...end mpi`.

The main differences lie in the fact that using the first alternative implies that the whole content of the script has to be replicated in each thread[34] and the MPI invocation needs to be executed outside the Gretl environment; whereas in the second one only the part inside the MPI block is run in parallel and the execution is not different from how any other Gretl script is run. For this reason the indirect application is particularly suitable when using parallelization inside a Gretl function, as in our case.

[31]When all three dynamics are not possible, the choice is restricted to only the ones available (uniformly again).

[32]Major details for MPI application can be found in the next subsection.

[33]Alternatives are under investigations.

[34]I recall very briefly the distinction between cores and threads: the core is the physical processing unit, the thread is the logical one: the term should be interchangeable as the physical core is also the logical one, but when technological progress did not allow for multiple core processors, the necessity of speeding up tasks led to the concept of *hyperthreading*. The idea is to split the core into two logical processors, and so enabling parallel procedures (any detail of how this is performed is skipped) even if we are using a single core machine. With the advent of multicore processor, the hyperthreading phenomenon has continued to be used, so a device with c cores, has $2c$ threads. In Gretl and Open MPI we refer explicitly to the threads of the machine. For any detail consult Cottrell and Lucchetti (2017).

Before analyzing the MPI features of the package, we will provide a short overview about how MPI should be implemented since an *ad-hoc* set up is requested and some "new" gretl MPI functions which will be (ab)used in our code are necessary.

Any parallelizable process, which starts with the declaration of the MPI block, in general, requires a preparation phase where variables need to be initialized in the root thread, operation which is available in our context through a conditioning with an `if` statement of this kind:

```
if $mpirank==0
  ...
endif
```

where `mpirank` represents the ID of the thread, where it is assumed that the root corresponds to ID $= 0$. Then the data are sent or shared across threads through the commands `mpibcast()` and `mpiscatter()`, where the former simply share either a scalar or a matrix to all the other threads; the latter splits a matrix into submatrices, each one destined to a thread[35]. Once these steps are accomplished all processors execute some work in parallel and finally the output of each one is processed again in the root which collects the results and apply a synthesis via the command `mpireduce`[36].

Notice that the parallelization of MCMCs poses a practical problem related to the generation of random numbers: using the same seed for random numbers ends up in replicating the same results in every core; whereas different seeds for each core using the standard Pseudo-Random Number Generator (PRNG) process may lead to producing sequences of numbers with arbitrary dependency. In Gretl the problem is handled by using the DCMT mechanism (Dynamic Creation of Mersenne Twisters) so that each MPI process gets its own, independent PRNG[37].

In the package two application of MPI are available: the vertical MCMC, which parallelizes the RJMCMC and then aggregates the result; and the preliminary computation of the score for the Shotgun Stochastic Search.

The framework of the vertical MCMC is exactly the same as the one introduced in section 2.4: in a preliminary phase the file `initial` is built, i.e. a bundle with all the inputs of `bma_glm` which is then broadcasted to all processors. Each thread, then, replicates the function `BMA` which contains the whole transdimensional model averaging procedure. Once concluded the MCMC, in each thread the sample mean and the within variance of the sampled parameters are computed before sending the whole information[38] to the root process which simply concatenate each result into a synthesis matrix.

In the end we will have three different matrices, each one stored in a different file:

- the matrix `sampled.mat` of dimension $2k \times$ `(nrep-burn)threads`, where k is the number of regressors in the full model. The first k rows represent the

[35]There are also other commands such as `mpisend` which allows to send an object (scalar, matrix) to a specific core, however this is not used in our code.

[36]`mpireduce` allows for vertical or horizontal concatenation of matrices; summation or product of scalars or matrices. For details consult the Cottrell and Lucchetti (2018).

[37]See Cottrell and Lucchetti (2017).

[38]Sampled parameters and sampled models.

sampled parameters obtained in each parallel process stacked horizontally; the other k rows are the correspondent sampled models in binary representation;

- a matrix named `id.mat` which contains in the first row the decimal ids of the sampled models (repeated only once), in the second row the number of times they were accepted.
- the matrix `synth.mat` which is obtained by vertically stacking a synthesis matrix provided in each thread. The synthesis matrix has the following structure:

$$\left(\begin{array}{c|c|c} \underbrace{\quad\bar{m}\quad}_{mean} & \underbrace{\quad V\quad}_{cov} & \underbrace{\quad p\quad}_{pip} \end{array}\right)$$

  where $\bar{m}$ is the sample mean of the parameters, V the (within) covariance matrix and p the vector containing the posterior inclusion probability (PIP), i.e. the number of times the parameter appears over the total amount of iteration minus the burn-in[39].

All the three matrices are stored in the final bundle, but both `id.mat` and `synth¬` `.mat` are also used in order to print the final result via the function `printres`. We will have a look on the output of this function in the empirical illustration section.

The second application of MPI provided in the function concerns Shotgun Stochastic Search. A vector named `sss.mat` is built with the computation of the score for each possible specification, the order of its elements follows the model decimal id, i.e. the first element is model with `id=1`, the second one the model with `id=2` and so on. Finally inside the `BMA`, if SSS is invoked, the entry of `sss.mat` is added to the relative model bundle.

2.5.6 A brief analysis of currently available RJMCMC and BMA packages

The implementation of RJMCMCs is quite challenging to be handled in practice, because the generality which is common requested for software functions is at odds with the details required for the transdimensional function $g()$: in other words the choice of a particular proposal is often problem-specific and the possibility of building an effective RJMCMC needs to face the trade-off of choosing some common proposal functions $g()$ with the cost of restricting the application to only some families of problems or building a very general framework with potential great applicability, but with an insane amount of complexity and often high computational costs.

For this reason RJMCMC softwares are quite rare, and apart from the original script AUTORJ by Green (2003) written in Fortran or its modified version due to Green and Hastie (2009) written in C, which, however, were closely related to only some common model exploration problems, other applications were not directly available since the recent R package "rjmcmc" by Gelling et al. (2018). This package, in particular, exploits the modified framework by Barker and Link (2013) which converts the Metropolis-Hastings framework into a Gibbs sampling one. Moreover

[39]It is obtained as the sum by row of the sampled model matrix in binary form. In this case we have the entry equal to 1 which represents presence; the entry 0 absence.

it has potentially no limits in the definition of the transdimensional function $g()$, since it is possible to specify any functional form (in symbolic notation) and the related Jacobian is numerically computed. The package, therefore, aims to be as general as possible, but at the same time some soft options are applied to maintain "low" computational costs: the models to consider inside the algorithm (M_i) needs to be specified a priori leading to a huge limitations in terms of quantity of explorable models, especially if nesting is assumed. Additionally, some preliminary information needs to be attached, remarkably a sample from the parameter posterior distribution under each specification: the package itself, however, does not provide such individual sampling schemes, as a consequence external functions which deal with this issue are necessary, making the whole "rjmcmc" dependent to other MCMC sampler algorithms.

The software implementation here presented, instead, is for sure not equally general, but aims to perform as efficiently as possible model explorations in nesting problems without the necessity of external samplers: GLMs are the core of the analysis, a good amount of flexibility is guaranteed thanks to the automated framework in subsection 2.3.1, and computational efficiency, which is a primary concern, is dealt with parallelization. Of course RJMCMCs can be extended to more complex scenarios, but for what concerns this work, RJMCMC is just a tool for analyzing model averaging in standard non-linear problems which can be encountered in Economics, hence this restriction to the class of GLMs seems to be well motivated. Furthermore, the limitation in terms of explorable models which derives from the package by Gelling et al. (2018) is far from being negligible in our context, as a consequence I have favored the possibility of analyzing wide model spaces "at the cost" of considering only some GLMs, providing an independent sampling scheme which does not rely on preliminary posterior draws or two-step procedures.

As regards the available softwares for BMA, many have been developed for linear models: some examples are the R packages "BMA" by Raftery et al. (2014), which exploits both Occam's window or MCMCs with BIC approximations for model posteriors; the package "BMS" by Zeugner and Feldkircher (2015) which, in turns, performs model averaging using a MCMC on the model space with great flexibility in the choice of prior or model proposal kernel; and finally the package "BAS" (Clyde et al., 2012) for Bayesian Adaptive Sampling[40]. In Gretl, Bayesian Model Averaging is implemented in the package BMA by Błażejowski and Kwiatkowski (2015), which deals with linear models using Markov Chain Monte Carlo Model Composition with the addition of jointness measures; Błażejowski and Kwiatkowski (2018) introduces the BACE package, which runs Bayesian Average of Classical Estimates (Sala-i Martin et al., 2004) for linear and time series (Augmented Distributed Lags) models.

As a matter of fact both R packages "BMA" and "BAS" can be applied to GLMs too, but several simplifications are used such as the use of frequentist estimators (Maximum-likelihood) instead of appropriate Bayesian counterparts or the lack of appropriate MCMC schemes: "BMA" provides the Occam's Window solution for GLM model averaging using BIC approximation for model posterior, with the MCMC alternative for the solely computation of Bayes Factor; "BAS", instead, performs the so-called *Bayesian Adaptive Sampling* (Clyde et al., 2012), which examines model

[40]A wide and complete analysis of the three packages is performed by Amini and Parmeter (2011)

space according to a tree search, demanding the use of MCMCs only for a preliminary definitions of model posteriors (standard MC^3 with Laplace approximations are used)[41].

The Gretl package that I have presented, provides a full exploration of both parameter and model space, a feature not considered in other packages, with the possibility of specifying some more general parameter prior structure and model proposal kernel. Parallelization, finally, improves considerably computational times and in particular this represents a true novelty since its contribution in BMA software functions has not been considered so far.

2.6 Empirical example

In this section I will illustrate an application of the package `bma_glm` using a dataset from Mroz (1987): I will focus on a probit estimation of the female labour force participation in 1975. The dataset contains information about 753 women, where our dicothomic dependent variable is named `LFP`, which is equal to 1 in case of labour participation; 0 otherwise.

The set of regressors used is:

- `KL6`, the number of children under the age of 6;
- `WA`, wife's age;
- `WE`, wife's education attainments, in years;
- `HA`, husband's age;
- `HE`, husband's education attainments;
- `HW`, husband's hourly wage;
- `MTR`, marginal tax rate facing the wife;
- `UN`, unemployment rate in the country of residence;
- `CIT`, dummy variable - 1 if living in a a large city, 0 otherwise;
- `AX`, actual years of wife's previous labour experience.

The output of a standard Probit estimation is the following:

[41]A comparison between BMA, BAS and the current `bma_glm` package is not provided because each one performs a different model averaging strategy which is differently affected by prior choices, and only asimptotically they should lead to similar conclusions. Moreover, as regards BMA and BAS, Amini and Parmeter (2011); Błażejowski and Kwiatkowski (2015) shows some comparison in linear models where under a similar set-up both packages produce quite different results with respect to the benchmark.

```
Model 1: Probit, using observations 1-753
Dependent variable: LFP
Standard errors based on Hessian

             coefficient   std. error      z      p-value
  --------------------------------------------------------
  const       0.238379     0.0523348     4.555    5.24e-06 ***
  KL6        -0.813479     0.117719     -6.910    4.83e-12 ***
  WA         -0.0604316    0.0145616    -4.150    3.32e-05 ***
  WE          0.117676     0.0299991     3.923    8.76e-05 ***
  HA         -0.00364750   0.0141875    -0.2571   0.7971
  HE         -0.0522828    0.0235334    -2.222    0.0263   **
  HW         -0.0897814    0.0198192    -4.530    5.90e-06 ***
  MTR        -5.57322      1.04103      -5.354    8.62e-08 ***
  UN          0.00303028   0.0170686     0.1775   0.8591
  CIT         0.0555055    0.117783      0.4713   0.6375
  AX          0.0693860    0.00752985    9.215    3.12e-20 ***

Mean dependent var   0.568393   S.D. dependent var   0.495630
McFadden R-squared   0.239902   Adjusted R-squared   0.218537
Log-likelihood      -391.3541   Akaike criterion     804.7083
Schwarz criterion    855.5730   Hannan-Quinn         824.3039
```

I would like to point out right now, that the example has a statistical motivation and meaning as the main interest is in identifying the most relevant variables and how the relative coefficients are shrunk in the model averaging procedure. For an economic interpretation we should consider the impact of endogeneity, a well known problem in this kind of studies[42].

Let us assume the following set up for the function: a diffuse prior on the constant term (`const_case=0`), a prior on the parameter defined as

$$\beta_i \sim N(\underline{0}, n(\tilde{X}_i^T \tilde{X}_i)^{-1})$$

where n is the total number of observations; $\tilde{X}_i$ is the matrix of demeaned regressors in model M_i[43]. The `dynamics` is set to 0, but similar conclusions are reached with different dynamics as well; `resamp=0` and `threads=1`. Finally the number of iterations and burn-in are respectively set to 100000 and 10000.

The algorithm is run in a Linux Debian Machine (server), with 20 physical processors.

[42]An economic application is provided in the next Chapter.

[43]Remember that in case of diffuse prior on the constant term, the constant is always included, and the overall prior distribution on the parameters is provided by (2.8).

The Gretl version is `2018_d git`, with Open MPI 1.6.5 for parallel computing. The correspondent Gretl script is:

```
open mroz87
#since I use diffuse const, no need to include it
list X= KL6 WA WE HA HE HW MTR UN CIT AX

modeltype = 1 # 1:probit, 2: logit
prior_mean = zeros(nelem(X), 1) #prior mean for coefficients
prior_var = {2,$nobs} # prior covariance for coefficients -
zellner prior UIP

prior_dist = 0 # 0 = uniform distribution for models

constant_special = 0 #diffuse prior for const
dynamics = 0 #MCMCMC
change_var=0 #max number of changing variable per model move
resamp = 0 #within model move
threads = 1

n_iter = 100000
burn_in = 10000

b=bma_glm(LFP, X, modeltype, prior_mean, prior_var, \
  prior_dist, phi, constant_special, \
  dynamics, change_var, resamp, threads, n_iter,burn_in)
```

The visual output of the function is the following:

```
  ------------------------------------------------------
  Bayesian Model Averaging with Generalized Linear Model
  ------------------------------------------------------
  Type of specification: Probit model
  Model Prior: P(M) ~ Uniform
  Model dynamics: MCMCMC - add/delete (1)var
  Prior on const: Diffuse const
  Resampling allowed: No
  MPI - threads: 1
  Number of iterations/burn-in: 100000/10000
  Elapsed time (in sec): 436.869
```

```
------------------------------------
Overall sampling statistics
            mean    stderr       pip
const    0.23398   0.05131   1.00000
  KL6   -0.81784   0.11592   1.00000
   WA   -0.06223   0.00878   1.00000
   WE    0.09487   0.03699   0.96013
   HA   -0.00022   0.00446   0.09808
   HE   -0.02009   0.02882   0.40062
   HW   -0.09095   0.01887   0.99997
  MTR   -5.44952   1.02652   1.00000
   UN    0.00020   0.00396   0.04893
  CIT    0.00214   0.02845   0.05391
   AX    0.06958   0.00739   1.00000
------------------------------------
Best specifications:

Model_615: P(M|D)=0.460167
const KL6 WA WE HW MTR AX

Model_631: P(M|D)=0.320544
const KL6 WA WE HE HW MTR AX
------------------------------------
```

where `mean`, `se` are respectively the model averaged mean (1.20) and the model averaged standard errors obtained from (1.21) obtained using sample counterparts; `pip` refers to the posterior probability of inclusion. Top models[44] with respect to their posterior model probability are also provided.

What can be inferred by comparing the two outputs perfectly conveys the idea of model averaging: starting by considering the probability of inclusion, it is possible to conclude that the most significant variables from the simple Probit estimation are also the ones which exhibits a `pip` equal or near to one. The related coefficient estimates are close to the model averaging expected values too. Less important variables have their model averaging means reflecting a shrinkage effect, which expresses the model uncertainty issue in a more evident manner. Finally, the top specifications which are encountered in the model averaging procedure are also the ones recognized as best ones according to Frequentist measures: `Model_615` is the best specification in terms of BIC and hypothesis testing with significance level at 0.01; whereas `Model_631` is the second best model according the BIC and the best one using tests at size 0.05 or Akaike's IC.

A possible extensions could be the analysis of how different priors (on models and parameters) affect the result, but for the moment this exceeds the scope of this part.

[44]By top models we mean every specification whose posterior probability exceeds the 10%.

2.6.1 The importance of being parallelized

The previous example performed the RJMCMC on a single physical core: since one of the main aim of this Chapter is ascertaining the importance of parallelization in MCMCs, it is surely interesting to replicate the previous example allowing, this time, for parallelization, in order to monitor the effective gain in CPU time.

Two different scenarios are proposed: in the first one I split the number of iterations per core up to a maximum of eight cores[45] assuming a *fixed* burn-in period, i.e. despite the fact that parallelization is enabled, in each core the same burn-in as the single-core experiment is used. In this case, in order to have the same amount of sampled units of the single-core experiment, it is necessary to balance the number of iterations per core. A fixed burn-in scenario represents a quite conservative choice because it assumes the ignorance of the real convergence speed of the chain, and imposing an equal amount of burn-in time as the one that would be used in a single long chain tends to assure certainly the stationarity of the sampled values.

In particular, I compare the results in terms of time, posterior model distributions of top models and Gelman and Brooks multivariate measure across a single core, two cores, four cores and eight cores simulations. Individual estimates of posterior moments are not provided, but the definition of the Gelman and Brooks statistic should assure that, if convergence is reached, quite the same conclusions as the long chain case are produced.

The results are presented in the following table:

cores	c=1	c=2	c=4	c=8
Iterations per core	100000	55000	32500	21250
Elapsed time (sec)	436.860	276.992	170.022	142.984
BG statistic		1.004	1.015	1.014
$P(M\|D)$				
Model 615	0.46	0.456	0.452	0.458
Model 631	0.32	0.32	0.316	0.311

Firstly, convergence is reached as we can note from the Brooks and Gelman (BG) statistic and from the posterior model distributions of the two top models. Once established this condition, it is possible to compare CPU time: the improvement of using two parallel chains instead of a single one is approximately 36%, whereas applying four cores or eight cores lead, respectively, to save the 60% and the 70% of the single-core CPU time, which is actually a huge enhancement. Of particular interest is the fact that the relative advantage of using more cores tends to manifest an upward tendency until a maximum point (four cores in the experiment) and then a downward one: this is in line with the current literature about vertical MCMCs when using a fixed burn-in time.

The second experiment introduces a *flexible* burn-in time equal to the 10% of the total iterations per core: the idea behind this assumption is that, in assuming

[45]Only physical cores are used and the analysis uses a maximum of eight cores in order to maintain the example replicable in common devices.

fixed burn-in time, we are induced to overestimate (or maybe underestimate) its appropriate value, so it may be appealing to ask whether using smaller burn-in leads to similar result in terms of convergence and whether there may be improvements in computational time[46]. As we can see from the following table:

threads	c=1	c=2	c=4	c=8
Iterations per core	100000	50000	25000	12500
Elapsed time (sec)	436.860	239.606	117.848	75.619
BG statistic		1.011	1.014	1.031
$P(M\|D)$				
Model 615	0.46	0.451	0.46	0.45
Model 631	0.32	0.309	0.31	0.326

even in this case convergence is reached (this is actually a symptom of quick convergence), so turning the attention to the computational time, we can end up with a time reduction of over 70% and 80% using respectively, four and eight cores, with respect to the single-core performance. This certainly derives from the fact that we do not need to balance iterations for the fixed burn-in, but the underlying idea is that if a smaller burn-in time can be employed and the convergence is fast, then parallelization leads to even bigger advantages. Notice that, even in this case, the relative advantage of using more cores exhibits a similar tendency to the one sketched before, but less evident.

It is clear from these examples that parallelization can drastically improve computational time as long as the convergence is quick and the requested burn-in is small. Even in the case of a fixed burn-in, which somehow represents a “safe” experiments in terms of convergence, the advantage is evident. What could be really appealing is determining whether such good results hold for bigger datasets, and, as we will see through the example of the next Chapter the answer is positive.

2.6.2 Summary

The package presented provides an application of Bayesian Model Averaging for Generalized Linear Model with the aim of making simple, even for a common user, the performance of an automatic model building procedure which accounts for model uncertainty. The potentiality of the model averaging procedure has been underlined, but its use in microeconomic problems is still unexplored, so applications in this context could provide a new flourishing area of interest.

The benefits of parallelization, moreover, suggest a remedy for many common computational issues typical in MCMC simulations, so parallelization should be encouraged not only for a better analysis of the parameter and model space but also for accelerating the CPU time even if this is somehow not typical of vertical MCMCs.

Future improvements could be the addition of more link functions; additional proposal kernels for models; alternative approximation for the posterior moments of parameters; different prior specifications and the possibility of averaging in not nested scenarios.

[46]Assuming a burn-in time which is tuned inside the MCMC is not recommended following the argument by Rosenthal (2000).

Chapter 3

A BMA analysis of Propensity Score Matching

Economic applications of model averaging primarily concern forecasting, due to the higher accuracy of the estimates: accounting for model uncertainty leads to a more robust evaluation and it could be reasonable to ask if this effect persists even in other possible scenarios.

The idea that will be studied in this Chapter is exactly whether Bayesian Model Averaging induces a more robust approach in Propensity Score Matching in terms of stability in the treatment evaluation when the choice of variables is crucial: in these cases parsimonious models lead to different conclusions (sometimes very different) with respect to more general ones and a compromise estimator, e.g. model averaging estimator, seems to be necessary.

