Computational Statistics (2024) 39:3547–3578
https://doi.org/10.1007/s00180-024-01466-5

ORIGINAL PAPER

Bayesian regression models in `gretl`: the `BayTool` package

Luca Pedini[1]

Received: 26 October 2023 / Accepted: 15 January 2024 / Published online: 21 February 2024

Abstract
This article presents the `gretl` package `BayTool` which integrates the software functionalities, mostly concerned with frequentist approaches, with Bayesian estimation methods of commonly used econometric models. Computational efficiency is achieved by pairing an extensive use of Gibbs sampling for posterior simulation with the possibility of splitting single-threaded experiments into multiple cores or machines by means of parallelization. From the user's perspective, the package requires only basic knowledge of `gretl` scripting to fully access its functionality, while providing a point-and-click solution in the form of a graphical interface for a less experienced audience. These features, in particular, make `BayTool` stand out as an excellent teaching device without sacrificing more advanced or complex applications.

Keywords `Gretl` · Bayesian methods · Gibbs sampling · Parallelization

1 Introduction

Bayesian methods, since their introduction in Econometrics (Bernardo and Smith 1994; Poirier 1995; Zellner 1971), have proposed a rigorous methodological alternative to classical estimations. However, the applicability of the Bayesian scheme is inherently a consequence of the computational advances of the last decades: the enhancement of computing machines on the one hand, the introduction of Markov Chain Monte Carlo (MCMC) simulators, on the other, have revolutionized the field, making the derivation of previously intractable formalization possible. In this sense Bayesian econometrics, and more generally Bayesian statistics, are "computationally-driven" sciences.

✉ Luca Pedini
l.pedini@staff.univpm.it

[1] Department of Economics and Social Sciences, Università Politecnica delle Marche, Piazzale R. Martelli, 8, 60121 Ancona, Marche, Italy

Given this premise, the availability of efficient statistical and econometric software which makes Bayesian estimation available to the user has been and is still a necessary ingredient for the success of the recipe: the key factor is the balance between the user's accessibility, flexibility and execution time.

In this paper I introduce a `gretl` package named `BayTool` which provides Bayesian estimation and post-estimation tools for widely used econometric models: the `gretl` ecosystem counts several frequentist methods, which are either natively supported or supplied by means of auxiliary function packages; Bayesian methods are a minority and pertain to only specific estimation aspects such as model averaging with the packages `BMA` (Błażejowski and Kwiatkowski 2015), `BACE` (Błazejowski and Kwiatkowski 2018), `ParMA` (Lucchetti and Pedini 2022), or more recently Vector Autoregressive specifications (the prototype package `BVAR`, Pedini and Schreiber 2023). However, a Bayesian counterpart for common specifications is missing and `BayTool` aims to bridge the gap.

The underlying idea is to provide intuitive functions that could mimic the classical `gretl` commands, guaranteeing a simple solution for different kinds of usage. `BayTool` provides a graphical interface (GUI), which truly stands out as an excellent medium for users who are new to the `gretl` software or to the Bayesian paradigm. On the other hand, using the package in scripting mode, which allows to access a higher degree of customization, is equally simple and requires only basic `gretl` knowledge: the implied workflow is transparent to the user and exploits a syntax which is shared by all included functions. These elements build an ideal workspace for teaching purposes: note that one of the major user base of `gretl` is inherently academic, with the software employed as a learning-teaching instrument. In adherence with this, `BayTool` provides several examples from all major Bayesian textbooks such as Chan et al. (2019), Koop (2003) via sample scripts.

Practitioners, however, may also find the package useful for more advanced uses: regularization methods such as the Bayesian LASSO by Park and Casella (2008) are provided; efficient computational schemes such as the Held and Holmes (2006) method for probit simulation are included; post-estimation and MCMC diagnostics are available; finally, parallelization is possible and is handled via the native Message Passing Interface (MPI) support.

The rest of the article is organized as follows: Sect. 2 reviews the competing software and function packages; Sect. 3 sketches the preliminary background of the models here considered; Sect. 4 describes the package and the included functions; Sect. 5 provides some replication exercises together with computational analysis; finally Sect. 6 concludes.

2 Review on competing software

The archetype of the Bayesian algorithm is generally a MCMC sampler which simulates pseudo-random numbers from the distribution of interest, namely the posterior.[1]

[1] For a detailed analysis of Monte Carlo methods see Gelman et al. (1995), Geweke (1999, 2005), Greenberg (2012), Koop (2003), Lancaster (2004), Robert et al. (1999).

Depending on the cases, this is often accomplished via Gibbs sampling (Gelfand and Smith 1990) or Metropolis-Hastings (Hastings 1970) methods. The former is computationally more efficient, since posteriors are iteratively sampled from their conditional distributions, but less flexible to the extent that closed-form conditionals impose specific prior choices. The Metropolis-Hastings scheme, instead, removes the limitation: the simulation relies on draws from a candidate distribution, whose proposals are judged via an accept-reject mechanism and updated accordingly. The generality is higher, but the quality of the sampling is heavily dependent on the choice of appropriate candidates, often difficult in practice. More recently, the need for a universal sampler, which could simulate any distribution at hand without sacrificing computational efficiency or accuracy, has led to the introduction of new schemes, namely adaptive Metropolis-Hastings (Haario et al. 2001; Roberts and Rosenthal 2007, 2009), the slice sampling (Neal 2003) and the Hamiltonian or Hybrid Monte Carlo methods (Betancourt 2017; Duane et al. 1987; Girolami and Calderhead 2011; Neal 1994).

This duality between efficiency and flexibility is reflected in the software panorama with the so-called case-specific algorithms on the one hand, and generic all-purposes ones on the other hand. As the name suggests, a case-specific algorithm is strictly dependent on the chosen specification (e.g., linear model or binary model), with priors following only specific distributional forms. This to provide sampling efficiency via the Gibbs method: an example could be the linear model with priors only from Normal-Inverse Gamma distributions (see Sect. 3.1). From a practical perspective, the major requirement for the user is the specification of the prior hyperparameters. Conversely, a generic sampler tries to automatically choose the ideal simulation scheme for each model in usage and employs advanced methods to approximate any given distribution. Priors are not restrained by the specification choices, but the requirements in terms of scripting skills are much more pronounced.[2]

The `BUGS` project (Lunn et al. 2012), with its declinations `WinBUGS` (Spiegelhalter et al. 2003) and `OpenBUGS` (Spiegelhalter et al. 2007), has probably been the first contribution in the direction of proper Bayesian software and a universal sampler. Both `WinBUGS` and `OpenBUGS` are written in Component Pascal and allow the user to compute estimation methods via point-and-click windows or via scripting following the `BUGS` language. This hinges upon some fundamental statements such as the `model` declaration, which sketches a statistical model in terms of blocks and nodes. Priors can come in any shape posing no restrictions to the user, while sampling is performed with Gibbs sampler when available, Metropolis-Hastings, adaptive schemes, slice sampling in the remaining cases. However, several limitations are in place: the impossibility of defining user-defined functions and the operating system dependency with `WinBUGS` available only on Windows and `OpenBUGS` on Linux too, led to the introduction of cross-platform software such as `JAGS` (Plummer 2017) or `Stan` (Carpenter et al. 2017; Stan Development Team 2023). Both are written in C++ and overcome the `BUGS` limitations: `JAGS` integrates and innovates the predecessor not only making it possible to invoke directly `BUGS` commands and files, but also introducing new

[2] A somehow more intuitive distinction between case-specific philosophy and the generic sampler one comes in the form of software functions: software which uses case-specific samplers provides several dedicated functions, each one related to specific model-prior setups; a generic sampler, on the other hand, is supplied in the form of a single function which can be tuned using either its arguments or external files.

functionalities in the form of internal modules for advanced or "exotic" applications. All the standard samplers together with slice sampling are implemented. `Stan`, on the other hand, relies on Hamiltonian Monte Carlo with the No-U-Turn Sampler (Hoffman et al. 2014): this scheme can approximate any continuous distribution at hand with the additional benefit of automatically tuning hyperparameters. Differently from `JAGS`, `Stan` language is not borrowed from `BUGS`, even if many similarities are present.

Besides these specific statistical software languages for Bayesian methods, many generic posterior simulators are provided as `R` packages: `Bayesiantools` (Hartig et al. 2023), `LaplacesDemon` (Hall et al. 2021), `MCMCpack` (Martin et al. 2022), `mcmc` (Geyer and Johnson 2020) and `nimble` (de Valpine et al. 2017, 2023) are probably the most notorious examples. Special mentions to those packages that port `JAGS` or `Stan` functionalities into `R` such as `greta` (Golding et al. 2022), `rjags` (Plummer et al. 2023), `R2jags` (Su and Yajima 2021), `runjags` (Denwood and Plummer 2023) or `brms` (Bürkner et al. 2023) and `rstan` (Guo et al. 2023).

As concerns case-specific algorithms, this concurring category has arisen from the necessities of applied scientists such as economists or applied econometricians who do not require the complexity of a universal sampler but prefer to be stuck to simple and efficient routines. Proprietary software such as `MATLAB` or `STATA` mostly adheres to this philosophy. `MATLAB`, for example, provides some internal routines for Bayesian linear models and regularized regressions with specific priors; the same occurs with the downloadable Lesage's Econometrics Toolbox (Lesage 1999) or Koops' scripts (Chan et al. 2019; Koop 2003). `STATA` implements case-specific functions which take advantage of adaptive MCMC schemes. These, in principle, can fit several prior distributions, but the user can also recur to default choices with minimal scripting effort. On the open-source side, `R` has some dedicated packages such as `arm` (Gelman et al. 2022) for linear and generalized linear models, `bayesm` (Rossi 2022) for microeconometric models or `monomvn` (Gramacy et al. 2023) for Bayesian shrinkage methods. As previously mentioned, Bayesian methods are not natively included in `gretl`, but function packages have integrated some of these functionalities over time, addressing peculiar user necessities: the packages `BMA` (Błażejowski and Kwiatkowski 2015), `BACE` (Błazejowski and Kwiatkowski 2018), `ParMA` (Lucchetti and Pedini 2022), for instance, add Bayesian model averaging and model selection tools.

