STATA BAYESIAN ANALYSIS REFERENCE MANUAL
RELEASE 14

A Stata Press Publication
StataCorp LP
College Station, Texas

Version 14

Published by Stata Press, 4905 Lakeway Drive, College Station, Texas 77845
Typeset in TEX

ISBN-10: 1-59718-149-8
ISBN-13: 978-1-59718-149-5

The suggested citation for this software is

StataCorp. 2015. *Stata: Release 14*. Statistical Software. College Station, TX: StataCorp LP.

Contents

Cross-referencing the documentation

When reading this manual, you will find references to other Stata manuals. For example,

[U] 26 Overview of Stata estimation commands
[R] regress
[D] reshape

The first example is a reference to chapter 26, *Overview of Stata estimation commands*, in the *User's Guide*; the second is a reference to the `regress` entry in the *Base Reference Manual*; and the third is a reference to the `reshape` entry in the *Data Management Reference Manual*.

All the manuals in the Stata Documentation have a shorthand notation:

[GSM]	*Getting Started with Stata for Mac*
[GSU]	*Getting Started with Stata for Unix*
[GSW]	*Getting Started with Stata for Windows*
[U]	*Stata User's Guide*
[R]	*Stata Base Reference Manual*
[BAYES]	*Stata Bayesian Analysis Reference Manual*
[D]	*Stata Data Management Reference Manual*
[FN]	*Stata Functions Reference Manual*
[G]	*Stata Graphics Reference Manual*
[IRT]	*Stata Item Response Theory Reference Manual*
[XT]	*Stata Longitudinal-Data/Panel-Data Reference Manual*
[ME]	*Stata Multilevel Mixed-Effects Reference Manual*
[MI]	*Stata Multiple-Imputation Reference Manual*
[MV]	*Stata Multivariate Statistics Reference Manual*
[PSS]	*Stata Power and Sample-Size Reference Manual*
[P]	*Stata Programming Reference Manual*
[SEM]	*Stata Structural Equation Modeling Reference Manual*
[SVY]	*Stata Survey Data Reference Manual*
[ST]	*Stata Survival Analysis Reference Manual*
[TS]	*Stata Time-Series Reference Manual*
[TE]	*Stata Treatment-Effects Reference Manual: Potential Outcomes/Counterfactual Outcomes*
[I]	*Stata Glossary and Index*
[M]	*Mata Reference Manual*

Title

intro — Introduction to Bayesian analysis

Description Remarks and examples References Also see

Description

This entry provides a software-free introduction to Bayesian analysis. See [BAYES] **bayes** for an overview of the software for performing Bayesian analysis and for an overview example.

Remarks and examples

Remarks are presented under the following headings:

The first five sections provide a general introduction to Bayesian analysis. The remaining sections provide a more technical discussion of the concepts of Bayesian analysis.

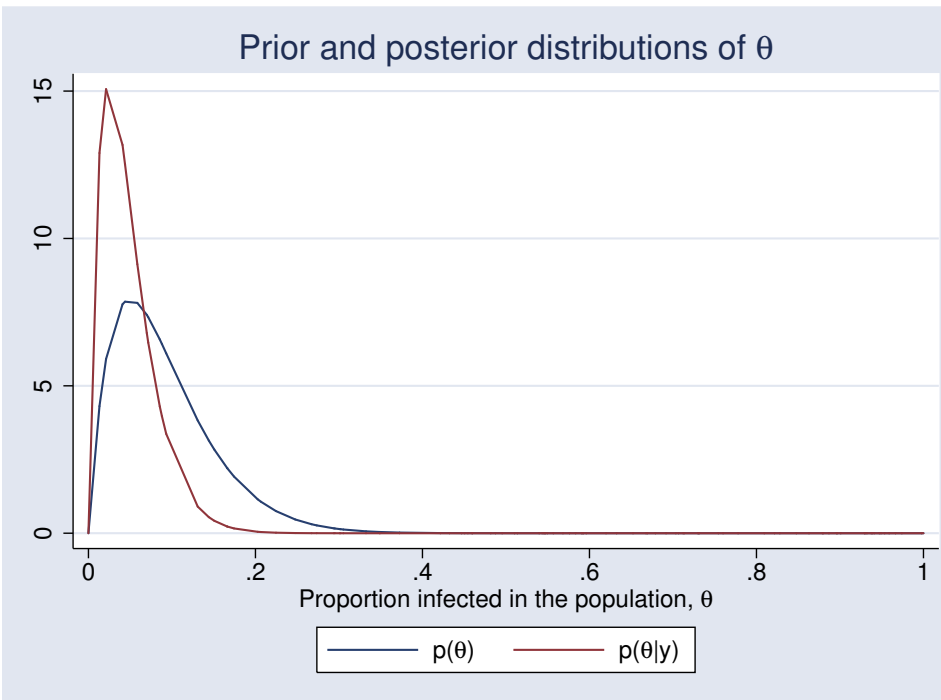
What is Bayesian analysis?

Bayesian analysis is a statistical analysis that answers research questions about unknown parameters of statistical models by using probability statements. Bayesian analysis rests on the assumption that all model parameters are random quantities and thus can incorporate prior knowledge. This assumption is in sharp contrast with the more traditional, also called frequentist, statistical inference where all parameters are considered unknown but fixed quantities. Bayesian analysis follows a simple rule of probability, the Bayes rule, which provides a formalism for combining prior information with evidence from the data at hand. The Bayes rule is used to form the so called posterior distribution of model parameters. The posterior distribution results from updating the prior knowledge about model parameters with evidence from the observed data. Bayesian analysis uses the posterior distribution to form various summaries for the model parameters including point estimates such as posterior means, medians, percentiles, and interval estimates such as credible intervals. Moreover, all statistical tests about model parameters can be expressed as probability statements based on the estimated posterior distribution.

As a quick introduction to Bayesian analysis, we use an example, described in Hoff (2009, 3), of estimating the prevalence of a rare infectious disease in a small city. A small random sample of 20 subjects from the city will be checked for infection. The parameter of interest $\theta \in [0,1]$ is the fraction of infected individuals in the city. Outcome y records the number of infected individuals in the sample. A reasonable sampling model for y is a binomial model: $y|\theta \sim \text{Binomial}(20,\theta)$. Based on the studies from other comparable cities, the infection rate ranged between 0.05 and 0.20, with an average prevalence of 0.10. To use this information, we must conduct Bayesian analysis. This information can be incorporated into a Bayesian model with a prior distribution for θ, which assigns a large probability between 0.05 and 0.20, with the expected value of θ close to 0.10. One potential prior that satisfies this condition is a $\text{Beta}(2,20)$ prior with the expected value of $2/(2+20) = 0.09$. So, let's assume this prior for the infection rate θ, that is, $\theta \sim \text{Beta}(2,20)$. We sample individuals and observe none who have an infection, that is, $y = 0$. This value is not that uncommon for a small sample and a rare disease. For example, for a true rate $\theta = 0.05$, the probability of observing 0 infections in a sample of 20 individuals is about 36% according to the binomial distribution. So, our Bayesian model can be defined as follows:

$$y|\theta \sim \text{Binomial}(20,\theta)$$
$$\theta \sim \text{Beta}(2,20)$$

For this Bayesian model, we can actually compute the posterior distribution of $\theta|y$, which is $\theta|y \sim \text{Beta}(2+0, 20+20-0) = \text{Beta}(2,40)$. The prior and posterior distributions of θ are depicted below.



The posterior density (shown in red) is more peaked and shifted to the left compared with the prior distribution (shown in blue). The posterior distribution combined the prior information about θ with

the information from the data, from which $y = 0$ provided evidence for a low value of θ and shifted the prior density to the left to form the posterior density. Based on this posterior distribution, the posterior mean estimate of θ is $2/(2 + 40) = 0.048$ and the posterior probability that, for example, $\theta < 0.10$ is about 93%.

If we compute a standard frequentist estimate of a population proportion θ as a fraction of the infected subjects in the sample, $\overline{y} = y/n$, we will obtain 0 with the corresponding 95% confidence interval $(\overline{y} - 1.96\sqrt{\overline{y}(1-\overline{y})/n}, \overline{y} + 1.96\sqrt{\overline{y}(1-\overline{y})/n})$ reducing to 0 as well. It may be difficult to convince a health policy maker that the prevalence of the disease in that city is indeed 0, given the small sample size and the prior information available from comparable cities about a nonzero prevalence of this disease.

We used a beta prior distribution in this example, but we could have chosen another prior distribution that supports our prior knowledge. For the final analysis, it is important to consider a range of different prior distributions and investigate the sensitivity of the results to the chosen priors.

For more details about this example, see Hoff (2009). Also see *Beta-binomial model* in [BAYES] **bayesmh** for how to fit this model using `bayesmh`.

Bayesian versus frequentist analysis, or why Bayesian analysis?

Why use Bayesian analysis? Perhaps a better question is when to use Bayesian analysis and when to use frequentist analysis. The answer to this question mainly lies in your research problem. You should choose an analysis that answers your specific research questions. For example, if you are interested in estimating the probability that the parameter of interest belongs to some prespecified interval, you will need the Bayesian framework, because this probability cannot be estimated within the frequentist framework. If you are interested in a repeated-sampling inference about your parameter, the frequentist framework provides that.

Bayesian and frequentist approaches have very different philosophies about what is considered fixed and, therefore, have very different interpretations of the results. The Bayesian approach assumes that the observed data sample is fixed and that model parameters are random. The posterior distribution of parameters is estimated based on the observed data and the prior distribution of parameters and is used for inference. The frequentist approach assumes that the observed data are a repeatable random sample and that parameters are unknown but fixed and constant across the repeated samples. The inference is based on the sampling distribution of the data or of the data characteristics (statistics). In other words, Bayesian analysis answers questions based on the distribution of parameters conditional on the observed sample, whereas frequentist analysis answers questions based on the distribution of statistics obtained from repeated hypothetical samples, which would be generated by the same process that produced the observed sample given that parameters are unknown but fixed. Frequentist analysis consequently requires that the process that generated the observed data is repeatable. This assumption may not always be feasible. For example, in meta-analysis, where the observed sample represents the collected studies of interest, one may argue that the collection of studies is a one-time experiment.

Frequentist analysis is entirely data-driven and strongly depends on whether or not the data assumptions required by the model are met. On the other hand, Bayesian analysis provides a more robust estimation approach by using not only the data at hand but also some existing information or knowledge about model parameters.

In frequentist statistics, estimators are used to approximate the true values of the unknown parameters, whereas Bayesian statistics provides an entire distribution of the parameters. In our example of a prevalence of an infectious disease from *What is Bayesian analysis?*, frequentist analysis produced one point estimate for the prevalence, whereas Bayesian analysis estimated the entire posterior distribution of the prevalence based on a given sample.

Frequentist inference is based on the sampling distributions of estimators of parameters and provides parameter point estimates and their standard errors as well as confidence intervals. The exact sampling distributions are rarely known and are often approximated by a large-sample normal distribution. Bayesian inference is based on the posterior distribution of the parameters and provides summaries of this distribution including posterior means and their MCMC standard errors (MCSE) as well as credible intervals. Although exact posterior distributions are known only in a number of cases, general posterior distributions can be estimated via, for example, Markov chain Monte Carlo (MCMC) sampling without any large-sample approximation.

Frequentist confidence intervals do not have straightforward probabilistic interpretations as do Bayesian credible intervals. For example, the interpretation of a 95% confidence interval is that if we repeat the same experiment many times and compute confidence intervals for each experiment, then 95% of those intervals will contain the true value of the parameter. For any given confidence interval, the probability that the true value is in that interval is either zero or one, and we do not know which. We may only infer that any given confidence interval provides a plausible range for the true value of the parameter. A 95% Bayesian credible interval, on the other hand, provides a range for a parameter such that the probability that the parameter lies in that range is 95%.

Frequentist hypothesis testing is based on a deterministic decision using a prespecified significance level of whether to accept or reject the null hypothesis based on the observed data, assuming that the null hypothesis is actually true. The decision is based on a p-value computed from the observed data. The interpretation of the p-value is that if we repeat the same experiment and use the same testing procedure many times, then given our null hypothesis is true, we will observe the result (test statistic) as extreme or more extreme than the one observed in the sample $(100 \times p\text{-value})\%$ of the times. The p-value cannot be interpreted as a probability of the null hypothesis, which is a common misinterpretation. In fact, it answers the question of how likely are our data given that the null hypothesis is true, and not how likely is the null hypothesis given our data. The latter question can be answered by Bayesian hypothesis testing, where we can compute the probability of any hypothesis of interest.

How to do Bayesian analysis

Bayesian analysis starts with the specification of a posterior model. The posterior model describes the probability distribution of all model parameters conditional on the observed data and some prior knowledge. The posterior distribution has two components: a likelihood, which includes information about model parameters based on the observed data, and a prior, which includes prior information (before observing the data) about model parameters. The likelihood and prior models are combined using the Bayes rule to produce the posterior distribution:

$$\text{Posterior} \propto \text{Likelihood} \times \text{Prior}$$

If the posterior distribution can be derived in a closed form, we may proceed directly to the inference stage of Bayesian analysis. Unfortunately, except for some special models, the posterior distribution is rarely available explicitly and needs to be estimated via simulations. MCMC sampling can be used to simulate potentially very complex posterior models with an arbitrary level of precision. MCMC methods for simulating Bayesian models are often demanding in terms of specifying an efficient sampling algorithm and verifying the convergence of the algorithm to the desired posterior distribution.

Inference is the next step of Bayesian analysis. If MCMC sampling is used for approximating the posterior distribution, the convergence of MCMC must be established before proceeding to inference. Point and interval estimators are either derived from the theoretical posterior distribution or estimated from a sample simulated from the posterior distribution. Many Bayesian estimators, such as posterior

mean and posterior standard deviation, involve integration. If the integration cannot be performed analytically to obtain a closed-form solution, sampling techniques such as Monte Carlo integration and MCMC and numerical integration are commonly used.

Bayesian hypothesis testing can take two forms, which we refer to as interval-hypothesis testing and model-hypothesis testing. In an interval-hypothesis testing, the probability that a parameter or a set of parameters belongs to a particular interval or intervals is computed. In model hypothesis testing, the probability of a Bayesian model of interest given the observed data is computed.

Model comparison is another common step of Bayesian analysis. The Bayesian framework provides a systematic and consistent approach to model comparison using the notion of posterior odds and related to them Bayes factors. See [BAYES] **bayesstats ic** for details.

Finally, prediction of some future unobserved data may also be of interest in Bayesian analysis. The prediction of a new data point is performed conditional on the observed data using the so-called posterior predictive distribution, which involves integrating out all parameters from the model with respect to their posterior distribution. Again, Monte Carlo integration is often the only feasible option for obtaining predictions. Prediction can also be helpful in estimating the goodness of fit of a model.

Advantages and disadvantages of Bayesian analysis

Bayesian analysis is a powerful analytical tool for statistical modeling, interpretation of results, and prediction of data. It can be used when there are no standard frequentist methods available or the existing frequentist methods fail. However, one should be aware of both the advantages and disadvantages of Bayesian analysis before applying it to a specific problem.

The universality of the Bayesian approach is probably its main methodological advantage to the traditional frequentist approach. Bayesian inference is based on a single rule of probability, the Bayes rule, which is applied to all parametric models. This makes the Bayesian approach universal and greatly facilitates its application and interpretation. The frequentist approach, however, relies on a variety of estimation methods designed for specific statistical problems and models. Often, inferential methods designed for one class of problems cannot be applied to another class of models.

In Bayesian analysis, we can use previous information, either belief or experimental evidence, in a data model to acquire more balanced results for a particular problem. For example, incorporating prior information can mitigate the effect of a small sample size. Importantly, the use of the prior evidence is achieved in a theoretically sound and principled way.

By using the knowledge of the entire posterior distribution of model parameters, Bayesian inference is far more comprehensive and flexible than the traditional inference.

Bayesian inference is exact, in the sense that estimation and prediction are based on the posterior distribution. The latter is either known analytically or can be estimated numerically with an arbitrary precision. In contrast, many frequentist estimation procedures such as maximum likelihood rely on the assumption of asymptotic normality for inference.

Bayesian inference provides a straightforward and more intuitive interpretation of the results in terms of probabilities. For example, credible intervals are interpreted as intervals to which parameters belong with a certain probability, unlike the less straightforward repeated-sampling interpretation of the confidence intervals.

Bayesian models satisfy the likelihood principle (Berger and Wolpert 1988) that the information in a sample is fully represented by the likelihood function. This principle requires that if the likelihood function of one model is proportional to the likelihood function of another model, then inferences from the two models should give the same results. Some researchers argue that frequentist methods that depend on the experimental design may violate the likelihood principle.

Finally, as we briefly mentioned earlier, the estimation precision in Bayesian analysis is not limited by the sample size—Bayesian simulation methods may provide an arbitrary degree of precision.

Despite the conceptual and methodological advantages of the Bayesian approach, its application in practice is still considered controversial sometimes. There are two main reasons for this—the presumed subjectivity in specifying prior information and the computational challenges in implementing Bayesian methods. Along with the objectivity that comes from the data, the Bayesian approach uses potentially subjective prior distribution. That is, different individuals may specify different prior distributions. Proponents of frequentist statistics argue that for this reason, Bayesian methods lack objectivity and should be avoided. Indeed, there are settings such as clinical trial cases when the researchers want to minimize a potential bias coming from preexisting beliefs and achieve more objective conclusions. Even in such cases, however, a balanced and reliable Bayesian approach is possible. The trend in using noninformative priors in Bayesian models is an attempt to address the issue of subjectivity. On the other hand, some Bayesian proponents argue that the classical methods of statistical inference have built-in subjectivity such as a choice for a sampling procedure, whereas the subjectivity is made explicit in Bayesian analysis.

Building a reliable Bayesian model requires extensive experience from the researchers, which leads to the second difficulty in Bayesian analysis—setting up a Bayesian model and performing analysis is a demanding and involving task. This is true, however, to an extent for any statistical modeling procedure.

Lastly, one of the main disadvantages of Bayesian analysis is the computational cost. As a rule, Bayesian analysis involves intractable integrals that can only be computed using intensive numerical methods. Most of these methods such as MCMC are stochastic by nature and do not comply with the natural expectation from a user of obtaining deterministic results. Using simulation methods does not compromise the discussed advantages of Bayesian approach, but unquestionably adds to the complexity of its application in practice.

For more discussion about advantages and disadvantages of Bayesian analysis, see, for example, Thompson (2012), Bernardo and Smith (2000), and Berger and Wolpert (1988).

Brief background and literature review

The principles of Bayesian analysis date back to the work of Thomas Bayes, who was a Presbyterian minister in Tunbridge Wells and Pierre Laplace, a French mathematician, astronomer, and physicist in the 18th century. Bayesian analysis started as a simple intuitive rule, named after Bayes, for updating beliefs on account of some evidence. For the next 200 years, however, Bayes's rule was just an obscure idea. Along with the rapid development of the standard or frequentist statistics in 20th century, Bayesian methodology was also developing, although with less attention and at a slower pace. One of the obstacles for the progress of Bayesian ideas has been the lasting opinion among mainstream statisticians of it being subjective. Another more-tangible problem for adopting Bayesian models in practice has been the lack of adequate computational resources. Nowadays, Bayesian statistics is widely accepted by researchers and practitioners as a valuable and feasible alternative.

Bayesian analysis proliferates in diverse areas including industry and government, but its application in sciences and engineering is particularly visible. Bayesian statistical inference is used in econometrics (Poirier [1995]; Chernozhukov and Hong [2003]; Kim, Shephard, and Chib [1998], Zellner [1997]); education (Johnson 1997); epidemiology (Greenland 1998); engineering (Godsill and Rayner 1998); genetics (Iversen, Parmigiani, and Berry 1999); social sciences (Pollard 1986); hydrology (Parent et al. 1998); quality management (Rios Insua 1990); atmospheric sciences (Berliner et al. 1999); and law (DeGroot, Fienberg, and Kadane 1986), to name a few.