3.1 Introduction

Propensity Score matching methods, since the seminal paper of Rosenbaum and Rubin (1983), have become a very popular approach to evaluate treatment effects. In observational studies, it is a well known fact how the counterfactual is not identified, so a plausible solution could be using for this scope a group of individuals (control group), not necessarily from the same population of the treated units, but with some chosen characteristics particularly similar to the treated ones. Unlike exact matching methods, where similarity between treated and control units is ascertained by directly comparing some observable characteristics and performing the pairing only when these exactly corresponds in the selected treated and control individual, with possible problems whenever continuous variables are included in the comparison or many variables are considered, Propensity Score matching proposes an intriguing alternative by allowing the evaluation of the similarity using a scalar measure which summarizes those individual features. In particular, such a measure is represented by the Propensity Score, i.e. the conditional probability of being treated given the comparison characteristics, which is commonly estimated via parametric methods such as binary choice models (logit or probit). The consecutive matching phase is then performed on these values via nearest neighbor or stratification and the average treatment effect is obtained accordingly.

However, in building Propensity Scores little attention is commonly devoted to selecting the variables to be included in the binary choice model: these variables identify the characteristics of interest which should guide the definition of the matched group, but estimation is often carried out either on every available covariate, ignoring possible complications in some assumptions which are requested in order to guarantee the reliability on Propensity Score matching[1], or on a subset arbitrarily chosen by the practitioner with some "knowledge-driven" approach.

A different strategy could be applying some variable selection procedures, but focusing exclusively in the best specification would condition the further analyses on this choice; it is quite obvious how a problem of model uncertainty arises: if a single specification can capture some aspects of the reality, it is probable that other models can capture other aspects, so a choice which favors only a single specification is obviously misleading. To acknowledge this uncertainty some different analysis based on Bayesian Model Averaging (BMA), using the previously illustrated `bma_glm` package, are here proposed: the choice for a Bayesian framework derives from its flexibility and its inferential and model comparisons properties, which will lead us to build different treatment effect estimators, which will be potentially precluded in Frequentist Model Averaging methods[2].

This methodology, in particular, will be applied to evaluate the economic impact of the Italian tax credit reform (Decree Law 66/2014), which introduced a monthly wage increase of about 80€ for all employees with an annual gross income between 8145€ and 26000€. A tax reduction is supposed to encourage household consumptions, however the effectiveness of tax credit policy is a debated topic in the literature (Shapiro and Slemrod, 2003a,b, 2009). In particular, I will try to replicate the approach proposed by Neri et al. (2017), based on *propensity score difference-in-differences* estimation and in doing so I will show how actually the results are sensitive to model specification in the propensity score definition. The application, therefore, exactly conveys the model uncertainty importance, and how taking into account for this aspect can contribute to improve the quality of the estimation in terms of reducing the "arbitrariness" in the choice of variables to use. Notice, moreover, that application of BMA techniques in Propensity Score problems, especially in Economics, is a novelty since previous works concerned with this topic belongs to Biostatistics (Kaplan and Chen, 2014; Zigler and Dominici, 2014).

A clarification is, however, necessary: the adherence to a Bayesian analysis in the present work, is in part restricted to the solely Propensity Score definition, since BMA is adopted in the variable choice of the related binary model, but the final treatment effect is computed in a standard Frequentist fashion. A purist Bayesian perspective would also have computed this element according to Bayesian guidelines, but this would have been far beyond the scope of this work, whose primarily aim is to ascertain the consequences of model uncertainty and to find an easy way to deal with them.

The Chapter is organized as follow: in the next Section a quick overview on Propensity score matching (both Frequentist and Bayesian) and Model Averaging applied in this context is provided; then I will illustrate some economic background

[1] See next section.

[2] This comes from the fact that in FMA, the definition of model probabilities is not straightforward.

to the tax credit policy, and then compare the study by Neri et al. (2017) with the "replica" here proposed; finally I will illustrate the Bayesian Model Averaging analysis on the same study with a short discussion concerning the possible implications.

3.2 Propensity Score matching and uncertainty

3.2.1 A quick overview on Propensity Score

Let D denote an indicator of treatment assignment, which takes value one in case the individual is assigned, and zero otherwise; further, assume a vector of covariates $\mathbf{x}$ and an outcome dependent variable y: it is common practice to define the outcome variable for a treated unit as y_1, whereas y_0 for a control one. The propensity score $p(\mathbf{x})$ is defined as the conditional probability measure of being treated given $\mathbf{x}$,

$$p(\mathbf{x}) = P(D = 1|\mathbf{x})$$

which can be simply computed using a probit or logit model. Two particular assumptions needs to hold:

- the *overlap* assumption, which simply states that $0 < p(\mathbf{x}) < 1$;
- the *balancing* condition, which implies $D \perp \mathbf{x}|p(\mathbf{x})$.

The overlap assumption guarantees that for each treated case there is another matched untreated one with a similar $\mathbf{x}$, in other words, the treated and untreated subsamples overlap; the balance condition, instead, allows for a random assignment of the treatment for individuals with the same propensity score.

Moreover Rosenbaum and Rubin (1983) prove that, if conditional independence assumptions hold, i.e. $y_0, y_1 \perp D|\mathbf{x}$, then:

$$y_0, y_1 \perp D|p(\mathbf{x})$$

These conditions ensure the possibility of using the observed outcome of the "correct" control unit to impute the counterfactual of the correspondent treated one. However, once defined the propensity score measure, what remains to be done is exactly the choice of the correct control unit per treated; this is performed via matching methods, some common choices are:

- *nearest neighbor* matching, which pairs each treated unit with the correspondent control whose propensity score is closest. A tolerance interval (*caliper*) on the maximum distance between treated-control propensity score is often introduced in order to avoid bad matches[3]; traditional nearest neighbor is performed without replacement, however allowing this feature has often good consequences but at the cost of a reduced set of controls. The order in which the matching is performed could be a crucial aspect (Austin, 2014).
- *caliper/radius* matching, starts from the same idea of nearest neighbor with a caliper, but instead of pairing the treated with a single control, the match is performed with the k nearest neighbors within the caliper (*k-nearest neighbors*) or with all the units inside the caliper;

[3]In case no matches are found the treated remains unmatched and excluded from further analysis.

- *optimal* matching, where a loss function is attached to the overall matching scheme; this translates into action by forming matches which minimize not the individual loss (like in nearest neighbor if we consider the distance as a loss function) but the overall one. The difference with traditional methods is particularly evident in optimal pairwise matching: once a matched coupled is established this one can be further modified by reassigning to the treated a new control unit, if for some other treated the previous control provides a reduction in the overall loss;

- *stratification* matching, in which the empirical distribution of the propensity score is split into a set of intervals (quantiles): Rosenbaum and Rubin (1983) show how the units assigned respectively to treated or control groups can be used efficiently to estimate the related treatment effect inside each stratum. The number of optimal strata is often set to 5 according to the argument by Rosenbaum and Rubin (1983), even though this choice may be rectified if the number of treated or control units is particularly small in some intervals. If balance property is not satisfied, a very common solution for stratification matching is the addition of dummies or interaction terms in the set of covariates for the propensity score model until the condition is verified: for this reason stratification matching is not particularly suggested in variable selection experiments.

The final step of the propensity score matching procedure is the computation of the treatment effect γ; following the matching schemes previously introduced, a general and comprehensive estimation of the average treatment effect on the treated, could be:

$$\gamma = \frac{1}{N_1} \sum_{i \in D=1} (y_i - \frac{1}{N_{0,i}} \sum_{j \in D=0} y_j) \tag{3.1}$$

where the indexes i and j refer respectively to the *i-th* treated unit and the *j-th* control; N_1 is the number of matched treated units and $N_{0,i}$ the number in the comparison group corresponding to the *i-th* observation[4].

An alternative method is to run a simple regression on the final group defined via the matching and retaining as estimate of the treatment impact $\hat{\gamma}$:

$$y = \mu + \gamma D + \varepsilon \tag{3.2}$$

where μ is the intercept and ε the usual error term[5].

3.2.2 The Bayesian Treatment evaluation

In a Bayesian perspective, the main differences with respect to the Frequentist framework concern the propensity score estimation and the evaluation of the treat-

[4]In case of radius matching, a weight to each control inside the radius could be attached with the aim of imposing different contribution according to the distance from the treated unit; in case of stratification matching (3.1) is, instead, computed per each stratum and then averaged to obtain the final average treatment effect on the treated.

[5]Even in this case additional attention is required in case of stratification or whenever more than one control is attached to each treated. Weighting the observations in the linear model could be an easy solution.

ment; the matching phase is often the same. As for the first element, the simple binary model estimation is substituted by an MCMC simulation which samples the coefficients of $\mathbf{x}$, let us say β. In this way a sample of parameters can be used to compute directly a sample of propensity scores: if the initial MCMC leads n sampled parameters, we will end up with n propensity scores, each one used to build a matching scheme. The final computation of the output model can be then performed either using a simple Frequentist estimation on (3.2) for each obtained propensity score and aggregating accordingly the result (An, 2010) or in a fully Bayesian fashion with a MCMC on each output model too (Kaplan and Chen, 2012).

In the first case we simply use, as treatment effect estimator over the whole propensity score distribution, the sample mean of $\hat{\gamma}_i$, where $\hat{\gamma}_i$ is the OLS estimate of the treatment effect in the output model corresponding to the *i-th* propensity score. The motivation of this choice lies in the assumptions that the treatment $\hat{\gamma}_i$ could be thought as a posterior mean of the parameter γ conditional on the data (the outcome variable y and the matching variables $\mathbf{x}$), the treatment assignment D and the parameter β (which actually represents the impact of the propensity score), in math terms $\hat{\gamma}_i = E(\gamma|\beta, y, \mathbf{x}, D)$. The overall treatment estimator $\hat{\gamma}$, therefore, is nothing but the the posterior mean of γ, given the outcome y, the covariates $\mathbf{x}$ and the treatment D:

$$\hat{\gamma} = E(\gamma|y, \mathbf{x}, D) = E(E(\gamma|\beta, y, \mathbf{x}, D)|y, \mathbf{x}, D)) = \frac{\sum_i^n \hat{\gamma}_i}{n} \tag{3.3}$$

where n is the total amount of sampled propensity scores; in a similar way the variance is computed as:

$$V(\gamma|y, \mathbf{x}, D) = E[V(\gamma|\beta, y, \mathbf{x}, D)|y, \mathbf{x}, D] + V[E(\gamma|\beta, y, \mathbf{x}, D)|y, \mathbf{x}, D] \tag{3.4}$$

where $V(\gamma|y, \mathbf{x}, D)$ is the posterior variance of the treatment effect, whereas $V(\gamma|\beta, y, \mathbf{x}, D)$ is the posterior variance conditioned to the propensity score via the parameter β, which following An (2010) is simply computed as the variance $\hat{\sigma}_i^2$ of the OLS estimator $\hat{\gamma}_i$ in the *i-th* outcome model. In other words:

$$V(\gamma|y, \mathbf{x}, D) = \frac{\sum_i^n \hat{\sigma}_i^2}{n} + \frac{\sum_i^n (\hat{\gamma}_i - \hat{\gamma})^2}{n-1}$$

where the first element of the right hand-side is the sample mean of the OLS variances of the treatment effects, and the second one the sample variance of the treatment estimates.

The fully Bayesian alternative, instead of assuming $\hat{\gamma}_i = E(\gamma|\beta, y, \mathbf{x}, D)$ and $V(\gamma|\beta, y, \mathbf{x}, D) = \hat{\sigma}_i^2$ runs *for each* sampled propensity score a separate MCMC for γ in the outcome model, conditioned on that particular propensity score. In this way both $E(\gamma|\beta, y, \mathbf{x}, D)$ and $V(\gamma|\beta, y, \mathbf{x}, D)$ are estimated by the related sample counterparts, and the computation of the overall treatment effect and its variance follow eqs. (3.3) and (3.4).

So far, I have considered a *sequential* estimation of the treatment effect which starts from the propensity score and ends with the outcome model by the way of the matching, but it is worth noticing that in the Bayesian analysis of Propensity Score and treatment evaluation an alternative and somehow intriguing approach is considered by McCandless et al. (2009, 2010) and partially by An (2010). Following the pioneer idea by McCandless et al. (2009), it is possible to jointly modeling propensity score and outcome model via a "simultaneous" sampling scheme in which propensity score model affects the outcome one and vice versa, allowing for quantities which would have been commonly included only in one of the two, to affect both. Notably, the outcome variable y would directly enter into the computation of the propensity score distribution, which is then used to define the treatment effect γ. Despite the authors provide some good motivations for their methodology, the fact that treatment determines propensity score via the outcome is highly questionable: not only overlapping or conditional independence assumptions could be violated, but also additional sources of biases are introduced as the propensity score distribution changes with the outcome variable y.

3.2.3 Model Uncertainty issues

As previously anticipated, Propensity Score is often computed with little attention paid on the variables included in the underlying binary model: common routines range from the inclusion of every possible covariate, ignoring in this way their real importance in the matching scheme, to the selection via hypothesis testing or Information Criteria of a sufficiently parsimonious specification, with the obvious drawback of potentially not including determinant variables which will then invalidate the outcome model analysis.

In such a scenario, it could be meaningful to ask whether considering model uncertainty could lead to a better definition of the matching problem and consequently of the causal effect estimation: in particular, with k variables, 2^k models are available and we can assume that among them there is a particular one, M^* which is not known, but which reflects all of the desirable conditions and leads to a correct treatment effect estimation. M^* could not be a parsimonious model, but assuming this additional condition often guarantees a sufficient control on bias-variance trade-off. We could then average model-specific estimates according to a weight which reflects the model probability to be the closest to reality, with the benefit of avoiding any choice of a single specification, which can be seen as a guess about M^*.

However, since the whole treatment evaluation is often a sequential procedure, in which in a first phase the propensity score is defined and then in the following ones matching and outcome model are performed, from a practical perspective it could be meaningful to ask how model averaging quantities are used: we could compute propensity score using the model averaged posterior mean (leading to what is commonly referred in the literature as a *plug-in* estimator) and performing the subsequent phases on this specific values; or we could build a more proper model averaging estimator as a weighted sum of model specific $\hat{\gamma}_{M_i} = E(\gamma|y, \mathbf{x}_i, D, M_i), \quad i = 1..M$ which according to the Bayesian framework of the previous subsection, is the posterior mean of the treatment effect under a particular propensity score model M_i with covariates $\mathbf{x}_i$,

$$\hat{\gamma} = E(\gamma|y, \mathbf{x}, D) = \sum_{i}^{M} E(\gamma|y, \mathbf{x}_i, D, M_i) P(M_i|\mathbf{x}_i, D) \tag{3.5}$$

where the weight are posterior model probabilities $P(M_i|\mathbf{x}_i, D)$, and

$E(\gamma|y, \mathbf{x}_i, D, M_i)$ can be computed using the OLS estimator of the treatment in that particular outcome model or more appropriate Bayesian alternatives.

The choice is not obvious, but a very reasonable guideline is to perform different model averaging estimator and compare their results; a very brief overview of current model averaging literature in propensity score matching is, therefore, considered, before explaining which particular choices have been made in this work.

BMA in Propensity Score matching

As already state, the topic of model uncertainty in Propensity Score is still largely unexplored in the literature: two notably examples can be found in Biostatistics and are respectively Zigler and Dominici (2014) and Kaplan and Chen (2014).

Zigler and Dominici (2014) propose three different Bayesian Model Averaging techniques[6]:

- a standard MC^3 paradigm which relies on Probit models for both propensity score and outcome model (a binary outcome model is used) with additional regularities assumptions in order to obtain analytical posterior model distribution (Albert and Chib, 1993). The joint modeling of propensity score and outcome *à la* McCandless et al. (2009) is implemented;

- a *pseudo-Bayesian* MC^3 scheme where a canonical sequential treatment evaluation is performed, from the propensity score to the outcome model. The term "pseudo-Bayesian" derives from the fact that the joint procedure is not used;

- Stochastic Search Variable Selection (SVSS), with again joint modeling of propensity score and outcome model.

The matching is performed in each case via stratification and the treatment is obtained via (3.5). The authors obtain very promising result in simulation studies with respect to common selection methods using all of the three procedures. In empirical examples, instead, model averaging seems to recover similar treatment effect estimation as standard selection routines, but this outcome is mainly driven by the design of the experiment. What is remarkable in this case, is the difference which emerges between the pseudo-Bayesian and the SVSS schemes, but these are motivated by the different sampling approaches, a sequential one and a joint one.

Kaplan and Chen (2014) propose a more standard analysis based on Propensity Score (sequential approach) computed with model averaged expected values using Occam's windows (R package "BMA") on the one hand and individual sampled parameters obtained in a two step procedure on the other. This second methodology aims to build the model averaged Propensity Score distribution, but it requires a method for sampling parameters from a model averaging procedure: the authors propose to explore the model space in a first phase, using Occam's window again with the sole purpose of discovering high probable models; and then, in a second step, the parameters are defined in separate MCMCs on each previously detected model, using as weight the posterior model distributions[7]. Propensity Scores are

[6]Not treatment effect estimators!

[7]The procedure is similar to sampling from a mixture, in particular, each posterior model distribution defines the weight, and the parameters are sampled from each specification accordingly.

then obtained for each sampled value and for each one an outcome model is defined. The final treatment is evaluated aggregating the single ones obtained from each Propensity Score through eqs. (3.3) and (3.4).

Since a sequential analysis of propesity score is adopted in this work, both applications provided by Kaplan and Chen (2014) seem to be extremely valuable for the idea that is developed throughout this Chapter, for this reason I will partially follow their guidelines: the treatment evaluation based on a Propensity Score defined via the model averaged posterior mean (the plug-in estimator) and the fully Bayesian procedure are applied. However, substantial differences are introduced too: the original work exploits the *classical* BMA point of view, hence the definition of sampled parameters needs to be addressed separately in additional MCMCs. Moreover, the implementation of the whole procedure is linked to the R package "BMA", which uses BIC approximations for model posteriors and, in case of GLM, frequentist estimators instead of proper Bayesian ones are applied. The modern BMA approach with RJMCMC is here proposed as alternative not only for the possibility of sampling simultaneously models and parameters (with tremendous advantages with respect to the two-step procedure), but also because more adherent to a Bayesian estimation. The previously presented Gretl package `bma_glm` will be the protagonist and I will explore the consequences of applying such techniques in an economic scenario: Kaplan and Chen (2014) present some examples where variables selection is not so influential in the treatment evaluation, and in fact, BMA leads to similar conclusions as using a full model or more specific ones; the empirical illustration of the present work will actually show how BMA handles cases where the inclusion of a variable rather than another one is determinant in the final evaluation. Moreover, an additional technique is shown, i.e. an average of the treatment effect estimators of the models encountered in the RJMCMC, using Propensity Scores computed with the model-specific probit estimates and weights equal to the posterior model distribution, which actually mimics equation (3.5) with $\hat{\gamma}_{M_i}$ as the OLS estimate of the outcome model defined via the propensity score model M_i. The fully Bayesian estimator is similar, since the main difference is the accounting for the Propensity Score distribution, which leads, following again equation (3.5), to define $\hat{\gamma}_{M_i}$ as the sample mean of the OLS estimate of the treatment effects obtained for each set of propensity score in model M_i.

These two methodologies, in particular, average the treatment estimation across the model space by construction; whereas the first one does not, so it could be meaningful to ask whether using the model averaging posterior mean of the parameter of interest to build Propensity Score can address equally the model uncertainty issue.

3.3 Empirical analysis

In order to investigate the validity of the afore-mentioned BMA procedure, an empirical study is conducted on an extremely debated economic subject: the effect of *tax rebates* on consumption. This topic, despite its problematic tractability, is renowned for the instability of results due to model specification, an aspect that should encourage the validity of model averaging.

The contribution of this analysis, in particular, aims to discover new interpretations of the 2014 Italian income tax rebate, which introduced an increase in individual monthly salary of 80€ to employees whose gross annual income was included in the brackets 8145 − 26000€.

The work by Neri et al. (2017) provides a good starting point, as it is concerned in a *propensity score difference-in-difference* analysis. Even if it is conceptually different from standard propensity score applications like the ones seen before, it could be considered, actually, as a slightly modification of the framework, so similar conclusion can be drawn. I will try to replicate their analysis of the tax credit bonus and then apply BMA in order to ascertain if accounting for model uncertainty could improve the final estimation of the policy.

3.3.1 The 80€ tax credit and some economic background

Tax rebate policies are common stabilizing instruments applied by policymakers to reduce the impact of business cycle: such fiscal interventions should induce economic agents to rise their propensity to consume, hence countering the negative effect of recessions.

The 2014 Italian tax credit was not an exception from this pattern: its introduction via Decree Law 66/2014 implied, according to Government estimates, a total transfer of 5.9€ billion to households (about 10 million employees), equal to 0.4% of GDP. From a technical point of view, the reform allowed for a reduction in the tax withheld[8] by the employer on behalf of the employee with the aim of increasing the monthly salary by 80€. The recipients were all payroll employees with a total annual income between 8145€ and 26000€; eligibility was defined on individual bases, so households could have more than one member benefitting from the bonus; finally the delivery of the rebate was automatic, since it was directly integrated in the monthly paycheck, starting from May 2014.

Since eligibility status referred to 2014 income, whose certainty have been assessed only in 2015, misclassifications of people near the lower threshold could have been occurred, in fact it was estimated that about 1.5 million people were falsely classified as eligible, with the consequence of reimbursing their bonus.

It is worth noting, finally, that the standard bonus amount of 80€ was reduced for those people whose income was 24000-26000€ and for those ones who obtained a job during 2014, as the bonus was proportional to the number of months spent in employment[9] across the year.