`BayTool`, as a consequence, is the first attempt to provide Bayesian functions of wider applicability in `gretl`: the package proposes a case-specific approach for Bayesian sampling in common-use econometric models, with dedicated functions that pursue low execution time and simplicity in usage. Finally, to fully understand the benefits of `BayTool` especially versus competing software, it should be noted that: (i) the sole fact that `gretl` belongs to the free software ecosystem induces *per se* a clear distinction with respect to `MATLAB` and `STATA`; (ii) the `gretl` community refers mainly to econometricians and economists who use the software for teaching but also for real-life applications, meaning a distinct audience to address in comparison with more general Bayesian inference software (e.g., `BUGS`, `JAGS`, `Stan`) or the `R` environment. The case-specific approach finds its intrinsic motivation in a user base who benefits from standardized and specialized routines with few coding prerequisites. Lastly, (iv) Bayesian econometrics novices may find the `gretl` accessibility appealing.

3 Preliminaries and notation

In this section, I will introduce the mathematical notation used throughout the paper and I will briefly report the statistical characterization of the analyzed models. For major details on the statistical framework I refer to specialized textbooks such as (Chan et al. 2019; Geweke 2005; Koop 2003; Lancaster 2004).

Start by considering a generic econometric model with $\mathcal{D}$ as the data of a given phenomenon of interest and θ, the unknown parameter which links the data to the model. Through Bayes' law,

$$P(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)P(\theta),$$

where $p(\mathcal{D}|\theta)$ identifies the likelihood, i.e., the data contribution to θ; $P(\theta)$ is the prior, which summarizes the non-data related information, while $P(\theta|\mathcal{D})$ represents the posterior distribution, which embodies all available knowledge about θ and therefore is the main object of investigation.

In order to keep the exposition concise, the specifications reported below are just a selection from `BayTool` included models. These are specifically intended to highlight different aspects of the package: the linear model as the universal benchmark for sampling accuracy; the Bayesian LASSO by Park and Casella (2008) due to the increasing interest in data regularization methods; the probit model for binary choices as a leading example of the latent variable approach.[3] Posterior computation is executed via Gibbs sampling.

3.1 Linear model

Consider the following linear specification:

$$y = X\beta + \varepsilon, \quad \varepsilon \sim N_n(0, \sigma^2 I_n) \tag{1}$$

where y is the $n \times 1$ vector of *i.i.d.* observations for the dependent variable; X is a $n \times k$ matrix of covariates; β the $k \times 1$ vector of covariate coefficients, ε the idiosyncratic error term. The notation $N_n(0, \sigma^2 I_n)$ identifies a multivariate Normal distribution of dimension n with mean 0 and variance-covariance matrix equal to the product of the parameter σ^2 and the $n \times n$ identity matrix.[4] Recalling the previous notation, $\theta = (\beta, \sigma^2)$ and $\mathcal{D} = (y, X)$, where under the canonical linear model assumptions, the dependency from X can be dropped.

Analytical solutions for the posteriors can be retrieved by specifying the so-called Normal-Inverse Gamma (NIG) conjugate prior, with $P(\beta|\sigma^2)P(\sigma^2)$ and $\beta|\sigma^2$ as Normal, σ^2 as Inverse Gamma. In this exposition, however, an independent NIG will be used,

[3] For the excluded models I refer to Sect. 4 or more generally to the package manual (Pedini 2023).

[4] The distribution has to be intended as univariate when the subscript is omitted.

$$P(\beta, \sigma^2) = P(\beta)P(\sigma^2),$$
$$\beta \sim N_k(\beta_0, V_0), \tag{2}$$

$$\sigma^2 \sim IG(a_0, b_0) \tag{3}$$

where β_0, V_0, a_0, b_0 are hyperparameters up to the user and $IG(a_0, b_0)$ indicates an Inverse Gamma with shape a_0 and scale b_0. In the context of linear models, such parameters admit a quite natural interpretation with the j-th element of β_0, namely $\beta_{0,j}$, and its related variance entry $V_{0,jj}$ being, respectively, the expected marginal effect of variable x_j on y and its variability, according to the researcher's *a priori* view. The matrix V_0 is commonly assumed diagonal due to the practical difficulties in eliciting covariances. The hyperparameters a_0 and b_0, instead, are indirectly derived from the moments of σ^2: the expected variability of the residuals should guide this choice, as reported in Sect. 5.1. Remember that prior information is "external" to the data at disposal and may derive from economic theory, common sense, experience or previous analysis on similar experiments.

Coming back to the framework adopted, the independent setup is not only more flexible than the conjugate, but also one of the most used in practice. The resulting posterior can be handled via a Gibbs sampler which loops through the *conditioned posteriors*: a Normal distribution for $\beta|\sigma^2, y$ and an Inverse Gamma for $\sigma^2|\beta, y$.

3.2 Bayesian LASSO

Reconsider the linear framework as in Eq. (1): in the context of statistical learning methods, a Bayesian alternative to the LASSO (Tibshirani 1996) is easily obtained using hierarchical priors. Following Park and Casella (2008),

$$\beta|\sigma^2, \tau_1^2, \tau_2^2, ..., \tau_k^2 \sim N_k(0, \sigma^2 D_\tau), \tag{4}$$

$$P(\sigma^2) = \frac{1}{\sigma^2}, \tag{5}$$

$$P(\tau_j^2) = \frac{\lambda^2}{2} \exp\left[-\lambda^2 \frac{\tau_j^2}{2}\right] \tag{6}$$

where D_τ is a diagonal matrix with entries $\tau_1^2, \tau_2^2, ..., \tau_k^2$, λ is a parameter which acts as a shrinkage factor; σ^2 has the same role as in Sect. 3.1 but with a diffuse prior; finally, $\tau_1^2, \tau_2^2, ..., \tau_k^2 > 0$. Differently from Eqs. (2)–(3), the above prior specification imposes a specific structure to both β, σ^2 via the introduction of τ. In this way, the sole tunable hyperparameter becomes λ. To mitigate the sensibility of the results to this, it is customary to elicit an additional prior:

$$\lambda^2 \sim G(r_0, \delta_0) \tag{7}$$

where $G(a, b)$ identifies a Gamma distribution with shape a and rate b. Equations (4)–(7) fully characterize the Bayesian LASSO and posterior derivation can be accom-

plished via Gibbs sampling, since conditional posteriors are known: a Normal for the β-conditional, an Inverse Gamma for σ^2 and an Inverse Gaussian for τ_j^2.

3.3 Binary choice model

Binary choice models are usually described in microeconometrics in terms of latent utility where a given action y_i is undertaken ($y_i = 1$) by the i-th agent if and only if the resulting net utility z_i is positive.

In symbols,

$$z_i = x_i^T \beta + \varepsilon_i, \quad \varepsilon_i \sim N(0, 1) \tag{8}$$

$$y_i = \begin{cases} 1 & z_i > 0 \\ 0 & z_i \leq 0 \end{cases} \tag{9}$$

with the subscript $i = 1, \ldots, n$ identifying the *i-th* observation. Since ε is assumed to be Normal, Eqs. (8)–(9) give rise to a probit model where the parameter of interest is β.[5]

Assuming a Normal prior for β as in Eq. (2) leads to a posterior which can be easily retrieved via MCMC methods: the most popular one follows Albert and Chib (1993) and builds a Gibbs sampler from the latent variable z,

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

then marginalizing for β. The result exploits $\beta|z, y$ being Normal as a consequence of the linear model in Eq. (8) and $z|\beta, y$ as a Truncated Normal. The simplicity, however, comes at the cost of heavily auto-correlated draws: `BayTool`, to counter this fact, proposes a supplementary sampler for probit specifications, i.e., the Held and Holmes (2006) scheme. This is a Gibbs sampler which jointly updates β and z by virtue of the following result:

$$P(\beta, z|y) = P(\beta|z, y) P(z|y)$$

3.4 Diagnostics

Alongside simulation algorithms, `BayTool` provides several diagnostics tools to measure the efficiency and the quality of MCMC samples: needless to say, monitoring the convergence to the stationary state and the mixing of the chain are key aspects necessary to draw reliable conclusions and inference.

In the literature, several indicators have been proposed ranging from graphical inspections to rigorous statistics. The traceplot, i.e., the time series plot of the MCMC draws, the running mean graph and the visualization of the autocorrelation function (ACF) belong to the first category. Even though immediate to interpret and extremely

[5] In a binary model β and σ^2 are not jointly identified. For this reason, it is assumed $\sigma^2 = 1$.

efficient for detecting issues, virtuous practice generally imposes to couple such measures with the second category: the widely used effective sample size (ESS) by Kass et al. (1998) is an example. ESS indicates the number of independent Monte Carlo iterations required to achieve the same precision as the current MCMC sample. Hypothetically, the closer the ESS is to the actual number of MCMC replicas, the better the chain is mixing.

More formally, following Vats et al. (2019), the ESS for a univariate sampled parameter β is defined as,

$$\text{ESS} = m\frac{\phi^2}{\sigma^2(\beta)}$$

where m is the number of MCMC iterations after burn-in, ϕ^2 and $\sigma^2(\beta)$ are respectively the sample variance and the true variance of β, with the latter commonly estimated via the batch method. The multivariate version (mESS) is instead defined as,

$$\text{mESS} = m\left(\frac{|\Phi|}{|\Sigma|}\right)^{1/k}$$

where the matrices Φ and Σ are the sample and true variance-covariance matrices of a $k \times 1$ parameter vector β; the operator $|\cdot|$ identifies the determinant. Again, Σ is consistently estimated via the batch method.

Another popular procedure, used especially in the context of parallel chains[6], is the Brook-Gelman convergence statistic (Brooks and Gelman 1998; Gelman and Rubin 1992): consider to have c parallel MCMCs of the same model, each one with $\overline{m} = m/c$ iterations after a constant burn-in. The sampled between-variance B (between parallel chains) of an univariate parameter β should approach 0 when convergence has taken place, leading to the total variance explained by the sole within-group one (W).

In symbols,

$$B = \frac{\overline{m}}{c-1}\sum_{j=1}^{c}(\bar{\beta}_j - \bar{\beta})^2; \qquad W = \frac{1}{c(\overline{m}-1)}\sum_{j=1}^{c}\sum_{l=1}^{\overline{m}}(\beta_{lj} - \bar{\beta}_j)^2$$

with β_{lj} as the *l-th* draw of β in chain j, $\bar{\beta}_j$ the related sample mean and $\bar{\beta}$ the total sample mean. The Gelman–Rubin univariate statistic (Gelman and Rubin 1992) computes the ratio of total and within variance:

$$R = \frac{V}{W}, \qquad V = \frac{\overline{m}-1}{\overline{m}}W + \frac{c+1}{c}\frac{B}{\overline{m}}$$

where $R < 1.2$ is universally considered as the condition for convergence. The multivariate version is proposed instead by Brooks and Gelman (1998), as

$$\mathbf{R} = \frac{\overline{m}-1}{\overline{m}} + \frac{c+1}{c}\gamma$$

[6] See Sect. 4.4.

where γ is the maximum eigenvalue of $\mathbf{W}^{-1}\mathbf{B}/\overline{m}$, with $\mathbf{W}$ and $\mathbf{B}$ as the multivariate correspondents of W and B. The same rule of thumb $\mathbf{R} < 1.2$ is applied. Notice that both Gelman–Rubin and Brooks–Gelman statistics signal the convergence to the stationary state, but do not properly acknowledge the correlation of the chain; similarly, the ESS indicates the degree of auto-correlation but indirectly ascertains the convergence. Consequently, both statistics should be coupled or at least backed with auxiliary analysis.