The subject of general statistics has been greatly influenced by the development of Bayesian ideas. Bayesian methodologies are now present in biostatistics (Carlin and Louis [2000]; Berry and Stangl [1996]); generalized linear models (Dey, Ghosh, and Mallick 2000); hierarchical modeling (Hobert 2000); statistical design (Chaloner and Verdinelli 1995); classification and discrimination (Neal [1996]; Neal [1999]); graphical models (Pearl 1998); nonparametric estimation (Müller and Vidakovic [1999]; Dey, Müller, and Sinha [1998]); survival analysis (Barlow, Clarotti, and Spizzichino 1993); sequential analysis (Carlin, Kadane, and Gelfand 1998); predictive inference (Aitchison and Dunsmore 1975); spatial statistics (Wolpert and Ickstadt [1998]; Besag and Higdon [1999]); testing and model selection (Kass and Raftery [1995]; Berger and Pericchi [1996]; Berger [2006]); and time series (Pole, West, and Harrison [1994]; West and Harrison [1997]).

Recent advances in computing allowed practitioners to perform Bayesian analysis using simulations. The simulation tools came from outside the statistics field—Metropolis et al. (1953) developed what is now known as a random-walk Metropolis algorithm to solve problems in statistical physics. Another landmark discovery was the Gibbs sampling algorithm (Geman and Geman 1984), initially used in image processing, which showed that exact sampling from a complex and otherwise intractable probability distribution is possible. These ideas were the seeds that led to the development of Markov chain Monte Carlo (MCMC)—a class of iterative simulation methods proved to be indispensable tools for Bayesian computations. Starting from the early 1990s, MCMC-based techniques slowly emerged in the mainstream statistical practice. More powerful and specialized methods appeared, such as perfect sampling (Propp and Wilson 1996), reversible-jump MCMC (Green 1995) for traversing variable dimension state spaces, and particle systems (Gordon, Salmond, and Smith 1993). Consequent widespread application of MCMC was imminent (Berger 2000) and influenced various specialized fields. For example, Gelman and Rubin (1992) investigated MCMC for the purpose of exploring posterior distributions; Geweke (1999) surveyed simulation methods for Bayesian inference in econometrics; Kim, Shephard, and Chib (1998) used MCMC simulations to fit stochastic volatility models; Carlin, Kadane, and Gelfand (1998) implemented Monte Carlo methods for identifying optimal strategies in clinical trials; Chib and Greenberg (1995) provided Bayesian formulation of a number of important econometrics models; and Chernozhukov and Hong (2003) reviewed some econometrics models involving Laplace-type estimators from an MCMC perspective. For more comprehensive exposition of MCMC, see, for example, Robert and Casella (2004); Tanner (1996); Gamerman and Lopes (2006); Chen, Shao, and Ibrahim (2000); and Brooks et al. (2011).

Bayesian statistics

Posterior distribution

To formulate the principles of Bayesian statistics, we start with a simple case when one is concerned with the interaction of two random variables, $\mathbf{A}$ and $\mathbf{B}$. Let $p(\cdot)$ denote either a probability mass function or a density, depending on whether the variables are discrete or continuous. The rule of conditional probability,

$$p(\mathbf{A}|\mathbf{B}) = \frac{p(\mathbf{A},\mathbf{B})}{p(\mathbf{B})}$$

can be used to derive the so-called Bayes's rule:

$$p(\mathbf{B}|\mathbf{A}) = \frac{p(\mathbf{A}|\mathbf{B})p(\mathbf{B})}{p(\mathbf{A})} \tag{1}$$

This rule also holds in the more general case when $\mathbf{A}$ and $\mathbf{B}$ are random vectors.

In a typical statistical problem, we have a data vector $\mathbf{y}$, which is assumed to be a sample from a probability model with an unknown parameter vector $\boldsymbol{\theta}$. We represent this model using the likelihood function $L(\boldsymbol{\theta};\mathbf{y}) = f(\mathbf{y};\boldsymbol{\theta}) = \prod_{i=1}^{n} f(y_i|\boldsymbol{\theta})$, where $f(y_i|\boldsymbol{\theta})$ denotes the probability density function of y_i given $\boldsymbol{\theta}$. We want to infer some properties of $\boldsymbol{\theta}$ based on the data $\mathbf{y}$. In Bayesian statistics, model parameters $\boldsymbol{\theta}$ is a random vector. We assume that $\boldsymbol{\theta}$ has a probability distribution $p(\boldsymbol{\theta}) = \pi(\boldsymbol{\theta})$, which is referred to as a prior distribution. Because both $\mathbf{y}$ and $\boldsymbol{\theta}$ are random, we can apply Bayes's rule (1) to derive the posterior distribution of $\boldsymbol{\theta}$ given data $\mathbf{y}$,

$$p(\boldsymbol{\theta}|\mathbf{y}) = \frac{p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{y})} = \frac{f(\mathbf{y};\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{m(\mathbf{y})} \tag{2}$$

where $m(\mathbf{y}) \equiv p(\mathbf{y})$, known as the marginal distribution of $\mathbf{y}$, is defined by

$$m(\mathbf{y}) = \int f(\mathbf{y};\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta} \tag{3}$$

The marginal distribution $m(\mathbf{y})$ in (3) does not depend on the parameter of interest $\boldsymbol{\theta}$, and we can, therefore, reduce (2) to

$$p(\boldsymbol{\theta}|\mathbf{y}) \propto L(\boldsymbol{\theta};\mathbf{y})\pi(\boldsymbol{\theta}) \tag{4}$$

Equation (4) is fundamental in Bayesian analysis and states that the posterior distribution of model parameters is proportional to their likelihood and prior probability distributions. We will often use (4) in the computationally more-convenient log-scale form

$$\ln\{p(\boldsymbol{\theta}|\mathbf{y})\} = l(\boldsymbol{\theta};\mathbf{y}) + \ln\{\pi(\boldsymbol{\theta})\} - c \tag{5}$$

where $l(\cdot;\cdot)$ denotes the log likelihood of the model. Depending on the analytical procedure involving the log-posterior $\ln\{p(\boldsymbol{\theta}|\mathbf{y})\}$, the actual value of the constant $c = \ln\{m(\mathbf{y})\}$ may or may not be relevant. For valid statistical analysis, however, we will always assume that c is finite.

Selecting priors

In Bayesian analysis, we seek a balance between prior information in a form of expert knowledge or belief and evidence from data at hand. Achieving the right balance is one of the difficulties in Bayesian modeling and inference. In general, we should not allow the prior information to overwhelm the evidence from the data, especially when we have a large data sample. A famous theoretical result, the Bernstein–von Mises theorem, states that in large data samples, the posterior distribution is independent of the prior distribution and, therefore, Bayesian and likelihood-based inferences should yield essentially the same results. On the other hand, we need a strong enough prior to support weak evidence that usually comes from insufficient data. It is always good practice to perform sensitivity analysis to check the dependence of the results on the choice of a prior.

The flexibility of choosing the prior freely is one of the main controversial issues associated with Bayesian analysis and the reason why some practitioners view the latter as subjective. It is also the reason why the Bayesian practice, especially in the early days, was dominated by noninformative priors. Noninformative priors, also called flat or vague priors, assign equal probabilities to all possible states of the parameter space with the aim of rectifying the subjectivity problem. One of the disadvantages of flat priors is that they are often improper; that is, they do not specify a legitimate probability distribution. For example, a uniform prior for a continuous parameter over an unbounded domain does

not integrate to a finite number. However, this is not necessarily a problem because the corresponding posterior distribution may still be proper. Although Bayesian inference based on improper priors is possible, this is equivalent to discarding the terms $\log \pi(\boldsymbol{\theta})$ and c in (5), which nullifies the benefit of Bayesian analysis because it reduces the latter to an inference based only on the likelihood. This is why there is a strong objection to the practice of noninformative priors. In recent years, an increasing number of researchers have advocated the use of sound informative priors, for example, Thompson (2014). For example, using informative priors is mandatory in areas such as genetics, where prior distributions have a physical basis and reflect scientific knowledge.

Another convenient preference for priors is to use conjugate priors. Their choice is desirable from technical and computational standpoints but may not necessarily provide a realistic representation of the model parameters. Because of the limited arsenal of conjugate priors, an inclination to overuse them severely limits the flexibility of Bayesian modeling.

Point and interval estimation

In Bayesian statistics, inference about parameters $\boldsymbol{\theta}$ is based on the posterior distribution $p(\boldsymbol{\theta}|\mathbf{y})$ and various ways of summarizing this distribution. Point and interval estimates can be used to summarize this distribution.

Commonly used point estimators are the posterior mean,

$$E(\boldsymbol{\theta}|\mathbf{y}) = \int \boldsymbol{\theta} p(\boldsymbol{\theta}|\mathbf{y}) d\boldsymbol{\theta}$$

and the posterior median, $q_{0.5}(\boldsymbol{\theta})$, which is the 0.5 quantile of the posterior; that is,

$$P\{\boldsymbol{\theta} \leq q_{0.5}(\boldsymbol{\theta})\} = 0.5$$

Another point estimator is the posterior mode, which is the value of $\boldsymbol{\theta}$ that maximizes $p(\boldsymbol{\theta}|\mathbf{y})$.

Interval estimation is performed by constructing so-called credible intervals (CRIs). CRIs are special cases of credible regions. Let $1 - \alpha \in (0, 1)$ be some predefined credible level. Then, an $\{(1 - \alpha) \times 100\}\%$ credible set R of θ is such that

$$\Pr(\theta \in R|\mathbf{y}) = \int_R p(\theta|\mathbf{y}) d\theta = 1 - \alpha$$

We consider two types of CRIs. The first one is based on quantiles. The second one is the highest posterior density (HPD) interval.

An $\{(1 - \alpha) \times 100\}\%$ quantile-based, or also known as an equal-tailed CRI, is defined as $(q_{\alpha/2}, q_{1-\alpha/2})$, where q_a denotes the ath quantile of the posterior distribution. A commonly reported equal-tailed CRI is $(q_{0.025}, q_{0.975})$.

HPD interval is defined as an $\{(1 - \alpha) \times 100\}\%$ CRI of the shortest width. As its name implies, this interval corresponds to the region of the posterior density with the highest concentration. For a unimodal posterior distribution, HPD is unique, but for a multimodal distribution it may not be unique. Computational approaches for calculating HPD are described in Chen and Shao (1999) and Eberly and Casella (2003).

Comparing Bayesian models

Model comparison is another important aspect of Bayesian statistics. We are often interested in comparing two or more plausible models for our data.

Let's assume that we have models M_j parameterized by vectors $\boldsymbol{\theta}_j$, $j = 1, \ldots, r$. We may have varying degree of belief in each of these models given by prior probabilities $p(M_j)$, such that $\sum_{j=1}^{r} p(M_j) = 1$. By applying Bayes's rule, we find the posterior model probabilities

$$p(M_j|\mathbf{y}) = \frac{p(\mathbf{y}|M_j)p(M_j)}{p(\mathbf{y})}$$

where $p(\mathbf{y}|M_j) = m_j(y)$ is the marginal likelihood of M_j with respect to $\mathbf{y}$. Because of the difficulty in calculating $p(\mathbf{y})$, it is a common practice to compare two models, say, M_j and M_k, using the posterior odds ratio

$$\text{PO}_{jk} = \frac{p(M_j|\mathbf{y})}{p(M_k|\mathbf{y})} = \frac{p(\mathbf{y}|M_j)p(M_j)}{p(\mathbf{y}|M_k)p(M_k)}$$

If all models are equally plausible, that is, $p(M_j) = 1/r$, the posterior odds ratio reduces to the so-called Bayes factors (BF) (Jeffreys 1935),

$$\text{BF}_{jk} = \frac{p(\mathbf{y}|M_j)}{p(\mathbf{y}|M_k)} = \frac{m_j(\mathbf{y})}{m_k(\mathbf{y})}$$

which are simply ratios of marginal likelihoods.

Jeffreys (1961) recommended an interpretation of BF_{jk} based on half-units of the log scale. The following table provides some rules of thumb:

$\log_{10}(\text{BF}_{jk})$	BF_{jk}	Evidence against M_k
0 to 1/2	1 to 3.2	Bare mention
1/2 to 1	3.2 to 10	Substantial
1 to 2	10 to 100	Strong
>2	>100	Decisive

The Schwarz criterion BIC (Schwarz 1978) is an approximation of BF in case of arbitrary but proper priors. Kass and Raftery (1995) and Berger (2006) provide a detailed exposition of Bayes factors, their calculation, and their role in model building and testing.

Posterior prediction

Prediction is another essential part of statistical analysis. In Bayesian statistics, prediction is performed using the posterior distribution. The probability of observing some future data $\mathbf{y}^*$ given the observed one can be obtained by the marginalization of

$$p(\mathbf{y}^*|\mathbf{y}) = \int p(\mathbf{y}^*|\mathbf{y}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{y})d\boldsymbol{\theta}$$

which, assuming that $\mathbf{y}^*$ is independent of $\mathbf{y}$, can be simplified to

$$p(\mathbf{y}^*|\mathbf{y}) = \int p(\mathbf{y}^*|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{y})d\boldsymbol{\theta} \tag{6}$$

Equation (6) is called a posterior predictive distribution and is used for Bayesian prediction.

Bayesian computation

An unavoidable difficulty in performing Bayesian analysis is the need to compute integrals such as those expressing marginal distributions and posterior moments. The integrals involved in Bayesian inference are of the form $E\{g(\boldsymbol{\theta})\} = \int g(\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{y})d\boldsymbol{\theta}$ for some function $g(\cdot)$ of the random vector $\boldsymbol{\theta}$. With the exception of a few cases for which analytical integration is possible, the integration is performed via simulations.

Given a sample from the posterior distribution, we can use Monte Carlo integration to approximate the integrals. Let $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_T$ be an independent sample from $p(\boldsymbol{\theta}|\mathbf{y})$.

The original integral of interest $E\{g(\boldsymbol{\theta})\}$ can be approximated by

$$\widehat{g} = \frac{1}{T}\sum_{t=1}^{T} g(\boldsymbol{\theta}_t)$$

Moreover, if g is a scalar function, under some mild conditions, the central limit theorem holds

$$\widehat{g} \approx N\left[E\{g(\boldsymbol{\theta})\}, \sigma^2/T\right]$$

where $\sigma^2 = \mathrm{Cov}\{g(\boldsymbol{\theta}_i)\}$ can be approximated by the sample variance $\sum_{t=1}^{T}\{g(\boldsymbol{\theta}_t) - \widehat{g}\}^2/T$. If the sample is not independent, then $\widehat{g}$ still approximates $E\{g(\boldsymbol{\theta})\}$ but the variance σ^2 is given by

$$\sigma^2 = \mathrm{Var}\{g(\boldsymbol{\theta}_t)\} + 2\sum_{k=1}^{\infty}\mathrm{Cov}\{g(\boldsymbol{\theta}_t), g(\boldsymbol{\theta}_{t+k})\} \tag{7}$$

and needs to be approximated. Moreover, the conditions needed for the central limit theorem to hold involve the convergence rate of the chain and can be difficult to check in practice (Tierney 1994).

The Monte Carlo integration method solves the problem of Bayesian computation of computing a posterior distribution by sampling from that posterior distribution. The latter has been an important problem in computational statistics and a focus of intense research. Rejection sampling techniques serve as basic tools for generating samples from a general probability distribution (von Neumann 1951). They are based on the idea that samples from the target distribution can be obtained from another, easy-to-sample distribution according to some acceptance–rejection rule for the samples from this distribution. It was soon recognized, however, that the acceptance–rejection methods did not scale well with the increase of dimensions, a problem known as the "curse of dimensionality", essentially reducing the acceptance probability to zero. An alternative solution was to use the Markov chains to generate sequences of correlated sample points from the domain of the target distribution and keeping a reasonable rate of acceptance. It was not long before Markov chain Monte Carlo methods were accepted as effective tools for approximate sampling from general posterior distributions (Tanner and Wong 1987).

Markov chain Monte Carlo methods

Every MCMC method is designed to generate values from a transition kernel such that the draws from that kernel converge to a prespecified target distribution. It simulates a Markov chain with the target distribution as the stationary or equilibrium distribution of the chain. By definition, a Markov chain is any sequence of values or states from the domain of the target distribution, such that each value depends on its immediate predecessor only. For a well-designed MCMC, the longer the chain, the closer the samples to the stationary distribution. MCMC methods differ substantially in their simulation efficiency and computational complexity.

The Metropolis algorithm proposed in Metropolis and Ulam (1949) and Metropolis et al. (1953) appears to be the earliest version of MCMC. The algorithm generates a sequence of states, each obtained from the previous one, according to a Gaussian proposal distribution centered at that state. Hastings (1970) described a more-general version of the algorithm, now known as a Metropolis–Hastings (MH) algorithm, which allows any distribution to be used as a proposal distribution. Below we review the general MH algorithm and some of its special cases.

Metropolis–Hastings algorithm

Here we present the MH algorithm for sampling from a posterior distribution in a general formulation. It requires the specification of a proposal probability distribution $q(\cdot)$ and a starting state $\boldsymbol{\theta}_0$ within the domain of the posterior, that is, $p(\boldsymbol{\theta}_0|\mathbf{y}) > 0$. The algorithm generates a Markov chain $\{\boldsymbol{\theta}_t\}_{t=0}^{T-1}$ such that at each step t 1) a proposal state $\boldsymbol{\theta}_*$ is generated conditional on the current state, and 2) $\boldsymbol{\theta}_*$ is accepted or rejected according to the suitably defined acceptance probability.

For $t = 1, \ldots, T-1$:

1. Generate a proposal state: $\boldsymbol{\theta}_* \sim q(\cdot|\boldsymbol{\theta}_{t-1})$.
2. Calculate the acceptance probability

$$\alpha(\boldsymbol{\theta}_*|\boldsymbol{\theta}_{t-1}) = \frac{p(\boldsymbol{\theta}_*|\mathbf{y})q(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}_*)}{p(\boldsymbol{\theta}_{t-1}|\mathbf{y})q(\boldsymbol{\theta}_*|\boldsymbol{\theta}_{t-1})}$$

3. Draw $u \sim \text{Uniform}(0, 1)$.
4. Set $\boldsymbol{\theta}_t = \boldsymbol{\theta}_*$ if $u < \min\{\alpha(\boldsymbol{\theta}_*|\boldsymbol{\theta}_{t-1}), 1\}$, and $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1}$ otherwise.

We refer to the iteration steps 1 through 4 as an MH update. By design, any Markov chain simulated using this MH algorithm is guaranteed to have $p(\boldsymbol{\theta}|\mathbf{y})$ as its stationary distribution.

Two important criteria measuring the efficiency of MCMC are the acceptance rate of the chain and the degree of autocorrelation in the generated sample. When the acceptance rate is close to 0, then most of the proposals are rejected, which means that the chain failed to explore regions of appreciable posterior probability. The other extreme is when the acceptance probability is close to 1, in which case the chain stays in a small region and fails to explore the whole posterior domain. An efficient MCMC has an acceptance rate that is neither too small nor too large and also has small autocorrelation. Gelman, Gilks, and Roberts (1997) showed that in the case of a multivariate posterior and proposal distributions, an acceptance rate of 0.234 is asymptotically optimal and, in the case of a univariate posterior, the optimal value is 0.45.

A special case of MH employs a Metropolis update with $q(\cdot)$ being a symmetric distribution. Then, the acceptance probability reduces to a ratio of posterior probabilities,

$$\alpha(\boldsymbol{\theta}_*|\boldsymbol{\theta}_{t-1}) = \frac{p(\boldsymbol{\theta}_*|\mathbf{y})}{p(\boldsymbol{\theta}_{t-1}|\mathbf{y})}$$

The symmetric Gaussian distribution is a common choice for a proposal distribution $q(\cdot)$, and this is the one used in the original Metropolis algorithm.

Another important MCMC method that can be viewed as a special case of MH is Gibbs sampling (Gelfand et al. 1990), where the updates are the full conditional distributions of each parameter given the rest of the parameters. Gibbs updates are always accepted. If $\boldsymbol{\theta} = (\theta^1, \ldots, \theta^d)$ and, for $j = 1 \ldots, d$, q_j is the conditional distribution of θ^j given the rest $\boldsymbol{\theta}^{\{-j\}}$, then the Gibbs algorithm is the following. For $t = 1, \ldots, T-1$ and for $j = 1, \ldots, d$: $\theta_t^j \sim q_j(\cdot|\boldsymbol{\theta}_{t-1}^{\{-j\}})$. This step is referred to as a Gibbs update.