As for the economic consequences of tax rebates, economic literature has not an unique interpretation: according to the *Life Cycle-Permanent Income Hypothesis* policy interventions which have a transitory nature should not affect the consumption. Economic agents are assumed to smooth consumption through lifetime, so tax rebate could be efficient only if they were perceived as permanent and so integrated in the lifetime income[10]. However history tells us of tax rebate interventions as provisional ones, so their real utility should be small. Additionally, as long as the policy

[8]Or in some cases in the pension contributions.

[9]Those employee who lost their job during 2014 may have had to repay part of the received bonus for the same reason.

[10]A proxy sometimes used to signal the temporariness of tax rebate policies is the simultaneous cut in government spending. This reflects the Barro/Ricardo equivalence: households integrate government budget in their choice, so consumption is increased only if government is not expected to raise taxes in the future.

is anticipated[11], consumers are expected to adjust their consumption pattern prior to the policy implementation, making the intervention even more difficult to analyze.

On the other hand, however, if we take into account that households may face liquidity constraints which prevent them to maintain their desired level of consumption, we could partially reconsider the impact of tax credits: we should expect that an income increase favors a realignment of the actual consumption to the the Life Cycle one, producing a positive effect. Behavioral literature, instead, considers factors such as "myopia", " rule of thumbs" or "mental accounting" as main drivers for a positive effect and in doing so, it explains how, in theory, the response to fiscal policies which increase disposable income should be *excessive*, implying the absorption of most of the bonus (Loewenstein and Thaler, 1989; Thaler, 1990).

If we turn our attention to empirical works, conclusions remain the same: uncertainty in the impact evaluation is still persistent and even more pronounced. Historically the first attempts (Modigliani et al., 1977; Blinder, 1981; Blinder et al., 1985; Poterba, 1988) focused only on a time series analysis, searching for a structural break which could convey information about policy effects. However in time series any other simultaneous or concurrent aggregate economic events could induce variation in spending, so the validity of the results were not so clear.

Micro-data analysis, instead, allows for a more genuine outlook, since the cross-sectional components can be studied: within this strand of literature two different, but somehow similar approaches are used. The first one uses directly Survey responses to evaluate the policy effect: Shapiro and Slemrod (2003b,a, 2009) are famous examples where, using additional questions enclosed to the Michigan Survey of Consumers, it is possible to directly verify the behavior of the consumer to the USA 2001 and 2008 tax credits, thorough his reply to the related question. Economists are quite skeptical about this methodology, as the response may not correspond to the real action, nonetheless more sophisticated estimation methods using similar dataset seem to confirm their findings: only the 21.8% of the interviewees seems to have increased significantly their spending, whereas the remaining part seems to have saved the tax rebate in order to pay down previous debt. Liquidity constraints seems to be irrelevant as well as possible lag effects between announcement and implementation.

The second approach, instead, relies on survey data equally, but, instead of focusing mainly in descriptive statistics and in individual responses, tries to elaborate a more complete method, such as treatment evaluation in order to study consumption and expenditure responses. Wilcox (1989); Parker (1999); Souleles (2002), for instance, using a regression like framework and difference-in-difference estimation evaluate significant responses to pre-announcement tax cut, leading to a potential counter-argument to the Life Cycle theory. Again, Souleles (1999) finds positive evidence for liquidity constraints significance: tax refunds seems to be used for non durables expenditure by low income/ low liquidity households; whereas high income and high liquidity individuals seems to prefer durables expenditure.

Johnson et al. (2006) confirms a positive effects of tax rebate policies, whose magnitude seems to be more pronounced in the case of liquidity constraints. The authors measure an estimated 20-40% of the rebate used for non durables goods in the quarter of the bonus receipt and then a lagged effect which covers about the 30% of the amount on the next quarter. Agarwal et al. (2007) using an innovative dataset

[11]This happens in practice when there is a lag between announcement and implementation.

related to credit card movements of households, detect a plausible new pattern in tax rebates use: consumers initially tend to save the bonus, by increasing their credit line use and thereby paying down debt; an increase in spending occurs only afterwards. Following this line of reasoning household balance sheet mechanisms seem to be extremely relevant in case of tax rebates. Finally Parker et al. (2013) find that households spend most of the rebate in durables goods, estimating a response of 50-90% of the payment.

However as highlighted by Heim (2007) additional care should be paid in such kind of studies: first of all it is extremely common to obtain different results by changing slightly both the methodology used and the model specification; moreover the significance of the result should be further investigated with greater attention: most studies obtain not *statistically* significant effects, and even in those cases where the significance is proved, the computation of standard error could be easily questioned. Finally differences in the data and in how some variables are built are crucial aspects too.

3.3.2 Dataset and a first analysis

In this subsection the analysis proposed by Neri et al. (2017) is studied in detail: as mentioned before, the idea is to follow their empirical methodology, firstly trying to be as adherent as possible to their framework and then noticing some possible source of uncertainty which will motivate our next steps.

The data come from the Survey on Household Income and Wealth (SHIW) held by the Bank of Italy every two years: the SHIW, in particular, consists of two different kind of datasets, the cross-sectional ones which reflect the biannual interviews and a panel one, which simply aggregates these covering the years 1977-2016[12]. However in the 2014 Survey, respondents were asked whether they received the tax rebates, and in case of positive reply, its amount and the way it was spent: of the 8156 interviewed households, 1514 declared to have at least a member who benefited from the bonus. The average amount received per household was equal approximately to 85€ and according to the reported replies, most of the bonus (about 90%) was devolved to consumption, even though additional information about possible lags in the expending pattern cannot be directly inferred through questionnaire answers. Neri et al. (2017) exploits for their treatment evaluation analysis the panel structure of the historical dataset, focusing only on years $2012 - 2014$.

In line with previous works (Brzozowski, 2007; Stuart et al., 2014), the methodology applied by the authors is based on *propensity score difference-in-differece*, that is at a first step a suitable sample of treated and control units is built using propensity score methods and then, on this sample, the difference-in-difference analysis is run.

Propensity score difference-in-difference is motivated by the fact that simple difference-in-difference is strongly related to the common trend assumption, however it could easily happen that, this is invalidated by group or time heterogeneity components: by group heterogeneity we mean the fact that some units belonging to a particular group (treated or control) are substituted by new ones across time;

[12]Households and individuals are aggregated via their number of questionnaire and number of order, respectively, which, for already registered units are not modified through time.

whereas time heterogeneity arises when, even if we have the same units, their behavioral pattern changes across time.

Introducing propensity score matching allows to partially disentangle difference-in-difference method from the common trend assumption, in order to build an *ad hoc* sample, which reflects, instead, propensity score features, such as the balancing property.

Despite the €80 tax rebate eligibility is deterministically defined, a propensity score analysis can be equally performed as long as it is used as a mere matching method i.e. establishing good matches between treated and untreated. In particular overlapping and balance assumptions should hold, and in general this is guaranteed by the choice of covariates in the binary choice model (probit model) and by additional diagnosis, prior to the matching scheme.

In their paper, Neri et al. (2017) refer almost each variable in the propensity score matching to 2012 (pre-treatment period); in this way, in theory, they further enhance the quality of propensity score estimates by avoiding that some not stochastic components in the treatment period (2014) affect the outcome. Group heterogeneity issue is avoided by building a sample of units which appear both in 2012 and in 2014; the authors account for time heterogeneity, instead, by introducing time variation covariates (see below).

The estimation of the treatment effect is carried out via an output model on the matched sample, in particular:

$$c_{i,t} = \beta_0 + \beta_1 \mathrm{TEMP}_t + \beta_2 \mathrm{BONUS}_i + \beta_3 \mathrm{TEMP}_t * \mathrm{BONUS}_i + \varepsilon_{i,t}$$

where $c_{i,t}$ is the average monthly consumption of the i-th household in time $t = \{2012; 2014\}$; TEMP is a time dummy equal to one in 2014; BONUS is a dummy equal to one if the i-th subject is treated and finally the interaction term, whose coefficient is the object of interest (monthly fraction of bonus spent on those goods). The authors consider three dependent variables $c_{i,t}$: the food consumption (as a proxy for non durable goods); transportation consumption (cars and other vehicles) and other durables.

Notice that the output model can be considered as the difference-in-difference counterpart of (3.2): in doing this we are implicitly computing (3.2) at each time and using the difference in the estimated $\hat{\gamma}$ as $\hat{\beta}_3$.

Starting from the premise that I cannot replicate their data because of the impossibility of determining exactly how some variables are obtained in the original work and due to the confidentiality of some information, the design of the study is the following: propensity score is performed in a dataset with 4458 households that were simultaneously observed in 2012 and 2014[13]; of which 864 were declared as eligible for the bonus (treated units).

[13]Missing observation were dropped.

The chosen covariates, which reflect demographic, socio-cultural characteristics as well as the general economic condition, are[14]:

- `Dcly1_1..5` is a group of dummy variables which identify the income class (each category corresponds to a fifth of the total distribution);
- `free_cash1` is the actual family income left over once actual rents and house mortgage are paid (in thousand euros);
- `nequ` is the normalized number of components in the household, according to the OCSE measure;
- `Darea3_1..3` is a group of geographical dummies which identify respectively North, Centre and South;
- `Deta5_1..5` are dummies for the age class (under 34, $35-44$, $45-54$, $55-64$, over 65);
- `Dstudio_1..6` are dummies for the educational level (None, Primary school, lower and upper secondary school, university and postgraduate degrees);
- `Dqualp3_1..3` are a set of dummies related to the employment activity (employee, self-employed, not employed);
- `con_1..3` are dummies which identify the ability of the household to make ends meet (with difficulty, with some difficulties, easily);
- `vd` is a dummy which reflects the quality of income responses (if evaluation is above 6, in a scale from 1 to 10, the household is considered reliable);
- `d_eta` is the variation occurred between 2014 and 2012 of older people in the household (by older we mean above 65);
- `d_edu` is the variation of high educated people (with at least university degrees) in 2014-2012;
- `d_employ` is the variation in the number of employees in 2014-2012;
- `d_weight` is the variation of the sample weights.

[14]Individual covariates refer to the head of the household; with respect to the original analysis the variable *delta income after bonus* is omitted.

The probit estimates are here summarized:

```
Model 1: Probit, using observations 1-4458
Dependent variable: bonus
Standard errors based on Hessian

               coefficient   std. error      z       p-value
  ----------------------------------------------------------
  const       -0.419927     0.283758     -1.480    0.1389
  Dcly1_2      0.504043     0.100149      5.033    4.83e-07 ***
  Dcly1_3      0.674237     0.105475      6.392    1.63e-10 ***
  Dcly1_4      0.828705     0.118678      6.983    2.89e-12 ***
  Dcly1_5      1.00555      0.157218      6.396    1.60e-10 ***
  free_cash1  -0.004189     0.002437     -1.719    0.0857   *
  nequ         0.312766     0.052769      5.927    3.08e-09 ***
  Darea3_2    -0.036931     0.067704     -0.5455   0.5854
  Darea3_3    -0.172973     0.061481     -2.813    0.0049   ***
  Deta5_2     -0.165574     0.158171     -1.047    0.2952
  Deta5_3     -0.444218     0.152357     -2.916    0.0035   ***
  Deta5_4     -0.678368     0.151385     -4.481    7.43e-06 ***
  Deta5_5     -1.18339      0.167406     -7.069    1.56e-12 ***
  Dstudio_2   -0.247115     0.194222     -1.272    0.2033
  Dstudio_3   -0.193147     0.193901     -0.9961   0.3192
  Dstudio_4   -0.435061     0.198435     -2.192    0.0283   **
  Dstudio_5   -0.624287     0.210175     -2.970    0.0030   ***
  Dstudio_6   -1.28717      0.323128     -3.983    6.79e-05 ***
  Dqualp3_2   -0.903884     0.092638     -9.757    1.72e-22 ***
  Dqualp3_3   -0.821156     0.066997     -12.26    1.55e-34 ***
  con2        -0.082903     0.064247     -1.290    0.1969
  con3        -0.375594     0.121211     -3.099    0.0019   ***
  vd           0.091344     0.121190      0.7537   0.4510
  d_eta       -0.141438     0.081679     -1.732    0.0833   *
  d_edu        0.069263     0.094296      0.7345   0.4626
  d_employ     0.427701     0.049609      8.621    6.62e-18 ***
  d_weight     0.041215     0.0412063     1.000    0.3172

Mean dependent var   0.193809   S.D. dependent var   0.395325
McFadden R-squared   0.242448   Adjusted R-squared   0.230131
Log-likelihood      -1660.549   Akaike criterion     3375.097
Schwarz criterion    3547.963   Hannan-Quinn         3436.037
```

Matching is then performed via nearest-neighbor without replacement, with a caliper of 0.01[15]; the treatment estimates (not winsored and winsored to the bottom/top 1%) are shown in Table 3.1, together with those by Neri et al. (2017).

[15]As for the order of units, the choice is the minimum distance order (Austin, 2014). Common support hypothesis is verified, even though this is not requested in nearest neighbor method, especially with small caliper. Balance properties are reported in the Appendix.

Table 3.1: Outcome model results: Full model specification

	Food	Cars	Other durables
Current Model	2.56/6.19	13.98/3.93	16.64/10.32*
	(21.85/20.53)	(22.51/18.70)	(10.49/6.24)
Baseline	**14.5**	**27.2**	**−0.1**
	(21.2)	(283.2)	(147.5)

Standard errors are in parenthesis: notice that the baseline values are clusterized at province level, however this cannot be reproduced due to the unavailability of the province information. Cars are obtained as net value between total purchases and total sales; for every entry two values are provided, the former is not winsorized, the latter is winsorized to the bottom and top 1%. Baseline results are provided only winsorized. Cars are obtained as net value between total purchases and total sales. $*$ significant at 10%, $**$ significant at 5%.

The results here obtained are not similar to the authors' ones[16]. Statistical significance is ascertained for other durables (only in the winsorized output), but this conclusion should be handled with extremely care, since standard error computation, in such kind of experiment, is extremely problematic: clusterization at province level should produce bigger standard errors than the ones obtained without clusters for durables categories, however province information is confidential; other types of corrections do not lead to substantial changes.

The discrepancy, in addition to the aforementioned data differences, could be also motivated by the model specification, so it could be interesting to verify what happens to the outcome estimation over different nested specifications. Many not significant variables can be detected in the probit model, so let us focus, as an example, only on those significant ones: with a simple stepwise procedure where we omit every variables whose significance is not guaranteed at a size of 1%[17] we end up with the output reported in Table 3.2:

Table 3.2: Reduced model

	Food	Cars	Other durables
Stepwise Model	20.25/20.11	25.00/11.07	11.35/9.01
	(21.64/20.24)	(25.08/20.01)	(10.06/6.34)

The results are quite different again, so a problem of model uncertainty arises; this leads to the possibility of applying model averaging tools.

3.3.3 A Bayesian Model Averaging counter-analysis

Let us assume the following set-up:

[16]The category *Other non durables* is not considered because highly variable: some plausible explanations range from the heterogeneity of goods which are included in this category (apparel, entertainment, charity...) to the impossibility of the methodology to catch this effect.

[17]Variables `free_cash1, Darea3_2, Deta5_2, Dstudio_2, Dstudio_3, con2, vd, d_eta, d¬ _edu, d_weight` are omitted.

- a normal prior on the parameter of interest β with a diffuse normal distribution attached to the parameter of the constant term α:

$$\beta_i \sim N(\underline{0}, n(\tilde{X}_i^T \tilde{X}_i)^{-1}), \qquad \alpha \sim N(0, 100)$$

  where n is the total number of observations; $\tilde{X}$ is the matrix of demeaned regressors[18], the subscript i identifies model M_i, which simply stands for the model with the subset of regressors $\tilde{X}_i$;

- a diffuse distribution on models, that is each model in the model space has the same prior probability (uniform);

- the number of iterations and burn-in are respectively 100000 and 10000.

The output of the RJMCMC is here summarized:

```
------------------------------------------------------
Bayesian Model Averaging with Generalized Linear Model
------------------------------------------------------
Overall sampling statistics
                 mean    stderr       pip
     const   -1.12347   0.02888   1.00000
   Dcly1_2    0.46436   0.09731   1.00000
   Dcly1_3    0.60534   0.10295   1.00000
   Dcly1_4    0.73129   0.11894   1.00000
   Dcly1_5    0.83886   0.16273   1.00000
free_cash1   -0.00165   0.00285   0.31307
      nequ    0.30866   0.05482   1.00000
  Darea3_2    0.00036   0.01122   0.02580
  Darea3_3   -0.06253   0.08079   0.43237
   Deta5_2   -0.00687   0.06325   0.08094
   Deta5_3   -0.29906   0.11040   0.95481
   Deta5_4   -0.56507   0.10197   1.00000
   Deta5_5   -1.04204   0.12488   1.00000
 Dstudio_2    0.00046   0.04509   0.05991
 Dstudio_3    0.02186   0.07437   0.14636
 Dstudio_4   -0.21730   0.10353   0.88927
 Dstudio_5   -0.44281   0.13882   0.96361
 Dstudio_6   -1.12579   0.28774   0.99410
 Dqualp3_2   -0.94225   0.09319   1.00000
 Dqualp3_3   -0.86087   0.06644   1.00000
      con2   -0.00375   0.02259   0.05540
      con3   -0.25892   0.16790   0.77974
        vd    0.00341   0.02799   0.03876
```

[18]Remember that in case of diffuse prior on the constant term, the constant is always included. The choice of a variance equal to 100 is a standard assumption.

```
     d_eta   -0.01749   0.05525   0.12049
     d_edu    0.00261   0.02299   0.03420
  d_employ    0.43664   0.04957   1.00000
  d_weight    0.00151   0.01111   0.04197
-----------------------------------
```

The `mean` column represents the model averaged estimation (1.20) where we can notice how variable parameters are shrunk according to their probability of inclusion (last column). The columns `stderr` and `pip` are respectively the model averaged standard errors (1.21) and posterior probability of inclusion, which represents the number of times the related variable appeared throughout the MCMC[19].

A similar analysis with respect to the one of the previous Chapter can be provided: variables with a posterior inclusion probability near or equal 1 have their model averaging posterior mean close to the standard Frequentist Probit coefficient[20]; other regressors exhibit a shrinkage effect with respect to the standard Probit proportional to their inclusion. It is very interesting to notice, moreover, how `Darea3_3` which is considered as an highly significant variable in the standard model, it is highly shrunk now. This could appear surprising, but it is possible to conclude, analyzing the model space, that the best specifications in terms of BIC are obtained exactly by removing this variable.

Three different BMA treatment effect estimators are evaluated:

- "BMA mean", which identifies the plug-in estimator based on a propensity score computed via the above model averaging posterior mean;
- "BMA Frequentist", which is based on equation (3.5), where the posterior model probability are provided by the RJMCMC, whereas the model specific treatments are obtained via the outcome models from the corresponding model specific propensity score, built using simple probit models;
- "BMA full", which replicates the fully Bayesian method by Kaplan and Chen (2014), and can be considered as the sample counterpart of equation (3.5), built on each outcome model derived from the sampled propensity scores[21].

In case of "BMA Frequentist" and "BMA full" the model specific treatment effects are obtained using the OLS estimators $\hat{\gamma}_i$ in the related outcome models; in "BMA mean" a single outcome model is defined and the related OLS estimator for the treatment effect is employed.

[19]The choice between using or not within movements (resampling) for the RJMCMC is secondary, as they lead to the same results; different thinning intervals have been applied too, as additional check, but the result does not change. For the example, I do not assume any thinning window. Same conclusions can be drawn for the proposal kernel for the movements between models: the standard one (addition and deletion of a single variable) has been used.

[20]With the exception of the constant term: this comes from the fact that I am assuming a diffuse prior with demeaned regressors; however this does not affect the further results in any way.

[21]Remember that the RJMCMC design allows to determine at each step a sampled parameter, which can be used, in turn, to define a sampled propensity score.

The estimators are presented in the following table, alongside the treatment effect for the best five specifications identified by the same BMA approach in order to highlight again the model uncertainty issue[22],

Table 3.3: Outcome model results: Model Averaging

	Food	Cars	Other durables
BMA (mean)	14.36/17.44	25.00/16.80	17.73*/13.08**
	(21.28/20.19)	(24.36/19.53)	(9.64/6.40)
BMA (Frequentist)	11.63/12.68	22.75/9.82	13.99/10.36
	(22.13/20.92)	(26.04/20.53)	(10.59/7.04)
BMA (full)	8.05/9.48	19.07/8.63	13.89/10.35
	(22.11/20.76)	(26.07/20.90)	(11.26/7.47)
I BMA model	9.59/11.89	34.37/17.65	15.74/11.53*
	(20.99/19.87)	(24.93/19.41)	(10.07/6.60)
II BMA model	20.25/20.11	25.00/11.07	11.35/9.01
	(21.64/20.24)	(25.08/20.01)	(10.06/6.34)
III BMA model	4.35/3.59	21.30/5.98	9.32/4.93
	(20.97/19.78)	(22.17/18.13)	(9.51/6.57)
IV BMA model	22.81/25.24	14.56/8.76	16.65*/11.51*
	(21.20/20.04)	(24.02/18.94)	(9.80/6.17)
V BMA model	8.75/11.42	18.33/7.33	17.11/13.03*
	(21.39/20.31)	(25.46/20.22)	(10.47/7.17)

Standard errors are in parenthesis: notice that the baseline values are clusterized at province level, however this cannot be reproduced due to the unavailability of the province information. Cars are obtained as net value between total purchases and total sales; for every entry two values are provided, the former is not winsorized, the latter is winsorized to the bottom and top 1%. Baseline results are provided only winsorized. Cars are obtained as net value between total purchases and total sales. $*$ significant at 10%, $**$ significant at 5%.