Additional indicators included in `BayTool` are the Geweke stationarity test (Geweke 1992) and the Heidelberger-Welch test (Heidelberger and Welch 1983): both of them appear as robust variants of mean difference statistics computed on particular windows of the sample. The Geweke statistic is obtained as

$$G = \frac{\bar{\beta}_{S_0} - \bar{\beta}_{S_1}}{\sqrt{(\tilde{\Omega}_{S_0}/0.1m) + (\tilde{\Omega}_{S_1}/0.5m)}} \sim N(0,1)$$

where β is the previous univariate sampled parameter; the subscripts S_0 and S_1 denote respectively two different sub-samples from the m draws, with S_0 as the first 10% and S_1 the last 50%. $\tilde{\Omega}$ corresponds to the long-run variance of β. When stationarity is achieved the test does not reject, signalling homogeneity between draws along the chain.

Heidelberger-Welch test, instead, is given as:

$$HW = \frac{\sum_{j=1}^{mt} \beta_j - \lfloor mt \rfloor \bar{\beta}}{\sqrt{m\tilde{\Omega}}}, \qquad 0 \leq t \leq 1$$

where $\lfloor \cdot \rfloor$ designates the floor operator. The test is asymptotically distributed as a Brownian Bridge and the Cramer-von Mises statistic is used to verify the stationarity of the sequence. The scalar t is assumed to be 1, but if a rejection occurs, its value is decreased to 0.9 and the first $1-t$ proportion of the sample discarded. This procedure is iterated till the test does not reject or $t = 0.5$ and HW still rejects. It is possible to conclude for the non-stationarity only when this last scenario occurs.

4 The package

Function packages in `gretl` are mainly meant to add to the native software non-trivial statistical functionalities in the form of estimation methods or testing procedures (Cottrell and Lucchetti 2023a, c). The general package structure follows the one of a `.zip` folder containing the function code as a `.gfn` file[7], a PDF manual, data files and supplementary materials such as additional datasets and sample scripts. Official packages are accessible and downloadable from the online repository via the `gretl` main window, under the path `File->Function packages->On server`. Alterna-

[7] A `.gfn` file is simply a XML conforming to the `gretl` standard as in the Document Type Definition `gretlfunc.dtd`. For more details, see Cottrell and Lucchetti (2023a)

tively, in scripting mode, the command `pkg install` performs the task as shown below:

```
? pkg install BayTool
```

Once installed, the package functions can be loaded into the current workspace with the command `include` (scripting mode),

```
? include BayTool.gfn
```

On the other hand, if the package is used via the GUI apparatus, the include step can be skipped.

The `BayTool` package provides: (i) functions primarily devoted to the posterior simulation and inference of commonly used econometric models; (ii) auxiliary functions which address the "post-estimation" analysis, such as plotting marginal posterior distributions or computing MCMC diagnostics. In the next sections, I will review the main features of the functions included: particular emphasis will be also placed on computational aspects such as the use of parallelization for MCMC experiments.

4.1 Main functions

Table 1 reports the list of `BayTool` main functions for posterior distribution computation: MCMC methods are employed (especially the Gibbs sampler) whenever analytical solutions are not available. To summarize, `BT_lm` deals with homoskedastic linear models assuming NIG prior specifications; `BT_lm_het`, instead, deals with heteroskedastic linear models allowing for different prior choices; `BT_lasso` provides the Bayesian LASSO following the work by Park and Casella (2008). In the context of latent variable models, `BT_probit` includes the canonical Bayesian probit with data augmentation and the subsequent upgrade due to Held and Holmes (2006); `BT_tobit` is used for Bayesian analysis of censored data.[8]

All functions share a common syntax in terms of the same argument types to establish a notational consistency. I will consider `BT_lm` as an example, but any of the above functions could have been used. The signature is the following,

```
function bundle BT_lm(series y, list X, bundle prior, bundle opt)
```

where

- `y`, is the series representing the dependent variable;
- `X`, is the list of covariates;

[8] This list reflects `BayTool` - version 0.7, which undergoes a rigorous moderation process by the `gretl` development team. Additional functions such as `BT_ordprobit` for ordered probit models and `BT_mnprobit` for multinomial ones are not reported despite unofficially available. These functions will be included in the next official release, but are also available upon request.

Table 1 Main `BayTool` functions and related details

Functions	Model	Priors allowed	Computation	`type` key
`BT_lm()`	Linear model	• $\beta \sim N_k(\beta_0, \sigma^2 V_0)$,	Analytical	`"conj"` (conjugate NIG)
	$y = X\beta + \varepsilon$,	$\sigma^2 \sim IG(a_0, b_0)$		
	$\varepsilon \sim N_n(0, \sigma^2 I)$	• $\beta \sim N(\beta_0, V_0)$,	Gibbs sampler	`"indep"` (independent NIG)
		$\sigma^2 \sim IG(a_0, b_0)$		
`BT_lm_het()`	Linear model (heterosk.)			
	• $y = X\beta + \varepsilon$,	• $\beta \sim N_k(\beta_0, V_0)$,	Gibbs sampler	`"fixed"` (known c_0)
	$\varepsilon \sim N_n(0, \sigma^2 diag(\psi_i))$	$\sigma^2 \sim IG(a_0, b_0)$, $\psi_i \sim G(c_0, c_0)$		
	• $y = X\beta + \varepsilon$,	• $\beta \sim N_k(\beta_0, V_0)$,	MwG	`"hyper"` (hyperprioron c_0)
	$\varepsilon \sim t_n(0, \sigma^2, c_0)$	$\sigma^2 \sim IG(a_0, b_0)$, $c_0 \sim NegExp(v_0)$		
`BT_lasso()`	Bayesian LASSO	• $\beta \sim N_k(0, \sigma^2 diag(\tau_j^2))$,	Gibbs sampler	`"fixed"`, (known λ)
	$y = X\beta + \varepsilon$,	$\tau_j^2 \sim (\lambda^2/2)\exp[-\lambda^2(\tau_j^2/2)]$,		`"marglik"`(auto-tuning of λ)
	$\varepsilon \sim N_n(0, \sigma^2 I)$	$\sigma^2 \sim \frac{1}{\sigma^2}$		
		• $\beta \sim N(0, \sigma^2 diag(\tau_j^2))$,	Gibbs sampler	`"hyper"` (hyperprior on λ^2)
		$\tau_j^2 \sim (\lambda^2/2)\exp[-\lambda^2(\tau_j^2/2)]$,		
		$\lambda^2 \sim G(r_0, \delta_0)$, $\sigma^2 \sim \frac{1}{\sigma^2}$		
`BT_probit()`	Probit model	$\beta \sim N_k(\beta_0, V_0)$	• Gibbs sampler	NA
	$z = X\beta + \varepsilon$,		• HH	
	$\varepsilon \sim N_n(0, I)$,			
	$y_i = \mathbb{1}(z_i > 0)$			
`BT_tobit()`	Tobit model	$\beta \sim N_k(\beta_0, V_0)$,	Gibbs sampler	NA
	$z = X\beta + \varepsilon$,	$\sigma^2 \sim IG(a_0, b_0)$		
	$\varepsilon \sim N_n(0, \sigma^2 I)$,			
	$y_i = \max(0, z_i)$			

MwG Metropolis-within-Gibbs. *HH* Held and Holmes (2006) Gibbs sampler

- `prior`, a bundle containing the prior distribution information;
- `opt`, a bundle for additional options. If omitted, the default choices are used.

While `y` and `X` are self-explanatory, the role of `prior` and `opt` deserves a clarification. The bundle `prior` governs the prior setup and plays a role similar to the `model` formula required by `BUGS, JAGS, Stan` files. However, a fundamental distinction occurs: the prior distributions are model-dependent and fixed to the functional forms shown in Table 1, with the possibility for the user to tune the related parameters via bundle keys. This is to preserve computational efficiency and consistency with standard `gretl` commands: `gretl` natively embraces the model-dependent philosophy for frequentist estimation methods with functions or commands such as `ols`, or `probit`. On the same line, `BayTool` proposes the same paradigm to foster the function accessibility for long-standing `gretl` users (potentially new to the Bayesian world) but also for those who search the simplicity in usage.

Recalling the previous linear model example, suppose to use an independent NIG with $\beta \sim N_5(0, I)$, $\sigma^2 \sim IG(5, 100)$, the related `gretl` script goes like

```
? bundle prior = defbundle()
? prior.type = "indep"
? prior.beta_mean = zeros(5,1)
? prior.beta_vcv = I(5)
? prior.sigma_a = 5
? prior.sigma_b = 100
```

In the first line, an empty bundle named `prior` is created; in the second the kind of prior specification is chosen via the key `type`: in the case of linear model two available strings are admitted, `"conj"` for the conjugate setup, `"indep"` for the independent prior case. Notice that `type` admits different values depending on the model specification used: the last column of Table 1 reports the alternatives.

Lines 3-6 show how to define prior parameters: the keys `beta_mean` and `beta_vcv` are universally valid for the prior mean and covariance matrix of β following a Normal distribution. Likewise `sigma_a` and `sigma_b` refer to the shape a_0 and scale b_0 of the Inverse Gamma for σ^2. The complete list of supported keys is reported in Table 2.

The bundle `opt` contains instead all the additional options,; the associate keys are

- `iter`, the number of MCMC iterations after the burn-in (default: 10000);
- `burn`, the number of burn-in replications (default: 1000);
- `thin`, an integer signalling the thinning interval (default = 0);
- `mpi`, the number of cores used for parallelization (default: 1);
- `mpi_seed`, the random number generator seed to use when MPI is active (default: none);
- `verbose`, a Boolean for printing the output on screen (default: 1).

While the above is common to all main functions, some options are specific such as

- `forecast`, a matrix containing the covariate values used for prediction. This key is available only for `BT_lm()` and `BT_lasso()`.