All MCMC methods share some limitations and potential problems. First, any simulated chain is influenced by its starting values, especially for short MCMC runs. It is required that the starting point has a positive posterior probability, but even when this condition is satisfied, if we start somewhere in a remote tail of the target distribution, it may take many iterations to reach a region of appreciable probability. Second, because there is no obvious stopping criterion, it is not easy to decide for how long to run the MCMC algorithm to achieve convergence to the target distribution. Third, the observations in MCMC samples are strongly dependent and this must be taken into account in any subsequent statistical inference. For example, the errors associated with the Monte Carlo integration should be calculated according to (7), which accounts for autocorrelation.

Adaptive random-walk Metropolis–Hastings

The choice of a proposal distribution $q(\cdot)$ in the MH algorithm is crucial for the mixing properties of the resulting Markov chain. The problem of determining an optimal proposal for a particular target posterior distribution is difficult and is still being researched actively. All proposed solutions are based on some form of an adaptation of the proposal distribution as the Markov chain progresses, which is carefully designed to preserve the ergodicity of the chain, that is, its tendency to converge to the target distribution. These methods are known as adaptive MCMC methods (Haario, Saksman, and Tamminen [2001]; Giordani and Kohn [2010]; and Roberts and Rosenthal [2009]).

The majority of adaptive MCMC methods are random-walk MH algorithms with updates of the form: $\boldsymbol{\theta}_* = \boldsymbol{\theta}_{t-1} + Z_t$, where Z_t follows some symmetric distribution. Specifically, we consider a Gaussian random-walk MH algorithm with $Z_t \sim N(0, \rho^2\Sigma)$, where ρ is a scalar controlling the scale of random jumps for generating updates and Σ is a d-dimensional covariance matrix. One of the first important results regarding adaptation is from Gelman, Gilks, and Roberts (1997), where the authors derive the optimal scaling factor $\rho = 2.38/\sqrt{d}$ and note that the optimal Σ is the true covariance matrix of the target distribution.

Haario, Saksman, and Tamminen (2001) proposes Σ to be estimated by the empirical covariance matrix plus a small diagonal matrix $\epsilon \times I_d$ to prevent zero covariance matrices. Alternatively, Roberts and Rosenthal (2009) proposed a mixture of the two covariance matrices,

$$\Sigma_t = \beta\widehat{\Sigma} + (1-\beta)\Sigma_0$$

for some fixed covariance matrix Σ_0 and $\beta \in [0, 1]$.

Because the proposal distribution of an adaptive MH algorithm changes at each step, the ergodicity of the chain is not necessarily preserved. However, under certain assumptions about the adaptation procedure, the ergodicity does hold; see Roberts and Rosenthal (2007), Andrieu and Moulines (2006), Atchadé and Rosenthal (2005), and Giordani and Kohn (2010) for details.

Blocking of parameters

In the original MH algorithm, the update steps of generating proposals and applying the acceptance–rejection rule are performed for all model parameters simultaneously. For high-dimensional models, this may result in a poor mixing—the Markov chain may stay in the tails of the posterior distribution for long periods of time and traverse the posterior domain very slowly. Suboptimal mixing is manifested by either very high or very low acceptance rates. Adaptive MH algorithms are also prone to this problem, especially when model parameters have very different scales. An effective solution to this problem is called *blocking*—model parameters are separated into two or more subsets or blocks and MH updates are applied to each block separately in the order that the blocks are specified.

Let's separate a vector of parameters into B blocks: $\boldsymbol{\theta} = \{\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^B\}$. The version of the Gaussian random-walk MH algorithm with blocking is as follows.

Let T_0 be the number of burn-in iterations, T be the number of MCMC samples, and $\rho_b^2\Sigma^b$, $b = 1, \ldots, B$, be block-specific proposal covariance matrices. Let $\boldsymbol{\theta}_0$ be the starting point within the domain of the posterior, that is, $p(\boldsymbol{\theta}_0|\mathbf{y}) > 0$.

1. At iteration t, let $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1}$.
2. For a block of parameters $\boldsymbol{\theta}_t^b$:
   2.1. Let $\boldsymbol{\theta}_* = \boldsymbol{\theta}_t$. Generate a proposal for the bth block: $\boldsymbol{\theta}_*^b = \boldsymbol{\theta}_{t-1}^b + \epsilon$, where $\epsilon \sim N(0, \rho_b^2\Sigma^b)$.
   2.2. Calculate the acceptance probability,

$$r = \frac{p(\boldsymbol{\theta}_*|\mathbf{y})}{p(\boldsymbol{\theta}_t|\mathbf{y})}$$

   where $\theta_* = (\boldsymbol{\theta}_t^1, \boldsymbol{\theta}_t^2, \ldots, \boldsymbol{\theta}_t^{b-1}, \boldsymbol{\theta}_*^b, \boldsymbol{\theta}_t^{b+1}, \ldots, \boldsymbol{\theta}_t^B)$.
   2.3. Draw $u \sim \text{Uniform}(0, 1)$.
   2.4. Let $\boldsymbol{\theta}_t^b = \boldsymbol{\theta}_*^b$ if $u < \min\{r, 1\}$.
3. Repeat step 2 for $b = 1, \ldots, B$.
4. Repeat steps 1 through 3 for $t = 1, \ldots, T + T_0 - 1$.
5. The final sequence is $\{\boldsymbol{\theta}_t\}_{t=T_0}^{T+T_0-1}$.

Blocking may not always improve efficiency. For example, separating all parameters in individual blocks (the so-called one-at-a-time update regime) can lead to slow mixing when some parameters are highly correlated. A Markov chain may explore the posterior domain very slowly if highly correlated parameters are updated independently. There are no theoretical results about optimal blocking, so you will need to use your judgment when determining the best set of blocks for your model. As a rule, parameters that are expected to be highly correlated are specified in one block. This will generally improve mixing of the chain unless the proposal correlation matrix does not capture the actual correlation structure of the block. For example, if there are two parameters in the block that have very different scales, adaptive MH algorithms that use the identity matrix for the initial proposal covariance may take a long time to approximate the optimal proposal correlation matrix. The user should, therefore, consider not only the probabilistic relationship between the parameters in the model, but also their scales to determine an optimal set of blocks.

Metropolis–Hastings with Gibbs updates

The original Gibbs sampler updates each model parameter one at a time according to its full conditional distribution. We have already noted that Gibbs is a special case of the MH algorithm. Some of the advantages of Gibbs sampling include its high efficiency, because all proposals are automatically accepted, and that it does not require any additional tuning for proposal distributions in MH algorithms. Unfortunately, for most posterior distributions in practice, the full conditionals are either not available or are very difficult to sample from. It may be the case, however, that for some model parameters or groups of parameters, the full conditionals are available and are easy to generate samples from. This is done in a hybrid MH algorithm, which implements Gibbs updates for only some blocks of parameters. A hybrid MH algorithm combines Gaussian random-walk updates with Gibbs updates to improve the mixing of the chain.

The MH algorithm with blocking allows different samplers to be used for updating different blocks. If there is a group of model parameters with a conjugate prior (or semiconjugate prior), we can place this group of parameters in a separate block and use Gibbs sampling for it. This can greatly improve the overall sampling efficiency of the algorithm.

For example, suppose that the data are normally distributed with a known mean μ and that we specify an inverse-gamma prior for σ^2 with shape α and scale β, which are some fixed constants.

$$y \sim N(\mu, \sigma^2), \quad \sigma^2 \sim \text{InvGamma}(\alpha,\ \beta)$$

The full conditional distribution for σ^2 in this case is also an inverse-gamma distribution, but with different shape and scale parameters,

$$\sigma^2 \sim \text{InvGamma}\left\{\widetilde{\alpha} = \alpha + \frac{n}{2},\ \widetilde{\beta} = \beta + \frac{1}{2}\sum_{i=1}^{n}(y_i - \mu)^2\right\}$$

where n is the data sample size. So, an inverse-gamma prior for the variance is a conjugate prior in this model. We can thus place σ^2 in a separate block and set up a Gibbs sampling for it using the above full conditional distribution.

See *Methods and formulas* in [BAYES] **bayesmh** for details.

Convergence diagnostics of MCMC

Checking convergence of MCMC is an essential step in any MCMC simulation. Bayesian inference based on an MCMC sample is valid only if the Markov chain has converged and the sample is drawn from the desired posterior distribution. It is important that we verify the convergence for all model parameters and not only for a subset of parameters of interest. One difficulty with assessing convergence of MCMC is that there is no single conclusive convergence criterion. The diagnostic usually involves checking for several necessary (but not necessarily sufficient) conditions for convergence. In general, the more aspects of the MCMC sample you inspect, the more reliable your results are.

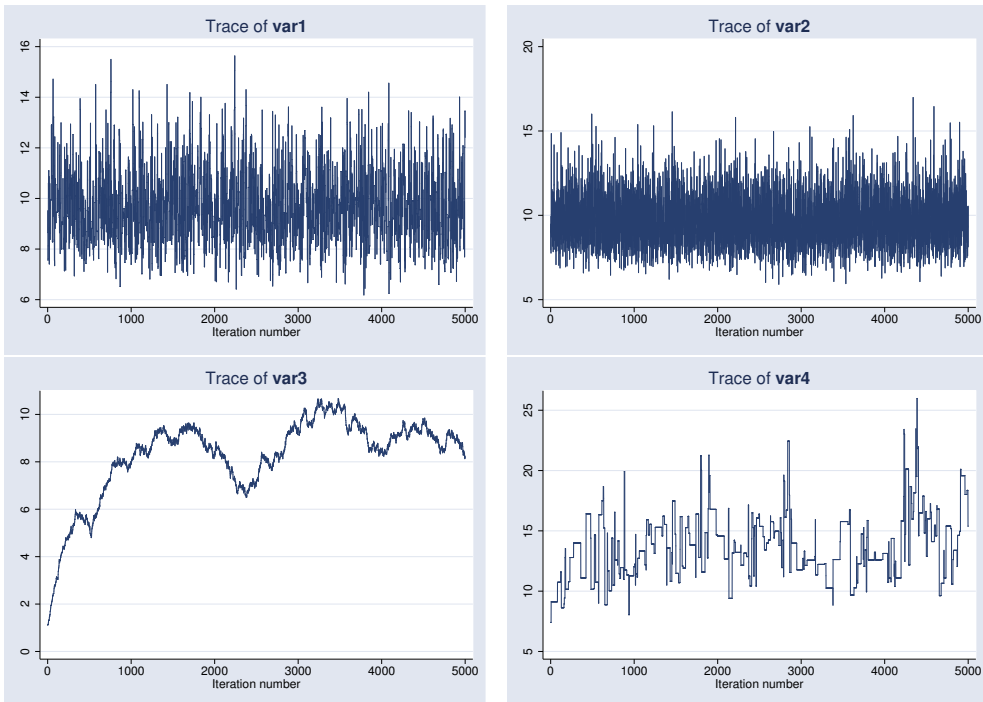
The most extensive review of the methods for assessing convergence is Cowles and Carlin (1996). Other discussions about monitoring convergence can be found in Gelman et al. (2014) and Brooks et al. (2011).

There are at least two general approaches for detecting convergence issues. The first one is to inspect the mixing and time trends within the chains of individual parameters. The second one is to examine the mixing and time trends of multiple chains for each parameter. The lack of convergence in a Markov chain can be especially difficult to detect in a case of pseudoconvergence, which often

occurs with multimodal posterior distributions. Pseudoconvergence occurs when the chain appears to have converged but it actually explored only a portion of the domain of a posterior distribution. To check for pseudoconvergence, Gelman and Rubin (1992) recommend running multiple chains from different starting states and comparing them.

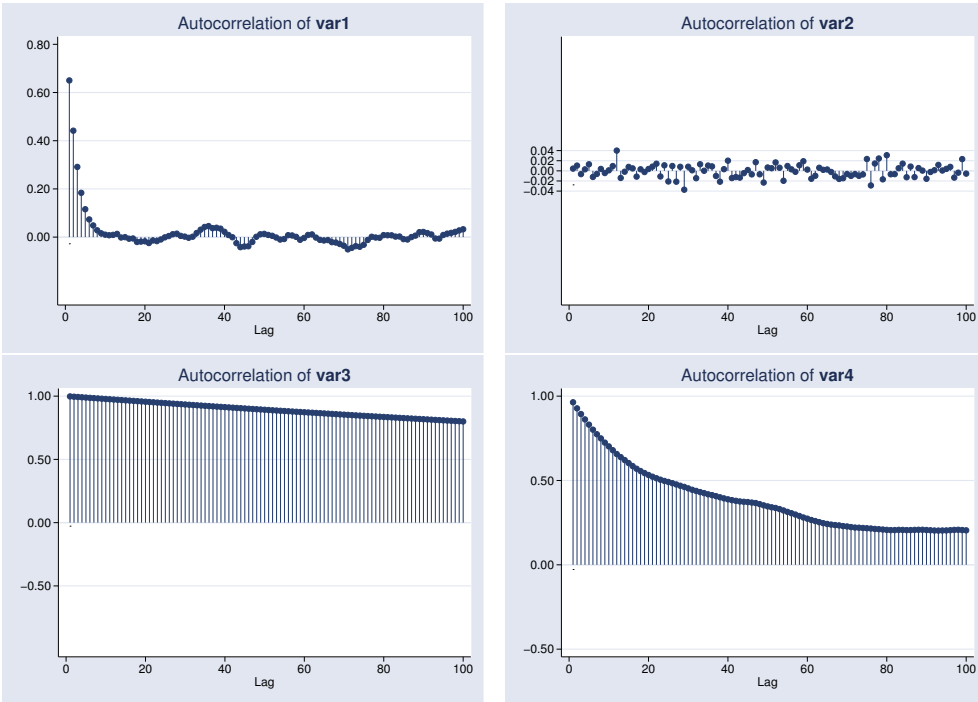
Trace plots are the most accessible convergence diagnostics and are easy to inspect visually. The trace plot of a parameter plots the simulated values for this parameter versus the iteration number. The trace plot of a well-mixing parameter should traverse the posterior domain rapidly and should have nearly constant mean and variance.

In the next figure, we show examples of trace plots for four parameters: `var1`, `var2`, `var3`, and `var4`. The first two parameters, `var1` and `var2`, have well-mixing chains, and the other two have poorly mixing chains. The chain for the parameter `var1` has a moderate acceptance rate, about 35%, and efficiency between 10% and 20%. This is a typical result for a Gaussian random-walk MH algorithm that has achieved convergence. The trace plot of `var2` in the top right panel shows almost perfect mixing—this is a typical example of Gibbs sampling with an acceptance rate close to 1 and efficiency above 95%. Although both chains traverse their marginal posterior domains, the right one does it more rapidly. On the downside, more efficient MCMC algorithms such as Gibbs sampling are usually associated with a higher computational cost.

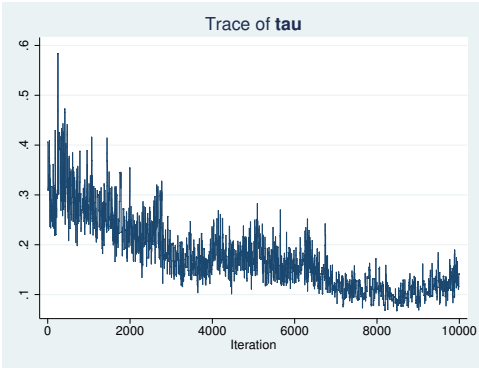


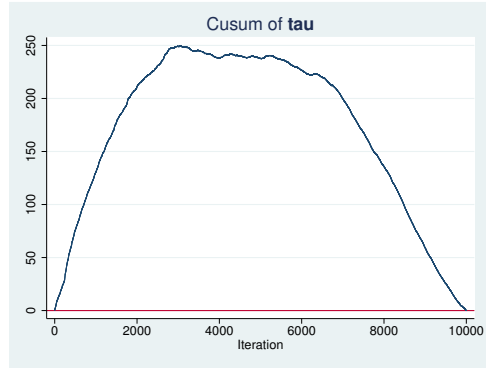
The bottom two trace plots illustrate cases of bad mixing and a lack of convergence. On the left, the chain for `var3` exhibits high acceptance rate but poor coverage of the posterior domain manifested by random drifting in isolated regions. This chain was produced by a Gaussian random-walk MH algorithm with a proposal distribution with a very small variance. On the right, the chain for the parameter `var4` has a very low acceptance rate, below 3%, because the used proposal distribution had a very large variance. In both cases, the chains do not converge; the simulation results do not represent the posterior distribution and should thus be discarded.

As we stated before, samples simulated using MCMC methods are correlated. The smaller the correlation, the more efficient the sampling process. Most of the MH algorithms typically generate highly correlated draws, whereas the Gibbs algorithm typically generates less-correlated draws. Below we show autocorrelation plots for the same four parameters using the same MCMC samples. The autocorrelation of `var1`, the one that comes from a well-mixing MH chain, becomes negligible fairly quickly, after about 10 lags. On the other hand, the autocorrelation of `var2` simulated using Gibbs sampling is essentially negligible for all positive lags. In the case of a poor mixing because of a small proposal variance (parameter `var3`), we observe very high positive correlation for at least 100 lags. The autocorrelation of `var4` is high but is lower than that of `var3`.

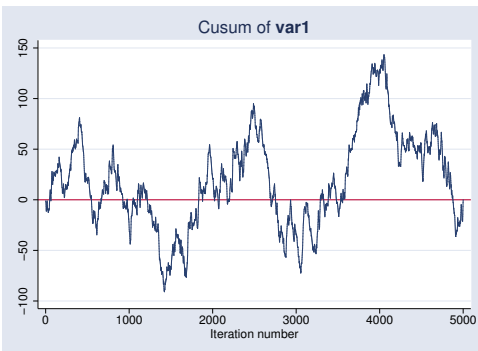


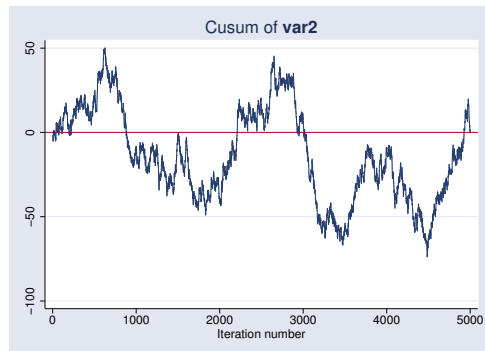
Yu and Mykland (1998) proposed a graphical procedure for assessing the convergence of individual parameters based on cumulative sums, also known as a cusum plot. By definition, any cusum plot starts at 0 and ends at 0. Cusum plots are useful for detecting drifts in the chain. For a chain without trend, the cusum plot should cross the x axis. For example, early drifts may indicate dependence on starting values. If we detect an early drift, we should discard an initial part of the chain and run it longer. Below, we show the trace plot of a poorly mixing parameter `tau` and its corresponding cusum plot on the right. There is an apparent positive drift for approximately the first half of the chain followed by the drift in the negative direction. As a result, the cusum plot has a distinctive mountain-like shape and never crosses the x axis.

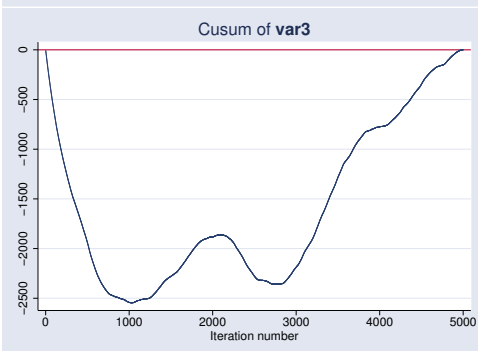


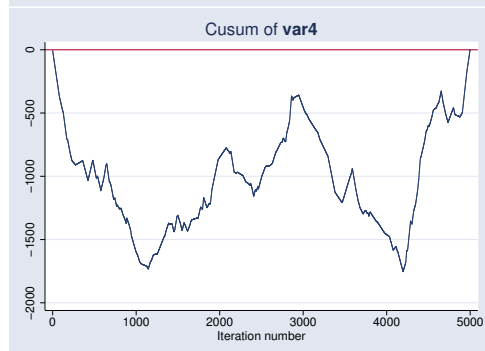


Cusum plots can be also used for assessing how fast the chain is mixing. The slower the mixing of the chain, the smoother the cusum plots. Conversely, the faster the mixing of the chain, the more jagged the cusum plots. Below, we demonstrate the cusum plots for the four variables considered previously. We can clearly see the contrast between the jagged lines of the fast mixing parameters `var1` and `var2` and the very smooth cusum line of the poorly mixing parameter `var3`.