At a first glance, all three BMA methods lead to different treatment estimations, but the difference is not so remarkable, in particular, it is quite curious the fact that "BMA Frequentist" provides an evaluation which is halfway between the one of BMA mean and the one of BMA full.

The fully Bayesian alternative balances individual treatment effects by default, as the BMA Frequentist does, but unlike this one, it introduces the uncertainty related to the Propensity Score. The difference between the two can be imputed to this element.

"BMA mean" produces a different effect which somehow takes into account model uncertainty, but not as the other two methods: the motivation lies in the fact that averaging the treatment across models is different than estimating the treatment evaluation on the shrinkage parameters, firstly because of the non-linearity induced by the Propensity Score function. Moreover, the treatment evaluation is a two step procedure, in which, in a first phase the propensity score is built, and then, in a second one the matching and the evaluation is performed, hence the treatment estimate depends heavily on the matching method too, in our case nearest neighbor, which may introduce additional uncertainty in the choice.

[22]Simple Probit estimations are used for the Propensity Score.

The fully Bayesian method implicitly incorporates these side-effects considering the propensity score distribution, so it should provide a more reliable evaluation: the smallest effects are detected, but as I will explain further on, they are also the more robust to different matching schemes.

Finally, we can notice how the policy effect is positive across the three different good categories, even if not in line with the original ones; winsoring implies a greater effect for goods and a lower one for durables classes. Standard errors, as anticipated, should be handled with extremely care, especially for cars and other durables, therefore any analysis concerning statistical significance is avoided.

Table 3.4: Best model specifications

	$P(M\|y)$	Variables
I BMA model	0.18	`const Dcly1_2 Dcly1_3 Dcly1_4 Dcly1_5 nequ Deta5_3 Deta5_4 Deta5_5 Dstudio_4 Dstudio_5 Dstudio_6 Dqualp3_2 Dqualp3_3 con3 d_employ`
II BMA model	0.14	`const Dcly1_2 Dcly1_3 Dcly1_4 Dcly1_5 nequ Darea3_3 Deta5_3 Deta5_4 Deta5_5 Dstudio_4 Dstudio_5 Dstudio_6 Dqualp3_2 Dqualp3_3 con3 d_employ`
III BMA model	0.05	`const Dcly1_2 Dcly1_3 Dcly1_4 Dcly1_5 free_cash1 nequ Deta5_3 Deta5_4 Deta5_5 Dstudio_4 Dstudio_5 Dstudio_6 Dqualp3_2 Dqualp3_3 con3 d_employ`
IV BMA model	0.04	`const Dcly1_2 Dcly1_3 Dcly1_4 Dcly1_5 free_cash1 nequ Deta5_3 Deta5_4 Deta5_5 Dstudio_4 Dstudio_5 Dstudio_6 Dqualp3_2 Dqualp3_3 d_employ`
V BMA model	0.04	`const Dcly1_2 Dcly1_3 Dcly1_4 Dcly1_5 free_cash1 nequ Darea3_3 Deta5_3 Deta5_4 Deta5_5 Dstudio_4 Dstudio_5 Dstudio_6 Dqualp3_2 Dqualp3_3 con3 d_employ`

3.3.4 More checks

In the previous example we have seen how three different BMA methodologies behave in front of an extremely model sensitive Propensity Score matching problem: the matching method was nearest neighbor without replacement within a caliper of 0.01, using the minimum distance principle as data ordering (Austin, 2014), i.e. matched couples are formed starting from the treated and control unit which are nearest and so on. Since pairwise matching methods generally tend to be sensitive to the choice of the caliper and to the data ordering, that is the order in which the treated units are chosen to be matched, in this subsection I will show how the previous results change when a different caliper or a different data ordering is used; finally radius matching is also considered for comparison purpose.

The tables report the treatment estimation for the three categories (not winsored and winsored) using BMA mean, BMA Frequentist and BMA full under nearest neighborhood, using the default order (minimum distance), the natural data order, a stochastic order and finally, reordering treated units in an increasing and decreasing order of Propensity Score. Three different caliper are analyzed: 0.1, 0.01 and 0.001.

Table 3.5: Outcome model results: Minimum distance order

		caliper		
		0.1	0.01	0.001
BMA mean	food	17.77/20.75	14.36/17.44	7.65/12.47
		(20.93/19.88)	(21.28/20.19)	(23.09/21.96)
	car	27.12/18.98	25.00/16.80	15.28/11.90
		(23.87/19.20)	(24.36/19.53)	(26.02/20.94)
	other	16.65/12.49	17.73/13.09	13.86/12.00
		(9.89/6.41)	(9.65/6.40)	(10.17/6.92)
BMA Frequentist	food	11.67/12.64	11.63/12.68	11.75/13.36
		(21.50/20.34)	(22.13/20.92)	(23.74/22.43)
	car	21.70/9.02	22.75/9.82	27.50/13.64
		(25.76/20.18)	(26.04/20.53)	(29.28/22.83)
	other	13.19/10.36	13.99/10.36	14.18/10.75
		(10.46/7.02)	(10.59/7.04)	(11.26/7.56)
BMA full	food	8.64/10.03	8.05/9.48	9.06/10.62
		(21.71/20.39)	(22.11/20.76)	(24.12/22.64)
	car	19.00/8.26	19.07/8.63	21.79/10.71
		(25.55/20.44)	(26.07/20.90)	(28.47/23.02)
	other	13.52/10.37	13.89/10.35	14.35/10.42
		(11.08/7.43)	(11.26/7.47)	(12.36/8.14)

Standard errors are in parenthesis; for every entry two values are provided, the former is not winsorized, the latter is winsorized to the bottom and top 1%.

Table 3.6: Outcome model results: Natural data order

		caliper		
		0.1	0.01	0.001
BMA mean	food	16.36/19.67	11.13/14.83	8.83/13.24
		(20.75/19.70)	(21.24/20.15)	(23.03/21.78)
	car	23.56/16.15	26.20/17.80	26.67/17.93
		(23.65/18.83)	(24.37/19.34)	(26.35/20.60)
	other	16.02/11.91	18.85/13.73	13.24/15.05
		(9.74/6.31)	(9.96/6.44)	(10.37/6.87)
BMA Frequentist	food	10.54/11.46	10.90/12.00	11.65/13.35
		(21.27/20.09)	(22.17/20.89)	(23.71/22.39)
	car	24.69/11.88	22.03/9.60	25.03/11.10
		(25.98/20.54)	(27.13/21.22)	(30.76/23.74)
	other	14.55/11.08	17.81/12.69	15.43/11.61
		(10.41/6.87)	(10.88/7.11)	(11.27/7.52)
BMA full	food	8.19/9.56	7.25/8.71	7.88/9.57
		(21.47/20.17)	(22.06/20.71)	(24.07/22.60)
	car	20.90/9.62	18.96/7.98	20.77/9.11
		(25.28/20.35)	(26.01/20.89)	(28.30/22.86)
	other	15.04/11.19	16.05/11.58	15.35/10.96
		(10.87/7.23)	(11.16/7.31)	(12.27/8.01)

Standard errors are in parenthesis; for every entry two values are provided, the former is not winsorized, the latter is winsorized to the bottom and top 1%.

Table 3.7: Outcome model results: Stochastic order

		caliper		
		0.1	0.01	0.001
BMA mean	food	17.57/21.16	14.33/18.04	8.84/13.26
		(20.91/19.86)	(21.27/20.18)	(22.86/21.72)
	car	22.76/14.28	26.39/17.86	14.42/9.65
		(23.69/18.71)	(24.39/19.51)	(25.71/20.97)
	other	16.29/11.94	17.57/12.24	13.26/13.90
		(9.84/6.40)	(9.98/6.49)	(9.76/6.48)
BMA Frequentist	food	10.77/11.91	10.83/11.99	8.46/10.20
		(21.25/20.10)	(22.02/20.82)	(23.66/22.30)
	car	22.17/8.50	25.87/11.74	28.07/12.73
		(25.43/20.03)	(26.71/21.22)	(28.97/22.43)
	other	15.19/11.41	16.82/11.73	16.25/11.98
		(10.56/7.08)	(10.66/7.03)	(11.26/7.88)
BMA full	food	8.35/9.76	7.60/9.10	8.15/9.74
		(21.56/20.26)	(22.10/20.76)	(24.13/22.66)
	car	21.97/10.70	20.49/9.97	23.10/11.51
		(25.68/20.70)	(26.32/21.13)	(28.61/23.03)
	other	13.51/10.40	13.67/10.29	13.84/10.33
		(10.97/7.36)	(11.26/7.52)	(12.42/8.41)

Standard errors are in parenthesis; for every entry two values are provided, the former is not winsorized, the latter is winsorized to the bottom and top 1%.

Table 3.8: Outcome model results: Increasing order

		caliper		
		0.1	0.01	0.001
BMA mean	food	13.99/17.77	11.27/15.32	12.76/17.37
		(21.08/20.02)	(21.10/20.06)	(22.72/21.48)
	car	22.75/17.70	13.39/9.09	10.24/7.46
		(23.67/18.69)	(23.96/19.09)	(25.16/20.05)
	other	10.19/9.12	13.01/12.51	13.51/11.58
		(9.67/6.82)	(9.78/6.89)	(9.61/6.29)
BMA Frequentist	food	9.88/11.09	10.04/11.37	11.26/12.95
		(21.85/20.58)	(21.82/20.56)	(23.61/22.19)
	car	22.86/10.76	20.84/9.02	26.47/13.01
		(26.47/20.80)	(26.90/21.17)	(31.07/23.73)
	other	13.96/11.01	14.41/11.22	15.95/11.62
		(10.77/7.41)	(10.84/7.36)	(11.59/7.97)
BMA full	food	7.59/9.00	8.01/9.47	8.91/10.49
		(22.00/20.67)	(22.07/20.74)	(24.05/22.59)
	car	19.02/8.47	17.98/7.95	21.39/10.08
		(25.57/20.41)	(25.96/20.78)	(28.29/22.84)
	other	14.19/11.11	14.21/10.71	14.84/10.81
		(11.24/7.70)	(11.27/7.52)	(12.30/8.11)

Standard errors are in parenthesis; for every entry two values are provided, the former is not winsorized, the latter is winsorized to the bottom and top 1%.

Table 3.9: Outcome model results: Decreasing order

		caliper		
		0.1	0.01	0.001
BMA mean	food	20.33/22.41	21.32/24.46	15.10/18.44
		(20.16/19.12)	(21.24/20.14)	(22.94/21.79)
	car	29.82/22.52	31.47/23.83	25.87/17.39
		(23.12/18.91)	(24.11/19.44)	(26.28/20.81)
	other	15.48/12.84	16.23/12.88	18.91/16.49
		(9.56/6.65)	(9.98/6.79)	(10.50/7.32)
BMA Frequentist	food	11.97/12.80	11.95/12.77	10.39/11.89
		(20.41/19.34)	(21.60/20.54)	(23.84/22.65)
	car	27.65/15.76	20.82/9.81	27.51/12.75
		(24.82/19.81)	(26.99/21.01)	(29.90/23.09)
	other	13.29/10.67	13.83/10.46	14.35/11.18
		(9.96/6.92)	(10.67/6.94)	(11.32/7.58
BMA full	food	10.56/11.92	8.23/9.70	8.75/10.37
		(20.70/19.46)	(21.88/20.56)	(24.07/22.59)
	car	24.55/13.45	18.55/8.42	21.19/10.07
		(24.40/19.70)	(25.77/20.66)	(28.35/22.84)
	other	13.42/10.53	14.08/10.60	14.75/10.75
		(10.41/7.04)	(11.11/7.39)	(12.30/8.11)

Standard errors are in parenthesis; for every entry two values are provided, the former is not winsorized, the latter is winsorized to the bottom and top 1%.

The treatment effects defined via the minimum distance, the natural data order, the stochastic one and partly the increasing one are generally similar *given each caliper* inside each estimation method class; whereas the decreasing order produces quite different estimates. However, considering how the results changes across different calipers, it is possible to notice how the BMA mean is particularly sensitive of the chosen caliper[23], whereas both BMA Frequentist and BMA full are more robust. Moreover, BMA mean sometimes shows some counterintuitive results given the overall tendency of the other estimates: the unwinsorized treatment effect for cars, which is generally detected as the highest compared to the other categories, is strangely small in the BMA mean case with caliper 0.001 for stochastic, increasing or default order.

Therefore, BMA Frequentist leads to a more robust estimate across data orders and especially calipers, even though some outliers are still detected, such as car estimate in decreasing data order with caliper 0.01 or default order with caliper 0.001; BMA full, instead, provides an even more robust output but, again, it has the smallest effects.

It could be meaningful to ask whether some differences in the results are due to the natural variability of the categories we are considering, so having a look at the winsored output, we can conclude how even in this case the previous differences are maintained but the differences are slighter for BMA Frequentist and BMA full.

Turning the discussion on radius matching, two different calipers are chosen, 0.01

[23]The effect for food is an example.

and 0.001, and the output of the three methods are here shown:

Table 3.10: Outcome model results: Radius matching

		caliper	
		0.01	0.001
BMA mean	food	6.22/8.67	3.32/7.02
		(31.30/30.49)	(36.81/35.93)
	car	22.12/14.13	19.67/9.93
		(23.33/17.88)	(22.63/17.42)
	other	13.68/8.81	12.29/8.38
		(9.21/5.74)	(9.45/6.33)
BMA Frequentist	food	8.52/10.50	9.26/11.13
		(31.53/30.83)	(37.39/36.33)
	car	19.84/11.95	27.05/14.43
		(24.63/18.78)	(26.78/20.80)
	other	12.12/7.41	14.40/8.54
		(9.31/5.52)	(13.34/6.47)
BMA full	food	7.86/9.66	8.07/9.78
		(31.56/30.86)	(36.69/35.76)
	car	20.54/12.43	23.81/12.65
		(25.18/19.07)	(27.93/21.52)
	other	11.87/7.58	11.26/7.12
		(9.65/5.66)	(11.93/6.77)

Standard errors are in parenthesis; for every entry two values are provided, the former is not winsorized, the latter is winsorized to the bottom and top 1%.

Notably, the BMA mean leads to a quite different food effect estimate with respect to the previous matching method, car and other durables are slightly smaller, instead; BMA Frequentist equally shows smaller effects, whereas BMA full recovers an incredibly similar estimate if compared to nearest neighbor ones.

A fact that deserves attention is how both BMA mean and BMA Frequentist outputs tend to be more in line with the ones provided by BMA full, as if considering more matched units per treated produces similar effects to considering the propensity score uncertainty, but further investigations are necessary to ascertain this effect.

In sum, BMA full treatment evaluation seems to be the more stable across different calipers and data ordering (and different matching scheme too), especially considering the winsored results. BMA Frequentist, except for some outliers (in particular in the decreasing order case) is stable too, but not as the previous technique. Finally, BMA mean represents a very intriguing case, since its result in the natural data, stochastic and default order are very close to each other given each caliper; however if we consider the two additional data ordering or the variability across calipers, BMA mean is the less stable method, with some “weird” results too.

3.3.5 An additional experiment

Let us try now, as a final experiment, to slightly change the model set-up proposed in Subsection 3.3.3: in particular, the dummy variable `vd`, which previously assigned the value 1 if the reliability signaled by the individual was above or equal to the score of 6, and 0 otherwise, is now split into:

- `vd1` assumes the value 1 if the income quality score is equal or above 6 and less than 9 (excluded), and 0 otherwise;
- `vd2` assumes value 1 for score equal or higher 9 and 0 otherwise.

The idea is to consider more income quality classes, but both the previous `vd` variable and the new ones are highly insignificant, so the treatment estimation produced by the BMA mean should not vary considerably as the shrinkage effect is high for those variables. In other words, it is interesting to verify whether adding, changing, or deleting not relevant variables affects the three methods, considering in this way the impact of this kind of miss-specification. The estimates of the model averaging procedure are reported in the Appendix, but as it could have been expected, the difference in terms of posterior mean, probability of inclusion and best models achieved are minimal with respect to the original experiment.

Both nearest neighborhood with minimum distance order within the 0.01 caliper and radius matching are proposed in the following table:

Table 3.11: Outcome model results: NN - Old vs. New

		Previous model	New model
BMA mean	food	14.36/17.44	10.77/12.84
		(20.93/19.88)	(21.79/20.58)
	car	25.00/16.80	17.40/5.16
		(24.36/19.53)	(22.97/18.59)
	other	17.73/13.09	8.08/7.60
		(9.65/6.40)	(10.49/6.79)
BMA Frequentist	food	11.63/12.68	12.35/13.32
		(22.13/20.92)	(22.16/20.95)
	car	22.75/9.82	24.18/10.66
		(26.04/20.53)	(26.05/20.49)
	other	13.99/10.36	14.03/10.46
		(10.59/7.04)	(10.54/7.00)
BMA full	food	8.05/9.48	7.76/9.21
		(22.11/20.76)	(22.13/20.78)
	car	19.07/8.63	19.29/8.71
		(26.07/20.90)	(26.00/20.83)
	other	13.89/10.35	13.86/10.31
		(11.26/7.47)	(11.21/7.44)

Standard errors are in parenthesis; for every entry two values are provided, the former is not winsorized, the latter is winsorized to the bottom and top 1%.

BMA mean leads to different estimates, and in part, this could have been predictable since pairwise matching is extremely sensitive to the numerical value of the propensity score, which is, of course, modified. BMA Frequentist and, above all, BMA Full, however, produce an extremely similar result even in this case; therefore using these methods guarantee an insurance for the miss-specification here presented.

In order to verify if the changes are amplified by nearest neighborhood choice, as it could be expected, radius matching within a caliper of 0.01 is also analyzed.

Table 3.12: Outcome model results: NN - Old vs. New

		Previous model	New model
BMA mean	food	6.22/8.67	5.31/7.75
		(31.30/30.49)	(31.34/30.54)
	car	22.12/14.13	20.56/12.83
		(23.33/17.88)	(23.40/17.81)
	other	13.68/8.81	13.47/8.51
		(9.21/5.74)	(9.17/5.63)
BMA Frequentist	food	8.52/10.50	8.69/10.69
		(31.53/30.83)	(31.55/30.86)
	car	19.84/11.95	19.97/12.12
		(24.63/18.78)	(24.66/18.74)
	other	12.12/7.41	12.11/7.44
		(9.31/5.52)	(9.29/5.53)
BMA full	food	7.86/9.66	7.88/9.68
		(31.56/30.86)	(31.53/30.82)
	car	20.54/12.43	20.57/12.50
		(25.18/19.07)	(25.23/19.06)
	other	11.87/7.58	11.86/7.58
		(9.65/5.66)	(9.66/5.66)

Standard errors are in parenthesis; for every entry two values are provided, the former is not winsorized, the latter is winsorized to the bottom and top 1%.

It is possible to notice how the variation is remarkably smaller for BMA mean, therefore the impact of the matching scheme has clearly its importance, however BMA Frequentist and especially BMA Full exhibit minimal changes.

3.3.6 Summary

In this Chapter we have explored the effect of BMA in Propensity Score matching: the choice of which variables should be included in the Propensity Score estimation is often ignored, but the consequences can be severe. Model averaging has been proposed as a plausible solution which avoids problems of miss-specification; in particular, three different techniques concerning BMA have been used: BMA mean, BMA Frequentist and BMA full. Using data from the Bank of Italy SHIW, a similar analysis to the one proposed by Neri et al. (2017) for the Italian 2014 tax rebate has been performed, with the aim of verifying the validity of BMA method and their robustness to different set-ups instead of merely replicating their results. BMA full in particular has shown the most robust results with respect to different calipers and data order for nearest neighborhood, radius matching and finally to miss-specification scenarios where some insignificant variables have been modified. The consequences of this fact can be remarkable, even though additional analyses are surely required: firstly the choice of priors for both parameters and models could be determinant, so a further study in this direction seems to be necessary and an extension, which embraces the so-called Spike and Slab priors could be very interesting. As for the matching methods, throughout the Chapter I have adopted pairwise matching and radius matching; considering different methods such as stratification, kernel or optimal matching should be appropriate. Finally, analyzing BMA methods for other datasets or simulated examples is necessary too.

Conclusion

Models have a central role in science: we have seen at the beginning of this "brief journey" how this fact is also true for Economics and Econometrics, where the human behavior is the object of the study. In particular, we would like to think of a model like a mathematical tool which helps to understand and interpret the reality: for what concerns this dissertation, I have assumed the model functional form as given, so that I have turned the attention to the choice of which variables should enter in this relationship and their causality.

Commonly, this is translated into practice via model selection, or more properly, with variable selection methods, which range from more traditional ones such as hypothesis testing or Information Criteria, to more advanced ones such as shrinkage regression; but all of them have in common the scarce care for uncertainty. Choosing a single best specification can be considered as an innocuous but necessary fiction, however it hides several traps: model averaging is the solution here proposed, since taking into account not just one, but more models clearly reduces the risk of miss-specifications.

The choice made has led us to Bayesian Model Averaging in Generalized Linear Models, where the Reversible Jump Markov Chain Monte Carlo design has come to the aid thanks to its extreme flexibility; of course the efficiency of the MCMC and the accuracy of the estimates could be smaller if compared to its main antagonist, i.e. Stochastic Search Variable Selection, but in this efficiency-flexibility trade off, I have favored the second aspect.

A Gretl algorithm, thus, has been provided with the aim of introducing a ready-to-use procedure for every kind of user; in order to circumvent possible problems with poor mixing, local maxima which temporarily halt the MCMC and the CPU time which can grow faster, several computational aspects have been taken into account, in particular the parallelization of the MCMC. Using a very simple example, it has been possible to show the advantages of parallelization in terms of CPU time gains and how the convergence has not been affected as well.