Table 2 Prior parameters admitted in `BayTool`

Bundle key	Distribution	Prior parameter
`beta_mean`	$\beta \sim N(\beta_0, V_0)$	β_0
`beta_vcv`		V_0
`sigma_a`	$\sigma^2 \sim IG(a_0, b_0)$	a_0
`sigma_b`		b_0
`precision_mean`	$1/\sigma^2 \sim G(a_0, b_0)$	a_0/b_0
`precision_df`		$2a_0$
`psi_c0`	$\psi_i \sim G(c_0, c_0)$	c_0
`c0_nu`	$c_0 \sim NegExp(\nu)$	ν_0
`lambda`	λ	λ
`lambda_shape`	$\lambda^2 \sim G(r_0, \delta_0)$	r_0
`lambda_rate`		δ_0

Prior setup and model formalization recall the notation presented in Table 1

- `stdize`, a Boolean flag which standardizes the covariates in case of 1 or simply center them if 0. The dependent variable is always centered (default = 1). Available for `BT_lasso()`;
- `method`, a string used for `BT_probit`. if `method="std"` the canonical Gibbs sampler *à la* Albert and Chib is used, if `method = "HH"`, the Held and Holmes (2006) algorithm is used (default = `"std"`);
- `part_eff`, a Boolean for the probit model. If 1 partial effects are automatically computed and reported in terms of partial effects on the mean and average partial effects. (default = 1);
- `rlimit` and `llimit`, scalars for the right and left censoring points in Tobit models (default: `llimit` = 0, `rlimit` = NA).

Each function returns a bundle with recurring keys regardless of the model used: most of these simply report basic information such as the dependent variable (`dependent`), the covariate list (`covariates`), the prior used (`prior_setup`); the key `sampled_coeff` grants access, instead, to a bundle containing the sampled posterior coefficients in matrix form. This can be further used to inspect directly the posterior distribution, especially if coupled with auxiliary functions such as `BT_paramplot`. Deriving functions of sampled parameters from `sampled_coeff` is immediate and, in the peculiar case of summary statistics, these can be directly retrieved via the utility bundle key `post_summ`.

4.2 Auxiliary functions

The package provides several auxiliary functions that integrate the posterior simulation part with utilities and "post-estimation" tools. `BT_printres` and `BT_paramplot`, for example, make possible the quick visualization of the main posterior features:

`BT_printres` prints on screen prior and posterior summary statistics.[9] The only argument required is the returning bundle from one of the previous posterior calculators. Notice that `BT_printres` is automatically invoked by these functions when `opt.verbose = 1`. `BT_paramplot` produces, instead, the marginal posterior distribution plots via kernel density estimates: the main argument is again the output bundle from a posterior sampler. It optionally admits as extra inputs the list of parameters to plot in the form of an array of strings and a bundle for graphical options (for major details see Pedini 2023). If they are omitted, the function plots on screen all the posteriors contained in the output bundle `sampled_coeff`.

Besides the graphical and summary tools, auxiliary functions mostly pertain to MCMC diagnostics as in Sect. 3.4: five different functions allow the user to explore and monitor the convergence quality of the simulated results. In particular, the included functions are:

- `BT_check_plot`;
- `BT_check_mESS`;
- `BT_check_G`;
- `BT_check_HW`;
- `BT_check_BG`.

Each one accepts as main input the matrix of sampled coefficients of interest: this can be easily retrieved by means of `sampled_coeff` bundle, however, the functionality is wider. Users can invoke diagnostics for any user-defined sampled coefficient matrix. In this sense the aim of `BayTool` is double: on the one hand, providing *ad hoc* Bayesian econometric models and, on the other hand, supplying procedures for monitoring simulation quality, which are suited for Bayesian methods, but not necessarily limited to these.

Coming to the diagnostic functions, `BT_check_plot` produces the traceplot, the running mean plot and the ACF plot for the selected parameters. Its signature is `BT_check_plot(matrix par_mat, strings names, bundle checkopt)`, where `par_mat` is the matrix of coefficients, `names`, an array of strings identifying a particular set of parameters to plot. The strings should correspond to the row names of `par_mat` or, alternatively, to the row numeric indices. The bundle `checkopt` contains some tunable options such as the number of lags for the ACF (`lag`).

`BT_check_mESS` allows the user to compute the numerical standard errors, the univariate and multivariate effective sample size using the batch method (Vats et al. 2019). The main input is `par_mat`, however an optional bundle with key `batch_size` can be supplied to modify the batch size. Finally, `BT_check_G`, `BT_check_HW` and `BT_check_BG` perform respectively the Geweke mean difference test (Geweke 1992), the Heidelberger and Welch stationarity test (Heidelberger and Welch 1983), the univariate and multivariate Brooks and Gelman convergence statistics (Brooks and Gelman 1998; Gelman and Rubin 1992). Both `BT_check_G` and `BT_check_HW` require the matrix `par_mat` as argument, while `BT_check_BG` introduces a slight change: the main input should correspond now to

[9] When available, the posterior results reported are both the ones obtained with non-informative and informative priors.

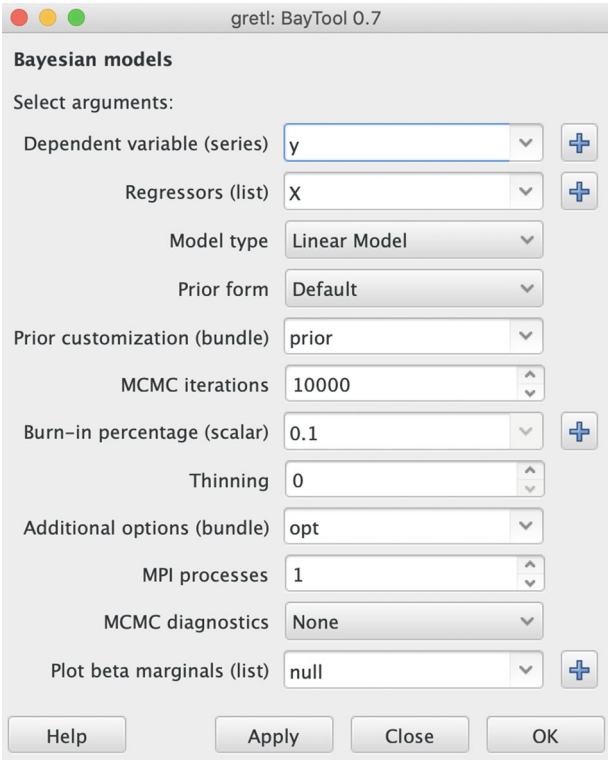

Fig. 1 `BayTool` dialog window

an array of matrices. The statistic is in fact often computed on different (possibly parallel) MCMC chains and the user has to explicitly provide their resulting coefficients. If the option `mpi` is enabled on any previous sampler, the matrix array can be easily obtained from `sampled_coeff` via the native function `msplitby`.

4.3 The GUI and the teaching support

When installed, the package offers the possibility to attach a graphical interface to the `gretl` main window under the path `Model->Bayesian models`: this represents an outstanding way to access the package functionalities, however, it should be noted that the highest degree of flexibility and customization could only be achieved via scripting.

Figure 1 reports the dialog box: most of the entries recall the main function arguments although some simplifications are in place. Firstly, the GUI explicitly imposes a workflow that starts with the model definition, it continues through the simulation step and concludes with diagnostics checking and marginal distribution plots.

The role of the bundle `prior` is here hugely reduced: the "Prior form" entry refers to the bundle key `prior.type`. The user can opt (i) for the "Default" alternative,

where the most efficient prior setup is automatically chosen depending on the model specification, or (ii) for autonomously defining the desired configuration. Posterior computation follows the default prior hyperparameter values, unless additional information is provided via the `prior` bundle. Notice that any bundle has to be initialized and filled with the appropriate quantities before the GUI invocation.

Some options, previously contained in the bundle `opt`, are now made explicit such as the number of MCMC iterations or the number of processes to use under MPI; non-included features can be tuned via the additional option bundle, which however is optional.

The last dialog box entries refer to the diagnostics and the plotting parts: the user can choose to use a single diagnostic check, all the available batteries (see Sect. 4.2) or none. Similarly, the marginal posteriors for the covariate coefficients can be directly obtained for the whole list or a specific choice.

Before concluding, it should be stressed the true GUI potential: after having loaded any kind of data in `gretl`, Bayesian analysis can be performed in just a few clicks and, optionally, just a few console inputs. In this way, the user can entirely focus on the inferential process without devoting time and attention to complex scripting. This aspect is extremely valuable when performing preliminary analysis, especially on the sensibility of prior choices. Students may especially benefit from this intuitiveness when approaching the Bayesian framework for the first time: the GUI represents an interactive tool which hugely simplifies the learning process, requiring almost null programming skills or knowledge.

At the same time, `BayTool` interface may represent the starting point for a more advanced usage: the minimal scripting involved, for example, in prior or option customization is only slightly less complicated than the full application via scripts (see Sect. 5); again, Bayesian novices may find this gradation in the learning approach extremely attractive, since the step required from the GUI usage to proper scripting is effortless.

4.4 Parallelization and MPI

Parallelization in MCMC experiments has become a standard routine supported in most Bayesian statistics software. The standard implementation, known as "embarrassingly parallel", splits the number of iterations across multiple chains, which are simultaneously executed (one per core), and combines the resulting sampled parameters.[10] The procedure clearly mimics the one used in independent Monte Carlo experiments but not negligible limitations take place. The sequential nature of MCMC experiments, in fact, poses a serious threat to the validity of parallelization, firstly because the burn-in period cannot be split and has to be kept fixed for each chain; secondly, because all chains have to converge to the stationary state, possibly with the less degree of autocorrelation. If these conditions are met, the computational benefit could be remarkable even though inferior to the one achieved by independent Monte Carlo experiments.

[10] Finer applications of parallelization are possible, but are often not directly available since at odds with the common degree of generality required.

The accessibility to parallel computing, however, is strictly dependent on the software architecture, with some solutions more intuitive and more immediate than others. JAGS or Stan when used in conjunction with R interfaces (consider rjags or rstan, for simplicity) require the only specification of the arguments n.chains or chains,cores to trigger MCMCs parallelization.[11] Stan also grants additional capillarity by allowing the use of *ad-hoc* functions for summation and "rectangular mapping" (Stan Development Team 2023) inside the .stan setup file. On the other hand, BUGS, despite the possibility of building multiple chains, didn't originally provide a parallel execution; the MultiBUGS project (Goudie et al. 2020) has handled the task introducing the modelDistribute command. As for BayTool "direct competitors" in econometric analysis, MATLAB does not include Bayesian samplers with tunable function arguments for multiple chains, but parallelization can be integrated manually via the Parallel Computing Toolbox (The Mathworks, Inc 2023). STATA proposes multiple-chain computation with a sequential execution: parallelization is in principle excluded even though multithread alternatives in the form of unofficial functions are available. The R packages for case-specific Bayesian estimation do not natively encompass parallel functionalities but, similarly to MATLAB, the user can manually recover multithread solutions by means of the parallel (The R Core Team 2023) or snow (Tierney et al. 2021) packages.