Besides graphical convergence diagnostics, there are some formal convergence tests (Geweke [1992]; Gelman and Rubin [1992]; Heidelberger and Welch [1983]; Raftery and Lewis [1992]; Zellner and Min [1995]).

Summary

Bayesian analysis is a statistical procedure that answers research questions by expressing uncertainty about unknown parameters using probabilities. Bayesian inference is based on the posterior distribution of model parameters conditional on the observed data. The posterior distribution is composed of a likelihood distribution of the data and the prior distribution of the model parameters. The likelihood model is specified in the same way it is specified with any standard likelihood-based analysis. The prior distribution is constructed based on the prior (before observing the data) scientific knowledge and results from previous studies. Sensitivity analysis is typically performed to evaluate the influence of different competing priors on the results.

Many posterior distributions do not have a closed form and must be simulated using MCMC methods such as MH methods or the Gibbs method or sometimes their combination. The convergence of MCMC must be verified before any inference can be made.

Marginal posterior distributions of the parameters are used for inference. These are summarized using point estimators such as posterior mean and median and interval estimators such as equal-tailed credible intervals and highest posterior density intervals. Credible intervals have an intuitive interpretation as fixed ranges to which a parameter is known to belong with a prespecified probability. Hypothesis testing provides a way to assign an actual probability to any hypothesis of interest. A number of criteria are available for comparing models of interest. Predictions are also available based on the posterior predictive distribution.

Bayesian analysis provides many advantages over the standard frequentist analysis, such as an ability to incorporate prior information in the analysis, higher robustness to sparse data, more-comprehensive inference based on the knowledge of the entire posterior distribution, and more intuitive and direct interpretations of results by using probability statements about parameters.

Thomas Bayes (1701(?)–1761) was a Presbyterian minister with an interest in calculus, geometry, and probability theory. He was born in Hertfordshire, England. The son of a Nonconformist minister, Bayes was banned from English universities and so studied at Edinburgh University before becoming a clergyman himself. Only two works are attributed to Bayes during his lifetime, both published anonymously. He was admitted to the Royal Society in 1742 and never published thereafter.

The paper that gives us "Bayes's Theorem" was published posthumously by Richard Price. The theorem has become an important concept for frequentist and Bayesian statisticians alike. However, the paper indicates that Bayes considered the theorem as relatively unimportant. His main interest appears to have been that probabilities were not fixed but instead followed some distribution. The notion, now foundational to Bayesian statistics, was largely ignored at the time.

Whether Bayes's theorem is appropriately named is the subject of much debate. Price acknowledged that he had written the paper based on information he found in Bayes's notebook, yet he never said how much he added beyond the introduction. Some scholars have also questioned whether Bayes's notes represent original work or are the result of correspondence with other mathematicians of the time.

Andrey Markov (1856–1922) was a Russian mathematician who made many contributions to mathematics and statistics. He was born in Ryazan, Russia. In primary school, he was known as a poor student in all areas except mathematics. Markov attended St. Petersburg University, where he studied under Pafnuty Chebyshev and later joined the physicomathematical faculty. He was a member of the Russian Academy of the Sciences.

Markov's first interest was in calculus. He did not start his work in probability theory until 1883 when Chebyshev left the university and Markov took over his teaching duties. A large and influential body of work followed, including applications of the weak law of large numbers and what are now known as Markov processes and Markov chains. His work on processes and chains would later influence the development of a variety of disciplines such as biology, chemistry, economics, physics, and statistics.

Known in the Russian press as the "militant academician" for his frequent written protests about the czarist government's interference in academic affairs, Markov spent much of his adult life at odds with Russian authorities. In 1908, he resigned from his teaching position in response to a government requirement that professors report on students' efforts to organize protests in the wake of the student riots earlier that year. He did not resume his university teaching duties until 1917, after the Russian Revolution. His trouble with Russian authorities also extended to the Russian Orthodox Church. In 1912, he was excommunicated at his own request in protest over the Church's excommunication of Leo Tolstoy.

References

Aitchison, J., and I. R. Dunsmore. 1975. *Statistical Prediction Analysis*. Cambridge: Cambridge University Press.

Andrieu, C., and É. Moulines. 2006. On the ergodicity properties of some adaptive MCMC algorithms. *Annals of Applied Probability* 16: 1462–1505.

Atchadé, Y. F., and J. S. Rosenthal. 2005. On adaptive Markov chain Monte Carlo algorithms. *Bernoulli* 11: 815–828.

Barlow, R. E., C. A. Clarotti, and F. Spizzichino, ed. 1993. *Reliability and Decision Making*. Cambridge: Chapman & Hall.

Berger, J. O. 2000. Bayesian analysis: A look at today and thoughts of tomorrow. *Journal of the American Statistical Association* 95: 1269–1276.

——. 2006. "Bayes factors." In *Encyclopedia of Statistical Sciences*, edited by Kotz, S., C. B. Read, N. Balakrishnan, and B. Vidakovic. Wiley. http://onlinelibrary.wiley.com/doi/10.1002/0471667196.ess0985.pub2/abstract.

Berger, J. O., and L. R. Pericchi. 1996. The intrinsic Bayes factor for model selection and prediction. *Journal of the American Statistical Association* 91: 109–122.

Berger, J. O., and R. L. Wolpert. 1988. *The Likelihood Principle: A Review, Generalizations, and Statistical Implications*. Hayward, CA: Institute of Mathematical Statistics.

Berliner, L. M., J. A. Royle, C. K. Wikle, and R. F. Milliff. 1999. Bayesian methods in atmospheric sciences. In Vol. 6 of *Bayesian Statistics: Proceedings of the Sixth Valencia International Meeting*, ed. J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, 83–100. Oxford: Oxford University Press.

Bernardo, J. M., and A. F. M. Smith. 2000. *Bayesian Theory*. Chichester, UK: Wiley.

Berry, D. A., and D. K. Stangl, ed. 1996. *Bayesian Biostatistics*. New York: Dekker.

Besag, J., and D. Higdon. 1999. Bayesian analysis for agricultural field experiments. *Journal of the Royal Statistical Society, Series B* 61: 691–746.

Brooks, S., A. Gelman, G. L. Jones, and X.-L. Meng, ed. 2011. *Handbook of Markov Chain Monte Carlo*. Boca Raton, FL: Chapman & Hall/CRC.

Carlin, B. P., J. B. Kadane, and A. E. Gelfand. 1998. Approaches for optimal sequential decision analysis in clinical trials. *Biometrics* 54: 964–975.

Carlin, B. P., and T. A. Louis. 2000. *Bayes and Empirical Bayes Methods for Data Analysis*. 2nd ed. Boca Raton, FL: Chapman & Hall/CRC.

Chaloner, K., and I. Verdinelli. 1995. Bayesian experimental design: A review. *Statistical Science* 10: 273–304.

Chen, M.-H., and Q.-M. Shao. 1999. Monte Carlo estimation of Bayesian credible and HPD intervals. *Journal of Computational and Graphical Statistics* 8: 69–92.

Chen, M.-H., Q.-M. Shao, and J. G. Ibrahim. 2000. *Monte Carlo Methods in Bayesian Computation*. New York: Springer.

Chernozhukov, V., and H. Hong. 2003. An MCMC approach to classical estimation. *Journal of Econometrics* 115: 293–346.

Chib, S., and E. Greenberg. 1995. Understanding the Metropolis–Hastings algorithm. *American Statistician* 49: 327–335.

Cowles, M. K., and B. P. Carlin. 1996. Markov chain Monte Carlo convergence diagnostic: A comparative review. *Journal of the American Statistical Association* 91: 883–904.

DeGroot, M. H., S. E. Fienberg, and J. B. Kadane. 1986. *Statistics and the Law*. New York: Wiley.

Dey, D. D., P. Müller, and D. Sinha, ed. 1998. *Practical Nonparametric and Semiparametric Bayesian Statistics*. New York: Springer.

Dey, D. K., S. K. Ghosh, and B. K. Mallick. 2000. *Generalized Linear Models: A Bayesian Perspective*. New York: Dekker.

Eberly, L. E., and G. Casella. 2003. Estimating Bayesian credible intervals. *Journal of Statistical Planning and Inference* 112: 115–132.

Gamerman, D., and H. F. Lopes. 2006. *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. 2nd ed. Boca Raton, FL: Chapman & Hall/CRC.

Gelfand, A. E., S. E. Hills, A. Racine-Poon, and A. F. M. Smith. 1990. Illustration of Bayesian inference in normal data models using Gibbs sampling. *Journal of the American Statistical Association* 85: 972–985.

Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. 2014. *Bayesian Data Analysis*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.

Gelman, A., W. R. Gilks, and G. O. Roberts. 1997. Weak convergence and optimal scaling of random walk Metropolis algorithms. *Annals of Applied Probability* 7: 110–120.

Gelman, A., and D. B. Rubin. 1992. Inference from iterative simulation using multiple sequences. *Statistical Science* 7: 457–472.

Geman, S., and D. Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6: 721–741.

Geweke, J. 1992. Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In Vol. 4 of *Bayesian Statistics: Proceedings of the Fourth Valencia International Meeting, April 15–20, 1991*, ed. J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, 169–193. Oxford: Clarendon Press.

——. 1999. Using simulation methods for Bayesian econometric models: Inference, development, and communication. *Econometric Reviews* 18: 1–73.

Giordani, P., and R. J. Kohn. 2010. Adaptive independent Metropolis–Hastings by fast estimation of mixtures of normals. *Journal of Computational and Graphical Statistics* 19: 243–259.

Godsill, S. J., and P. J. W. Rayner. 1998. *Digital Audio Restoration*. Berlin: Springer.

Gordon, N. J., D. J. Salmond, and A. F. M. Smith. 1993. novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings on Radar and Signal Processing* 140: 107–113.

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

Greenland, S. 1998. Probability logic and probabilistic induction. *Epidemiology* 9: 322–332.

Haario, H., E. Saksman, and J. Tamminen. 2001. An adaptive Metropolis algorithm. *Bernoulli* 7: 223–242.

Hastings, W. K. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 57: 97–109.

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

Hobert, J. P. 2000. Hierarchical models: A current computational perspective. *Journal of the American Statistical Association* 95: 1312–1316.

Hoff, P. D. 2009. *A First Course in Bayesian Statistical Methods*. New York: Springer.

Iversen, E., Jr, G. Parmigiani, and D. A. Berry. 1999. Validating Bayesian Prediction Models: a Case Study in Genetic Susceptibility to Breast Cancer. In *Case Studies in Bayesian Statistics*, ed. J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, vol. IV, 321–338. New York: Springer.

Jeffreys, H. 1935. Some tests of significance, treated by the theory of probability. *Mathematical Proceedings of the Cambridge Philosophical Society* 31: 203–222.

——. 1961. *Theory of Probability*. 3rd ed. Oxford: Oxford University Press.

Johnson, V. E. 1997. An alternative to traditional GPA for evaluating student performance. *Statistical Science* 12: 251–269.

Kass, R. E., and A. E. Raftery. 1995. Bayes factors. *Journal of the American Statistical Association* 90: 773–795.

Kim, S., N. Shephard, and S. Chib. 1998. Stochastic volatility: Likelihood inference and comparison with ARCH models. *The Reviews of Economic Studies* 65: 361–393.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics* 21: 1087–1092.

Metropolis, N., and S. Ulam. 1949. The Monte Carlo method. *Journal of the American Statistical Association* 44: 335–341.

Müller, P., and B. Vidakovic, ed. 1999. *Bayesian Inference in Wavelet-Based Models*. New York: Springer.

Neal, R. M. 1996. *Bayesian Learning for Neural Networks*. New York: Springer.

——. 1999. Regression and classification using gaussian process priors. In Vol. 6 of *Bayesian Statistics: Proceedings of the Sixth Valencia International Meeting*, ed. J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, 475–501. Oxford: Oxford University Press.

Parent, E., P. Hubert, B. Bobee, and J. Miquel. 1998. *Statistical and Bayesian Methods in Hydrological Sciences*. Paris: UNESCO Press.

Pearl, J. 1998. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA: Morgan Kaufmann.

Poirier, D. J. 1995. *Intermediate Statistics and Econometrics: A Comparative Approach*. Cambridge, MA: MIT Press.

Pole, A., M. West, and J. Harrison. 1994. *Applied Bayesian Forecasting and Time Series Analysis*. Boca Raton, FL: Chapman and Hall.

Pollard, W. E. 1986. *Bayesian Statistics for Evaluation Research: An Introduction*. Newbury Park, CA: Sage.

Propp, J. G., and D. B. Wilson. 1996. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms* 9: 223–252.

Raftery, A. E., and S. M. Lewis. 1992. How many iterations in the Gibbs sampler? In Vol. 4 of *Bayesian Statistics: Proceedings of the Fourth Valencia International Meeting, April 15–20, 1991*, ed. J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, 763–773. Oxford: Clarendon Press.

Rios Insua, D. 1990. *Sensitivity Analysis in Multi-Objective Decision Making*. New York: Springer.

Robert, C. P., and G. Casella. 2004. *Monte Carlo Statistical Methods*. 2nd ed. New York: Springer.

Roberts, G. O., and J. S. Rosenthal. 2007. Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. *Journal of Applied Probability* 44: 458–475.

——. 2009. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics* 18: 349–367.

Schwarz, G. 1978. Estimating the dimension of a model. *Annals of Statistics* 6: 461–464.

Tanner, M. A. 1996. *Tools for Statistical Inference: Methods for the Exploration of Posterior Distributions and Likelihood Functions*. 3rd ed. New York: Springer.

Tanner, M. A., and W. H. Wong. 1987. The calculation of posterior distributions by data augmentation (with discussion). *Journal of the American Statistical Association* 82: 528–550.

Thompson, J. 2014. *Bayesian Analysis with Stata*. College Station, TX: Stata Press.

Thompson, S. K. 2012. *Sampling*. 3rd ed. Hoboken, NJ: Wiley.

Tierney, L. 1994. Markov chains for exploring posterior distributions. *Annals of Statistics* 22: 1701–1728.

von Neumann, J. 1951. Various techniques used in connection with random digits. Monte Carlo methods. *Journal of Research of the National Bureau of Standards* 12: 36–38.

West, M., and J. Harrison. 1997. *Bayesian Forecasting and Dynamic Models*. 2nd ed. New York: Springer.

Wolpert, R. L., and K. Ickstadt. 1998. Poisson/gamma random field models for spatial statistics. *Biometrika* 85: 251–267.

Yu, B., and P. Mykland. 1998. Looking at Markov samplers through cusum path plots: A simple diagnostic idea. *Statistics and Computing* 8: 275–286.

Zellner, A. 1997. *Bayesian Analysis in Econometrics and Statistics: The Zellner View and Papers*. Northampton, MA: Edward Elgar.

Zellner, A., and C.-K. Min. 1995. Gibbs sampler convergence criteria. *Journal of the American Statistical Association* 90: 921–927.

Also see

[BAYES] **bayes** — Introduction to commands for Bayesian analysis

[BAYES] **Glossary**

Title

bayes — Introduction to commands for Bayesian analysis

Description

This entry describes commands to perform Bayesian analysis. Bayesian analysis is a statistical procedure that answers research questions by expressing uncertainty about unknown parameters using probabilities. It is based on the fundamental assumption that not only the outcome of interest but also all the unknown parameters in a statistical model are essentially random and are subject to prior beliefs.

Estimation

`bayesmh`	Bayesian regression using MH
`bayesmh evaluators`	User-written Bayesian models using MH

Convergence tests and graphical summaries

`bayesgraph`	Graphical summaries

Postestimation statistics

`bayesstats ess`	Effective sample sizes and related statistics
`bayesstats summary`	Bayesian summary statistics
`bayesstats ic`	Bayesian information criteria and Bayes factors

Hypothesis testing

`bayestest model`	Hypothesis testing using model posterior probabilities
`bayestest interval`	Interval hypothesis testing

Remarks and examples

This entry describes commands to perform Bayesian analysis. See [BAYES] **intro** for an introduction to the topic of Bayesian analysis.

The `bayesmh` command is the main command of the Bayesian suite of commands. It fits a variety of Bayesian regression models and estimates parameters using an adaptive MH Markov chain Monte Carlo (MCMC) method. You can choose from a variety of supported Bayesian models by specifying the `likelihood()` and `prior()` options. Or you can program your own Bayesian models by supplying a program evaluator for the posterior distributions of model parameters in the `evaluator()` option; see [BAYES] **bayesmh evaluators** for details.

After estimation, you can use bayesgraph to check convergence of MCMC visually. You can also use bayesstats ess to compute effective sample sizes and related statistics for model parameters and functions of model parameters to assess the efficiency of the sampling algorithm and autocorrelation in the obtained MCMC sample. Once convergence is established, you can use bayesstats summary to obtain Bayesian summaries such as posterior means and standard deviations of model parameters and functions of model parameters and bayesstats ic to compute Bayesian information criteria and Bayes factors for models. You can use bayestest model to test hypotheses by comparing posterior probabilities of models. You can also use bayestest interval to test interval hypotheses about parameters and functions of parameters.

Below we provide an overview example demonstrating the Bayesian suite of commands. For more examples, see *Remarks and examples* in [BAYES] **bayesmh**.

Overview example

Consider an example from Kuehl (2000, 551) about the effects of exercise on oxygen uptake. The research objective is to compare the impact of the two exercise programs—12 weeks of step aerobic training and 12 weeks of outdoor running on flat terrain—on maximal oxygen uptake. Twelve healthy men were randomly assigned to one of the two groups, the "aerobic" group or the "running" group. Their changes in maximal ventilation (liters/minute) of oxygen for the 12-week period were recorded.

oxygen.dta contains 12 observations of changes in maximal ventilation of oxygen, recorded in variable change, from two groups, recorded in variable group. Additionally, ages of subjects are recorded in variable age, and an interaction between age and group is stored in variable interaction.

```
. use http://www.stata-press.com/data/r14/oxygen
(Oxygen Uptake Data)
. describe
Contains data from http://www.stata-press.com/data/r14/oxygen.dta
  obs:            12                          Oxygen Uptake Data
 vars:             4                          20 Jan 2015 15:56
 size:            84                          (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
change          float   %9.0g                 Change in maximal oxygen uptake
                                                (liters/minute)
group           byte    %8.0g      grouplab   Exercise group (0: Running, 1:
                                                Aerobic)
age             byte    %8.0g                 Age (years)
ageXgr          byte    %9.0g                 Interaction between age and group
-------------------------------------------------------------------------------
Sorted by:
```

Kuehl (2000) uses analysis of covariance to analyze these data. We use linear regression instead,

$$\texttt{change} = \beta_0 + \beta_{\text{group}}\texttt{group} + \beta_{\text{age}}\texttt{age} + \epsilon$$

where ϵ is a random error with zero mean and variance σ^2. Also see Hoff (2009) for Bayesian analysis of these data.

▷ Example 1: OLS

Let's fit OLS regression to our data first.

```
. regress change group age
```

Source	SS	df	MS	Number of obs	=	12
				F(2, 9)	=	41.42
Model	647.874893	2	323.937446	Prob > F	=	0.0000
Residual	70.388768	9	7.82097423	R-squared	=	0.9020
				Adj R-squared	=	0.8802
Total	718.263661	11	65.2966964	Root MSE	=	2.7966

change	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
group	5.442621	1.796453	3.03	0.014	1.378763	9.506479
age	1.885892	.295335	6.39	0.000	1.217798	2.553986
_cons	-46.4565	6.936531	-6.70	0.000	-62.14803	-30.76498

From the table, both `group` and `age` are significant predictors of the outcome in this model.