In order to investigate the utility of BMA procedure in typical econometric routines, an application on Propensity Score matching for treatment evaluation has been proposed: when the choice of variables is crucial, in the sense that different models lead to different Propensity Scores, hence different treatment estimations, the necessity of a model averaged technique is impelling. The economic consequences of tax rebates, with the particular case of the recent Italian Tax rebate, has been studied, and three different methodologies concerning BMA have been used: BMA mean, BMA Frequentist and BMA Full. BMA Full has shown an intrinsic robustness to different matching set-ups, and even though it has produced the smallest estimates, on the other hand, considering the propensity score distribution takes into account an additional source of uncertainty ignored by the other methods. It could be interesting to compare this result with matching methods not considered in this work, i.e. optimal matching or kernel matching methods; as well as considering a simulated design experiment, where it would be possible to identify the effect in an ad-hoc environment.

As for the possible developments of the Gretl algorithm, different extensions would be interesting (and currently in process) such as integrating more link functions for even more GLMs, considering additional model proposal kernels, adaption schemes and further applications of parallelization (including horizontal MCMCs). The first aspect deserves a closer look, since the flexibility of the RJMCMC allows for even greater designs: for practical purposes, more general and complex models than GLMs can be studied, extending the scheme for not-nested cases as well. A potential guideline in this direction could be Barker and Link (2013), who modified the RJMCMC into a more general Markov Chain Monte Carlo simulation with Gibbs sampling. It could be desirable, moreover, to perform detailed studies on how different prior choices impact on the averaging procedure, analyzing the Spike and Slab cases which are somehow extremely famous in the literature. Finally, extensions of BMA to other applications such as Heckit models (Heckman, 1979) or endogeneity issues seems equally valuable.

To conclude, I sincerely hope that in these pages the reader have had the chance to taste a bit of the world of model building, and why not, find it a little amusing. Rephrasing the introduction of *"The Evolution of Physics"* (Einstein and Infeld, 1961), in imagination there exists the *perfect mystery story*, a story in which all clues lead the reader, if he follows the plot carefully, to the ultimate riddle solution just before the author provides it; but is it possible to compare this kind of reader to a scientist? The answer is negative, as the theory may explain many facts, but the general solution compatible with all clues is far from being discovered. The more we read from our incomplete "book of Nature", the more we appreciate it, though a complete solution seems to recede as we advance.

Bibliography

Agarwal, S., C. Liu, and N. S. Souleles (2007). The reaction of consumer spending and debt to tax rebates. Evidence from consumer credit data. *Journal of political Economy 115*(6), 986–1019.

Akaike, H. (1998). Information Theory and an extension of the Maximum Likelihood principle. In *Selected Papers of Hirotugu Akaike*, pp. 199–213. Springer.

Albert, J. H. and S. Chib (1993). Bayesian analysis of binary and polychotomous response data. *Journal of the American statistical Association 88*(422), 669–679.

Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pp. 483–485. ACM.

Amini, S. M. and C. F. Parmeter (2011). Bayesian model averaging in r. *Journal of Economic and Social Measurement 36*(4), 253–287.

An, W. (2010). Bayesian Propensity Score Estimators: Incorporating Uncertainties in Propensity Scores into Causal Inference. *Sociological Methodology 40*(1), 151–189.

Anderson, T. W. (1984). An introduction to multivariate statistical methods. *Wiley.Box Class of Likelihood Ratio Criteria. J. of the American Statist. Assoc 86*, 437–440.

Ando, T. and K.-C. Li (2017). A weight-relaxed model averaging approach for high-dimensional generalized linear models. *The Annals of Statistics 45*(6), 2654–2679.

Angrist, J. D. and J.-S. Pischke (2010). The credibility revolution in empirical Economics: How better research design is taking the con out of Econometrics. *The Journal of Economic Perspectives 24*(2), 3–30.

Atchadé, Y. F. and J. S. Rosenthal (2005). On Adaptive Markov Chain Monte Carlo algorithms. *Bernoulli 11*(5), 815–828.

Atkinson, A. C. (1981). Likelihood ratios, Posterior odds and Information Criteria. *Journal of Econometrics 16*(1), 15–20.

Austin, P. C. (2014). A comparison of 12 algorithms for matching on the propensity score. *Statistics in medicine 33*(6), 1057–1069.

Barker, R. J. and W. A. Link (2013). Bayesian multimodel inference by rjmcmc: A gibbs sampling approach. *The American Statistician 67*(3), 150–156.

Belloni, A., V. Chernozhukov, and C. Hansen (2011). Inference for high-dimensional sparse econometric models. *arXiv preprint arXiv:1201.0220*.

Bernardo, J. M. and A. F. Smith (1994). *Bayesian theory.* John Wiley & Sons.

Błażejowski, M. and J. Kwiatkowski (2015). Bayesian model averaging and jointness measures for gretl. *Journal of Statistical Software 68*(1), 1–24.

Błażejowski, M. and J. Kwiatkowski (2018). Bayesian averaging of classical estimates (bace) for gretl. Technical report, Universita'Politecnica delle Marche (I), Dipartimento di Scienze Economiche e Sociali.

Blinder, A. S. (1981). Temporary income taxes and consumer spending. *Journal of Political Economy 89*(1), 26–53.

Blinder, A. S., A. Deaton, R. E. Hall, and R. G. Hubbard (1985). The time series consumption function revisited. *Brookings Papers on Economic Activity 1985*(2), 465–521.

Bollen, K. A., S. Ray, J. Zavisca, and J. J. Harden (2012). A comparison of Bayes Factor approximation methods including two new methods. *Sociological Methods & Research 41*(2), 294–324.

Bottolo, L. and S. Richardson (2010). Evolutionary Stochastic Search for Bayesian model exploration. *Bayesian Analysis 5*(3), 583–618.

Box, G. and G. Jenkins (1970). *Time series analysis: forecasting and control.* Holden-Day, London.

Brock, W. A. and S. N. Durlauf (2001). What have we learned from a decade of empirical research on growth? growth empirics and reality. *The World Bank Economic Review 15*(2), 229–272.

Brockwell, A. E. (2006). Parallel Markov Chain Monte Carlo simulation by pre-fetching. *Journal of Computational and Graphical Statistics 15*(1), 246–261.

Brooks, S. P. and A. Gelman (1998). General methods for monitoring convergence of iterative simulations. *Journal of computational and graphical statistics 7*(4), 434–455.

Brooks, S. P., P. Giudici, and G. O. Roberts (2003). Efficient construction of reversible jump markov chain monte carlo proposal distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 65*(1), 3–39.

Brzozowski, M. (2007). Welfare reforms and consumption among single mother households: evidence from Canadian provinces. *Canadian Public Policy 33*(2), 227–250.

Buckland, S. T., K. P. Burnham, and N. H. Augustin (1997). Model selection: an integral part of inference. *Biometrics*, 603–618.

Burnham, K. P. and D. R. Anderson (2003). *Model selection and multimodel inference: a practical information-theoretic approach.* Springer Science & Business Media.

Burnham, K. P. and D. R. Anderson (2004). Multimodel inference: understanding AIC and BIC in model selection. *Sociological methods & research 33*(2), 261–304.

Campos, J., D. F. Hendry, and H.-M. Krolzig (2003). Consistent model selection by an automatic Gets approach. *Oxford Bulletin of Economics and Statistics 65*(s1), 803–819.

Castle, J. L., J. A. Doornik, and D. F. Hendry (2013). Model selection in equations with many "small" effects. *Oxford Bulletin of Economics and Statistics 75*(1), 6–22.

Chatfield, C. (1995). Model Uncertainty, Data Mining and Statistical Inference. *Journal of the Royal Statistical Society 158*(3), 419–466.

Chen, M.-H., L. Huang, J. G. Ibrahim, and S. Kim (2008). Bayesian variable selection and computation for generalized linear models with conjugate priors. *Bayesian analysis (Online) 3*(3), 585.

Chen, M.-H. and J. G. Ibrahim (2003). Conjugate priors for generalized linear models. *Statistica Sinica*, 461–476.

Chib, S. (1995). Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association 90*(432), 1313–1321.

Chipman, H., E. I. George, R. E. McCulloch, M. Clyde, D. P. Foster, and R. A. Stine (2001). The practical implementation of bayesian model selection. *Lecture Notes-Monograph Series*, 65–134.

Claeskens, G. and N. L. Hjort (2008). *Model selection and model averaging*, Volume 330. Cambridge University Press Cambridge.

Clyde, M. and E. I. George (2004). Model uncertainty. *Statistical science*, 81–94.

Clyde, M., M. Littman, and J. Ghosh (2012). BAS: Bayesian Model Averaging Using Bayesian Adaptive Sampling. *R package version 1.*

Clyde, M. A., J. Ghosh, and M. L. Littman (2011). Bayesian Adaptive Sampling for variable selection and model averaging. *Journal of Computational and Graphical Statistics 20*(1), 80–101.

Clyde, M. A. and E. S. Iversen (2013). Bayesian model averaging in the m-open framework. *Bayesian theory and applications*.

Cogley, T. and T. J. Sargent (2005). The conquest of us inflation: Learning and robustness to model uncertainty. *Review of Economic dynamics 8*(2), 528–563.

Cottrell, A. and R. Lucchetti (2017). Gretl + MPI.

Cottrell, A. and R. Lucchetti (2018). Gretl user's guide.

Danilov, D. and J. R. Magnus (2004). On the harm that ignoring pretesting can cause. *Journal of Econometrics 122*(1), 27–46.

De Luca, G., J. R. Magnus, and F. Peracchi (2018). Balanced variable addition in linear models. *Journal of Economic Surveys 32*(4), 1183–1200.

Doornik, J. A. (2009a). Autometrics. In *in Honour of David F. Hendry*. Citeseer.

Doornik, J. A. (2009b). Econometric model selection with more variables than observations. *Unpublished paper). Economics Department, University of Oxford*.

Durlauf, S. (2018). Institutions, development and growth: Where does evidence stand. Technical report, Working Paper WP18/04.1, Economic Development & Institutions.

Durlauf, S. N., A. Kourtellos, and C. M. Tan (2008). Are any growth theories robust? *The Economic Journal 118*(527), 329–346.

Eicher, T. S. and M. Newiak (2013). Intellectual property rights as development determinants. *Canadian Journal of Economics/Revue canadienne d'économique 46*(1), 4–22.

Eicher, T. S., C. Papageorgiou, and A. E. Raftery (2011). Default priors and predictive performance in Bayesian model averaging, with application to growth determinants. *Journal of Applied Econometrics 26*(1), 30–55.

Einstein, A. and L. Infeld (1961). *The Evolution of Physics: The Growths of Ideas from Early Concepts to Relativity and Quanta*. Cambridge University Press.

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

Fernandez, C., E. Ley, and M. F. Steel (2001b). Model uncertainty in cross-country growth regressions. *Journal of applied Econometrics 16*(5), 563–576.

Fosdick, L. D. (1959). Calculation of order parameters in a binary alloy by the monte carlo method. *Physical Review 116*(3), 565.

Foster, D. P. and E. I. George (1994). The risk inflation criterion for multiple regression. *The Annals of Statistics*, 1947–1975.

Fouskakis, D., I. Ntzoufras, and D. Draper (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.

Frühwirth-Schnatter, S. and R. Frühwirth (2007). Auxiliary mixture sampling with applications to logistic models. *Computational Statistics & Data Analysis 51*(7), 3509–3528.

Frühwirth-Schnatter, S. and R. Frühwirth (2010). Data augmentation and MCMC for binary and multinomial logit models. In *Statistical modelling and regression structures*, pp. 111–132. Springer.

Frühwirth-Schnatter, S. and H. Wagner (2006). Auxiliary mixture sampling for parameter-driven models of time series of counts with applications to state space modelling. *Biometrika 93*(4), 827–841.

Frühwirth-Schnatter, S. and H. Wagner (2010). Stochastic model specification search for Gaussian and partial non-Gaussian state space models. *Journal of Econometrics 154*(1), 85–100.

Gamerman, D. (1997). Sampling from the posterior distribution in Generalized Linear Mixed Models. *Statistics and Computing 7*(1), 57–68.

Gelfand, A. E. and A. F. Smith (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association 85*(410), 398–409.

Gelling, N., M. R. Schofield, and B. R. J (2018). R package rjmcmc: The calculation of posterior model probabilities from mcmc output.

Gelman, A., A. Jakulin, M. G. Pittau, and Y.-S. Su (2008). A weakly informative default prior distribution for logistic and other regression models. *The Annals of Applied Statistics 2*(4), 1360–1383.

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

George, E. I. and R. E. McCulloch (1993). Variable selection via Gibbs sampling. *Journal of the American Statistical Association 88*(423), 881–889.

George, E. I., D. Sun, and S. Ni (2008). Bayesian stochastic search for var model restrictions. *Journal of Econometrics 142*(1), 553–580.

Godsill, S. J. (2001). On the relationship between markov chain monte carlo methods for model uncertainty. *Journal of computational and graphical statistics 10*(2), 230–248.

Godsill, S. J. (2003). Proposal densities and product-space methods. *OXFORD STATISTICAL SCIENCE SERIES*, 199–202.

Granger, C. W. and D. F. Hendry (2005). A dialogue concerning a new instrument for econometric modeling. *Econometric Theory 21*(1), 278–297.

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

Green, P. J. (2003). Trans-dimensional Markov Chain Monte Carlo. *Oxford Statistical Science Series*, 179–198.

Green, P. J. and D. I. Hastie (2009). Reversible jump MCMC. *Genetics 155*(3), 1391–1403.

Greene, W. H. (2000). *Econometric analysis (International edition).* Pearson US Imports & PHIPEs.

Hannan, E. J. and B. G. Quinn (1979). The determination of the order of an autoregression. *Journal of the Royal Statistical Society. Series B (Methodological)*, 190–195.

Hans, C., A. Dobra, and M. West (2007). Shotgun Stochastic Search for "large p" regression. *Journal of the American Statistical Association 102*(478), 507–516.

Hansen, B. E. (2007). Least Squares Model Averaging. *Econometrica 75*(4), 1175–1189.

Hansen, B. E. (2014). Model averaging, asymptotic risk, and regressor groups. *Quantitative Economics 5*(3), 495–530.

Hansen, B. E. and J. S. Racine (2012). Jackknife model averaging. *Journal of Econometrics 167*(1), 38–46.

Hansen, P. R., A. Lunde, and J. M. Nason (2011). The Model Confidence Set. *Econometrica 79*(2), 453–497.

Hanson, T. E., A. J. Branscum, and W. O. Johnson (2014). Informative g-priors for logistic regression. *Bayesian Analysis 9*(3), 597–612.

Hastie, D. (2005). *Towards automatic Reversible Jump Markov Chain Monte Carlo*. Ph. D. thesis, University of Bristol.

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

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov Chains and their applications. *Biometrika 57*(1), 97–109.

Heckman, J. J. (1979). Sample selection bias as a specification error. *Econometrica 47*(1), 153–161.

Heim, B. T. (2007). The effect of tax rebates on consumption expenditures: evidence from state tax rebates. *National Tax Journal*, 685–710.

Hendry, D. and H.-M. Krolzig (2004). Sub-sample model selection procedures in general-to-specific modelling. *Contemporary Issues in Economics and Econometrics: Theory and Application*, 53–74.

Hendry, D. F. (1980). Econometrics-alchemy or science? *Economica*, 387–406.

Hendry, D. F. and H.-M. Krolzig (1999). Improving on "Data mining reconsidered" by KD Hoover and SJ Perez. *The econometrics journal 2*(2), 202–219.

Hjort, N. L. and G. Claeskens (2003). Frequentist model average estimators. *Journal of the American Statistical Association 98*(464), 879–899.

Hoeting, J. A., D. Madigan, A. E. Raftery, and C. T. Volinsky (1999). Bayesian Model Averaging: a tutorial. *Statistical science*, 382–401.

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

Hoover, K. D. and S. J. Perez (1999). Data mining reconsidered: encompassing and the general-to-specific approach to specification search. *The econometrics journal 2*(2), 167–191.

Hoshino, T. (2008). A Bayesian propensity score adjustment for latent variable modeling and MCMC algorithm. *Computational Statistics & Data Analysis 52*(3), 1413–1429.

Hurvich, C. M. and C.-L. Tsai (1989). Regression and time series model selection in small samples. *Biometrika 76*(2), 297–307.

Johansen, S. (1988). Statistical analysis of cointegration vectors. *Journal of economic dynamics and control 12*(2-3), 231–254.

Johnson, D. S., J. A. Parker, and N. S. Souleles (2006). Household expenditure and the income tax rebates of 2001. *American Economic Review 96*(5), 1589–1610.

Judge, G. G. and M. E. Bock (1978). The statistical implicatinos of Pre-Test and Stein-Rule Estimators in Econometrics.

Kaplan, D. and J. Chen (2012). A two-step Bayesian approach for propensity score analysis: Simulations and case study. *Psychometrika 77*(3), 581–609.

Kaplan, D. and J. Chen (2014). Bayesian model averaging for propensity score analysis. *Multivariate behavioral research 49*(6), 505–517.

Kass, R. E. and A. E. Raftery (1995). Bayes Factors. *Journal of the american statistical association 90*(430), 773–795.

Kass, R. E. and S. K. Vaidyanathan (1992). Approximate Bayes Factors and orthogonal parameters, with application to testing equality of two binomial proportions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 129–144.

Kass, R. E. and L. Wasserman (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.

Kass, R. E. and L. Wasserman (1996). The selection of Prior distributions by formal rules. *Journal of the American Statistical Association 91*(435), 1343–1370.

Koop, G. (2017). Bayesian methods for empirical macroeconomics with big data. *Review of Economic Analysis 9*(1), 33–56.

Koop, G., E. Ley, J. Osiewalski, and M. F. Steel (1997). Bayesian analysis of long memory and persistence using arfima models. *Journal of Econometrics 76*(1-2), 149–169.

Koop, G., D. J. Poirier, and J. L. Tobias (2007). *Bayesian econometric methods.* Cambridge University Press.

Kuhn, T. S. (1962). *The structure of scientific revolutions.* University of Chicago press.

Kullback, S. and R. A. Leibler (1951). On Information and Sufficiency. *The annals of mathematical statistics 22*(1), 79–86.

Lakatos, I. (1976). Falsification and the methodology of scientific research programmes. In *Can Theories be Refuted?*, pp. 205–259. Springer.

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

Lamnisos, D., J. E. Griffin, and M. F. Steel (2012). Cross-validation prior choice in bayesian probit regression with many covariates. *Statistics and Computing 22*(2), 359–373.

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

Lancaster, T. (2004). *An introduction to modern Bayesian econometrics.* Blackwell Oxford.

Leamer, E. E. (1974). False models and post-data model construction. *Journal of the American Statistical Association 69*(345), 122–131.

Leamer, E. E. (1983). Let's take the con out of Econometrics. *The American Economic Review 73*(1), 31–43.

Lenkoski, A., T. S. Eicher, and A. E. Raftery (2014). Two-stage Bayesian model averaging in endogenous variable models. *Econometric reviews 33*(1-4), 122–151.

Liang, F., R. Paulo, G. Molina, M. A. Clyde, and J. O. Berger (2008). Mixtures of g priors for bayesian variable selection. *Journal of the American Statistical Association 103*(481), 410–423.

Liu, Q., R. Okui, and A. Yoshimura (2016). Generalized least squares model averaging. *Econometric Reviews*, 1–61.

Loewenstein, G. and R. H. Thaler (1989). Anomalies: intertemporal choice. *Journal of Economic perspectives 3*(4), 181–193.

Lu, X. and L. Su (2015). Jackknife model averaging for quantile regressions. *Journal of Econometrics 188*(1), 40–58.

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

Madigan, D., J. York, and D. Allard (1995). Bayesian graphical models for discrete data. *International Statistical Review/Revue Internationale de Statistique*, 215–232.

Magnus, J. R. and G. De Luca (2016). Weighted-average least squares (wals): a survey. *Journal of Economic Surveys 30*(1), 117–148.

Magnus, J. R. and J. Durbin (1999). Estimation of regression coefficients of interest when other regression coefficients are of no interest. *Econometrica 67*(3), 639–643.

Magnus, J. R., O. Powell, and P. Prüfer (2010). A comparison of two model averaging techniques with an application to growth empirics. *Journal of econometrics 154*(2), 139–153.

Magnus, J. R. and A. L. Vasnev (2007). Local sensitivity and diagnostic tests. *The Econometrics Journal 10*(1), 166–192.

McCandless, L. C., I. J. Douglas, S. J. Evans, and L. Smeeth (2010). Cutting feedback in Bayesian regression adjustment for the propensity score. *The international journal of biostatistics 6*(2).

McCandless, L. C., P. Gustafson, and P. C. Austin (2009). Bayesian propensity score analysis for observational data. *Statistics in medicine 28*(1), 94–112.

Modigliani, F., C. Steindel, S. H. Hymans, and F. T. Juster (1977). Is a tax rebate an effective tool for stabilization policy? *Brookings Papers on Economic Activity 1977*(1), 175–209.

Moral-Benito, E. (2012). Determinants of economic growth: a bayesian panel data approach. *Review of Economics and Statistics 94*(2), 566–579.

Moral-Benito, E. (2015). Model averaging in economics: An overview. *Journal of Economic Surveys 29*(1), 46–75.

Mroz, T. A. (1987). The sensitivity of an empirical model of married women's hours of work to economic and statistical assumptions. *Econometrica: Journal of the Econometric Society*, 765–799.