In gretl, parallelization of MCMCs via "embarrassingly parallel" chains has been previously employed in the package ParMA (Lucchetti and Pedini 2022) which deals with Reversible Jump Markov Chain Monte Carlo (Green 1995; Lamnisos et al. 2009) for model averaging in Generalized Linear Models. In particular, ParMA uses the MPI apparatus natively provided by gretl: even though different modes of parallelization can be configured in gretl (OpenMP, for instance), MPI functionalities are directly integrated in scripting via built-in commands and functions, totally transparent to the user; the only requirement is a suitable MPI library such as MPICH or Open MPI for Unix systems or MS-MPI for Windows ones (Cottrell and Lucchetti 2023b, Section 9 - "Platform specifics"). This implies not only the possibility of exploiting all the benefits deriving from such infrastructure (e.g., parallelization through cores or through networked machines) but also an extreme flexibility of usage.[12]

Similarly to ParMA, BayTool relies on MPI functionalities for "embarrassingly parallel" chains: MPI commands are accessed internally via the function package parallel_func (Schreiber 2023) which is automatically loaded during BayTool declaration. Finally, the user has to simply choose the number of simultaneous chains via the bundle key opt.mpi, available in all BayTool sampler; in case of a GUI user, the MPI box is directly selectable in the dialogue window. Remember however that, to fully take advantage of parallelization, diagnostic checking is even more fundamental here than in single-threaded executions: stationarity and convergence have to be properly acknowledged. The aforementioned Brook and Gelman statistic is a natural choice, but nothing prevents the use of graphical analysis or other tools such as Geweke or Hidelberger-Welch tests for stationarity and ESS for correlation.

[11] Some prerequisites are often in place, such as the installation of suitable libraries or packages for enabling multithreading. MPI binaries are an example.

[12] See the dedicated manual Cottrell and Lucchetti (2023b) for major details about MPI usage in gretl.

5 Applications

In this section, I provide some empirical illustrations covering the specifications reported in Sect. 3. To emphasize the contribution that the package could bring to teaching purposes, I will replicate examples and exercises from well-known Bayesian textbooks or published articles.

Moreover, since a scripting approach is here favored over the GUI, the function invocations are reported alongside all preliminary setups: `BayTool` requires only a few necessary declarations to work, with the previously shown prior elicitation as the prominent one. The typical workflow can be summarized as follow:

1. load `BayTool`;
2. open a dataset;
3. define the list of covariates to use;
4. define and build the `prior` bundle: the user, in general, has to specify the `type` to use and, possibly, the prior hyperparameters;
5. (optional) define and build the `opt` bundle for MCMC options;
6. invoke the main function;
7. (optional) post-estimation analysis.

Only basic `gretl` coding is involved and no particular model formalizations are required other than the prior form and the choice for the posterior sampler function.[13] In this sense, the procedure is highly "standardized" regardless of the main functions used.

As for the presented applications, Sect. 5.1 deals with linear models and provides a preliminary benchmark for testing the package quality. Section 5.2 introduces a Bayesian LASSO example by comparing the function `BT_lasso` with a competing software; while Sect. 5.3 considers binary choice models shedding light on the computational scheme: the traditional Albert and Chib algorithm is contrasted with Held and Holmes (2006) method. MCMC diagnostic measures are introduced to highlight the convergence quality of the chains. Finally, Sect. 5.4 inspects the impact of parallelization in MCMCs.

5.1 Linear model example

Consider the Bayesian linear model example with independent priors proposed by Koop (2003), Chapter 4.2.7, on house prices. The dataset comes from Anglin and Gencay (1996) and contains 576 observations on house sales in Windsor, Canada, in 1987: the dependent variable `y` represents the sale price of a given house, while the covariates are the lot size in square feet (`lot_size`), the number of bedrooms (`n_bedroom`), the number of bathrooms (`n_bathroom`) and the number of storeys (`n_storeys`).

The prior for (β, σ^2) follows an independent NIG, with

$$\beta \sim N_5(\beta_0, V_0),$$

[13] For a GUI user, the only essential part is step 2, with steps 3-4-5 as optional and executable via scripts or console.

$$\beta_0 = \begin{bmatrix} 0 \\ 10 \\ 5000 \\ 10000 \\ 10000 \end{bmatrix} \quad V_0 = \begin{bmatrix} 10000^2 & 0 & 0 & 0 & 0 \\ 0 & 5^2 & 0 & 0 & 0 \\ 0 & 0 & 2500^2 & 0 & 0 \\ 0 & 0 & 0 & 5000^2 & 0 \\ 0 & 0 & 0 & 0 & 5000^2 \end{bmatrix}$$

where the first coefficient refers to the intercept while the others refer to the previously presented covariates (in order of appearance). Instead of using directly σ^2, Koop (2003) reparametrizes in terms of precision $1/\sigma^2$, which follows a Gamma distribution with mean 4×10^{-8} and 5 degrees of freedom. The interpretation of prior hyperparameters is the following: consider $\beta_{\texttt{n\_bedroom}}$, the expected effect of an additional bedroom in the house price (given the other variables constant) is an increase of 5000 dollars, i.e., $\beta_{0,\texttt{n\_bedroom}}$. Attaching a prior standard deviation of 2500 implies a marginal effect inside the region [100, 9900] with 95% probability.[14] Such a large variance reflects a rather crude approximation of the regression coefficient impact, hinting at a cautious evaluation of the real estate market. As for σ^2, residuals from a well-fitting model will likely range in $[-10000, 10000]$. Assuming such an interval as the 95% probability region of the Normal error distribution leads to $\sigma \approx 5000$ and consequently $1/\sigma^2 = 4 \times 10^{-8}$. The degrees of freedom are set to a small value in order to have relatively loose information on σ^2.

The Gibbs sampler is run for 10000 iterations with an initial burn-in of 1000. The `gretl` script is reported below

```
? include BayTool.gfn
? set seed 1234567

? open AnglinGencay.gdt --quiet --frompkg=BayTool

? series y = price
? list X = const lot_size n_bedroom
n_bathroom n_storeys
```

where in the first line the package functions are included in the workspace (step 1), while in the second the seed for pseudo-random number generation is set (optional). In the third line, the dataset is retrieved (step 2), and in the subsequent declarations the dependent and the independent variables are collected (step 3). Prior specification (step 4) is handled via the `bundle prior` in the following way

```
? bundle prior = defbundle{}
? prior.type = "indep"
? prior.beta_mean = {0; 10; 5000; 10000; 10000}
? prior.beta_cov = {10000^2; 25; 2500^2; 5000^2; 5000^2} .* I(5)
? prior.precision_mean = 4e-08
```

[14] The computation follows from the Normal distribution: the 95% probability interval (centered on the mean) is around $[-1.96 * sd(\beta) + E(\beta); 1.96 * sd(\beta) + E(\beta)]$.

```
? prior.precision_df = 5
```

while additional options (step 5) are passed through the `bundle opt`

```
? bundle opt = defbundle()
? opt.forecast = {1, 5000, 2, 2, 1}
```

where `opt.forecast` represents a vector containing the covariate values used to build a single prediction. Notice that the prior parameters for σ^2 can be elicited directly in terms of shape and scale of the Inverse Gamma or, alternatively, in terms of precision mean and degrees of freedom as above.

The function invocation (step 6) is

```
? bundle out = BT_lm(y, X, prior)
```

which produces the following output[15]

```
------------------------------BAYESIAN LINEAR MODEL------------------------------

Dependent variable: y
Prior specification: Independent NIG prior
Estimation method: Gibbs Sampler - 10000 iter (1000 burn-in, 0 thin)

Summary statistics:
              Prior_m    Prior_se   NI-Post_m  NI-Post_se    I-Post_m   I-Post_se
     const      0.000   10000.000   -4009.550    3609.788   -4131.248    3269.358
  lot_size     10.000       5.000       5.429       0.370       5.453       0.366
 n_bedroom   5000.000    2500.000    2824.614    1217.059    3224.680    1068.047
n_bathroom  10000.000    5000.000   17105.174    1737.649   16123.061    1624.835
 n_storeys  10000.000    5000.000    7634.897    1009.843    7691.850     974.032

    Sigma2   4.17e+07    5.89e+07    3.35e+08    2.04e+07    3.32e+08    2.01e+07

-----------------------------------------------
Prediction summary statistics
                  Mean       Std. Err
 Forecast# 1      69644.224      18242.876
```

The first two columns report the prior mean and standard deviations of the coefficients, the third and fourth are the non-informative ones (frequentist estimates), while the last columns represent the informative posterior statistics obtained used the `bundle prior` information.

For comparison, Table 3 collects the main posterior summary statistics from the original estimates in Koop (2003)[16] where, for simplicity, those previously obtained with `BayTool` are added: as can be noted, the results are almost identical.

[15] Execution time: approximately 0.60 seconds.

[16] The values here reported refer to Table 4.1, Chapter 4 of Koop (2003). Similar results can be also obtained by running the corresponding replication MATLAB code `chapter4a.m` available at https://www.wiley.com/legacy/wileychi/koopbayesian/.