For example, we reject the hypothesis of H_0: $\beta_{\text{group}} = 0$ at a 5% level based on the p-value of 0.014. The actual interpretation of the reported p-value is that if we repeat the same experiment and use the same testing procedure many times, then given our null hypothesis of no effect of group, we will observe the result (test statistic) as extreme or more extreme than the one observed in this sample (`t` $= 3.03$) only 1.4% of the times. The p-value cannot be interpreted as a probability of the null hypothesis, which is a common misinterpretation. In fact, it answers the question of how likely our data are, given that the null hypothesis is true, and not how likely the null hypothesis is, given our data. The latter question can be answered using Bayesian hypothesis testing, which we demonstrate in example 8.

Confidence intervals are popular alternatives to p-values that eliminate some of the p-value shortcomings. For example, the 95% confidence interval for the coefficient for `group` is [1.38, 9.51] and does not contain the value of 0, so we consider `group` to be a significant predictor of `change`. The interpretation of a 95% confidence interval is that if we repeat the same experiment many times and compute confidence intervals for each experiment, then 95% of those intervals will contain the true value of the parameter. Thus we cannot conclude that the true coefficient for `group` lies between 1.38 and 9.51 with a probability of 0.95—a common misinterpretation of a confidence interval. This probability is either 0 or 1, and we do not know which for any particular confidence interval. All we know is that [1.38, 9.51] is a plausible range for the true value of the coefficient for `group`. Intervals that can actually be interpreted as probabilistic ranges for a parameter of interest may be constructed within the Bayesian paradigm; see example 8.

◁

▷ Example 2: Bayesian normal linear regression with noninformative prior

In example 1, we stated that frequentist methods cannot provide probabilistic summaries for the parameters of interest. This is because in frequentist statistics, parameters are viewed as unknown but fixed quantities. The only random quantity in a frequentist model is an outcome of interest. Bayesian statistics, on the other hand, in addition to the outcome of interest, also treats all model parameters as random quantities. This is what sets Bayesian statistics apart from frequentist statistics and enables one to make probability statements about the likely values of parameters and to assign probabilities to hypotheses of interest.

Bayesian statistics focuses on the estimation of various aspects of the posterior distribution of a parameter of interest, an initial or a prior distribution that has been updated with information about a parameter contained in the observed data. A posterior distribution is thus described by the prior distribution of a parameter and the likelihood function of the data given the parameter.

Let's now fit a Bayesian linear regression to `oxygen.dta`. To fit a Bayesian parametric model, we need to specify the likelihood function or the distribution of the data and prior distributions for all model parameters. Our Bayesian linear model has four parameters: three regression coefficients and the variance of the data. We assume a normal distribution for our outcome, `change`, and start with a noninformative Jeffreys prior for the parameters. Under the Jeffreys prior, the joint prior distribution of the coefficients and the variance is proportional to the inverse of the variance.

We can write our model as follows,

$$\texttt{change} \sim N(X\boldsymbol{\beta}, \sigma^2)$$
$$(\boldsymbol{\beta}, \sigma^2) \sim \frac{1}{\sigma^2}$$

where X is our design matrix, and $\boldsymbol{\beta} = (\beta_0, \beta_{\text{group}}, \beta_{\text{age}})'$, which is a vector of coefficients.

We use the `bayesmh` command to fit our Bayesian model. Let's consider the specification of the model first.

```
bayesmh change group age, likelihood(normal({var}))    ///
    prior({change:}, flat) prior({var}, jeffreys)
```

The specification of the regression function in `bayesmh` is the same as in any other Stata regression command—the name of the dependent variable follows the command, and the covariates of interest are specified next. Likelihood or outcome distribution is specified in the `likelihood()` option, and prior distributions are specified in the `prior()` options, which are repeated options.

All model parameters must be specified in curly braces, `{}`. `bayesmh` automatically creates parameters associated with the regression function—regression coefficients—but it is your responsibility to define the remaining model parameters. In our example, the only parameter we need to define is the variance parameter, which we define as `{var}`. The three regression coefficients `{change:group}`, `{change:age}`, and `{change:_cons}` are automatically created by `bayesmh`.

The last step is to specify the likelihood and the prior distributions. `bayesmh` provides several different built-in distributions for the likelihood and priors. If a certain distribution is not available or you have a particularly complicated Bayesian model, you may consider writing your own evaluator for the posterior distribution; see [BAYES] **bayesmh evaluators** for details. In our example, we specify distribution `normal({var})` in option `likelihood()` to request the likelihood function of the normal model with the variance parameter `{var}`. This specification together with the regression specification defines the likelihood model for our outcome `change`. We assign the `flat` prior, a prior with a density of 1, to all regression coefficients with `prior({change:}, flat)`, where `{change:}` is a shortcut for referring to all parameters with equation name `change`, our regression coefficients. Finally, we specify prior `jeffreys` for the variance parameter `{var}` to request the density $1/\sigma^2$.

Let's now run our command. `bayesmh` uses MCMC sampling, specifically, an adaptive random-walk MH MCMC method, to estimate marginal posterior distributions of parameters. Because `bayesmh` is using an MCMC method, which is stochastic, we must specify a random-number seed for reproducibility of our results. For consistency and simplicity, we use the same random seed of 14 in all of our examples throughout the manual.

```
. set seed 14
. bayesmh change group age, likelihood(normal({var}))
> prior({change:}, flat) prior({var}, jeffreys)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  change ~ normal(xb_change,{var})
Priors:
  {change:group age _cons} ~ 1 (flat)                                    (1)
                     {var} ~ jeffreys
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_change.
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         12
                                                Acceptance rate  =      .1371
                                                Efficiency:  min =     .02687
                                                             avg =     .03765
Log marginal likelihood = -24.703776                         max =     .05724
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
change						
group	5.429677	2.007889	.083928	5.533821	1.157584	9.249262
age	1.8873	.3514983	.019534	1.887856	1.184714	2.567883
_cons	-46.49866	8.32077	.450432	-46.8483	-62.48236	-30.22105
var	10.27946	5.541467	.338079	9.023905	3.980325	25.43771

First, `bayesmh` provides a summary for the specified model. It is particularly useful for complicated models with many parameters and hyperparameters. In fact, we recommend that you first specify the `dryrun` option, which provides only the summary of the model without estimation, to verify the specification of your model and then proceed with estimation. You can then use the `nomodelsummary` option during estimation to suppress the model summary, which may be rather long.

Next, `bayesmh` provides a header with various model summaries on the right-hand side. It reports the total number of MCMC iterations, 12,500, including the default 2,500 burn-in iterations, which are discarded from the analysis MCMC sample, and the number of iterations retained in the MCMC sample, or MCMC sample size, which is 10,000 by default. These default values should be viewed as initial estimates and further adjusted for the problem at hand to ensure convergence of the MCMC; see example 5.

An acceptance rate and a summary of the parameter-specific efficiencies are also part of the output header. An acceptance rate specifies the proportion of proposed parameter values that was accepted by the algorithm. An acceptance rate of 0.14 in our example means that 14% out of 10,000 proposal parameter values were accepted by the algorithm. For the MH algorithm, this number rarely exceeds 50% and is typically below 30%. A low acceptance rate (for example, below 10%) may indicate convergence problems. In our example, the acceptance rate is a bit low, so we may need to investigate this further. In general, MH tends to have lower efficiencies compared with other MCMC methods. For example, efficiencies of 10% and higher are considered good. Efficiencies below 1% may be a source of concern. The efficiencies are somewhat low in our example, so we may consider tuning our MCMC sampler; see *Improving efficiency of the MH algorithm—blocking of parameters*.

Finally, `bayesmh` reports a table with a summary of the results. The `Mean` column reports the estimates of posterior means, which are means of the marginal posterior distributions of the parameters. The posterior mean estimates are pretty close to the OLS estimates obtained in example 1. This is expected, provided MCMC converged, because we used a noninformative prior. That is, we did not provide any additional information about parameters beyond that contained in the data.

The next column reports estimates of posterior standard deviations, which are standard deviations of the marginal posterior distribution. These values describe the variability in the posterior distribution of the parameter and are comparable to our OLS standard errors.

The precision of the posterior mean estimates is described by their Monte Carlo standard errors. These numbers should be small, relative to the scales of the parameters. Increasing the MCMC sample size should decrease these numbers.

The `Median` column provides estimates of the median of the posterior distribution and can be used to assess the symmetries of the posterior distribution. At a quick glance, the estimates of posterior means and medians are pretty close for the regression coefficients, so we suspect that their posterior distributions may be symmetric.

The last two columns provide credible intervals for the parameters. Unlike confidence intervals, as discussed in example 1, these intervals have a straightforward probabilistic interpretation. For example, the probability that the coefficient for `group` is between 1.16 and 9.25 is about 0.95. The lower bound of the interval is greater than 0, so we conclude that there is an effect of the exercise program on the change in oxygen uptake. We can also use Bayesian hypothesis testing to test effects of parameters; see example 8.

Before any interpretation of the results, however, it is important to verify the convergence of MCMC; see example 5.

◁

▷ Example 3: Bayesian linear regression with informative prior

In example 2, we considered a noninformative prior for the model parameters. The strength (as well as the weakness) of Bayesian modeling is specifying an informative prior distribution, which may improve results. The strength is that if we have reliable prior knowledge about the distribution of a parameter, incorporating this in our model will improve results and potentially make certain analysis that would not be possible to perform in the frequentist domain feasible. The weakness is that a strong incorrect prior may lead to results that are not supported by the observed data. As with any modeling task, Bayesian or frequentist, a substantive research of the process generating the data and its parameters will be necessary for you to find appropriate models.

Let's consider an informative conjugate prior distribution for our normal regression model.

$$(\boldsymbol{\beta}|\sigma^2) \sim \text{i.i.d. } N(0, \sigma^2)$$
$$\sigma^2 \sim \text{InvGamma}(2.5, 2.5)$$

Here, for simplicity, all coefficients are assumed to be independently and identically distributed as normal with zero mean and variance σ^2, and the variance parameter is distributed according to the above inverse gamma distribution. In practice, a better prior would be to allow each parameter to have a different variance, at least for parameters with different scales.

Let's fit this model using `bayesmh`. Following the model above, we specify the `normal(0,{var})` prior for the coefficients and the `igamma(2.5,2.5)` prior for the variance.

```
. set seed 14
. bayesmh change group age, likelihood(normal({var}))
> prior({change:}, normal(0, {var}))
> prior({var}, igamma(2.5, 2.5))
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  change ~ normal(xb_change,{var})
Priors:
  {change:group age _cons} ~ normal(0,{var})                              (1)
                     {var} ~ igamma(2.5,2.5)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_change.
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         12
                                                Acceptance rate  =      .1984
                                                Efficiency:  min =     .03732
                                                             avg =     .04997
Log marginal likelihood = -49.744054                         max =     .06264
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
change						
group	6.510807	2.812828	.129931	6.50829	.9605561	12.23164
age	.2710499	.2167863	.009413	.2657002	-.1556194	.7173697
_cons	-6.838302	4.780343	.191005	-6.683556	-16.53356	2.495631
var	28.83438	10.53573	.545382	26.81462	14.75695	54.1965

The results from this model are substantially different from the results we obtained in example 2. Considering that we used this simple prior for demonstration purposes only and did not use any external information about model parameters based on prior studies, we would be reluctant to trust the results from this model.

◁

▷ Example 4: Bayesian normal linear regression with multivariate prior

Continuing with informative priors, we will consider Zellner's g-prior (Zellner 1986), which is one of the more commonly used priors for the regression coefficients in a normal linear regression. Hoff (2009) provides more details about this example, and he includes the interaction between age and group as in example 7. Here we concentrate on demonstrating how to fit our model using `bayesmh`.

The mathematical formulation of the priors is the following,

$$(\boldsymbol{\beta}|\sigma^2) \sim \text{MVN}(0, g\sigma^2(X'X)^{-1})$$
$$\sigma^2 \sim \text{InvGamma}(\nu_0/2, \nu_0\sigma_0^2/2)$$

where g reflects prior sample size, ν_0 is the prior degrees of freedom for the inverse gamma distribution, and σ_0^2 is the prior variance for the inverse gamma distribution. This prior incorporates dependencies between coefficients. We use values of the parameters similar to those in Hoff (2009): $g = 12$, $\nu_0 = 1$, and $\sigma_0^2 = 8$.

bayesmh provides the zellnersg0() prior to accommodate the above prior. The first argument is the dimension of the distribution, which is 3 in our example, the second argument is the prior degrees of freedom, which is 12 in our example, and the last argument is the variance parameter, which is {var} in our example. The mean is assumed to be a zero vector of the corresponding dimension. (You can use zellnersg() if you want to specify a nonzero mean vector; see [BAYES] **bayesmh**.)

```
. set seed 14
. bayesmh change group age, likelihood(normal({var}))
> prior({change:}, zellnersg0(3,12,{var}))
> prior({var}, igamma(0.5, 4))
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  change ~ normal(xb_change,{var})
Priors:
  {change:group age _cons} ~ zellnersg(3,12,0,{var})                    (1)
                     {var} ~ igamma(0.5,4)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_change.
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         12
                                                Acceptance rate  =     .06169
                                                Efficiency:  min =      .0165
                                                             avg =     .02018
Log marginal likelihood = -35.356501                         max =     .02159
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
change						
group	4.988881	2.260571	.153837	4.919351	.7793098	9.775568
age	1.713159	.3545698	.024216	1.695671	1.053206	2.458556
_cons	-42.31891	8.239571	.565879	-41.45385	-59.30145	-27.83421
var	12.29575	6.570879	.511475	10.3609	5.636195	30.93576

These results are more in agreement with results from example 2 than with results of example 3, but our acceptance rate and efficiencies are low and require further investigation.

◁

❑ Technical note

We can reproduce what zellnersg0() does above manually. First, we must compute $(X'X)^{-1}$. We can use Stata's matrix functions to do that.

```
. matrix accum xTx = group age
(obs=12)
. matrix S = syminv(xTx)
```

We now specify the desired multivariate normal prior for the coefficients, mvnormal0(3,12*{var}*S). The first argument of mvnormal0() specifies the dimension of the distribution, and the second argument specifies the variance–covariance matrix. A mean of zero is assumed for all dimensions. One interesting feature of this specification is that the variance–covariance matrix is specified as a function of {var}.

```
. set seed 14
. bayesmh change group age, likelihood(normal({var}))
> prior({change:}, mvnormal0(3,12*{var}*S))
> prior({var}, igamma(0.5, 4))
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  change ~ normal(xb_change,{var})
Priors:
  {change:group age _cons} ~ mvnormal(3,0,12*{var}*S)                    (1)
                     {var} ~ igamma(0.5,4)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_change.
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         12
                                                Acceptance rate  =     .06169
                                                Efficiency:  min =      .0165
                                                             avg =     .02018
Log marginal likelihood = -35.356501                         max =     .02159
------------------------------------------------------------------------------
             |                                                Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
change       |
       group |  4.988881   2.260571   .153837   4.919351   .7793098   9.775568
         age |  1.713159   .3545698   .024216   1.695671   1.053206   2.458556
       _cons | -42.31891   8.239571   .565879  -41.45385  -59.30145  -27.83421
-------------+----------------------------------------------------------------
         var |  12.29575   6.570879   .511475    10.3609   5.636195   30.93576
------------------------------------------------------------------------------
```

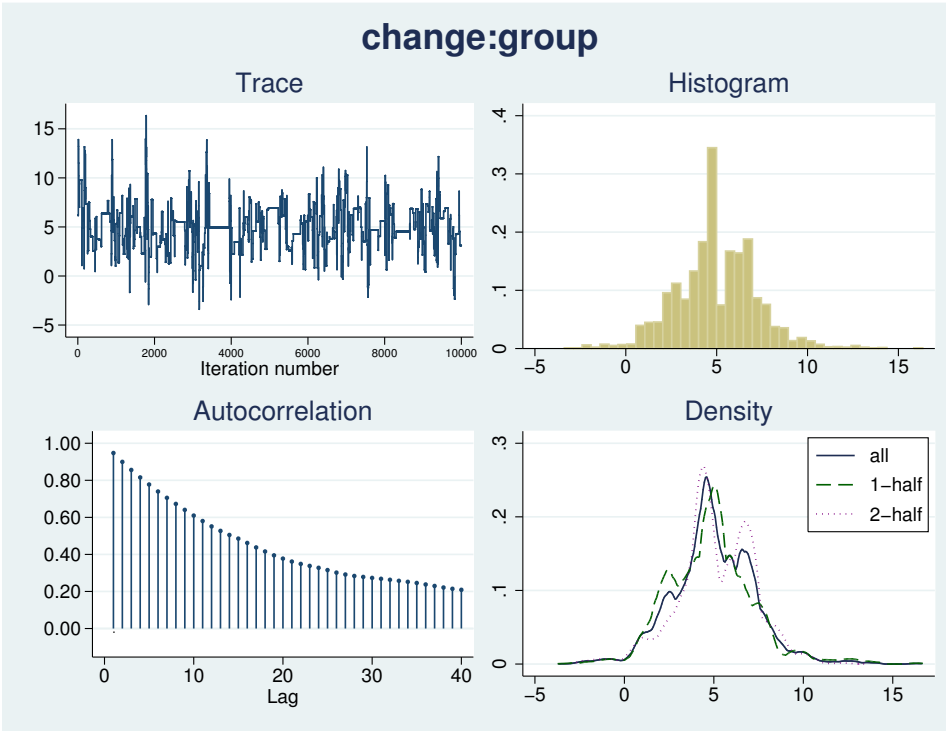
❑

▷ Example 5: Checking convergence

We can use the bayesgraph command to visually check convergence of MCMC of parameter estimates. bayesgraph provides a variety of graphs. For several commonly used visual diagnostics displayed in a compact form, use bayesgraph diagnostics.

For example, we can look at graphical diagnostics for the coefficient for group.

```
. bayesgraph diagnostics {change:group}
```



The displayed diagnostics include a trace plot, an autocorrelation plot, a histogram, and a kernel density estimate overlaid with densities estimated using the first and the second halves of the MCMC sample. Both the trace plot and the autocorrelation plot demonstrate high autocorrelation. The shape of the histogram is not unimodal. We definitely have some convergence issues in this example.

Similarly, we can look at diagnostics for other model parameters. To see all graphs at once, type

```
bayesgraph diagnostics _all
```

Other useful summaries are effective sample sizes and statistics related to them. These can be obtained by using the bayesstats ess command.

```
. bayesstats ess
Efficiency summaries     MCMC sample size =     10,000
```

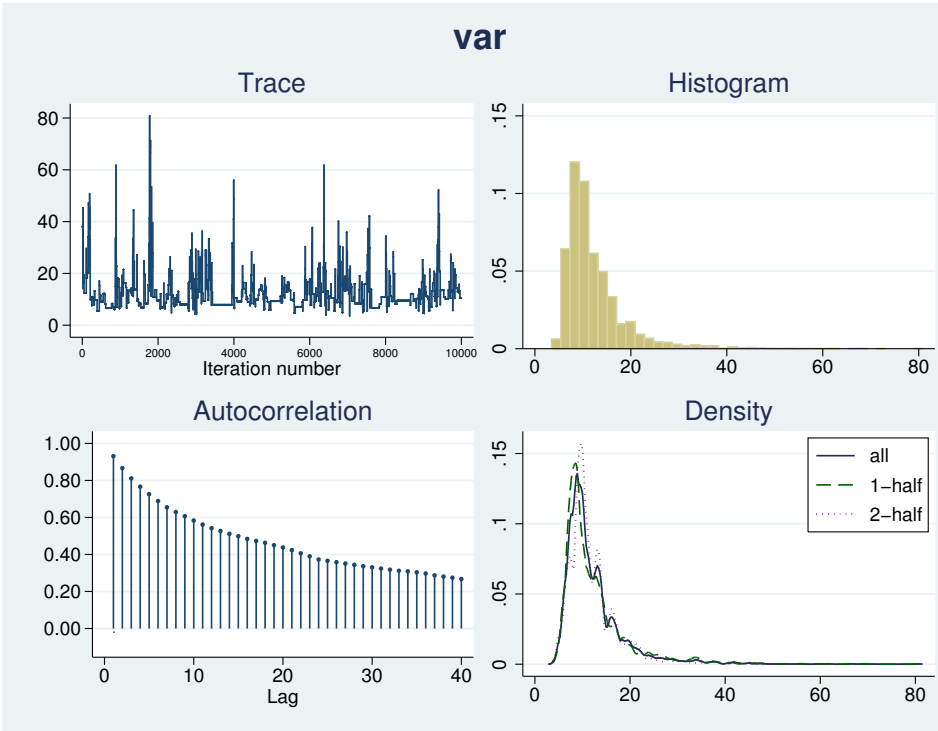
	ESS	Corr. time	Efficiency
change			
group	215.93	46.31	0.0216
age	214.39	46.64	0.0214
_cons	212.01	47.17	0.0212
var	165.04	60.59	0.0165

The closer ESS estimates are to the MCMC sample size, the less correlated the MCMC sample is, and the more precise our estimates of parameters are. Do not expect to see values close to the MCMC sample size with the MH algorithm, but values below 1% of the MCMC sample size are certainly red flags. In our example, ESS for `{var}` is somewhat low, so we may need to look into improving its sampling efficiency. For example, blocking on `{var}` should improve the efficiency for the variance; see *Improving efficiency of the MH algorithm—blocking of parameters*. It is usually a good idea to sample regression coefficients and the variance in two separate blocks.

Correlation times may be viewed as estimates of autocorrelation lags in the MCMC samples. For example, correlation times of the coefficients range between 46 and 47, and the correlation time for the variance parameter is higher, 61. Consequently, the efficiency for the variance is lower than for the regression coefficients. More investigation of the MCMC for `{var}` is needed.

Indeed, the MCMC for the variance has very poor mixing and very high autocorrelation.