Neri, A., C. Rondinelli, and F. Scoccianti (2017). Household spending out of a tax rebate: Italian "€80 tax bonus".

Parker, J. A. (1999). The reaction of household consumption to predictable changes in social security taxes. *American Economic Review 89*(4), 959–973.

Parker, J. A., N. S. Souleles, D. S. Johnson, and R. McClelland (2013). Consumer spending and the economic stimulus payments of 2008. *American Economic Review 103*(6), 2530–53.

Poterba, J. M. (1988). Are consumers forward looking? evidence from fiscal experiments. *The American Economic Review 78*(2), 413–418.

Raftery, A., J. Hoeting, C. Volinsky, I. Painter, and K. Yeung (2014). Bayesian Model Averaging–Package BMA. *URL http://CRAN. R-project. org/package= BMA 15.*

Raftery, A. E. (1995). Bayesian model selection in social research. *Sociological methodology*, 111–163.

Raftery, A. E. (1996). Approximate Bayes Factors and accounting for model uncertainty in Generalised Linear Models. *Biometrika 83*(2), 251–266.

Raftery, A. E. and Y. Zheng (2003). Discussion: Performance of bayesian model averaging. *Journal of the American Statistical Association 98*(464), 931–938.

Roberts, G. O. and J. S. Rosenthal (2007). Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. *Journal of applied probability 44*(02), 458–475.

Roberts, G. O. and J. S. Rosenthal (2009). Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics 18*(2), 349–367.

Rosenbaum, P. R. and D. B. Rubin (1983). The central role of the propensity score in observational studies for causal effects. *Biometrika 70*(1), 41–55.

Rosenthal, J. S. (2000). Parallel computing and Monte Carlo algorithms. *Far east journal of theoretical statistics 4*(2), 207–236.

Sala-i Martin, X., G. Doppelhofer, and R. I. Miller (2004). Determinants of long-term growth: A bayesian averaging of classical estimates (bace) approach. *American economic review 94*(4), 813–835.

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics 6*(2), 461–464.

Scott, S. L., A. W. Blocker, F. V. Bonassi, H. A. Chipman, E. I. George, and R. E. McCulloch (2016). Bayes and big data: The consensus Monte Carlo algorithm. *International Journal of Management Science and Engineering Management 11*(2), 78–88.

Shapiro, M. D. and J. Slemrod (2003a). Consumer response to tax rebates. *American Economic Review 93*(1), 381–396.

Shapiro, M. D. and J. Slemrod (2003b). Did the 2001 tax rebate stimulate spending? evidence from taxpayer surveys. *Tax policy and the economy 17*, 83–109.

Shapiro, M. D. and J. Slemrod (2009). Did the 2008 tax rebates stimulate spending? *American Economic Review 99*(2), 374–79.

Sickles, R. C. (2005). Panel estimators and the identification of firm-specific efficiency levels in parametric, semiparametric and nonparametric settings. *Journal of econometrics 126*(2), 305–334.

Souleles, N. S. (1999). The response of household consumption to income tax refunds. *American Economic Review 89*(4), 947–958.

Souleles, N. S. (2002). Consumer response to the Reagan tax cuts. *Journal of Public Economics 85*(1), 99–120.

Steel, M. F. (2018). Model averaging and its use in economics. *MPRA Paper No. 90110*.

Strachan, R. W. and H. K. Van Dijk (2013). Evidence on features of a dsge business cycle model from bayesian model averaging. *International Economic Review 54*(1), 385–402.

Stuart, E. A., H. A. Huskamp, K. Duckworth, J. Simmons, Z. Song, M. E. Chernew, and C. L. Barry (2014). Using propensity scores in difference-in-differences models to estimate the effects of a policy change. *Health Services and Outcomes Research Methodology 14*(4), 166–182.

Sugiura, N. (1978). Further analysts of the data by Akaike's Information Criterion and the finite corrections: Further analysts of the data by Akaike's. *Communications in Statistics-Theory and Methods 7*(1), 13–26.

Takeuchi, K. (1976). Distribution of informational statistics and a criterion of model fitting. *Suri-kagaku (Mathematical Sciences) 153*, 12–18.

Thaler, R. H. (1990). Anomalies: Saving, fungibility, and mental accounts. *Journal of economic perspectives 4*(1), 193–205.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288.

Van de Geer, S., P. Bühlmann, Y. Ritov, and R. Dezeure (2014). On asymptotically optimal confidence regions and tests for high-dimensional models. *The Annals of Statistics 42*(3), 1166–1202.

Van den Broeck, J., G. Koop, J. Osiewalski, and M. F. Steel (1994). Stochastic frontier models: A bayesian perspective. *Journal of Econometrics 61*(2), 273–303.

Wan, A. T., X. Zhang, and G. Zou (2010). Least squares model averaging by mallows criterion. *Journal of Econometrics 156*(2), 277–283.

Wang, H., X. Zhang, and G. Zou (2009). Frequentist model averaging estimation: a review. *Journal of Systems Science and Complexity 22*(4), 732.

White, H. (1990). A consistent model selection procedure based on m-testing. *Modelling economic series: Readings in econometric methodology*, 369–383.

White, H. (2000). A reality check for data snooping. *Econometrica 68*(5), 1097–1126.

Wilcox, D. W. (1989). Social security benefits, consumption expenditure, and the life cycle hypothesis. *Journal of Political Economy 97*(2), 288–304.

Zellner, A. (1986). On assessing prior distributions and bayesian regression analysis with g-prior distributions. *Bayesian inference and decision techniques: Essays in Honor of Bruno De Finetti 6*, 233–243.

Zellner, A. and A. Siow (1980). Posterior odds ratios for selected regression hypotheses. *Trabajos de estadística y de investigación operativa 31*(1), 585–603.

Zeugner, S. and M. Feldkircher (2015). Bayesian model averaging employing fixed and flexible priors: The bms package for r. *Journal of Statistical Software 68*(4), 1–37.

Zhang, X., D. Yu, G. Zou, and H. Liang (2016). Optimal model averaging estimation for generalized linear models and generalized linear mixed-effects models. *Journal of the American Statistical Association 111*(516), 1775–1790.

Zigler, C. M. and F. Dominici (2014). Uncertainty in propensity score estimation: Bayesian methods for variable selection and model-averaged causal effects. *Journal of the American Statistical Association 109*(505), 95–107.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association 101*(476), 1418–1429.

Appendices

Appendix A

Balancing properties: full probit model and BMA mean

The balancing properties for the Full model and BMA mean are reported in the following tables: using simple t-tests for the equality of the means between the two populations (treated and control units)[1], it is possible to conclude how in the full model the balance is achieved.

In the BMA mean case the same conclusions can be drawn, even though, apparently, the variables `free_cash` and `Darea3_3` reject at a significance level of 5% and 10%, respectively; however we should take into account the shrinkage effect imposed by the averaging procedure.

If we consider the posterior probability of inclusion (the last column in table A.2, `pip`) as additional element in the balancing property check, both `free_cash` and `Darea3_3` appear the 30% and the 40% of times across the MCMC, so their impact in the final matching scheme should not be the same as the one of those variables with the highest `pip`.

In this regards, all test statistics should be "re-scaled" by taking into account this quantity, hence, the balancing properties are verified for BMA mean too.

[1]Other methods such as *Cohen's d* or McFadden R test have been applied but not reported as they lead to the same conclusions.

Table A.1: Balancing properties - Full model

	mean (treated)	sd (treated)	mean (control)	sd (control)	mean diff	t statistic	p-value
Dcly1_2	0.163	0.370	0.145	0.352	0.018	0.986	0.324
Dcly1_3	0.209	0.407	0.189	0.392	0.019	0.956	0.339
Dcly1_4	0.255	0.436	0.258	0.438	−0.003	−0.116	0.907
Dcly1_5	0.290	0.454	0.312	0.464	−0.022	−0.943	0.346
free_cash	32.892	19.236	33.980	21.399	−1.087	−1.050	0.294
nequ	1.941	0.550	1.984	0.567	−0.043	−1.517	0.129
Darea3_2	0.198	0.399	0.201	0.401	−0.003	−0.127	0.899
Darea3_3	0.346	0.476	0.363	0.481	−0.017	−0.691	0.489
Deta5_2	0.157	0.364	0.174	0.379	−0.017	−0.891	0.373
Deta5_3	0.321	0.467	0.311	0.463	0.010	0.438	0.662
Deta5_4	0.398	0.490	0.407	0.492	−0.009	−0.363	0.717
Deta5_5	0.084	0.278	0.076	0.266	0.008	0.562	0.575
Dstudio_2	0.085	0.280	0.083	0.276	0.003	0.183	0.855
Dstudio_3	0.440	0.497	0.437	0.496	0.004	0.154	0.878
Dstudio_4	0.342	0.475	0.338	0.473	0.004	0.161	0.872
Dstudio_5	0.114	0.318	0.122	0.327	−0.008	−0.473	0.636
Dstudio_6	0.008	0.088	0.012	0.107	−0.004	−0.778	0.437
Dqualp3_2	0.066	0.249	0.078	0.268	−0.012	−0.886	0.376
Dqualp3_3	0.277	0.448	0.281	0.450	−0.004	−0.170	0.865
con2	0.591	0.492	0.585	0.493	0.005	0.207	0.836
con3	0.057	0.232	0.063	0.244	−0.006	−0.535	0.593
vd	0.959	0.199	0.951	0.216	0.008	0.734	0.463
d_eta	0.022	0.235	0.023	0.283	−0.001	−0.098	0.922
d_edu	0.012	0.241	0.014	0.307	−0.003	−0.184	0.854
d_employ	−0.019	0.497	0.001	0.566	−0.021	−0.764	0.445
d_weight	0.072	0.574	0.100	0.607	−0.028	−0.945	0.345

Table A.2: Balancing properties - BMA mean

	mean (treated)	sd (treated)	mean (control)	sd (control)	mean diff	t statistic	p-value	pip
Dcly1_2	0.159	0.366	0.153	0.360	0.006	0.349	0.727	1.000
Dcly1_3	0.211	0.408	0.185	0.388	0.026	1.272	0.204	1.000
Dcly1_4	0.259	0.439	0.263	0.441	−0.004	−0.173	0.863	1.000
Dcly1_5	0.288	0.453	0.320	0.467	−0.032	−1.377	0.169	1.000
free_cash	32.734	19.036	34.841	20.952	−2.106	−2.077	0.038	0.313
nequ	1.935	0.539	1.975	0.566	−0.040	−1.421	0.155	1.000
Darea3_2	0.204	0.403	0.184	0.387	0.021	1.025	0.305	0.026
Darea3_3	0.332	0.471	0.379	0.485	−0.046	−1.906	0.057	0.432
Deta5_2	0.159	0.366	0.167	0.373	−0.008	−0.411	0.681	0.081
Deta5_3	0.325	0.469	0.320	0.467	0.005	0.217	0.828	0.955
Deta5_4	0.393	0.489	0.402	0.491	−0.009	−0.362	0.717	1.000
Deta5_5	0.083	0.277	0.078	0.269	0.005	0.371	0.710	1.000
Dstudio_2	0.083	0.277	0.091	0.288	−0.008	−0.538	0.590	0.060
Dstudio_3	0.435	0.496	0.403	0.491	0.032	1.283	0.200	0.146
Dstudio_4	0.348	0.477	0.356	0.479	−0.008	−0.318	0.750	0.889
Dstudio_5	0.116	0.320	0.139	0.346	−0.023	−1.369	0.171	0.964
Dstudio_6	0.008	0.087	0.004	0.062	0.004	1.003	0.316	0.994
Dqualp3_2	0.065	0.248	0.063	0.243	0.003	0.207	0.836	1.000
Dqualp3_3	0.275	0.447	0.276	0.447	−0.001	−0.057	0.955	1.000
con2	0.598	0.491	0.629	0.483	−0.031	−1.249	0.212	0.055
con3	0.058	0.233	0.062	0.241	−0.004	−0.321	0.749	0.780
vd	0.959	0.199	0.955	0.207	0.004	0.374	0.708	0.039
d_eta	0.019	0.234	0.026	0.307	−0.006	−0.464	0.643	0.120
d_edu	0.012	0.240	0.003	0.268	0.009	0.696	0.486	0.034
d_employ	−0.010	0.499	−0.035	0.558	0.024	0.909	0.363	1.000
d_weight	0.069	0.571	0.046	0.709	0.023	0.703	0.482	0.042

Appendix B

RJMCMC output for the miss-specification example

In this Appendix the function output for the experiment in Subsection 3.3.5 is reported; the set-up for the function i.e the choice of the priors, number of iterations etc. are the same ones presented in the leading example (Subsection 3.3.3).

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

Overall sampling statistics
                 mean         se        pip
      const   -1.12301   0.02969    1.00000
    Dcly1_2    0.46721   0.10240    1.00000
    Dcly1_3    0.60731   0.10602    1.00000
    Dcly1_4    0.73610   0.12186    1.00000
    Dcly1_5    0.84987   0.16778    1.00000
 free_cash1   -0.00170   0.00290    0.31726
       nequ    0.30784   0.05439    1.00000
   Darea3_2    0.00020   0.01331    0.02708
   Darea3_3   -0.06188   0.08158    0.42278
    Deta5_2   -0.00554   0.06297    0.07964
    Deta5_3   -0.29880   0.11066    0.95670
    Deta5_4   -0.56334   0.10374    1.00000
    Deta5_5   -1.04278   0.12603    1.00000
  Dstudio_2   -0.00107   0.03435    0.04929
  Dstudio_3    0.01620   0.06021    0.10906
  Dstudio_4   -0.22259   0.09340    0.90952
  Dstudio_5   -0.45010   0.12832    0.97216
  Dstudio_6   -1.11800   0.28091    0.99428
  Dqualp3_2   -0.94132   0.09530    1.00000
  Dqualp3_3   -0.85862   0.06792    1.00000
```

```
      con2   -0.00325   0.02110    0.04712
      con3   -0.25676   0.16889    0.77322
       vd1    0.00110   0.01157    0.02011
       vd2   -0.00043   0.00895    0.01393
     d_eta   -0.01713   0.05531    0.11906
     d_edu    0.00148   0.01678    0.02249
  d_employ    0.43891   0.05020    1.00000
  d_weight    0.00144   0.01102    0.03797
------------------------------------
Best specifications:

Model_35114543: P(M|D)=0.195311
--
const Dcly1_2 Dcly1_3 Dcly1_4 Dcly1_5 nequ Deta5_3
Deta5_4 Deta5_5 Dstudio_4 Dstudio_5 Dstudio_6 Dqualp3_2
Dqualp3_3 con3 d_employ
------------------------------------
Model_35114671: P(M|D)=0.151022
--
const Dcly1_2 Dcly1_3 Dcly1_4 Dcly1_5 nequ Darea3_3
Deta5_3 Deta5_4 Deta5_5 Dstudio_4 Dstudio_5 Dstudio_6
Dqualp3_2 Dqualp3_3 con3 d_employ
```

Appendix C

Parallelization in the tax rebate example

A comparison between CPU time and the impact of parallelization is here proposed using the dataset and model proposed in Chapter 3 for Propensity Score matching.

The experiment is similar to the one of subsection 2.6.1, a fixed and a flexible burn-in design is proposed with 100000 iterations; the single core, 4-cores and 8-cores performances are compared in terms of CPU time and convergence.

Table C.1: Fixed burn-in set-up

cores	c=1	c=4	c=8
Iterations per core	100000	32500	21250
Elapsed time (min)	5128.96	1175.30	820.33
BG statistic		1.06	1.08
$P(M\|D)$			
Model I	0.18	0.19	0.19
Model II	0.14	0.14	0.13

The fixed burn-in design leads to better convergence diagnosis as we could have expected since the same burn-in time is elapsed in each core; however, moving from 4 cores to 8 cores implies a worsening of the multivariate Brooks and Gelman statistics. As for the time gain, the 4 core experiment allows to save up to the 78% of the time of the single core run; the 8 core experiment up to the 84%.

Even in this case, the relative advantage of using more cores exhibits a downward tendency when moving from 4 to 8 cores.

Turning the discussion to the flexible burn-in experiment, the CPU time advantage is more pronounced, in particular the 8 core design leads to an improvement of about 90% of the single core performance, but at a cost: the BG convergence statistics exceeds the 1.1 threshold, which is considered as a first signal of convergence problems.

In many cases this is not a big issue, as real convergence issues are detected over

Table C.2: Flexible burn-in set-up

cores	c=1	c=4	c=8
Iterations per core	100000	25000	12500
Elapsed time (min)	5128.96	836.10	404.10
BG statistic		1.08	1.11
$P(M\|D)$			
Model I	0.18	0.19	0.19
Model II	0.14	0.14	0.13

the 1.2 threshold; however the deterioration of the quality of the MCMC output due to smaller burn-in time seems to be not negligible.

To conclude, when many variables are used, the impact of parallelization seems to be more evident in both fixed and flexible burn-in designs, even though convergence statistics tend to worsen especially in the flexible set-up, where the small burn-in period may be inadequate. However further investigations are surely required, especially when much more variables are involved.

Table C.3: Best model specifications

	Variables
Model I	`const Dcly1_2 Dcly1_3 Dcly1_4 Dcly1_5 nequ Deta5_3 Deta5_4 Deta5_5 Dstudio_4 Dstudio_5 Dstudio_6 Dqualp3_2 Dqualp3_3 con3 d_employ`
Model II	`const Dcly1_2 Dcly1_3 Dcly1_4 Dcly1_5 nequ Darea3_3 Deta5_3 Deta5_4 Deta5_5 Dstudio_4 Dstudio_5 Dstudio_6 Dqualp3_2 Dqualp3_3 con3 d_employ`

Appendix D

RJMCMC algorithm - the package

```
#private functions

function void glm_stuff(matrix eta  "linear predictor: X'b",
                        const matrix Y    "dependent variable",
                        const matrix X_0  "matrix of regressors",
                        int type[1:4]  "type of link function for glm",
                        matrix *XWX,
                        matrix *XWz,
                        matrix *b,
                        matrix *V)

     #THIS FUNCTION COMPUTES SOME BASIC QUANTITIES FOR GENERALIZED LINEAR MODELS
     #Linear models are not considered in this function


     if type==1  #probit
          mu = cnorm(eta)  #mean
          d = 1/dnorm(eta)  #derivative of eta w.r.t. mu
          v = mu .* (1 - mu)  #~variance~
     elif type==2  #logit
          mu = logistic(eta)
          d = 1./(mu.*(1-mu))
          v = mu .* (1 - mu)
     elif type==3  #cloglog (binary)
          mu = 1-exp(-exp(eta))
          d = 1./exp(eta-exp(eta))
          v = mu .* (1 - (mu))
     elif type==4  #poisson
          mu=exp(eta)
          d = 1./exp(eta)
          v = mu
     endif

     z = eta + (Y - mu).*d  #transformation of 'y'

     w = 1./((d.^2) .* v)


     XW = X_0 .* w
     XWX = XW'X_0     # correspondent of (X'X) for glm
     XWz = XW'z # correspondent of (X'y) for glm

     if type==3
          V=invpd(XWX)  #covariance matrix
          b=V*XWz #parameter estimator (b_hat) -- frequentist
     endif

end function

function void glm(series y  "dependent variable, as a series",
                  list X "list of regressors",
                  const matrix X_0    "matrix of regressors",
                  int type[1:4],
                  matrix *b "vector containing parameter estimation",
                  matrix *v     "matrix containing covariance estimation")

    #FREQUENTIST ESTIMATION OF GLM
    #we use std Gretl functions for each model except for 'cloglog'
    #where  Iterative Weighted Least Squares are used



     if type == 1
          probit y X --quiet
          b = $coeff
          v= $vcv
     elif type == 2
          logit y X --quiet
          b = $coeff
          v= $vcv
     elif type == 4
          poisson y X --quiet
          b= $coeff
          v= $vcv
     else #cloglog case
          matrix Y = {y}
          matrix XWX={}
          matrix XWz={}
          matrix thres=1
          loop while thres>1.0e-20 #iterated weighted least squares

               b_1=b
               eta = X_0*b
               glm_stuff(eta, Y, X_0, type, &XWX, &XWz, &b, &v)
               thres=abs(b-b_1)

          endloop

     endif

end function
```

```
function matrix multi_norm(const matrix meanx  "mean of multivariate normal",
                           const matrix A  "Cholesky decomposition  covariance matrix")

     #THE FUNCTION GENERATES A RANDOM VECTOr
     #FROM A MULTIVARIATE NORMAL WITH MEAN "meanx" AND VARIANCE "V=AA'"

     matrix x = mnormal(rows(meanx),1)
     matrix Y=meanx + A*x

     return Y

end function


function void fillthebundle(bundles *MOD "bundles of models",
                            matrix model "binary vector representing model",
                            scalar id "id of the selected model",
                            series y,
                            list X,
                            const matrix X_0,
                            const matrix pr_mean,
                            const matrix pr_var,
                            int type,
                            bool const_case "diffuse(0) or informative(1) prior const")

     #GIVEN THAT EACH MODEL "m" CORRESPONDS TO A SINGLE BUNDLE IN THE ARRAY "MOD",
      #THIS FUNCTION FILLS EACH MODEL BUNDLE WITH SOME
     #QUANTITIES OF INTEREST (INITIALIZATION)

     bundle m = MOD[id]

     if inbundle(m, "already")==0  #Verify that the bundle in not already initialized
          m.already=1
          m.flag=1 #for resampling
          if const_case ==0 #adapt for diffuse prior on const
               model=1|model
          endif

          matrix Z_0=selifc(X_0,model') #matrix of regressors of id-model
          scalar k=cols(Z_0) #number of regressors of id-model

          list Z = X #update Z
          loop j=1..rows(model) --quiet
               if model[$j,1]== 0
                    Z -= X[$j]
               endif
           endloop
           # or with indexes
           #matrix var_x=X
           #var_z=selifc(var_x,model')
           #list Z = var_z
```