Table 3 Posterior statistics comparison (mean and standard deviation): Koop (2003) textbook results versus `BayTool`

	Koop (2003)		`BT_lm - BayTool`	
	Posterior mean	Posterior sd	Posterior mean	Posterior sd
`const`	−4063.08	3259.00	−4131.25	3269.36
`lot_size`	5.44	0.37	5.45	0.37
`n_bedroom`	3214.09	1057.67	3224.68	1068.05
`n_bathroom`	16132.78	1617.34	16123.06	1624.84
`n_storeys`	7680.50	979.09	7691.85	974.03
`forecast`	69750	18402	69644	18242

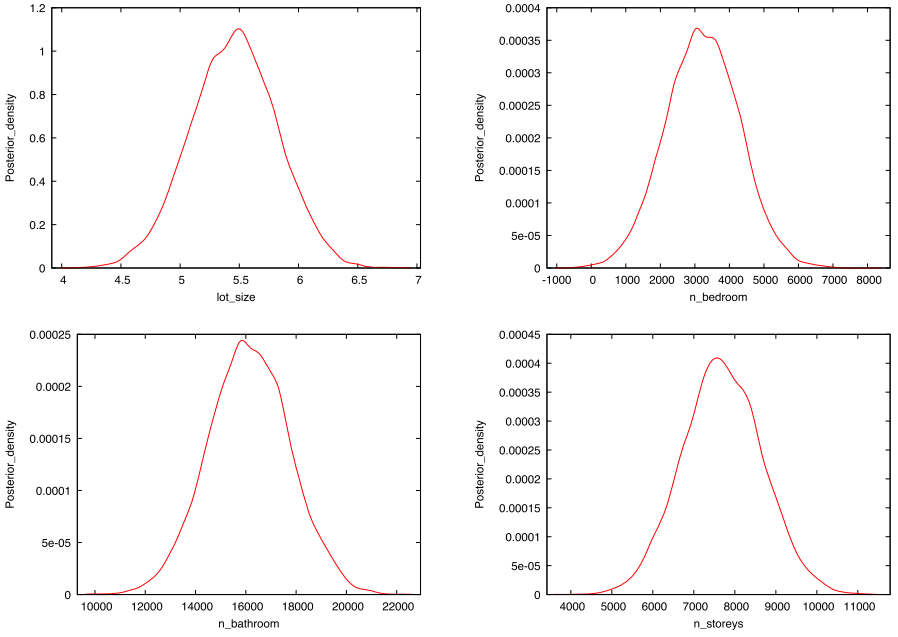

Fig. 2 Marginal posteriors for main covariate coefficients

To further illustrate the package functionality, the marginal posterior distributions can be produced (step 7) using the dedicated function `BT_paramplot`,

```
? strings names = varnames(X)
? BT_paramplot(out, names)
```

which returns the graphs reported in Fig. 2.

At this point, it is also possible to access additional information such as posterior predictions and to produce the related histogram in a few scripting commands,

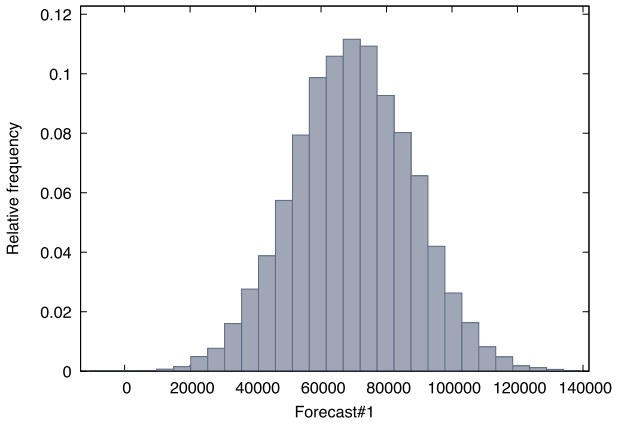

Fig. 3 Posterior predictive distribution

```
? matrix yhat1 = out.post_pred.sampled_pred'
? freq --matrix=yhat1 --plot=display
```

with the corresponding result plotted in Fig. 3.

5.2 Bayesian LASSO example

In the following, a replica of the diabetes example as in Park and Casella (2008) is proposed: `BayTool` is here compared to the `R` function `blasso` from the package `monomvn`.

The dataset is due to Efron et al. (2004) and collects information on diabete disease progression (`y`) in 443 patients. The regressor list contains: age (`age`), sex (`sex`), body mass index (`bmi`), average blood pressure (`map`) and six blood serum measurements (`tc`, `ldl`, `hdl`, `tch`, `ltg`, `glu`). The posterior derivation is based on 10000 iterations after a 10% burn-in and a thinning window of 10, with a Gamma hyperprior on λ^2 with shape and rate respectively equal to 1 and 1.78.

The `gretl` script is the following:

```
? include BayTool.gfn
? set seed 1234567

? open diabetes.gdt --quiet --frompkg=BayTool

? list X = age sex bmi map tc ldl hdl tch ltg glu

? bundle prior = defbundle()
? prior.type = "hyper"
? prior.lambda_shape = 1
? prior.lambda_rate = 1.78

? bundle out = BT_lasso(y, X, prior, defbundle("stdize", 0,\
```

Table 4 Posterior statistics comparison (main quantiles): `R` function `blasso` from `monomvn` versus `BayTool - BT_lasso`

	`blasso - R`			`BT_lasso - BayTool`		
	I Quartile	Median	III Quartile	I Quartile	Median	III Quartile
`age`	−37.89	−2.38	32.07	−36.84	−2.16	33.41
`sex`	−251.29	−209.77	−167.54	−247.59	−204.79	−163.43
`bmi`	477.30	524.20	568.10	474.18	521.08	567.88
`map`	260.59	303.43	348.77	256.62	305.40	354.33
`tc`	−270.23	−151.83	−50.14	−274.84	−153.03	−46.51
`ldl`	−91.16	−12.43	73.07	−90.25	−9.75	74.20
`hdl`	−236.96	−156.66	−77.64	−226.75	−156.66	−79.26
`tch`	14.31	88.30	170.68	16.40	84.37	169.49
`ltg`	449.10	513.10	580.00	457.66	513.21	583.00
`glu`	20.02	60.51	103.79	22.32	63.65	108.73

The CPU time is nearly the same: ≈ 1 second for `blasso` and ≈ 1.5 for `BT_lasso`

```
"thin", 10))
```

Table 4 reports the main quantiles of the parameters coefficients computed via the `R` function `blasso` and `BT_lasso`: the results are extremely similar, therefore the example is successfully replicated.[17]

Notice that not all quantiles are, by default, supplied as summary statistics in the returning bundle of `BT_lasso` (`out.post_summ`), however obtaining these quantities is trivial from the sampled β in `out.sampled_coeff.sampled_beta`, as suggested in the following lines

```
? matrix coeff = out.sampled_coeff.sampled_beta
? matrix q1 = quantile(coeff', 0.25)'
? matrix q2 = quantile(coeff', 0.75)'
? matrix med = out.post_summ.post_betamedian

? matrix summ = q1 ~ med ~ q2
? rnameset(summ, varnames(X))
? cnameset(summ, defarray("q025", "q050", "q075"))
? print summ
```

[17] Some notes about the replication exercise: the covariates are already rescaled to have unit-norm and mean zero, as a consequence no normalization is required; `BT_lasso` automatically "demeans" the dependent variable, while `blasso` provides an estimate for the intercept term; the thinning interval used follows the automatic choice proposed by `blasso`. Additionally, recall that Bayesian LASSO does not shrink parameter to zero as the frequentist counterpart.

5.3 Bayesian Probit example

Binary choice models are common in microeconometrics and this experiment shows how a Bayesian estimation can be performed using two different computational schemes. The example comes from Chan et al. (2019), Chapter 14, Exercise 14.3, and re-examines the application by Fair (1978) on extramarital affairs: the dataset contains 601 responses from the Psychology Today Survey where the dependent variable is the decision to have an extramarital affair (`y`) and the covariates are a male dummy (`male`), the number of years married (`ys_married`), a dummy for children (`kids`), a dummy for classifying whether the interviewed is religious, years of education (`ed`) and a dummy for denoting a happy marriage (`happy`).

In the first part of the script the common data augmentation scheme *à la* Albert and Chib for Bayesian probit is performed using a Gibbs sampler with 2000 iterations, after 500 replicas of burn-in. The prior for covariate coefficients is specified as

$$\beta \sim N_7(0, 100 \cdot I_7)$$

where an intercept term is added to the variable list.

To further inspect the quality of the result (which is notoriously affected by high autocorrelation) some diagnostics are included such as the ESS (both univariate and multivariate) and a graphical analysis of some selected coefficients. Remember that these are available in the matrix of sampled coefficients (`out1.sampled_coeff.sampled_beta`). The script is here reported

```
? include BayTool.gfn
? set seed 5783236

? open Fair_Koop.gdt --quiet --frompkg=BayTool

? list X = const male ys_married kids religious ed happy

? bundle prior = defbundle()
? prior.beta_mean = zeros(7,1)
? prior.beta_vcv =  100 * I(7)

? bundle opt = defbundle()
? opt.iter = 2000
? opt.burn = 500

? out1 = BT_probit(affair, X, prior, opt)

? diagn1 =  BT_check_mESS(out1.sampled_coeff.sampled_beta)
? BT_check_plot(out1.sampled_coeff.sampled_beta)
```

Table 5 Posterior statistics comparison (mean and standard deviation): Chan et al. (2019) textbook results versus `BayTool`

	Chan et al. (2019)		`BT_probit - std`		`BT_probit - HH`	
	Mean	Sd	Mean	Sd	Mean	Sd
`const`	−0.726	0.417	−0.734	0.419	−0.737	0.422
`male`	0.154	0.131	0.153	0.127	0.145	0.129
`ys_married`	0.029	0.013	0.029	0.013	0.029	0.013
`kids`	0.256	0.159	0.245	0.162	0.244	0.159
`religious`	−0.514	0.124	−0.513	0.122	−0.512	0.122
`ed`	0.005	0.026	0.006	0.026	0.006	0.027
`happy`	−0.514	0.125	−0.519	0.127	−0.516	0.125

Table 6 Effective Sample Size from `BT_probit`, standard algorithm

	ESS
`const`	618
`male`	537
`ys_married`	657
`kids`	722
`religious`	489
`ed`	824
`happy`	686
Multivariate ESS	731

Table 5 summarizes the posterior means and standard deviations of the sampled coefficients produced by the function `BT_probit` (columns 3–4) alongside the estimates reported in Chan et al. (2019) (columns 1–2).[18]

Both posterior means and posterior standard deviations are replicated.[19] However, as highlighted in Table 6, a high degree of correlation can be spotted in the coefficients: the ESS is considerably smaller than the total amount of iterations. Further confirmation of this fact can be acknowledged via Fig. 4 where the traceplots and the autocorrelation functions for the parameters of `ys_married` and `happy` are shown.

To counter this fact, in the second part of the script the Held and Holmes (2006) algorithm is implemented. The `gretl` script goes as follow:

```
? opt.method = "HH"
? out2 = BT_probit(affair, X, prior, opt)

? diagn2 =  BT_check_mESS(out2.sampled_coeff.sampled_beta)
? BT_check_plot(out2.sampled_coeff.sampled_beta)
```

[18] The authors provide a `MATLAB` replication script at https://web.ics.purdue.edu/~jltobias/second_edition/Chapter14/chapt14.html.