```
. bayesgraph diagnostics {var}
```



One remedy is to update the variance parameter separately from the regression coefficients by putting the variance parameter in a separate block; see *Improving efficiency of the MH algorithm—blocking of parameters* for details about this procedure. In `bayesmh`, this can be done by specifying the `block()` option.

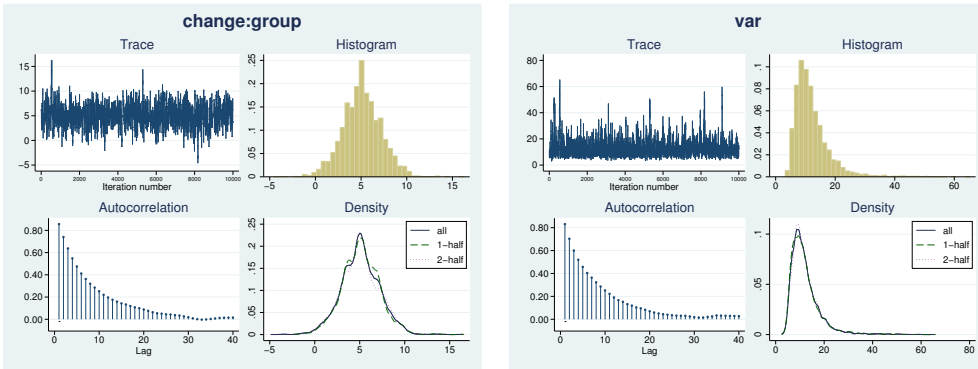
```
. set seed 14
. bayesmh change group age, likelihood(normal({var}))
> prior({change:}, zellnersg0(3,12,{var}))
> prior({var}, igamma(0.5, 4)) block({var})
> saving(agegroup_simdata)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  change ~ normal(xb_change,{var})
Priors:
  {change:group age _cons} ~ zellnersg(3,12,0,{var})                    (1)
                     {var} ~ igamma(0.5,4)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_change.
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         12
                                                Acceptance rate  =      .3232
                                                Efficiency:  min =     .06694
                                                             avg =      .1056
Log marginal likelihood = -35.460606                         max =      .1443
------------------------------------------------------------------------------
             |                                                Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
change       |
       group |   5.080653   2.110911   .080507   5.039834   .8564619   9.399672
         age |   1.748516   .3347172   .008875   1.753897   1.128348   2.400989
       _cons |  -43.12425   7.865979   .207051   -43.2883  -58.64107  -27.79122
-------------+----------------------------------------------------------------
         var |   12.09916   5.971454   .230798   10.67555   5.375774   27.32451
------------------------------------------------------------------------------
file agegroup_simdata.dta saved
. estimates store agegroup
```

Our acceptance rate and efficiencies are now higher.

In this example, we also used `estimates store agegroup` to store current estimation results as `agegroup` for future use. To use `estimates store` after `bayesmh`, we had to specify the `saving()` option with `bayesmh` to save the `bayesmh` simulation results to a permanent Stata dataset; see *Storing estimation results after bayesmh*.

The MCMC chains are now mixing much better. We may consider increasing the default MCMC sample size to achieve even lower autocorrelation.

```
. bayesgraph diagnostics {change:group} {var}
```



Example 6: Postestimation summaries

We can use the `bayesstats summary` command to compute postestimation summaries for model parameters and functions of model parameters. For example, we can compute an estimate of the standardized coefficient for `change`, which is $\widehat{\beta}_{\text{group}} \times \sigma_x/\sigma_y$, where σ_x and σ_y are sample standard deviations of `group` and `change`, respectively.

We use `summarize` (see [R] **summarize**) to compute sample standard deviations and store them in respective scalars.

```
. summarize group
```

Variable	Obs	Mean	Std. Dev.	Min	Max
group	12	.5	.522233	0	1

```
. scalar sd_x = r(sd)
. summarize change
```

Variable	Obs	Mean	Std. Dev.	Min	Max
change	12	2.469167	8.080637	-10.74	17.05

```
. scalar sd_y = r(sd)
```

The standardized coefficient is an expression of the model parameter `{change:group}`, so we specify it in parentheses.

```
. bayesstats summary (group_std:{change:group}*sd_x/sd_y)
Posterior summary statistics                     MCMC sample size =    10,000
   group_std : {change:group}*sd_x/sd_y
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
group_std	.3283509	.1364233	.005203	.3257128	.0553512	.6074792

The posterior mean estimate of the standardized `group` coefficient is 0.33 with a 95% credible interval of [0.055, 0.61].

▷ Example 7: Model comparison

As we can with frequentist analysis, we can use various information criteria to compare different models. There is great flexibility in which model can be compared: you can compare models with different distributions for the outcome, you can compare models with different priors, you can compare models with different forms for the regression function, and more. The only requirement is that the same data are used to fit the models. Comparisons using Bayes factors additionally require that parameters be sampled from the complete posterior distribution, which includes the normalizing constant.

Let's compare our reduced model with the full model including an interaction term. We again use a multivariate Zellners-g prior for the coefficients and an inverse gamma prior for the variance. We use the same values as in example 4 for prior parameters. (We use the interaction variable in this example for notational simplicity. We could have used the factor-variable notation `c.age#i.group` to include this interaction directly in our model; see **[U] 11.4.3 Factor variables**.)

```
. set seed 14
. bayesmh change group age ageXgr, likelihood(normal({var}))
> prior({change:}, zellnersg0(4,12,{var}))
> prior({var}, igamma(0.5, 4)) block({var})
> saving(full_simdata)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  change ~ normal(xb_change,{var})
Priors:
  {change:group age ageXgr _cons} ~ zellnersg(4,12,0,{var})              (1)
                          {var} ~ igamma(0.5,4)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_change.
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         12
                                                Acceptance rate  =      .3113
                                                Efficiency:  min =      .0562
                                                             avg =     .06425
Log marginal likelihood = -36.738363                         max =     .08478
------------------------------------------------------------------------------
             |                                                Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
change       |
       group |   11.94079   16.74992   .706542   12.13983  -22.31056   45.11963
         age |   1.939266   .5802772   .023359   1.938756   .7998007   3.091072
      ageXgr |  -.2838718   .6985226   .028732   -.285647  -1.671354   1.159183
       _cons |  -47.57742    13.4779    .55275  -47.44761  -74.64672  -20.78989
-------------+----------------------------------------------------------------
         var |   11.72886    5.08428   .174612   10.68098   5.302265   24.89543
------------------------------------------------------------------------------
file full_simdata.dta saved
. estimates store full
```

We can use the bayesstats ic command to compare the models. We list the names of the corresponding estimation results following the command name.

```
. bayesstats ic full agegroup
Bayesian information criteria
```

	DIC	log(ML)	log(BF)
full	65.03326	-36.73836	.
agegroup	63.5884	-35.46061	1.277756

```
Note: Marginal likelihood (ML) is computed
      using Laplace-Metropolis approximation.
```

The smaller that DIC is and the larger that log(ML) is, the better. The model without interaction, agegroup, is preferred according to these statistics. The log Bayes-factor for the agegroup model relative to the full model is 1.28. Kass and Raftery (1995) provide a table of values for Bayes factors; see, for example, *Bayes factors* in [BAYES] **bayesstats ic**. According to their scale, because $2 \times 1.28 = 2.56$ is greater than 2 (slightly), there is some mild evidence that model agegroup is better than model full.

◁

▷ Example 8: Hypothesis testing

Continuing with example 7, we can compute the actual probability associated with each of the models. We can use the bayestest model command to do this.

Similar to bayesstats ic, this command requires the names of estimation results corresponding to the models of interest.

```
. bayestest model full agegroup
Bayesian model tests
```

	log(ML)	P(M)	P(M|y)
full	-36.7384	0.5000	0.2179
agegroup	-35.4606	0.5000	0.7821

```
Note: Marginal likelihood (ML) is computed using
      Laplace-Metropolis approximation.
```

Under the assumption that both models are equally probable a priori, the model without interaction, agegroup, has the probability of 0.78, whereas the full model has the probability of only 0.22. Despite the drastic disparity in the probabilities, according to the results from example 7, model agegroup is only slightly preferable to model full. To have stronger evidence against full, we would expect to see higher probabilities (above 0.9) for agegroup.

We may be interested in testing an interval hypothesis about the parameter of interest. For example, for a model without interaction, let's compute the probability that the coefficient for group is between 4 and 8. We use estimates restore (see [R] **estimates store**) to load the results of the agegroup model back into memory.

```
. estimates restore agegroup
(results agegroup are active now)
. bayestest interval {change:group}, lower(4) upper(8)
Interval tests     MCMC sample size =    10,000
       prob1 : 4 < {change:group} < 8
```

	Mean	Std. Dev.	MCSE
prob1	.6159	0.48641	.0155788

The estimated probability or, technically, its posterior mean estimate is 0.62 with a standard deviation of 0.49 and Monte Carlo standard errors of 0.016.

◁

▷ Example 9: Erasing simulation datasets

After you are done with your analysis, remember to erase any simulation datasets that you created using `bayesmh` and no longer need. If you want to save your estimation results to disk for future reference, use `estimates save`; see [R] **estimates save**.

We are done with our analysis, and we do not need the datasets for future reference, so we remove both simulation files we created using `bayesmh`.

```
. erase agegroup_simdata.dta
. erase full_simdata.dta
```

◁

Acknowledgments

We thank John Thompson of the Department of Health Sciences at the University of Leicester, UK, and author of *Bayesian Analysis with Stata*, and Matthew J. Baker of Hunter College and the Graduate Center, CUNY for their software and contributions to Bayesian analysis in Stata.

References

Baker, M. J. 2014. Adaptive Markov chain Monte Carlo sampling and estimation in Mata. *Stata Journal* 14: 623–661.

Hoff, P. D. 2009. *A First Course in Bayesian Statistical Methods*. New York: Springer.

Kass, R. E., and A. E. Raftery. 1995. Bayes factors. *Journal of the American Statistical Association* 90: 773–795.

Kuehl, R. O. 2000. *Design of Experiments: Statistical Principles of Research Design and Analysis*. 2nd ed. Belmont, CA: Duxbury.

Zellner, A. 1986. On assessing prior distributions and Bayesian regression analysis with g-prior distributions. In Vol. 6 of *Bayesian Inference and Decision Techniques: Essays in Honor of Bruno De Finetti (Studies in Bayesian Econometrics and Statistics)*, ed. P. K. Goel and A. Zellner, 233–343. Amsterdam: North-Holland.

Also see

[BAYES] **intro** — Introduction to Bayesian analysis

[BAYES] **bayesmh** — Bayesian regression using Metropolis–Hastings algorithm

[BAYES] **bayesmh postestimation** — Postestimation tools for bayesmh

[BAYES] **Glossary**

Title

bayesmh — Bayesian regression using Metropolis–Hastings algorithm

Description

`bayesmh` fits a variety of Bayesian models using Metropolis–Hastings (MH) algorithm. It provides various likelihood models and prior distributions for you to choose from. Likelihood models include univariate normal linear and nonlinear regressions, multivariate normal linear and nonlinear regressions, generalized linear models such as logit and Poisson regressions, and multiple-equations linear models. Prior distributions include continuous distributions such as uniform, Jeffreys, normal, gamma, multivariate normal, and Wishart and discrete distributions such as Bernoulli and Poisson. You can also program your own Bayesian models; see [BAYES] **bayesmh evaluators**.

Quick start

Bayesian normal linear regression of `y1` on `x1` with flat priors for coefficient on `x1` and the intercept and with a Jeffreys prior on the variance parameter `{var}`

```
bayesmh y1 x1, likelihood(normal({var}))  ///
     prior({y1: x1 _cons}, flat) prior({var}, jeffreys)
```

Add binary variable `a` using factor-variable notation

```
bayesmh y1 x1 i.a, likelihood(normal({var}))  ///
     prior({y1: x1 i.a _cons}, flat) prior({var}, jeffreys)
```

Same as above

```
bayesmh y1 x1 i.a, likelihood(normal({var}))  ///
     prior({y1:}, flat) prior({var}, jeffreys)
```

Specify a different prior for `a` = 1

```
bayesmh y1 x1 i.a, likelihood(normal({var}))                        ///
     prior({y1:x1  _cons}, flat) prior({y1: 1.a}, normal(0,100))  ///
     prior({var}, jeffreys)
```

Specify a starting value of 1 for parameter `{var}`

```
bayesmh y1 x1 i.a, likelihood(normal({var}))  ///
     prior({y1:}, flat) prior({var}, jeffreys) initial({var} 1)
```

Same as above

```
bayesmh y1 x1 i.a, likelihood(normal({var=1}))  ///
     prior({y1:}, flat) prior({var}, jeffreys)
```

A normal prior with $\mu = 2$ and $\sigma^2 = 0.5$ for the coefficient on `x1`, a normal prior with $\mu = -40$ and $\sigma^2 = 100$ for the intercept, and an inverse-gamma prior with shape parameter of 0.1 and scale parameter of 1 for `{var}`

```
bayesmh y1 x1, likelihood(normal({var})) ///
     prior({y1:x1}, normal(2,.5))        ///
     prior({y1:_cons}, normal(-40,100))  ///
     prior({var}, igamma(0.1,1))
```

Place {var} into a separate block

```
bayesmh y1 x1, likelihood(normal({var}))  ///
     prior({y1:x1}, normal(2,.5))         ///
     prior({y1:_cons}, normal(-40,100))   ///
     prior({var}, igamma(0.1,1)) block({var})
```

Zellner's g prior to allow {y1:x1} and {y1:_cons} to be correlated, specifying 2 dimensions, df $= 30$, $\mu = 2$ for {y1:x1}, $\mu = -40$ for {y1:_cons}, and variance parameter {var}

```
bayesmh y1 x1, likelihood(normal({var}))  ///
     prior({var}, igamma(0.1,1))          ///
     prior({y1:}, zellnersg(2,30,2,-40,{var}))
```

Model for dichotomous dependent variable y2 regressed on x1 with a logit likelihood

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100))
```

As above, and save model results to simdata.dta, and store estimates in memory as m1

```
bayesmh y2 x1, likelihood(logit) prior({y2:},  ///
     normal(0,100)) saving(simdata.dta)
estimates store m1
```

As above, but save the results on replay

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100))
bayesmh, saving(simdata.dta)
estimates store m1
```

Show model summary without performing estimation

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100)) dryrun
```

Fit model without showing model summary

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100))  ///
     nomodelsummary
```

As above, and set the random-number seed for reproducibility

```
set seed 1234
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100))
```

Same as above

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100))  ///
     rseed(1234)
```

Specify 20,000 MCMC samples, and set length of the burn-in period to 5,000

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100))  ///
     mcmcsize(20000) burnin(5000)
```

Specify that only observations $1 + 5k$, for $k = 0, 1, \ldots$, be saved to the MCMC sample

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100))  ///
     thinning(5)
```

Set the maximum number of adaptive iterations of the MCMC procedure to 30, and specify that adaptation of the MCMC procedure be attempted every 25 iterations

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100))  ///
     adaptation(maxiter(30) every(25))
```

Request that a dot be displayed every 100 simulations

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100))  ///
     dots(100)
```

Also request that an iteration number be displayed every 1,000 iterations

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100))  ///
     dots(100, every(1000))
```

Same as above

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100))  ///
     dots
```

Request that the 90% equal-tailed credible interval be displayed

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100))  ///
     clevel(90)
```

Request that the default 95% highest posterior density credible interval be displayed

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100)) hpd
```

Use the batch-means estimator of MCSE with the length of the block of 5

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100))  ///
     batch(5)
```

Multivariate normal regression of `y1` and `y3` on `x1` and `x2`, using normal priors with $\mu = 0$ and $\sigma^2 = 100$ for the regression coefficients and intercepts, an inverse-Wishart prior for the covariance matrix parameter `{S, matrix}` of dimension 2, df = 100, and an identity scale matrix

```
bayesmh y1 y3 = x1 x2, likelihood(mvnormal({S, matrix}))  ///
     prior({y1:} {y3:}, normal(0,100))                    ///
     prior({S, matrix}, iwishart(2,100,I(2)))
```

As above, but use abbreviated declaration for the covariance matrix

```
bayesmh y1 y3 = x1 x2, likelihood(mvnormal({S,m}))  ///
     prior({y1:} {y3:}, normal(0,100))              ///
     prior({S,m}, iwishart(2,100,I(2)))
```

As above, and specify starting values for matrix `{S,m}` using previously defined matrix `W`

```
bayesmh y1 y3 = x1 x2, likelihood(mvnormal({S,m}))  ///
     prior({y1:} {y3:}, normal(0,100))              ///
     prior({S,m}, iwishart(2,100,I(2))) initial({S,m} W)
```

Multivariate normal regression with outcome-specific regressors

```
bayesmh (y1 x1 x2) (y3 x1 x3), likelihood(mvnormal({S,m}))  ///
    prior({y1:} {y3:}, normal(0,100))                       ///
    prior({S,m}, iwishart(2,100,I(2)))
```

Linear multiple-equation model of `y1` on `x1` and of `y3` on `y1`, `x1`, and `x2` with separate variance parameters for each equation

```
bayesmh (y1 x1, likelihood(normal({var1})))      ///
     (y3 y1 x1 x2, likelihood(normal({var2}))),  ///
     prior({y1:} {y3:}, flat)                    ///
     prior({var1}, jeffreys) prior({var2}, jeffreys)
```

Nonlinear model with parameters {a}, {b}, {c}, and {var} specified using a substitutable expression

```
bayesmh y1 = ({a}+{b}*x1^{c}), likelihood(normal({var}))  ///
    prior({a b}, normal(0,100)) prior({c}, normal(0,2))   ///
    prior({var}, igamma(0.1,1))
```

Multivariate nonlinear model with distinct parameters in each equation

```
bayesmh (y1 = ({a1} + {b1}*x1^{c1}))                          ///
    (y3 = ({a2} + {b2}*x1^{c2})), likelihood(mvnormal({S,m}))  ///
    prior({a1 a2 b1 b2}, normal(0,100))                       ///
    prior({c1 c2}, normal(0,2)) prior({S,m}, iwishart(2,100,I(2)))
```

Menu

Statistics > Bayesian analysis > Estimation

Syntax

Univariate linear models

```
bayesmh depvar [indepvars] [if] [in] [weight],
        likelihood(modelspec) prior(priorspec) [options]
```

Multivariate linear models

Multivariate normal linear regression with common regressors

```
bayesmh depvars = [indepvars] [if] [in] [weight],
        likelihood(mvnormal(...)) prior(priorspec) [options]
```

Multivariate normal regression with outcome-specific regressors

```
bayesmh ([eqname1:]depvar1 [indepvars1])
        ([eqname2:]depvar2 [indepvars2]) [...] [if] [in] [weight],
        likelihood(mvnormal(...)) prior(priorspec) [options]
```

Multiple-equation linear models

```
bayesmh (eqspec) [(eqspec)] [...] [if] [in] [weight], prior(priorspec) [options]
```

Nonlinear models

Univariate nonlinear regression

```
bayesmh depvar = (subexpr) [if] [in] [weight],
        likelihood(modelspec) prior(priorspec) [options]
```

Multivariate normal nonlinear regression

```
bayesmh (depvars1 = (subexpr1))
        (depvars2 = (subexpr2)) [...] [if] [in] [weight],
        likelihood(mvnormal(...)) prior(priorspec) [options]
```

The syntax of *eqspec* is

varspec [*if*] [*in*] [*weight*], likelihood(*modelspec*) [noconstant]

The syntax of *varspec* is one of the following:

for single outcome

[*eqname*:]*depvar* [*indepvars*]

for multiple outcomes with common regressors

depvars = [*indepvars*]

for multiple outcomes with outcome-specific regressors

([*eqname1*:]*depvar1* [*indepvars1*]) ([*eqname2*:]*depvar2* [*indepvars2*]) [...]

subexpr, *subexpr1*, *subexpr2*, and so on are substitutable expressions; see *Substitutable expressions* for details.