```
m.regr_list= Z #list of regressors of id-model
m.regr_matrix= Z_0

m.numb = k

m.acctimes=0   #number of times that the model is accepted in the MCMC
m.rejtimes=0   #number of times that the model is rejected in the MCMC

matrix p_mean = selifr(pr_mean, model)
if const_case==0
    if pr_var[1]==1
        p_varino = pr_var[2]*I(k-1)
        p_var = diagcat(100,p_varino)
    elif pr_var[1]==2
        p_varino = pr_var[2]*inv(Z_0[,2:]'Z_0[,2:])
        p_var = diagcat(100, p_varino)
    endif
else
    if pr_var[1]==1
        p_var = pr_var[2]*I(k)

    elif pr_var[1]==2
        p_var = pr_var[2]*inv(Z_0'Z_0)

    endif
endif


m.priormean = p_mean #prior mean of parameter of model id
m.priorvar = p_var #prior var of parameter of model id
inv_pv=invpd(p_var)
m.invpv=inv_pv  #inv of prior var -- for computation

matrix b=zeros(k,1)
matrix V=zeros(k,k)

glm(y, Z, Z_0, type, &b, &V) #freq estimates
ivar=invpd(V)

###bayesian average and covariance matrix based on frequentist estimation
VAR_freq=invpd(inv_pv+ivar)
Chol_freq=cholesky(VAR_freq)

ot_va=invpd(V+p_var)
b_freq=b-(V*ot_va*(b-p_mean))

m.bsamp=b                  #frequentist initialization
m.varfreq=VAR_freq         #bayesian VAR of paramete on freq. estimation
m.cholfreq=Chol_freq       #Chol decomposition of bayesian VAR
m.avfreq=b_freq            #bayesian mean of parameter on frequ. estiamtion

#bayesian average and covariance for MCMC re-sampling
```

```
          m.avchain=b_freq
          m.varchain=VAR_freq
          m.cholchain=Chol_freq

          #bayesian average and covariance for MCMC re-sampling,
          # 1-step ahead -- to speed up computations
          m.avchain_1=b_freq
          m.varchain_1=VAR_freq
          m.cholchain_1=Chol_freq


          MOD[id] = m
     endif

end function

#THE FOLLOWING TWO FUNCTIONS CONVERT A MODEL FROM DECIMAL TO BINARY NOTATION AND VICEVERSA
function scalar convert(matrix one)
     n = nelem(one)
    return (2 .^seq(0,n-1))*vec(one)
end function

function matrix reconv(scalar num, scalar n)
    x = num
    ret = zeros(n, 1)
    i = 1
    loop while x>0 --quiet
        y = x % 2
        ret[i++] =  y
        x -= y
        x /= 2
    endloop
    return ret
end function

function void sampling(series y, bundles *MOD, scalar id, int type, bool const_case, scalar U1)
     #IT COMPUTES A STEP (ITERATION) OF GAMERMAN'S BAYESIAN
     #ITERATED WEIGHTED LEAST SQUARES FOR GLM
     #I use this function to sample the initial vector of parameters and then
     #for resampling when no new movement is accepted
     #to speed uo the procedure we use the chain_1 values, avoiding useless repetition.
     #Note that this is true only when we stay for more than 1 times.
     #The first time needs some adjustments which are provided via the bool 'flag'

     bundle m = MOD[id]
     matrix Y = {y}
     matrix Z_0=m.regr_matrix
     matrix prvar=m.priorvar
     matrix invpv=m.invpv
     matrix prmean=m.priormean

     b_old=m.bsamp
```

```
 if  m.flag==0 #it guarantees that when a new model is accepted
 #and then a resample occurs the beta used is the current one for the model

     eta_1 = Z_0*b_old

     matrix ZWZ_1={}
     matrix ZWz_1={}
     matrix b_1={}
     matrix V_1={}
     glm_stuff(eta_1, Y, Z_0, type, &ZWZ_1, &ZWz_1, &b_1, &V_1)

     varchain=inv(invpv+ZWZ_1)

     varx_1=0.5*(varchain+varchain')
     b_new_ch_1=cholesky(varx_1)
     b_new_av_1 = varx_1*(invpv*prmean + ZWz_1)

     m.avchain_1=b_new_av_1
     m.varchain_1=varx_1
     m.cholchain_1=b_new_ch_1

     m.flag=1
endif

 b_av=m.avchain_1
 b_var=m.varchain_1
 b_ch=m.cholchain_1

 b_samp=multi_norm(b_av, b_ch)

 eta = Z_0*b_samp #update

 matrix ZWZ={}
 matrix ZWz={}
 matrix b={}
 matrix V={}
 glm_stuff(eta, Y, Z_0, type, &ZWZ, &ZWz, &b, &V)

 b_new_var=inv(invpv+ZWZ)

 varx=0.5*(b_new_var+b_new_var')
 b_new_ch=cholesky(varx)
 b_new_av = varx*(invpv*prmean + ZWz)

 AR=A_R(y, b_samp, b_old, prmean, prvar, type, Z_0, b_av, b_var, \
  b_new_av, b_new_var, const_case)#Metropolis-Hastings AR
 alpha=xmin(1,AR)

 if U1 <= alpha #accept --> update values in the bundle
     m.bsamp=b_samp

     m.avchain=b_av
     m.varchain=b_var
```

```
            m.cholchain=b_ch

            m.avchain_1=b_new_av
            m.varchain_1=b_new_var
            m.cholchain_1=b_new_ch

        endif
        MOD[id] = m
end function

function scalar ln_LikPrior(series y, const matrix beta, const matrix X, \
 const matrix prmean, const matrix prvar, int type, bool const_case)

    #ln(likelihood*prior) CALCULATED ON beta
    #The function can be also used to compute directly the ratio
    #between two different betas (provided they have the same dimension):
    #in this case beta must have two columns
    #representing each one the parameters

    scalar k = rows(beta)
    matrix i_prvar = inv(prvar)
    matrix ld_prvar = ldet(prvar)
    scalar lK = -k/2*ln(2*$pi) - 0.5 * ld_prvar
    pr = lK - 0.5 * qform((beta .- prmean)', i_prvar) #prior on beta (log)

    if cols(beta)==2
        pr0 = pr[1,1]
        pr1 = pr[2,2]
    endif

    matrix eta= X*beta

    if type==1
        mu = cnorm(eta)
    elif type==2  #logit
        mu = logistic(eta)
    elif type==3  #cloglog
        mu = 1-exp(-exp(eta))
    elif type==4  #poisson
        mu=exp(eta)
    endif

    if cols(beta)==1
        if type == 1 || type==2 || type==3 #likelihood   #1.binary models

            lik=sum(y.*log(mu)+(1-y).*log(1-mu))

        elif type == 4 #2.poisson

            lik=sum(y.*log(mu)-lngamma(y+1).-mu)

        endif
        ret=pr+lik

    elif cols(beta) ==2

        if type == 1 || type==2 || type==3

            lik0 = sum(y.*log(mu[,1])+(1-y).*log(1-mu[,1]))
            lik1 = sum(y.*log(mu[,2])+(1-y).*log(1-mu[,2]))

        elif type == 4
            matrix lgy = lngamma(y+1) .+ mu

            lik0 = sum(y.*log(mu[,1]) - lgy[,1])
            lik1 = sum(y.*log(mu[,2]) - lgy[,2])

        endif

        ret = (pr0 + lik0) - (pr1 + lik1)

    endif

    return ret
end function



function scalar A_R(series y, matrix beta, matrix b, matrix pr_mean, matrix pr_var, \
int type, matrix X_0, matrix M_star_1, matrix V_star_1,\
matrix M_star, matrix V_star, bool const_case)

     # M_star and V_star are respectively the current proposal posterior mean and variance
     # M_star_1 and V_star_1 are the old ones

     #THE FUNCTION COMPUTES PART OF THE TRANSITION PROBABILITY USED TO ACCEPT-
REJECT THE NEW SAMPLED beta

     f1 = exp(ln_LikPrior(y, beta ~ b, X_0, pr_mean, pr_var, type, const_case))

     scalar k = rows(beta)

     matrix i_posvar_1 = inv(V_star_1)
     matrix i_posvar_1 = 0.5*(i_posvar_1+i_posvar_1')#force simmetry
     matrix d_posvar_1 = ldet(V_star_1)
     scalar lK_1 = -k/2*ln(2*$pi) - 0.5 * d_posvar_1
     d1 = lK_1 -0.5 * qform((beta-M_star_1)',i_posvar_1) #proposal density -
- transitional kernel

     matrix i_posvar = inv(V_star)
     matrix i_posvar= 0.5*(i_posvar+i_posvar')
     matrix d_posvar = ldet(V_star)
     scalar lK = -k/2*ln(2*$pi) - 0.5 * d_posvar
     d2 = lK -0.5 * qform((b-M_star)',i_posvar)
     d = exp(d2 - d1) #ratio between proposal

     scalar a_r= f1*d #transition probability
     return a_r
end function

function matrix random_perm(matrix beta)
    n=rows(beta)
    a = seq(1,n)'
    b = msortby(muniform(n,1) ~ a,1)
    perm=b[,2]
    return beta[perm]
end function


function matrix b_born(bundles MOD,
                       scalar id "id of the model in the previous iteration step",
                       scalar idnew "id of the new proposed model",
                       matrix *u "transdimensional variable",
                       scalar *det_ratio "ratio between cholesky var of the two model",
                       scalar *g_u "distribution of u")

     ###TRANSFORMATION OF BETA THROUGH DIFFERENT MODELS --
REVERSIBLE JUMP MCMC DIMENSIONAL FUNCTION


     bundle m_old=MOD[id] #previous-step model
     k_old=m_old.numb
     b_old=m_old.bsamp

     av_old=m_old.avchain
     ch_old=m_old.cholchain

     bundle m_new=MOD[idnew]
     k_new=m_new.numb
     av_new=m_new.avchain
     ch_new=m_new.cholchain
     B1=det(ch_new)

     matrix iCh_old=inv(ch_old)
     B2=det(iCh_old)

     matrix b_std=iCh_old*(b_old-av_old) #standardize beta of the previous iteration
        #print b_std
     if k_old>k_new

          matrix b_std1 =random_perm(b_std)
          matrix u=b_std1[k_new+1:k_old,1]
          matrix b_std1=b_std1[1:k_new]
          #print b_std1
          #print u

     elif k_old<k_new
```

```
          matrix u = mnormal(k_new-k_old,1)

          matrix b_std1 = b_std | u
          matrix b_std1=random_perm(b_std1)

     else

          matrix b_std1=random_perm(b_std)

     endif

     scalar det_ratio=B1*B2 #part of accept-reject probability in transdimensional case

     if k_old!=k_new
          densit=dnorm(u) #u is a (multivariate) std normal
          g=prodc(densit)

          g_u=k_new<k_old ? g : 1/g #part of accept-
reject probability in transdimensional case
     endif

     matrix b_trans=ch_new*b_std1
     matrix b_new= av_new + b_trans #new beta

     return b_new
end function


function scalar ratio(series y,
                      const matrix newb "new sampled beta",
                      bundles MOD,
                      scalar newid "bundle of the new model",
                      scalar id,
                      scalar det_ratio,
                      scalar g_u,
                      int type,
                      bool const_case,
                      scalar prop_ratio)


     #PART OF ACCEPT-REJECT PROBABILITY FOR RJMCMC.
     #See Green(2003) for details

     bundle m_old=MOD[id]
     matrix Z_old=m_old.regr_matrix
     matrix prvar_old=m_old.priorvar
     matrix prmean_old=m_old.priormean
     matrix b=m_old.bsamp

     bundle m_new=MOD[newid]
     matrix Z_new=m_new.regr_matrix
     matrix prvar_new=m_new.priorvar
     matrix prmean_new=m_new.priormean

     P_1 = ln_LikPrior(y, newb, Z_new, prmean_new, prvar_new, type, const_case)
     P_0 = ln_LikPrior(y, b, Z_old, prmean_old, prvar_old, type, const_case)

     alpha=exp(P_1 - P_0)
     alpha_dyn=alpha*det_ratio*g_u*prop_ratio


     return alpha_dyn

end function

function scalar model_prior( matrix Phi, const matrix model)
     #IT COMPUTES MODEL PRIOR IN CASE OF BINOMIAL
     #'Phi' can be a scalar (average inclusion prob) or a vector \
     (each probability is specified)
     #Notice that we accept values in the range (0,1), with extremes excluded
     #if const_case==0 Phi in vector should not include the const term,\
    as it is always included!

     if nelem(Phi)==1
          Phi=Phi.*ones(rows(model),1)
     endif

     w=log(Phi)'model
     cw=log(1-Phi)'(1-model)

     pro=w+cw
     prior=exp(pro)

     return prior

end function

function void accept(matrix *model, matrix modelnew, scalar *id, scalar newid,\
 matrix bnew, bundles *MOD)
     #It updates the new values inside the selected bundle when the movement to
    a new model is accepted

     matrix model = modelnew
     scalar id = newid
     MOD[id].bsamp=bnew

     MOD[id].flag=0


end function



###The following parts are related to the model dynamics
```

```
function scalar binomial_coeff(int n, int k)
    num=gammafun(n+1)
    den=gammafun(k+1)*gammafun(n-k+1)
    return num/den
end function

function void Unif_II(const matrix model, matrix *modelnew, scalar change_var,\
 int dynamics, scalar adaption, scalar *prop_ratio, bool const_case)

    #movement from uniform, with delete, add, switching
    #dynamics ==1 you must specify the number of changing variable
    #if dynamics==2 this number is determined automatically,
    # you must specify the total number of potential variable to be changed

    k=nelem(model)
    #print model
    if dynamics==1
        change_var=randint(1,change_var)
    elif dynamics == 2
        change_var=randgen1(b,adaption,change_var-1)+1  #adaption case
    endif
  #print change_var
  if sum(model)>change_var #we specify the moves available
       #print "prim"
        if k-sum(model)>=change_var
            move=randint(1,3)
            qm=1/3
            #print "p1"
       else
            #print "p2"
            move=1
            qm=1
        endif
   elif sum(model)<=change_var && change_var<=k-sum(model)
        #print "second"
        if sum(model)==change_var && const_case==1
            move = randint(2,3)
            qm=0.5
            #print "s1"
       elif sum(model)==change_var && const_case==0
            move = randint(1,3)
            qm = 1/3
            #print "s2"
       elif sum(model)<change_var
            move=3
            qm=1
            #print "s3"
        endif
    elif sum(model)<=change_var && change_var>k-sum(model) && const_case==0
         move=1
         qm=1
         #print "terz"
    endif
```

```
if move==1 #proposal of new models
    #focus only on the  available variables --1s in model
     matrix subset = {}
     loop i=1..k --quiet
         if model[$i]==1
             subset~=$i
         endif
     endloop

     loop while sum(modelnew)!=(sum(model)-change_var) --quiet
         proposal=randint(1,nelem(subset))
         modelnew[subset[proposal]]=0
     endloop

     neigh_p=(1/ncount_move(model, move, k, change_var))
     #q_go= (dynamics==1)? qm*neigh_p : q*qm*neigh_p
     q_go=qm*neigh_p
     aw=ncount_move(model, move, k, change_var)

 elif move==3
     #focus only on the missing variables -- 0s in model
     matrix subset = {}
     loop i=1..k --quiet
         if model[$i]==0
             subset~=$i
         endif
     endloop

     loop while sum(modelnew)!=(sum(model)+change_var) --quiet
         proposal=randint(1,nelem(subset))
         modelnew[subset[proposal]]=1
     endloop

     neigh_p=(1/ncount_move(model, move, k, change_var))
     #q_go= (dynamics==1) ? qm*neigh_p : q*qm*neigh_p
     q_go=qm*neigh_p

 else
     #switch movement
     matrix modelnew_1=modelnew
     matrix modelnew_0=modelnew
     matrix subset_1 = {}
     matrix subset_0={}
     loop i=1..k --quiet
         if model[$i]==0
             subset_0~=$i
         else
             subset_1~=$i
         endif
     endloop

     loop while sum(modelnew_0)!=(sum(model)+change_var) --quiet
```

```
            proposal=randint(1,nelem(subset_0))
            modelnew_0[subset_0[proposal]]=1
            modelnew[subset_0[proposal]]=1
        endloop

        loop while sum(modelnew_1)!=(sum(model)-change_var) --quiet
            proposal=randint(1,nelem(subset_1))
            modelnew_1[subset_1[proposal]]=0
            modelnew[subset_1[proposal]]=0
        endloop

    endif

    #return movement
    if sum(modelnew)>change_var
         if k-sum(modelnew)>=change_var
              qm_r=1/3
         else
              qm_r=1
         endif
    elif sum(modelnew)<=change_var && change_var<=k-sum(modelnew)
         if sum(modelnew)==change_var && const_case==1
              qm_r=0.5
         elif sum(modelnew)==change_var && const_case==0
              qm_r=1/3
         elif sum(modelnew)<change_var

              qm_r=1
         endif
    elif sum(modelnew)<=change_var && change_var>k-
sum(modelnew) && const_case==0
         qm_r=1

    endif

    if move==1
        move_r=3
        neigh_pr=(1/ncount_move(modelnew, move_r, k, change_var))
        q_ret=qm_r*neigh_pr

        prop_ratio=q_ret/q_go
    elif move==3
        move_r=1
        neigh_pr=(1/ncount_move(modelnew, move_r, k, change_var))
        q_ret=qm_r*neigh_pr

        prop_ratio=q_ret/q_go
    else
        prop_ratio=1
   endif
   #print modelnew
```

```
end function

function void neighbor(bundle *m, bundles MOD,const matrix modelnew,\
 scalar k, bool const_case)
    #Computes the neighborhood of a selected model, for dynamics==3
    if inbundle(m, "neigh")==0
        set_red={}
        set_switch={}
        set_add={}

        loop i=1..k --quiet #delete/add moves
            if modelnew[$i]==1 #deletion move neightborhood
                modelnew_del=modelnew
                modelnew_del[$i]=0
                id_red=convert(modelnew_del)
                if id_red!=0 && const_case==1
                     prob_red=MOD[id_red].sss_prob
                     r=id_red | prob_red
                     set_red ~= r
                elif const_case==0
                     if id_red==0
                          id_red=2^k
                     endif
                     prob_red=MOD[id_red].sss_prob
                     r=id_red | prob_red
                     set_red ~= r
                endif

            else #addition move neighbborhood
                modelnew_add=modelnew
                modelnew_add[$i]=1
                id_add=convert(modelnew_add)
                prob_add=MOD[id_add].sss_prob
                a=id_add | prob_add
                set_add ~= a
            endif
        endloop

        loop i=1..k --quiet #switch moves
            if modelnew[$i]==1
                modelnew_switch=modelnew
                modelnew_switch[$i]=0
                loop j=1..k --quiet
                    if modelnew[$j]==0
                        mod_switch=modelnew_switch
                        mod_switch[$j]=1
                        id_switch=convert(mod_switch)
                        prob_switch=MOD[id_switch].sss_prob
                        s = id_switch|prob_switch
                        set_switch~=s
                    endif
                endloop
            endif
        endloop

        neigh=set_red~set_add~set_switch

        row=neigh[2,]/sum(neigh[2,])
        neigh|=row

        #reduce neighborhood dimension
        neigh=-msortby(-neigh', 3)'
        reduction = (neigh[3,].>0.001) #maybe to change
        if sum(reduction)>=1
             #neigh=neigh[,1:sum(reduction)]
            neigh=selifc(neigh, reduction)
            neigh[3,]=neigh[2,]/sum(neigh[2,])
        endif

        m.neigh=neigh
    endif

end function

function scalar ncount_move(const matrix model, int move, int k, int change_var)
    #IT COUNTS THE NUMBER OF POSSIBLE MOVEMENTS (MODELS)\
     FROM 'model' WITH 'chang_var' AS NUMBER OF VAR CHANGED.

    n_ones=sum(model)
    n_zeros=k-n_ones

    if move==1 #deletion case
        count=binomial_coeff(n_ones, change_var)
    elif move==3 #addition
        count=binomial_coeff(n_zeros, change_var)
    endif #In case of switch the number of moves has no meanings
    # because it simplifies with the return move

    return count
end function

function matrix modelmove(const matrix model, int dynamics, int k, int change_var,\
scalar adaption, scalar *prop_ratio, bundles *MOD, bool const_case)
#THE FUNCTION DEFINES THE TYPE OF MOVEMENT IN MODEL SPACE (THE PROPOSED NEW MODEL)
    modelnew = model

    if dynamics==0 #std MC3 dynamics, a variable more or a variable less

         potentialvar=randint(1,k) #number of var in list Z

         if modelnew[potentialvar]== 1 && sum(modelnew)>1 && const_case==1
              modelnew[potentialvar]= 0
         elif modelnew[potentialvar]==1 && const_case==0
              modelnew[potentialvar]= 0
         else
              modelnew[potentialvar]= 1
         endif

    elif dynamics== 1 || dynamics==2 #generalization of dynamics == 0

         Unif_II(model, &modelnew, change_var, dynamics, adaption, &prop_ratio, const_case)

    else #dynamics ==3 Shotgun Stochastic Search

         id=convert(model)
         if id==0
              id=2^k
         endif

        bundle m = MOD[id]
        neighbor(&m, MOD, model, k, const_case)