[19] `BT_probit` running time: ≈ 1 second.

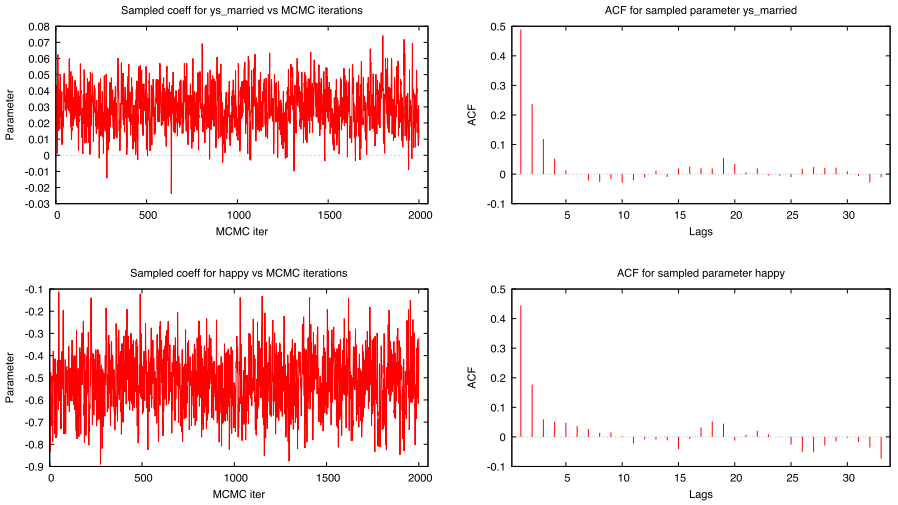

Fig. 4 Traceplot and ACF function for the sampled parameters of `ys_married` and `happy` - standard Gibbs sampler (*à la* Albert and Chib)

Table 7 Effective Sample Size from `BT_probit`, Held and Holmes (2006) algorithm

	ESS
`const`	1615
`male`	1510
`ys_married`	1226
`kids`	1024
`religious`	1506
`ed`	1744
`happy`	1778
Multivariate ESS	1683

As can be noted in Table 5 from columns 5-6, the posterior quantities are also replicated in this context; however, Table 7 and Fig. 5 reveal a remarkably different quality of the sample: the autocorrelation is now hugely decreased with benefits in the mixing of the chain.

5.4 Parallelization in action

In order to determine the computational advantage due to parallelization, I will replicate the previous examples on the linear model, LASSO and the probit model[20] with 100000 iterations after a 10% burn-in, using up to 20 parallel chains (`opt.mpi`). Following the results reported in Cottrell and Lucchetti (2023b) and in Lucchetti and Pedini (2022), hyper-threading, that is the use of logical cores, has been avoided due to the proven CPU time deterioration.

[20] For the sake of simplicity and brevity only the standard algorithm has been analyzed.

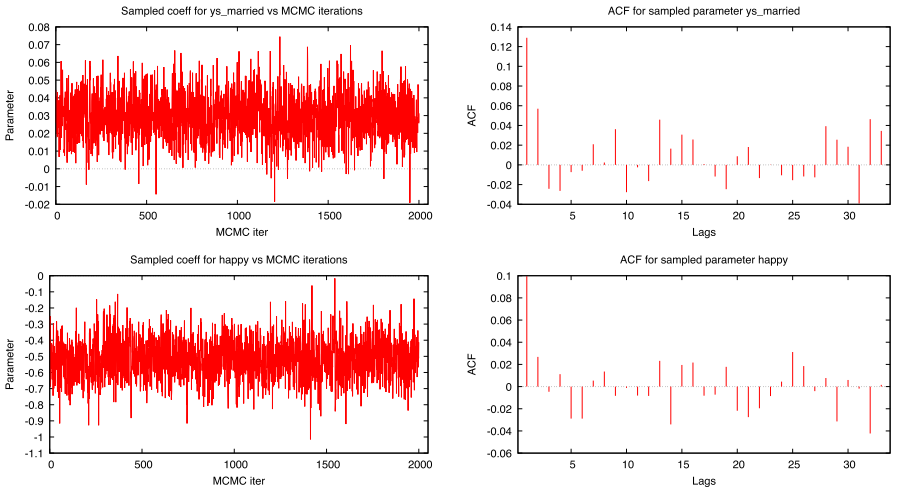

Fig. 5 Traceplot and ACF function for the sampled parameters of `ys_married` and `happy` - Held and Holmes (2006) algorithm

Table 8 CPU time performance (in seconds) of the three considered `BayTool` functions across different numbers of cores (MPI) employed

Cores (`opt.mpi`)	CPU time (seconds)		
	`BT_lm`	`BT_lasso`	`BT_probit`
1	34.864	136.57	235.97
2	13.805	54.530	113.10
3	8.7025	33.535	68.858
4	6.9334	25.463	49.697
5	6.0533	21.555	39.964
6	5.5346	19.154	33.939
7	**5.2621**	17.936	30.421
8	5.2959	16.979	27.880
9	5.2928	16.533	26.341
10	5.3889	15.906	25.039
11	5.4258	15.829	24.663
12	5.4743	15.406	23.809
13	5.5069	15.531	23.665
14	5.6491	15.446	23.059
15	5.7131	15.499	23.160
16	5.7945	**15.207**	22.544
17	5.8437	15.640	22.740
18	6.0087	15.365	22.459
19	6.0875	15.890	22.622
20	6.3431	15.748	**22.383**

Bold is used for indicating the best performance

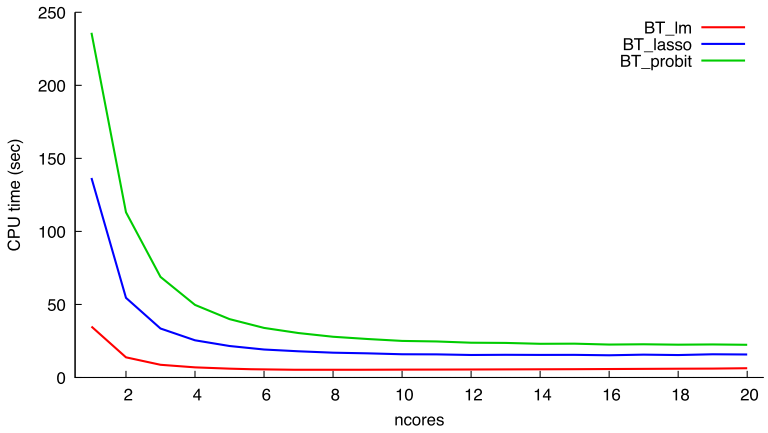

Fig. 6 CPU elapsed time (seconds) in the three scenarios considered

Table 8 collects the results[21]: in all scenarios considered parallelization leads to massive CPU time reductions. Simply switching from a single-threaded MCMC to two parallel chains offers a 50–60% time saving which reaches its peak of 85–90% in the case of 7 cores for the linear model, 16 for the LASSO and 20 for the probit. Surprisingly, exploiting all available cores seems to induce the most valuable performance only in the probit example. Before attempting any explanation, it should be noticed that the samplers show merely identical results starting from given thresholds: 5–6 chains in the linear model, 10 in the LASSO case and 16 for the probit. From here on, the variation is mostly ascribable to computational noise.

As Fig. 6 clarifies, the CPU performance follows a hyperbolic function mostly declining in the first part of the x-axis, but then stabilizing to a steady path in the second one. The linear case, for example, can reach meaningful results with the smallest amount of parallel chains, showing an almost constant trend after 6–7 simultaneous chains. Once reached this point, the cost of invoking additional cores is not totally offset by the potential speed-up of having an extra chain. Similar conclusions apply to the other two functions. A plausible determinant of this harsh reduction of marginal benefits has to be searched in the extremely quick convergence of the MCMCs, which require few iterations to reach valuable results.

In conclusion, the impact of parallelization is far from being negligible, with an impressive time reduction in all scenarios considered. What is less intuitive, is finding the ideal number of parallel chains to use. The solution is not straightforward, because *i)* exploiting all available power is not always the *first-best* choice, as previously shown; moreover *ii)* the performance is inherently case-to-case dependent. Practical guidance is often found in running some preliminary smaller MCMCs to detect the CPU scalability of the simulation.

[21] The Brook and Gelman convergence test has been used to ascertain convergence to the stationary state: in all cases the statistics ≈ 1.

6 Conclusion

The BayTool package collects Bayesian estimation methods of commonly used econometric models. Computational efficiency is reached by combining a case-specific sampling approach, which ensures Gibbs sampling applicability, with the parallelization of MCMCs via MPI. As shown via the replication exercises, the package can reach remarkable results both in terms of accuracy and execution time. Moreover, the simplicity in the usage, fostered by a graphical interface, makes BayTool an extremely user-friendly solution for Bayesian econometrics, which, on the one hand, encourages the consolidated user base to experiment with the now-integrated Bayesian routines, previously absent, and on the other hand, looks at a new audience which may find the package profitable for learning the basics of Bayesian methods. Last but not least, the package is intended as a long-term project with recurrent updates that will enrich its functionalities: following the case-specific philosophy, BayTool will provide various specialized routines for micro- and macroeconometric models; at the same time, this clearly will not exclude the possibility of a more general applicability with more prior choices per function. The current state of development includes the forthcoming release of ordinal and multinomial probit, logit and count models.

Acknowledgements I wish to thank Riccardo Lucchetti, all the participants at the 2023 gretl Conference and two anonymous referees for their useful remarks and observations.

Funding Open access funding provided by Università Politecnica delle Marche within the CRUI-CARE Agreement.

Data availability The datasets can be retrieved directly from BayTool. All experiments have been run and tested on gretl 2023a-2023b-2023c. Examples on Sect. 5.1, 5.2 and 5.3 have been executed on a macOS machine (Intel); while the experiment reported on Sect. 5.4 has been computed on a Linux server.

Code Availability Scripts are provided as attached files. Alternatively, most of these can be retrieved from BayTool sample scripts.

Declarations

Conflict of interest The author has no relevant financial or non-financial interest to disclosure.