The syntax of *modelspec* is

model [, *modelopts*]

model	Description
Continuous	
normal(*var*)	normal regression with variance *var*
lognormal(*var*)	lognormal regression with variance *var*
lnormal(*var*)	synonym for lognormal()
exponential	exponential regression
mvnormal(*Sigma*)	multivariate normal regression with covariance matrix *Sigma*
Discrete	
probit	probit regression
logit	logistic regression
logistic	logistic regression; synonym for logit
binlogit(*n*)	binomial regression with logit link
oprobit	ordered probit regression
ologit	ordered logistic regression
poisson	Poisson regression
Generic	
llf(*subexpr*)	substitutable expression for observation-level log-likelihood function

A distribution argument is a number for scalar arguments such as *var*; a variable name, *varname* (except for matrix arguments); a matrix for matrix arguments such as *Sigma*; a model parameter, *paramspec*; an expression, *expr*; or a substitutable expression, *subexpr*. See *Specifying arguments of likelihood models and prior distributions*.

modelopts	Description
offset(*varname*$_o$)	include *varname*$_o$ in model with coefficient constrained to 1; not allowed with normal() and mvnormal()
exposure(*varname*$_e$)	include ln(*varname*$_e$) in model with coefficient constrained to 1; allowed only with poisson
noglmtransform	do not apply GLM-type transformation to the linear predictor or nonlinear specification; allowed only with exponential, poisson, or binlogit()

The syntax of *priorspec* is

paramref , *priordist*

where the simplest specification of *paramref* is

paramspec [*paramspec* [...]]

Also see *Referring to model parameters* for other specifications.

The syntax of *paramspec* is

{ [*eqname*:]*param*[, matrix] }

where the parameter label *eqname* and parameter name *param* are valid Stata names. Model parameters are either scalars such as {var}, {mean}, and {shape:alpha}, or matrices such as {Sigma, matrix} and {Scale:V, matrix}. For scalar parameters, you can use {param=#} to specify an initial value. For example, you can specify, {var=1}, {mean=1.267}, or {shape:alpha=3}.

priordist	Description
Univariate continuous	
`normal(`*mu*`,`*var*`)`	normal with mean *mu* and variance *var*
`lognormal(`*mu*`,`*var*`)`	lognormal with mean *mu* and variance *var*
`lnormal(`*mu*`,`*var*`)`	synonym for `lognormal()`
`uniform(`*a*`,`*b*`)`	uniform on (*a*, *b*)
`gamma(`*alpha*`,`*beta*`)`	gamma with shape *alpha* and scale *beta*
`igamma(`*alpha*`,`*beta*`)`	inverse gamma with shape *alpha* and scale *beta*
`exponential(`*beta*`)`	exponential with scale *beta*
`beta(`*a*`,`*b*`)`	beta with shape parameters *a* and *b*
`chi2(`*df*`)`	central χ^2 with degrees of freedom *df*
`jeffreys`	Jeffreys prior for variance of a normal distribution
Multivariate continuous	
`mvnormal(`*d*`,`*mean*`,`*Sigma*`)`	multivariate normal of dimension *d* with mean vector *mean* and covariance matrix *Sigma*; *mean* can be a matrix name or a list of *d* means separated by comma: mu_1, mu_2, ..., mu_d
`mvnormal0(`*d*`,`*Sigma*`)`	multivariate normal of dimension *d* with zero mean vector and covariance matrix *Sigma*
`mvn0(`*d*`,`*Sigma*`)`	synonym for `mvnormal0()`
`zellnersg(`*d*`,`*g*`,`*mean*`,{`*var*`})`	Zellner's *g*-prior of dimension *d* with *g* degrees of freedom, mean vector *mean*, and variance parameter `{`*var*`}`; *mean* can be a matrix name or a list of *d* means separated by comma: mu_1, mu_2, ..., mu_d
`zellnersg0(`*d*`,`*g*`,{`*var*`})`	Zellner's *g*-prior of dimension *d* with *g* degrees of freedom, zero mean vector, and variance parameter `{`*var*`}`
`wishart(`*d*`,`*df*`,`*V*`)`	Wishart of dimension *d* with degrees of freedom *df* and scale matrix *V*
`iwishart(`*d*`,`*df*`,`*V*`)`	inverse Wishart of dimension *d* with degrees of freedom *df* and scale matrix *V*
`jeffreys(`*d*`)`	Jeffreys prior for covariance of a multivariate normal distribution of dimension *d*
Discrete	
`bernoulli(`*p*`)`	Bernoulli with success probability *p*
`index(`p_1`,...,`p_k`)`	discrete indices 1, 2, ..., *k* with probabilities p_1, p_2, ..., p_k
`poisson(`*mu*`)`	Poisson with mean *mu*
Generic	
`flat`	flat prior; equivalent to `density(1)` or `logdensity(0)`
`density(`*f*`)`	generic density *f*
`logdensity(`*logf*`)`	generic log density *logf*

Dimension *d* is a positive number #.

A distribution argument is a number for scalar arguments such as *var*, *alpha*, *beta*; a Stata matrix for matrix arguments such as *Sigma* and *V*; a model parameter, *paramspec*; an expression, *expr*; or a substitutable expression, *subexpr*. See *Specifying arguments of likelihood models and prior distributions*.

f is a nonnegative number, #; an expression, *expr*; or a substitutable expression, *subexpr*.

logf is a number, #; an expression, *expr*; or a substitutable expression, *subexpr*.

options	Description
Model	
noconstant	suppress constant term; not allowed with ordered and nonlinear models
*likelihood(*modelspec*)	distribution for the likelihood model
*prior(*priorspec*)	prior for model parameters; this option may be repeated
dryrun	show model summary without estimation
Model 2	
block(*paramref* [, *blockopts*])	specify a block of model parameters; this option may be repeated
initial(*initspec*)	initial values for model parameters
nomleinitial	suppress the use of maximum likelihood estimates as starting values
Simulation	
mcmcsize(#)	MCMC sample size; default is mcmcsize(10000)
burnin(#)	burn-in period; default is burnin(2500)
thinning(#)	thinning interval; default is thinning(1)
rseed(#)	random-number seed
exclude(*paramref*)	specify model parameters to be excluded from the simulation results
Adaptation	
adaptation(*adaptopts*)	control the adaptive MCMC procedure
scale(#)	initial multiplier for scale factor; default is scale(2.38)
covariance(*cov*)	initial proposal covariance; default is the identity matrix
Reporting	
clevel(#)	set credible interval level; default is clevel(95)
hpd	display HPD credible intervals instead of the default equal-tailed credible intervals
batch(#)	specify length of block for batch-means calculations; default is batch(0)
nomodelsummary	suppress model summary
noexpression	suppress output of expressions from model summary
blocksummary	display block summary
dots	display dots every 100 iterations and iteration numbers every 1,000 iterations
dots(# [, every(#)])	display dots as simulation is performed
noshow(*paramref*)	specify model parameters to be excluded from the output
notable	suppress estimation table
noheader	suppress output header
title(*string*)	display *string* as title above the table of parameter estimates
saving(*filename* [, replace])	save simulation results to *filename*.dta
display_options	control spacing, line width, and base and empty cells
Advanced	
search(*search_options*)	control the search for feasible initial values
corrlag(#)	specify maximum autocorrelation lag; default varies
corrtol(#)	specify autocorrelation tolerance; default is corrtol(0.01)

*Options likelihood() and prior() are required. prior() must be specified for all model parameters.

Options prior() and block() can be repeated.

indepvars and *paramref* may contain factor variables; see [U] **11.4.3 Factor variables**.

With multiple-equations specifications, a local *if* specified within an equation is applied together with the global *if* specified with the command.

Only fweights are allowed; see [U] **11.1.6 weight**.

With multiple-equations specifications, local weights or (weights specified within an equation) override global weights (weights specified with the command).

See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

blockopts	Description
gibbs	requests Gibbs sampling; available for selected models only and not allowed with scale(), covariance(), or adaptation()
split	requests that all parameters in a block be treated as separate blocks
scale(#)	initial multiplier for scale factor for current block; default is scale(2.38); not allowed with gibbs
covariance(*cov*)	initial proposal covariance for the current block; default is the identity matrix; not allowed with gibbs
adaptation(*adaptopts*)	control the adaptive MCMC procedure of the current block; not allowed with gibbs

Only tarate() and tolerance() may be specified in the adaptation() option.

adaptopts	Description
every(#)	adaptation interval; default is every(100)
maxiter(#)	maximum number of adaptation loops; default is maxiter(25) or max{25, floor(burnin()/every())} whenever default values o options are modified
miniter(#)	minimum number of adaptation loops; default is miniter(5)
alpha(#)	parameter controlling acceptance rate (AR); default is alpha(0.75)
beta(#)	parameter controlling proposal covariance; default is beta(0.8)
gamma(#)	parameter controlling adaptation rate; default is gamma(0)
*tarate(#)	target acceptance rate (TAR); default is parameter specific
*tolerance(#)	tolerance for AR; default is tolerance(0.01)

*Only starred options may be specified in the adaptation() option specified within block().

Options

Model

noconstant suppresses the constant term (intercept) from the regression model. By default, bayesmh automatically includes a model parameter {*depname*:_cons} in all regression models except ordered and nonlinear models. Excluding the constant term may be desirable when there is a factor variable, the base level of which absorbs the constant term in the linear combination.

likelihood(*modelspec*) specifies the distribution of the data in the regression model. This option specifies the likelihood portion of the Bayesian model. This option is required. *modelspec* specifies

one of the supported likelihood distributions. A location parameter of these distributions is automatically parameterized as a linear combination of the specified independent variables and needs not be specified. Other parameters may be specified as arguments to the distribution separated by commas. Each argument may be a real number (#), a variable name (except for matrix parameters), a predefined matrix, a model parameter specified in {}, a Stata expression, or a substitutable expression containing model parameters; see *Declaring model parameters* and *Specifying arguments of likelihood models and prior distributions*.

For some likelihood models, option likelihood() provides suboptions *subopts* in likelihood(..., *subopts*). *subopts* is offset(), exposure(), and baseoutcome().

offset(*varname*$_o$) specifies that *varname*$_o$ be included in the regression model with the coefficient constrained to be 1. This option is available with probit, logit, binlogit(), oprobit, ologit, and poisson.

exposure(*varname*$_e$) specifies a variable that reflects the amount of exposure over which the *depvar* events were observed for each observation; ln(*varname*$_e$) with coefficient constrained to be 1 is entered into the log-link function. This option is available with poisson.

noglmtransform requests that the GLM-type transformation not be applied to linear and nonlinear predictors of Poisson, exponential, and binomial logistic regression models. If this option is not specified, Poisson, exponential, and binomial logistic regressions are assumed, which apply the corresponding GLM transformation to the linear predictor to model the mean function. The GLM transformation is $\mu = E(Y|X) = h(X'\boldsymbol{\beta})$, where $h(\cdot)$ is an inverse link function. For example, for Poisson and exponential regression models, $h(x) = \exp(x)$. For generalized nonlinear models, $\mu = h\{f(X, \boldsymbol{\beta})\}$, where $f(\cdot)$ is a nonlinear function of predictors. If this option is specified, the linear and nonlinear specifications are assumed to directly model the mean μ of the Poisson distribution, the scale parameter $b = 1/\mu$ of the exponential distribution, and the probability of success μ/n of the binomial distribution. This option is useful with constant-only models for modeling outcome distributions directly; see *Beta-binomial model*.

prior(*priorspec*) specifies a prior distribution for model parameters. This option is required and may be repeated. A prior must be specified for each model parameter. Model parameters may be scalars or matrices but both types may not be combined in one prior statement. If multiple scalar parameters are assigned a single univariate prior, they are considered independent, and the specified prior is used for each parameter. You may assign a multivariate prior of dimension d to d scalar parameters. Also see *Referring to model parameters* and *Specifying arguments of likelihood models and prior distributions*.

All likelihood() and prior() combinations are allowed, but they are not guaranteed to correspond to proper posterior distributions. You need to think carefully about the model you are building and evaluate its convergence thoroughly.

dryrun specifies to show the summary of the model that would be fit without actually fitting the model. This option is recommended for checking specifications of the model before fitting the model.

Model 2

block(*paramref* [, *blockopts*]) specifies a group of model parameters for the blocked MH algorithm. By default, all parameters except matrices are treated as one block, and each matrix parameter is viewed as a separate block. You can use the block() option to separate scalar parameters in multiple blocks. Technically, you can also use block() to combine matrix parameters in one block, but this is not recommended. The block() option may be repeated to define multiple blocks. Different types of model parameters, such as scalars and matrices, may not be specified in one block(). Parameters within one block are updated simultaneously, and each block of parameters

is updated in the order it is specified; the first specified block is updated first, the second is updated second, and so on. See *Improving efficiency of the MH algorithm—blocking of parameters*.

blockopts include `gibbs`, `split`, `scale()`, `covariance()`, and `adaptation()`.

`gibbs` option specifies to use Gibbs sampling to update parameters in the block. This option is allowed only for specific combinations of likelihood models and prior distributions; see *Gibbs sampling for some likelihood-prior and prior-hyperprior configurations*. For more information, see *Gibbs and hybrid MH sampling*. `gibbs` may not be combined with `scale()`, `covariance()`, or `adaptation()`.

`split` specifies that all parameters in a block are treated as separate blocks. This may be useful for levels of factor variables.

`scale(#)` specifies an initial multiplier for the scale factor corresponding to the specified block. The initial scale factor is computed as $\#/\sqrt{n_p}$ for continuous parameters and as $\#/n_p$ for discrete parameters, where n_p is the number of parameters in the block. By default, # is equal to 2.38 (that is, `scale(2.38)`) is the default. If specified, this option overrides the respective setting from the `scale()` option specified with the `bayesmh` command. `scale()` may not be combined with `gibbs`.

`covariance(`*matname*`)` specifies a scale matrix *matname* to be used to compute an initial proposal covariance matrix corresponding to the specified block. The initial proposal covariance is computed as *rho*×*Sigma*, where *rho* is a scale factor and *Sigma* = *matname*. By default, *Sigma* is the identity matrix. If specified, this option overrides the respective setting from the `covariance()` option specified with the `bayesmh` command. `covariance()` may not be combined with `gibbs`.

`adaptation(tarate())` and `adaptation(tolerance())` specify block-specific TAR and acceptance tolerance. If specified, they override the respective settings from the `adaptation()` option specified with the `bayesmh` command. `adaptation()` may not be combined with `gibbs`.

`initial(`*initspec*`)` specifies initial values for the model parameters to be used in the simulation. You can specify a parameter name, its initial value, another parameter name, its initial value, and so on. For example, to initialize a scalar parameter `alpha` to 0.5 and a 2x2 matrix `Sigma` to the identity matrix `I(2)`, you can type

```
bayesmh ... , initial({alpha} 0.5 {Sigma,m} I(2)) ...
```

You can also specify a list of parameters using any of the specifications described in *Referring to model parameters*. For example, to initialize all regression coefficients from equations `y1` and `y2` to zero, you can type

```
bayesmh ... , initial({y1:} {y2:} 0) ...
```

The general specification of *initspec* is

paramref # [*paramref* # [...]]

Curly braces may be omitted for scalar parameters but must be specified for matrix parameters. Initial values declared using this option override the default initial values or any initial values declared during parameter specification in the `likelihood()` option. See *Specifying initial values* for details.

`nomleinitial` suppresses using maximum likelihood estimates (MLEs) starting values for regression coefficients. By default, when no initial values are specified, MLE values (when available) are used as initial values. If `nomleinitial` is specified and no initial values are provided, `bayesmh` uses ones for positive scalar parameters, zeros for other scalar parameters, and identity matrices for

matrix parameters. `nomleinitial` may be useful for providing an alternative starting state when checking convergence of MCMC.

Simulation

`mcmcsize(#)` specifies the target MCMC sample size. The default MCMC sample size is `mcmcsize(10000)`. The total number of iterations for the MH algorithm equals the sum of the burn-in iterations and the MCMC sample size in the absence of thinning. If thinning is present, the total number of MCMC iterations is computed as `burnin()` + (`mcmcsize()` − 1) × `thinning()` + 1. Computation time of the MH algorithm is proportional to the total number of iterations. The MCMC sample size determines the precision of posterior summaries, which may be different for different model parameters and will depend on the efficiency of the Markov chain. Also see *Burn-in period and MCMC sample size*.

`burnin(#)` specifies the number of iterations for the burn-in period of MCMC. The values of parameters simulated during burn-in are used for adaptation purposes only and are not used for estimation. The default is `burnin(2500)`. Typically, burn-in is chosen to be as long as or longer than the adaptation period. Also see *Burn-in period and MCMC sample size* and *Convergence of MCMC*.

`thinning(#)` specifies the thinning interval. Only simulated values from every $(1+k\times\#)$th iteration for $k = 0, 1, 2, \ldots$ are saved in the final MCMC sample; all other simulated values are discarded. The default is `thinning(1)`; that is, all simulation values are saved. Thinning greater than one is typically used for decreasing the autocorrelation of the simulated MCMC sample.

`rseed(#)` sets the random-number seed. This option can be used to reproduce results. `rseed(#)` is equivalent to typing `set seed #` prior to calling `bayesmh`; see [R] **set seed** and *Reproducing results*.

`exclude(`*paramref*`)` specifies which model parameters should be excluded from the final MCMC sample. These model parameters will not appear in the estimation table, and postestimation features will not be available for these parameters. This option is useful for suppressing nuisance model parameters. For example, if you have a factor predictor variable with many levels but you are only interested in the variability of the coefficients associated with its levels, not their actual values, then you may wish to exclude this factor variable from the simulation results. If you simply want to omit some model parameters from the output, see the `noshow()` option.

Adaptation

`adaptation(`*adaptopts*`)` controls adaptation of the MCMC procedure. Adaptation takes place every prespecified number of MCMC iterations and consists of tuning the proposal scale factor and proposal covariance for each block of model parameters. Adaptation is used to improve sampling efficiency. Provided defaults are based on theoretical results and may not be sufficient for all applications. See *Adaptation of the MH algorithm* for details about adaptation and its parameters.

adaptopts are any of the following options:

`every(#)` specifies that adaptation be attempted every #th iteration. The default is `every(100)`. To determine the adaptation interval, you need to consider the maximum block size specified in your model. The update of a block with k model parameters requires the estimation of $k \times k$ covariance matrix. If the adaptation interval is not sufficient for estimating the $k(k+1)/2$ elements of this matrix, the adaptation may be insufficient.

`maxiter(#)` specifies the maximum number of adaptive iterations. Adaptation includes tuning of the proposal covariance and of the scale factor for each block of model parameters. Once the TAR is achieved within the specified tolerance, the adaptation stops. However, no more than # adaptation steps will be performed. The default is variable and is computed as max{25, `floor(burnin()/adaptation(every()))`}.

`maxiter()` is usually chosen to be no greater than (`mcmcsize()` + `burnin()`)/ `adaptation(every())`.

`miniter(#)` specifies the minimum number of adaptive iterations to be performed regardless of whether the TAR has been achieved. The default is `miniter(5)`. If the specified `miniter()` is greater than `maxiter()`, then `miniter()` is reset to `maxiter()`. Thus, if you set `maxiter(0)`, then no adaptation will be performed.