        MOD[id]=m
        proposal = randgen1(u, 0,1)
        neigh_model=m.neigh
        print neigh_model
        cum_neigh=cum(neigh_model[3,]')'
        loop i=1..cols(neigh_model) --quiet
            if cum_neigh[$i]>=proposal
                 newid=neigh_model[1,$i]
                 if newid==2^k
                      modelnew=zeros(k,1)
                 else

                      modelnew=reconv(newid,k)
                 endif

                q_go=neigh_model[3,$i]

                break
            endif
        endloop

        bundle mnew=MOD[newid]
        neighbor(&mnew, MOD, modelnew, k, const_case)
        MOD[newid]=mnew
        neigh_modelnew=mnew.neigh
        q_ret=0
        loop i=1..cols(neigh_modelnew) --quiet
            if neigh_modelnew[1,$i]==id
                q_ret=neigh_modelnew[3,$i]
                break
            endif
        endloop

        prop_ratio=q_ret/q_go

    endif
```

```
    return modelnew

end function

function matrix score(const matrix Y,const matrix X_0, list X, const matrix M,int type,\
const matrix pr_mean,const matrix pr_var, int prior_mod,\
const matrix Phi, bool const_case)

    #THE FUNCTION COMPUTES THE SCORE OF EACH SINGULAR MODEL FOR SSS
    matrix SC={}

    loop i=1..cols(M) --quiet
         id = M[$i]
         if const_case==0
              if id==2^(cols(X_0)-1)
                   model = 1 | zeros(cols(X_0)-1,1)
              else

                   model=reconv(id, cols(X_0)-1)

                   model=1|model
              endif

         else
              model=reconv(id, cols(X_0))
         endif

        matrix Z_0=selifc(X_0,model') #matrix of regressors of id-model
        scalar k=cols(Z_0) #number of regressors of id-model

        list Z = X #update Z
        loop j=1..rows(model) --quiet
            if model[$j,1]== 0
                Z -= X[$j]
            endif
        endloop

        matrix p_mean = selifr(pr_mean, model)
          if const_case==0
              if pr_var[1]==1
                  p_varino = pr_var[2]*I(k-1)
                  p_var = diagcat(100,p_varino)
              elif pr_var[1]==2
                  p_varino = pr_var[2]*inv(Z_0[,2:]'Z_0[,2:])
                  p_var = diagcat(100, p_varino)
              endif
          else
              if pr_var[1]==1
                  p_var = pr_var[2]*I(k)

              elif pr_var[1]==2
                  p_var = pr_var[2]*inv(Z_0'Z_0)
              endif
          endif


            matrix b=zeros(k,1)
            matrix V=zeros(k,k)
            series y = Y

            glm(y, Z, Z_0, type, &b, &V) #freq estimates

            matrix P_bm=ln_LikPrior(y, b, Z_0,  p_mean, p_var,type, const_case)

            if prior_mod==1
                 if const_case==0
                      Pm=log(model_prior(Phi, model[2:]))
                 else
                      Pm=log(model_prior(Phi, model))
                 endif

                P_bm+=Pm
            endif

        SC~=P_bm

    endloop

    return SC

end function

###print the results
function void printres(bundle *res, list X, int type, int prior_mod, \
int dynamics,scalar change_var, scalar adapt_rate,int threads, bool resamp, bool const_case)

     b_mean=res.sampled_mean
     pip_mean=res.sampled_pip
     b_var=res.sampled_var
     nrep=res.nrep
     burn=res.burnin
     tim=res.temp

     ### heading

     if   type==1
          string ty="Probit model"
     elif type==2
          string ty="Logit model"
     elif type==3
          string ty="Cloglog model"
     elif type==4
          string ty="Poisson model"
     endif
```

```
if prior_mod==0
     string mopr="P(M) ~ Uniform"
else
     string mopr="P(M) ~ Binomial(Phi)"
endif

if dynamics ==0
     string dyn="MCMCMC - add/delete (1)var"
elif dynamics ==1
    string dyn=sprintf("MCMCMC - add/delete/switch (%d)var\n", change_var)
elif dynamics ==2
    string dyn = "Adaptive MCMCMC"
else
    string dyn= "Shotgun Stochastic Search"
endif


if const_case==0
     string conca="Diffuse const"
else
     string conca="Informative const"
endif

if resamp
     string resampling="Yes"
else
     string resampling="No"
endif

#### overall stat
average=meanr(b_mean) #overall sampled mean

if threads==1
     std=sqrt(diag(b_var))
else

     k_variables=rows(b_mean)
     n_obs=nrep-burn
     #within variance
     matrix W=zeros(k_variables, k_variables)
     loop i=1..threads --quiet
          #b_part=b_var[(($i-1)*k_variables)+1:$i*k_variables,]

          W+=b_var[(($i-1)*k_variables)+1:$i*k_variables,]
     endloop

     W/=threads #mean of covariance
     n_rat=(n_obs-1)/n_obs
     W1=W*n_rat #adjustment

     #between variance
     B1=mcov(b_mean')*(threads-1)/(threads)
     B2=mcov(b_mean')
     #total variance
     V=W1+B1
     #throds=(threads*n_obs)/(threads*n_obs-1)
     #V*=throds #correction of variance

     std=sqrt(diag(V))

     #variance in Brook & Gelman
     correct=(threads+1)/(threads)
     V_hat=W1+(correct*B2)

     std_hat=sqrt(diag(V_hat))
endif

#v=mcov(b')
#std=sqrt(diag(v))

pip=meanr(pip_mean) #overall pip

if threads ==1

     matrix result=average~std~pip

     string title="mean stderr pip"
else
     matrix result = average~std~std_hat~pip

     string title="mean se adj_se pip"
endif

string co=strsub(varname(X),",","," ")

colnames(result, title)
rownames(result, co)

res.overall_result=result

#Brooks & Gelman statistics of convergence

if threads !=1
     iW=inv(W)
     WB=iW*B2
     eigenvalue= eigengen(WB)#here
     if cols(eigenvalue)==2
          mod=eigenvalue[1,].^2 + eigenvalue[2,].^2
          eigenvalue=sqrt(mod)
     endif

     lambda = max(eigenvalue)

     GB=n_rat + (correct*lambda)
```

```
endif

#top models
id=res.sampled_id

models=values(id[1,])'

mod_count={}
loop i=1..nelem(models) --quiet
     corresp=id.*(id[1,].=models[$i])

     mod_count~=sumr(corresp[2,]) #counting of total models
endloop

post_model= mod_count./sum(mod_count) #posterior model distribution

tot_models= models|post_model

top_models= -msortby(-tot_models', 2)' #ordered models by their posterior \ prob (decreasi

res.listmodels=top_models

#I focus in the best three specifications, if available,\
provided their posterior is above 0.1

threshold=top_models[2,].>0.1

threshold=threshold[1:3]
#print threshold
name_vars=varnames(X)

loop i=1..sum(threshold)  --quiet
     if const_case ==0
          if top_models[1,$i]==2^(nelem(X)-1)
               model_$i= 1|zeros(nelem(X)-1,1)

          else

               model_$i=reconv(top_models[1,$i],nelem(X)-1)
               model_$i=1|model_$i
          endif

     else

          model_$i=reconv(top_models[1,$i],nelem(X))
     endif

     string var_names$i
     loop j=1..nelem(X) --quiet

          if model_$i[$j]==1
               var_names$i ~= name_vars[$j]
```

```
                    var_names$i ~= ","
               endif
          endloop

          var_names$i=strsub(var_names$i,",","," ")

     endloop

     #print
     print "--------------------------------------------------"
     print "Bayesian Model Averaging with Generalized Linear Model"
     print "--------------------------------------------------"
     printf "Type of specification: %s\n", ty
     printf "Model Prior: %s\n", mopr
     printf "Model dynamics: %s \n", dyn
     printf "Prior on const: %s \n", conca
     printf "Resampling allowed: %s\n", resampling
     printf "MPI - threads: %d\n", threads
     printf "Number of iterations/burn-in: %d/%d\n", nrep, burn
     printf "Elapsed time (in sec): %6.3f \n", tim
     print "-----------------------------------"
     print "Overall sampling statistics"
     printf "%10.5f", result
     print "-----------------------------------"
     print "Best specifications:"
     printf "\n"
     loop i=1..sum(threshold) --quiet
          printf "Model_%d: P(M|D)=%f\n", top_models[1,$i], top_models[2,$i]
          print var_names$i
          printf "\n"
          #print "-----------------------------------"
     endloop
     print "-----------------------------------"
     if threads!=1
          printf "Gelman & Brooks convergence statistics: %3.3f \n", GB
          print "-----------------------------------"
     endif

end function

#################
#####
#In order to perform mpi we need to build and save a bundle with all initial information

function bundle initial_setup(const matrix Y,const matrix X_0, int type,\
const matrix pr_mean, const matrix pr_var, int prior_mod, const matrix Phi,\
bool const_case,int dynamics,int change_var, scalar adapt_rate,\
bool resamp, int threads, int nrep, int burn)

     bundle setup

     setup.depvar = Y
     setup.matregr = X_0
```

```
    setup.link = type
    setup.pm = pr_mean
    setup.pv = pr_var
    setup.pmod = prior_mod
    setup.phi = Phi
    setup.cons = const_case
    setup.dyn = dynamics
    setup.change = change_var
    setup.adaptrate = adapt_rate
    setup.resamp = resamp
    #setup.core = threads
    setup.iterat = nrep
    setup.burnin = burn

    return setup

end function


function bundle BMA(matrix Y, matrix X_1, scalar type,\
matrix pr_mean, matrix pr_var, scalar prior_mod, matrix Phi, scalar const_case,\
scalar dynamics, int change_var, scalar adapt_rate,\
bool resamp, scalar nrep, scalar burn, matrix points)

   series y = Y
   list X = X_1
   matrix X_0 = {X}

   scalar k=cols(X_0)
   scalar obz=rows(X_0)

   matrix b_samp=zeros(k,nrep) #matrix which contains the beta sampled at each step
   matrix m_samp=zeros(k,nrep) #matrix which contains model specifications
   matrix id_samp=zeros(1,nrep-burn)

   if const_case==0 #adapt for diffuse const
   #notice that this influences only the binary representation,\
   but not the decimal one!
        k-=1
        scalar max_modnumb= 2^k
        #we also  consider the null model, with only the const in this case
   else
        scalar max_modnumb= 2^k-1
        #total number of possible models (void model excluded!)
   endif

   bundles MOD = array(max_modnumb) #array of bundles, each one is a single model

   if dynamics == 3
       loop j=1..max_modnumb --quiet
           MOD[$j].sss_prob=points[$j]
       endloop
   endif
```

```
matrix U1 = muniform(nrep,1)#for param sampling
matrix U2 = muniform(nrep,1)#for model sampling

####initial model, full model
matrix model=ones(k,1)
scalar id = convert(model)

fillthebundle(&MOD, model, id,  y, X, X_0, pr_mean, pr_var, type, const_case)

sampling(y, &MOD, id, type, const_case, randgen1(u,0,1))

scalar prop_ratio=1 #ratio between model proposal distribution for MCMC use

scalar adaption_0 = 0.5
scalar adaption = adaption_0

loop i=1..nrep --quiet

    #model movement
    modelnew=modelmove(model, dynamics, k, change_var,\
    adaption,  &prop_ratio, &MOD, const_case)

    newid=convert(modelnew)

    if const_case==0
         newid = (newid==0) ? max_modnumb : newid #the null model has id = maxmodnumb
    endif

    fillthebundle(&MOD, modelnew, newid, y, X, X_0, pr_mean, pr_var, type, const_case)

    scalar det_ratio=1
    scalar g_u= 1
    matrix u={}

    #transformation

    newb = b_born(MOD, id, newid, &u, &det_ratio, &g_u)

    #judge the new model
    scalar alpha = ratio(y, newb, MOD, newid, id, det_ratio, g_u, type,\
    const_case, prop_ratio)

    if prior_mod != 0 #not uniform
         Pm_0= model_prior(Phi, model)
         Pm_1= model_prior(Phi, modelnew)
         Pm= Pm_1/Pm_0
         rho=xmin(1,alpha*Pm)
    else #uniform
         rho=xmin(1,alpha)
    endif

    if dynamics==2 #adaption!
```

```
          if adapt_rate==0
               sgen=adaption_0/$i
               adaption = adaption + sgen*(rho-0.3)
               if adaption>1
                    adaption=1
               elif adaption < 0
                    adaption =0
               endif
               #print adaption
          else
               prog= $i/nrep*100

               if !(prog%adapt_rate)==1 && prog!=100
                    sgen=adaption_0/$i
                    adaption = adaption + sgen*(rho-0.3)
                    if adaption>1
                         adaption=1
                    elif adaption < 0
                         adaption =0
                    endif
               endif

          endif

     endif


     ################
     if U2[i] <= rho
          if i>burn
               MOD[newid].acctimes+=1
               MOD[id].rejtimes+=1
          endif

          accept(&model, modelnew, &id, newid, newb, &MOD)

     else
          if resamp      #if move not accepted resampling
               sampling(y, &MOD, id, type, const_case, U1[$i])
          endif

          if i>burn
               MOD[id].acctimes+=1
               MOD[newid].rejtimes+=1
          endif

     endif

     if const_case==0
          m_samp[,$i]=1|model
     else
          m_samp[,$i]=model
     endif
```

```
            matrix bar=MOD[id].bsamp

            if i>burn
                 id_samp[i-burn]=id
            endif


            scalar count=0
            loop j=1..rows(m_samp) --quiet  #build the b extraction according to its model

                 if m_samp[$j,$i] == 1
                      b_samp[$j,$i]=bar[$j-count]
                 else
                      count += 1
                 endif
            endloop

  endloop

  #bundle ultima

  b_samp=b_samp[,burn+1:]
  m_samp=m_samp[,burn+1:]

  #now let us count number of times the model are occured
  id_s=values(id_samp)'
  matrix id_count=zeros(1,cols(id_s))

  loop i=1..cols(id_s) --quiet
       id_count[$i]= MOD[id_s[$i]].acctimes
  endloop

  bundle BMAstuff
  BMAstuff.coeffmodel=b_samp|m_samp
  BMAstuff.ids=id_s|id_count

  return BMAstuff
end function


####################################################################################
#public function ###################################################################
####################################################################################


function bundle bma_glm(series y            "dep variable",
                        list X              "list of regressors",
                        int type[1:4]       "link function",
                        matrix pr_mean      "prior mean",
                        matrix pr_var       "prior var",
                        int prior_mod[0:2]  "model prior type",
                        matrix Phi          "parameter of prior on beta",
                        bool const_case[0]  "const prior distribution",
                        int dynamics        "movement kernel (model)"",
                        int change_var[1]   "n of var to change",
                        scalar adapt_rate[0]"regeneration time in adaption",
                        bool resamp[0]      "resampling",
                        int threads[1]      "number of threads(core) mpi",
                        int nrep            "n of MCMC simul",
                        int burn            "burn-in sample")



         ######################################### START ######################

      set stopwatch



      #adjust the regressor in case of diffuse prior on const
      if const_case == 0
           inside=inlist(X,const)
           if inside!=0
                con=ones(nelem(X),1)
                con[inside]=0
                list X-=const
                matrix pr_mean = selifr(pr_mean, con)
            endif


                if pr_var[1]==1

                    loop foreach i X --quiet
                        $i = (X.$i - mean(X.$i))/sd(X.$i) #stdize regressors
                        X[i] =$i
                    endloop
                elif pr_var[1]==2

                    loop foreach i X --quiet
                        $i = X.$i - mean(X.$i) #demeaned regressors --
see Fernandez et al(2001)
                        X[i] =$i
                    endloop
                endif


           # the diffuse case is different in linear and in glm
           list X = const || X

           pr_mean = 0 | pr_mean


      endif

      #Initialization
```

```
matrix Y = y
matrix X_0 = X
bundle setup = initial_setup(Y, X_0, type, pr_mean, pr_var, prior_mod,\
Phi, const_case, dynamics,change_var, \
adapt_rate, resamp, threads,  nrep,  burn )

bwrite(setup, "initial")

if dynamics==3 #In Shotgun we need a first mpi exploration for computing the score

    mpi --send-functions --send-data

        matrix Y_s
        matrix X0_s
        scalar type_s
        matrix pr_mean_s
        matrix pr_var_s
        scalar prior_mod_s
        matrix Phi_s
        scalar const_case_s
        matrix S
        matrix mod

        if $mpirank == 0
            bundle Ini = bread("initial")
            Y_s = Ini.depvar
            X0_s = Ini.matregr
            type_s = Ini.link
            pr_mean_s =  Ini.pm
            pr_var_s = Ini.pv
            prior_mod_s = Ini.pmod
            Phi_s = Ini.phi
            const_case_s = Ini.cons

            scalar k=cols(X0_s)
            if const_case_s==0
                 k-=1
                 scalar max_modnumb= 2^k
            else
                 scalar max_modnumb= 2^k-1
            endif

            matrix mod = seq(1,max_modnumb)

        endif

        mpibcast(&Y_s)
        mpibcast(&X0_s)
        mpibcast(&type_s)
        mpibcast(&pr_mean_s)
        mpibcast(&pr_var_s)
        mpibcast(&prior_mod_s)
        mpibcast(&Phi_s)
        mpibcast(&const_case_s)

        mpiscatter(&mod, bycols)

        list Z_s= X0_s
        matrix Z0_s={Z_s}

        S=score(Y_s,Z0_s, Z_s, mod,type_s, pr_mean_s, pr_var_s,\
        prior_mod_s, Phi_s, const_case_s)

        mpireduce(&S,hcat)

        if $mpirank == 0

                 S=exp(S-max(S))
                 #for computational easiness we subtract a same quantity
                 to all values


            mwrite(S, "sss.mat")
        endif
    end mpi
endif

mpi --send-functions --send-data

    matrix Y
    matrix X_1
    scalar type
    matrix pr_mean
    matrix pr_var
    scalar prior_mod
    matrix Phi
    scalar const_case
    scalar dynamics
    scalar change_var
    scalar adapt_rate
    scalar resamp
    scalar nrep
    scalar burn
    matrix RES_coeffmod
    matrix RES_synth
    matrix RES_ids

    if $mpirank == 0
        bundle Big = bread("initial")
        Y = Big.depvar
        X_1 = Big.matregr
        type = Big.link
        pr_mean =  Big.pm
        pr_var = Big.pv
        prior_mod = Big.pmod
        Phi = Big.phi
        const_case = Big.cons
        dynamics = Big.dyn
        change_var=Big.change
        adapt_rate=Big.adaptrate
        resamp=Big.resamp
        nrep = Big.iterat
        burn = Big.burnin

     endif

     mpibcast(&Y)
     mpibcast(&X_1)
     mpibcast(&type)
     mpibcast(&pr_mean)
     mpibcast(&pr_var)
     mpibcast(&prior_mod)
     mpibcast(&Phi)
     mpibcast(&const_case)
     mpibcast(&dynamics)
     mpibcast(&change_var)
     mpibcast(&adapt_rate)
     mpibcast(&resamp)
     mpibcast(&nrep)
     mpibcast(&burn)

     if dynamics == 3
         matrix points=mread("sss.mat")
     else
         points=0
     endif


     RES_partial = BMA( Y, X_1, type, pr_mean, pr_var,  prior_mod, Phi,\
    const_case,  dynamics, change_var, adapt_rate,\
    resamp, nrep, burn, points)

     matrix RES_coeffmod=RES_partial.coeffmodel
     matrix RES_ids=RES_partial.ids

     #in order to build the matrix of syntesis measures we have to
    perform some calculus

     matrix mean_coeff=meanr(RES_coeffmod[1:floor(0.5*(rows(RES_coeffmod))),])
     matrix cov_coeff=mcov(RES_coeffmod[1:floor(0.5*(rows(RES_coeffmod))),]')
     matrix pip_coeff=meanr(RES_coeffmod[floor(0.5*(rows(RES_coeffmod)))+1:,])
     #print mean_coeff cov_coeff pip_coeff
     matrix RES_synth=mean_coeff~cov_coeff~pip_coeff

     mpireduce(&RES_coeffmod, hcat)
     mpireduce(&RES_ids, hcat)
     mpireduce(&RES_synth, vcat)
     if $mpirank == 0
```

```
            #eval RES
            mwrite(RES_coeffmod, "sampled.mat")
            mwrite(RES_ids, "id.mat")
            mwrite(RES_synth, "synthesis.mat")
        endif
    end mpi --np=threads

  ##########################################################################
  mat_coeffmod=mread("sampled.mat")
  mat_synth=mread("synthesis.mat")
  mat_id=mread("id.mat")

  #now rearrange matrix of synthesis measure - for computation easiness
  mat_sampmean=mat_synth[,1] #first col are sampled means
  mat_sampmean=mshape(mat_sampmean, nelem(X), threads)

  mat_sampvar=mat_synth[,2:nelem(X)+1] #the within matrix are covariances

  mat_samppip=mat_synth[,nelem(X)+2] #last col are sampled pips
  mat_samppip=mshape(mat_samppip, nelem(X), threads)

  bundle ultima
  tim=$stopwatch

  ultima.sampled_coeff=mat_coeffmod[1:floor(0.5*(rows(mat_coeffmod))),]
  ultima.sampled_model=mat_coeffmod[floor(0.5*(rows(mat_coeffmod)))+1:,]
  ultima.sampled_id=mat_id
  ultima.sampled_mean=mat_sampmean
  ultima.sampled_var=mat_sampvar
  ultima.sampled_pip=mat_samppip
  ultima.nrep=nrep
  ultima.burnin=burn
  ultima.temp=tim

  printres(&ultima, X, type, prior_mod, dynamics, change_var,\
  adapt_rate, threads, resamp, const_case)

  return ultima

end function


#########################################################################################
```