References

Albert JH, Chib S (1993) Bayesian analysis of binary and polychotomous response data. J Am Stat Assoc 88(422):669–679

Anglin PM, Gencay R (1996) Semiparametric estimation of a hedonic price function. J Appl Economet 11(6):633–648
Bernardo JM, Smith AF (1994) Bayesian Theory, vol 405. Wiley, Chichester
Betancourt M (2017) A conceptual introduction to Hamiltonian Monte Carlo. arXiv:1701.02434
Błażejowski M, Kwiatkowski J (2015) Bayesian model averaging and jointness measures for gretl. J Stat Softw 68:1–24
Błazejowski M, Kwiatkowski J (2018) BACE: A gretl Package for Model Averaging in limited dependent variable models. gretl working papers 6
Brooks SP, Gelman A (1998) General methods for monitoring convergence of iterative simulations. J Comput Graph Stat 7(4):434–455
Bürkner P-C, Gabry J, Weber S, Johnson A, Modrak M, Badr HS, Wild S (2023) `brms`: Bayesian Regression Models using 'Stan'. https://cran.r-project.org/package=brmsR package version 2.20.1
Carpenter B, Gelman A, Hoffman MD, Lee D, Goodrich B, Betancourt M, Brubaker MA, Guo J, Li P, Riddell A (2017) Stan: a probabilistic programming language. J Stat Softw 76:1
Chan J, Koop G, Poirier DJ, Tobias JL (2019) Bayesian econometric methods, vol 7. Cambridge University Press, Cambridge
Cottrell A, Lucchetti R (2023a) Gretl function package guide [Computer software manual]. https://sourceforge.net/projects/gretl/files/manual/pkgbook.pdf/download
Cottrell A, Lucchetti R (2023b) Gretl + MPI [Computer software manual]. https://sourceforge.net/projects/gretl/files/manual/gretl-mpi.pdf/download
Cottrell A, Lucchetti R (2023c) Gretl User's Guide [Computer software manual]. https://sourceforge.net/projects/gretl/files/manual/gretl-guide.pdf/download
Denwood M, Plummer M (2023) `runjags`: interface utilities, model templates, parallel computing methods and additional distributions for MCMC models in JAGS. https://CRAN.R-project.org/package=runjagsR package version 2.2.2-1.1
de Valpine P, Paciorek C, Turek D, Michaud N, Anderson-Bergman C, Obermeyer F, Sujan P (2023) `nimble`: Mcmc, particle filtering, and programmable hierarchical modeling. https://CRAN.R-project.org/package=nimblecR package version 1.0.1
de Valpine P, Turek D, Paciorek CJ, Anderson-Bergman C, Lang DT, Bodik R (2017) Programming with models: writing statistical algorithms for general model structures with NIMBLE. J Comput Graph Stat 26(2):403–413
Duane S, Kennedy AD, Pendleton BJ, Roweth D (1987) Hybrid Monte Carlo. Phys Lett B 195(2):216–222
Efron B, Hastie T, Johnstone I, Tibshirani R (2004) Least angle regression. Ann Stat 32(2):407–499
Fair RC (1978) A theory of extramarital affairs. J Politi Econ 86(1):45–61
Gelfand AE, Smith AF (1990) Sampling-based approaches to calculating marginal densities. J Am Stat Assoc 85(410):398–409
Gelman A, Carlin JB, Stern HS, Rubin DB (1995) Bayesian data analysis. Chapman and Hall/CRC, Boca Raton
Gelman A, Rubin DB (1992) Inference from iterative simulation using multiple sequences. Stat Sci 7:457–472
Gelman A, Su Y-S, Yajima M, Hill J, Pittau MG, Kerman J, ... Dorie V (2022) `arm`: Data Analysis Using Regression and Multilevel/Hierarchical Models. https://CRAN.R-project.org/package=armR package version 1.13-1
Geweke J (1992) Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In: Bernardo JM, Berger JO, Dawid AP, Smith AFM (eds) Bayesian statistics, vol 4. Clarendon Press, Oxford, pp 169–193
Geweke J (1999) Using simulation methods for Bayesian econometric models: inference, development, and communication. Econ Rev 18(1):1–73
Geweke J (2005) Contemporary Bayesian econometrics and statistics. Wiley, London
Geyer CJ, Johnson LT (2020) `mcmc`: Markov Chain Monte Carlo. https://CRAN.R-project.org/package=mcmcR package version 0.9-7
Girolami M, Calderhead B (2011) Riemann manifold Langevin and Hamiltonian Monte Carlo methods Riemann manifold Langevin and Hamiltonian Monte Carlo methods. J R Stat Soc Ser B Stat Methodol 73(2):123–214
Golding N, Tierney N, Dirmeier S, Fleischhacker A, Glander S, Ingram M, Yen J (2022) `greta`: simple and scalable statistical modelling in R. https://CRAN.R-project.org/package=gretaR package version 0.4.3

Goudie RJ, Turner RM, De Angelis D, Thomas A (2020) MultiBUGS: a parallel implementation of the BUGS modelling framework for faster Bayesian inference. J Stat Softw 95:7
Gramacy RB, Moler C, Turlach BA (2023) monomvn: Estimation for MVN and student-t data with Monotone Missingness. https://CRAN.R-project.org/package=monomvnR package version 1.9-18
Green PJ (1995) Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika 82(4):711–732
Greenberg E (2012) Introduction to Bayesian Econometrics. Cambridge University Press, Cambridge
Guo J, Gabry J, Goodrich B, Johnson A, Weber SW, Lee D, ... Kormanyos B (2023) Christophe rand Steve. rstan: R Interface to Stan. https://CRAN.R-project.org/package=rstanR package version 2.26.23
Haario H, Saksman E, Tamminen J (2001) An adaptive Metropolis algorithm. Bernoulli 7:223–242
Hall B, Hall M, Statisticat LLC, Brown E, Hermanson R, Charpentier E, Singmann H (2021) LaplacesDemon: Complete environment for bayesian inference. https://CRAN.R-project.org/package=LaplacesDemonR package version 16.1.6
Hartig F, Minunno F, Paul S, Cameron D, Ott TO, Pichelr M (2023) Bayesiantools: General-purpose mcmc and smc samplers and tools for bayesian statistics. https://CRAN.R-project.org/package=BayesianToolsR package version 0.1.8
Hastings W (1970) Monte Carlo sampling methods using Markov chains and their applications. Biometrika pp 97–109
Heidelberger P, Welch PD (1983) Simulation run length control in the presence of an initial transient. Op Res 31(6):1109–1144
Held L, Holmes CC (2006) Bayesian auxiliary variable models for binary and multinomial regression. Bayesian Anal 11:145–168
Hoffman MD, Gelman A et al (2014) The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. J Mach Learn Res 15(1):1593–1623
Kass RE, Carlin BP, Gelman A, Neal RM (1998) Markov chain Monte Carlo in practice: a roundtable discussion. Am Stat 52(2):93–100
Koop G (2003) Bayesian econometrics. Wiley, London
Lamnisos D, Griffin JE, Steel MF (2009) Transdimensional sampling algorithms for Bayesian variable selection in classification problems with many more variables than observations. J Comput Graph Stat 18(3):592–612
Lancaster T (2004) An introduction to modern Bayesian econometrics. Blackwell Oxford, Oxford
Lesage JP (1999) Applied Econometrics using MATLAB. https://www.spatial-econometrics.com/
Lucchetti RJ, Pedini L (2022) ParMA: parallelized Bayesian model averaging for generalized linear models. J Stat Softw 104:1–39
Lunn D, Jackson C, Best N, Thomas A, Spiegelhalter D (2012) The BUGS book: a practical introduction to Bayesian analysis. CRC Press, Boca Raton
Martin AD, Quinn KM, Park JH, Vieilledent G, Malecki M, Blackwell M, ... Nishimura T (2022) MCMCpack: Markov chain monte carlo (mcmc) package. https://CRAN.R-project.org/package=MCMCpackR package version 1.6-3
Neal RM (1994) An improved acceptance procedure for the hybrid Monte Carlo algorithm. J Comput Phys 111(1):194–203
Neal RM (2003) Slice sampling. Ann Stat 31(3):705–767
Park T, Casella G (2008) The Bayesian Lasso. J Ame Stat Assoc 103(482):681–686
Pedini L (2023) BayTool: Bayesian regression models. https://gretl.sourceforge.net/current_fnfiles/BayTool.zip package version 0.7
Pedini L, Schreiber S (2023) BVAR: Bayesian VARs in gretl. https://gretl.sourceforge.net/current_fnfiles/BVAR.zip package version 0.1
Plummer M (2017) JAGS Version 4.3.0 user manual [Computer software manual]. Retrieved from https://sourceforge.net/projects/mcmc-jags/files/Manuals/4.x/jags_user_manual.pdf/download
Plummer M, Stukalov A, Denwood M (2023) rjags: Bayesian graphical models using mcmc. https://CRAN.R-project.org/package=rjagsR package version 4-14
Poirier DJ (1995) Intermediate statistics and econometrics: a comparative approach. Mit Press, Cambridge
Robert CP, Casella G, Casella G (1999) Monte Carlo Statistical Methods, vol 2. Springer, Berlin
Roberts GO, Rosenthal JS (2007) Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. J Appl Prob 44(2):458–475
Roberts GO, Rosenthal JS (2009) Examples of adaptive MCMC. J Comput Graph Stat 18(2):349–367

Rossi P (2022) `bayesm`: Bayesian inference for marketing/micro-econometrics. https://CRAN.R-project.org/package=bayesm`R` package version 3.1-5
Schreiber S (2023) `parallel_func`: parallelized execution of a (matrices-returning) function. https://gretl.sourceforge.net/current_fnfiles/parallel_func.gfn package version 0.4
Spiegelhalter D, Thomas A, Best N, Lunn D (2003) WinBUGS user manual. Citeseer
Spiegelhalter D, Thomas A, Best N, Lunn D (2007) OpenBUGS user manual (Vol 3) (no 2)
Stan Development Team (2023) Stan user's guide [Computer software manual]. https://mc-stan.org/docs/2_32/stan-users-guide-2_32.pdf version 2.32
Su Y-S, Yajima M (2021) `R2jags`: Using r to run 'jags'. https://CRAN.R-project.org/package=R2jags`R` package version 0.7-1
The Mathworks, Inc. (2023) Parallel computing toolbox
The R Core Team (2023) Package `parallel`. `R` package version 4.3.2
Tibshirani R (1996) Regression shrinkage and selection via the Lasso. J R Stat Soc Ser B (Methodological) 58(1):267–288
Tierney L, Rossini AJ, Sevcikova HNL (2021) `snow`: simple network of workstations . https://CRAN.R-project.org/package=snow`R` package version 0.4-4
Vats D, Flegal JM, Jones GL (2019) Multivariate output analysis for Markov Chain Monte Carlo. Biometrika 106(2):321–337
Zellner A (1971) An introduction to Bayesian inference in econometrics. Wiley, New York