`alpha(#)` specifies a parameter controlling the adaptation of the AR. `alpha()` should be in $[0, 1]$. The default is `alpha(0.75)`.

`beta(#)` specifies a parameter controlling the adaptation of the proposal covariance matrix. `beta()` must be in [0,1]. The closer `beta()` is to zero, the less adaptive the proposal covariance. When `beta()` is zero, the same proposal covariance will be used in all MCMC iterations. The default is `beta(0.8)`.

`gamma(#)` specifies a parameter controlling the adaptation rate of the proposal covariance matrix. `gamma()` must be in [0,1]. The larger the value of `gamma()`, the less adaptive the proposal covariance. The default is `gamma(0)`.

`tarate(#)` specifies the TAR for all blocks of model parameters; this is rarely used. `tarate()` must be in (0,1). The default AR is 0.234 for blocks containing continuous multiple parameters, 0.44 for blocks with one continuous parameter, and 1/*n_maxlev* for blocks with discrete parameters, where *n_maxlev* is the maximum number of levels for a discrete parameter in the block.

`tolerance(#)` specifies the tolerance criterion for adaptation based on the TAR. `tolerance()` should be in (0,1). Adaptation stops whenever the absolute difference between the current and TARs is less than `tolerance()`. The default is `tolerance(0.01)`.

`scale(#)` specifies an initial multiplier for the scale factor for all blocks. The initial scale factor is computed as $\#/\sqrt{n_p}$ for continuous parameters and $\#/n_p$ for discrete parameters, where n_p is the number of parameters in the block. By default, # is equal to 2.38; that is, `scale(2.38)` is the default.

`covariance(`*cov*`)` specifies a scale matrix *cov* to be used to compute an initial proposal covariance matrix. The initial proposal covariance is computed as $\rho \times \Sigma$, where ρ is a scale factor and $\Sigma =$ *matname*. By default, Σ is the identity matrix. Partial specification of Σ is also allowed. The rows and columns of *cov* should be named after some or all model parameters. According to some theoretical results, the optimal proposal covariance is the posterior covariance matrix of model parameters, which is usually unknown.

Reporting

`clevel(#)` specifies the credible level, as a percentage, for equal-tailed and HPD credible intervals. The default is `clevel(95)` or as set by [BAYES] **set clevel**.

`hpd` specifies the display of HPD credible intervals instead of the default equal-tailed credible intervals.

`batch(#)` specifies the length of the block for calculating batch means, batch standard deviation, and MCSE using batch means. The default is `batch(0)`, which means no batch calculations. When `batch()` is not specified, MCSE is computed using effective sample sizes instead of batch means. Option `batch()` may not be combined with `corrlag()` or `corrtol()`.

`nomodelsummary` suppresses the detailed summary of the specified model. Model summary is reported by default.

`noexpression` suppresses the output of expressions from the model summary. Expressions (when specified) are reported by default.

`blocksummary` displays the summary of the specified blocks. This option is useful when `block()` is specified and may not be combined with `dryrun`.

`dots` and `dots(#)` specify to display dots as simulation is performed. `dots(#)` displays a dot every # iterations. During the adaptation period, a symbol `a` is displayed instead of a dot. If `dots(..., every(#))` is specified, then an iteration number is displayed every #th iteration instead of a dot or `a`. `dots(, every(#))` is equivalent to `dots(1, every(#))`. `dots` displays dots every 100 iterations and iteration numbers every 1,000 iterations; it is a synonym for `dots(100), every(1000)`. By default, no dots are displayed (`dots(0)`).

`noshow(`*paramref*`)` specifies a list of model parameters to be excluded from the output. Do not confuse this option with `exclude()`, which excludes the specified parameters from the MCMC sample.

`notable` suppresses the estimation table from the output. By default, a summary table is displayed containing all model parameters except those listed in the `exclude()` and `noshow()` options. Regression model parameters are grouped by equation names. The table includes six columns and reports the following statistics using the MCMC simulation results: posterior mean, posterior standard deviation, MCMC standard error or MCSE, posterior median, and credible intervals.

`noheader` suppresses the output header either at estimation or upon replay.

`title(`*string*`)` specifies an optional title for the command that is displayed above the table of the parameter estimates. The default title is specific to the specified likelihood model.

`saving(`*filename*`[ , replace ])` saves simulation results in *filename*`.dta`. The `replace` option specifies to overwrite *filename*`.dta` if it exists. If the `saving()` option is not specified, `bayesmh` saves simulation results in a temporary file for later access by postestimation commands. This temporary file will be overridden every time `bayesmh` is run and will also be erased if the current estimation results are cleared. `saving()` may be specified during estimation or on replay.

The saved dataset has the following structure. Variance `_index` records iteration numbers. `bayesmh` saves only states (sets of parameter values) that are different from one iteration to another and the frequency of each state in variable `_frequency`. (Some states may be repeated for discrete parameters.) As such, `_index` may not necessarily contain consecutive integers. Remember to use `_frequency` as a frequency weight if you need to obtain any summaries of this dataset. Values for each parameter are saved in a separate variable in the dataset. Variables containing values of parameters without equation names are named as `eq0_p#`, following the order in which parameters are declared in `bayesmh`. Variables containing values of parameters with equation names are named as `eq#_p#`, again following the order in which parameters are defined. Parameters with the same equation names will have the same variable prefix `eq#`. For example,

```
. bayesmh y x1, likelihood(normal({var})) saving(mcmc) ...
```

will create a dataset `mcmc.dta` with variable names `eq1_p1` for `{y:x1}`, `eq1_p2` for `{y:_cons}`, and `eq0_p1` for `{var}`. Also see macros `e(parnames)` and `e(varnames)` for the correspondence between parameter names and variable names.

In addition, `bayesmh` saves variable `_loglikelihood` to contain values of the log likelihood from each iteration and variable `_logposterior` to contain values of log posterior from each iteration.

display_options: `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fvwrap(#)`, `fvwrapon(`*style*`)`, and `nolstretch`; see [R] **estimation options**.

Advanced

`search(`*search_options*`)` searches for feasible initial values. `search_options` are `on`, `repeat(#)`, and `off`.

`search(on)` is equivalent to `search(repeat(500))`. This is the default.

`search(repeat(`k`))`, $k > 0$, specifies the number of random attempts to be made to find a feasible initial-value vector, or initial state. The default is `repeat(500)`. An initial-value vector is feasible if it corresponds to a state with positive posterior probability. If feasible initial values are not found after k attempts, an error will be issued. `repeat(0)` (rarely used) specifies that no random attempts be made to find a feasible starting point. In this case, if the specified initial vector does not correspond to a feasible state, an error will be issued.

`search(off)` prevents `bayesmh` from searching for feasible initial values. We do not recommend specifying this option.

`corrlag(#)` specifies the maximum autocorrelation lag used for calculating effective sample sizes. The default is $\min\{500, \texttt{mcmcsize()}/2\}$. The total autocorrelation is computed as the sum of all lag-k autocorrelation values for k from 0 to either `corrlag()` or the index at which the autocorrelation becomes less than `corrtol()` if the latter is less than `corrlag()`. Options `corrlag()` and `batch()` may not be combined.

`corrtol(#)` specifies the autocorrelation tolerance used for calculating effective sample sizes. The default is `corrtol(0.01)`. For a given model parameter, if the absolute value of the lag-k autocorrelation is less than `corrtol()`, then all autocorrelation lags beyond the kth lag are discarded. Options `corrtol()` and `batch()` may not be combined.

Remarks and examples

Remarks are presented under the following headings:

Examples are presented under the following headings:

For a quick overview example of all Bayesian commands, see *Overview example* in [BAYES] **bayes**.

Using bayesmh

The `bayesmh` command for Bayesian analysis includes three functional components: setting up a posterior model, performing MCMC simulation, and summarizing and reporting results. The first component, the model-building step, requires some experience in the practice of Bayesian statistics and, as any modeling task, is probably the most demanding. You should specify a posterior model that is statistically correct and that represents the observed data. Another important aspect is the computational feasibility of the model in the context of the MH MCMC procedure implemented in `bayesmh`. The provided MH algorithm is adaptive and, to a degree, can accommodate various statistical models and data structures. However, careful model parameterization and well-specified initial values and MCMC sampling scheme are crucial for achieving a fast-converging Markov chain and consequently good results. Simulation of MCMC must be followed by a thorough investigation of the convergence of the MCMC algorithm. Once you are satisfied with the convergence of the simulated chains, you may proceed with posterior summaries of the results and their interpretation. Below we discuss the three major steps of using `bayesmh` and provide recommendations.

Setting up a posterior model

Any posterior model includes a likelihood model that specifies the conditional distribution of the data given model parameters and prior distributions for all model parameters. The prior distribution of a parameter can itself be specified conditional on other parameters, also referred to as *hyperparameters*. We will refer to their prior distributions as *hyperpriors*.

Likelihood model

The likelihood model describes the data. You build your likelihood model the same way you do this in frequentist likelihood-based analysis.

The `bayesmh` command provides various likelihood models, which are specified in the `likelihood()` option. For a univariate response, there are normal models, generalized linear models for binary and count response, and more. For a multivariate model, you may choose between a multivariate normal model with covariates common to all variables and with covariates specific to each variable. You can also build likelihood models for multiple variables by specifying a distribution and a regression function for each variable by using `bayesmh`'s multiple-equation specification.

`bayesmh` is designed for fitting regression models. As we said above, you specify the likelihood or outcome distribution in the `likelihood()` option. The regression specification of the model is the same as for other regression commands. For a univariate response, you specify the dependent and all independent variables following the command name. (Here we also include the `prior()` option that specifies prior distributions to emphasize that it is required in addition to `likelihood()`. See the next subsection for details about this option.)

```
. bayesmh y x1 x2, likelihood() prior() ...
```

For a multivariate response, you separate the dependent variables from the independent variables with the equal sign.

```
. bayesmh y1 y2 = x1 x2, likelihood(mvnormal(...)) prior() ...
```

With the multiple-equation specification, you follow the syntax for the univariate response, but you specify each equation in parentheses and you specify the `likelihood()` option within each equation.

```
. bayesmh (y1 x1, likelihood()) (y2 x2, likelihood()),  prior() ...
```

In the above models, the regression function is modeled using a linear combination of the specified independent variables and regression coefficients. The constant is included by default, but you can specify the `noconstant` option to omit it from the linear predictor.

`bayesmh` also allows you to model the regression function as a nonlinear function of independent variables and regression parameters. In this case, you must use the equal sign to separate the dependent variable from the expression and specify the expression in parentheses:

```
. bayesmh y = ({a}+{b}*x^{c}), likelihood(normal()) prior() ...

. bayesmh (y1 = ({a1}+{b1}*x^{c1})   ///
          (y2 = ({a2}+{b2}*x^{c2}), likelihood(mvnormal()) prior() ...
```

For a not-supported or nonstandard likelihood, you can use the `llf()` option within `likelihood()` to specify a generic expression for the observation-level likelihood function; see *Substitutable expressions*. When you use the `llf()` option, it is your responsibility to ensure that the provided expression corresponds to a valid density. For more complicated Bayesian models, you may consider writing your own likelihood or posterior function evaluators; see [BAYES] **bayesmh evaluators**.

Prior distributions

In addition to the likelihood, you must also specify prior distributions for all model parameters in a Bayesian model. Prior distributions or priors are key components in a Bayesian model specification and should be chosen carefully. They are used to quantify some expert knowledge or existing information about model parameters. For example, priors can be used for constraining the domain of some parameters to localize values that we think are more probable for reasons that are not considered in the likelihood specification. Improper priors (priors with densities that do not integrate to finite numbers) are also allowed, as long as they yield valid posterior distributions. Priors are often categorized as informative (subjective) or noninformative (objective). Noninformative priors are also known as vague priors. Uniform distributions are often used as noninformative priors and can even be applied to parameters with unbounded domains, in which case they become improper priors. Normal and gamma distributions with very large variances relative to the expected values of the parameters are also used as noninformative priors. Another family of noninformative priors, often chosen for their invariance under reparameterization, are so-called Jeffreys priors, named after Harold Jeffreys (Jeffreys 1946). For example, the `bayesmh` command provides built-in Jeffreys priors for the normal family of distributions. Jeffreys priors are usually improper. As discussed by many researchers, however, the overuse of noninformative priors contradicts the principles of Bayesian approach—analysis of a posterior model with noninformative priors would be close to one based on the likelihood only. Noninformative priors may also negatively influence the MCMC convergence. It is thus important to find good priors based on earlier studies and use them in the model as well as perform sensitivity analysis for competing priors. A good choice of prior should minimize the MCMC standard errors of the parameter estimates.

As for likelihoods, the bayesmh command provides several priors you can choose from by specifying the prior() options. For example, continuous univariate priors include normal, lognormal, uniform, inverse gamma, and exponential; discrete priors include Bernoulli and Poisson; multivariate priors include multivariate normal and inverse Wishart. There are also special priors: jeffreys and jeffreys(#), which specify Jeffreys priors for the variance of the normal and multivariate normal distributions, and zellnersg() and zellnersg0(), which specify multivariate priors for regression coefficients (Zellner and Revankar 1969).

The prior() option is required and can be repeated. You can use the prior() option for each parameter or you can combine multiple parameters in one prior() specification.

For example, we can specify different priors for parameters {y:x} and {y:_cons} by

```
. bayesmh y x, ... prior({y:x}, normal(10,100)) prior({y:_cons}, normal(20,200)) ...
```

or the same univariate prior using one prior() statement, using

```
. bayesmh y x, ... prior({y:x _cons}, normal(10,100)) ...
```

or a multivariate prior with zero mean and fixed variance–covariance S, as follows:

```
. bayesmh y x, ... prior({y:x _cons}, mvnormal0(2,S)) ...
```

In the prior() option, we list model parameters following any of the specifications described in *Referring to model parameters* and then, following the comma, we specify one of the prior distributions *priordist*.

If you want to specify a nonstandard prior or if the prior you need is not supported, you can use the density() or logdensity() option within the prior() option to specify an expression for a generic density or log density of the prior distribution; see *Substitutable expressions*. When you use the density() or logdensity() option, it is your responsibility to ensure that the provided expression corresponds to a valid density. For a complicated Bayesian model, you may consider writing your own posterior function evaluator; see [BAYES] **bayesmh evaluators**.

Sometimes, you may need to specify a flat prior (a prior with the density equal to one) for some of the parameters. This is often needed when specifying a noninformative prior. You can specify the flat option instead of the prior distribution in the prior() option to request the flat prior. This option is equivalent to specifying density(1) or logdensity(0) in prior().

The specified likelihood model for the data and prior distributions for the parameters are not guaranteed to result in proper posterior distributions of the parameters. Therefore, unless you are using one of the standard Bayesian models, you should always check the validity of the posterior model you specified.

Declaring model parameters

Model parameters are typically declared, meaning first introduced, in the arguments of distributions specified in options likelihood() and prior(). We will refer to model parameters that are declared in the prior distributions (and not the likelihood distributions) as hyperparameters. Model parameters may also be declared within the parameter specification of the prior() option, but this is more rare.

bayesmh distinguishes between two types of model parameters: scalar and matrix. All parameters must be specified in curly braces, {}. There are two ways for declaring a scalar parameter: {*param*} and {*eqname*:*param*}, where *param* and *eqname* are valid Stata names.

The specification of a matrix parameter is similar, but you must use the matrix suboptions: {*param*, matrix} and {*eqname*:*param*, matrix}. The most common application of matrix model parameters is for specifying the variance–covariance matrix of a multivariate normal distribution.

All matrices are assumed to be symmetric and only the elements in the lower diagonal are reported in the output. Only a few multivariate prior distributions are available for matrix parameters: `wishart()`, `iwishart()`, and `jeffreys()`. In addition to being symmetric, these distributions require that the matrices be positive definite.

It is your responsibility to declare all parameters of your model, except regression coefficients in linear models. For a linear model, `bayesmh` automatically creates a regression coefficient with the name {*depvar*:*indepvar*} for each independent variable *indepvar* in the model and, if `noconstant` is not specified, an intercept parameter {*depvar*:`_cons`}. In the presence of factor variables, `bayesmh` will create a parameter {*depvar*:*level*} for each level indicator *level* and a parameter {*depvar*:*inter*} for each interaction indicator *inter*; see **[U] 11.4.3 Factor variables**. (It is still your responsibility, however, to specify prior distributions for the regression parameters.)

For example,

```
. bayesmh y x, ...
```

will automatically have two regression parameters: {`y:x`} and {`y:_cons`}, whereas

```
. bayesmh y x, noconstant ...
```

will have only one: {`y:x`}.

For a univariate normal linear regression, we may want to additionally declare the scalar variance parameter by

```
. bayesmh y x, likelihood(normal({sig2})) ...
```

We can label the variance parameter, as follows:

```
. bayesmh y x, likelihood(normal({var:sig2})) ...
```

We can declare a hyperparameter for {`sig2`} using

```
. bayesmh y x, likelihood(normal({sig2})) prior({sig2}, igamma({df},2)) ...
```

where the hyperparameter {`df`} is declared in the inverse-gamma prior distribution for {`sig2`}.

For a multivariate normal linear regression, in addition to four regression parameters declared automatically by `bayesmh`: {`y1:x`}, {`y1:_cons`}, {`y2:x`}, and {`y2:_cons`}, we may also declare a parameter for the variance–covariance matrix:

```
. bayesmh y1 y2 = x, likelihood(mvnormal({Sigma, matrix})) ...
```

or abbreviate `matrix` to `m` for short:

```
. bayesmh y1 y2 = x, likelihood(mvnormal({Sigma, m})) ...
```

Referring to model parameters

After a model parameter is declared, we may need to refer to it in our further model specification. We will definitely need to refer to it when we specify its prior distribution. We may also need to use it as an argument in the prior distributions of other parameters or need to specify it in the `block()` option for blocking of model parameters; see *Improving efficiency of the MH algorithm—blocking of parameters*.

To refer to one parameter, we simply use its definition: {*param*}, {*eqname*:*param*}, {*param*, `matrix`}, or {*eqname*:*param*, `matrix`}. There are several ways in which you can refer to multiple parameters. You can refer to multiple model parameters in the parameter specification *paramref* of the `prior(`*paramref*`, ...)` option, of the `block(`*paramref*`, ...)` option, or of the `initial(`*paramref* `#)` option.

The most straightforward way to refer to multiple scalar model parameters is to simply list them individually, as follows:

```
{param1} {param2} ...
```

but there are shortcuts.

For example, the alternative to the above is

```
{param1 param2} ...
```

where we simply list the names of all parameters inside one set of curly braces.

If parameters have the same equation name, you can refer to all of the parameters with that equation name as follows. Suppose that we have three parameters with the same equation name `eqname`, then the specification

```
{eqname:param1} {eqname:param2} {eqname:param3}
```

is the same as the specification

```
{eqname:}
```

or the specification

```
{eqname:param1 param2 param3}
```

The above specification is useful if we want to refer to a subset of parameters with the same equation name. For example, in the above, if we wanted to refer to only `param1` and `param2`, we could type

```
{eqname:param1 param2}
```

If a factor variable is used in the specification of the regression function, you can use the same factor-variable specification within *paramref* to refer to the coefficients associated with the levels of that factor variable; see **[U] 11.4.3 Factor variables**.

For example, factor variables are useful for constructing multilevel Bayesian models. Suppose that variable `id` defines the second level of hierarchy in a two-level random-effects model. We can fit a Bayesian random-intercept model as follows.

```
. bayesmh y x i.id, likelihood(normal({var})) prior({y:i.id}, normal(0,{tau})) ...
```

Here we used `{y:i.id}` in the prior specification to refer to all levels of `id`.

Similarly, we can add a random coefficient for a continuous covariate `x` by typing

```
. bayesmh y c.x##i.id, likelihood(normal({var}))
>                      prior({y:i.id}, normal(0,{tau1}))
>                      prior({y:c.x#i.id}, normal(0,{tau2})) ...
```

You can mix and match all of the specifications above in one parameter specification, *paramref*.

To refer to multiple matrix model parameters, you can use {*paramlist*, `matrix`} to refer to matrix parameters with names *paramlist* and {*eqname*:*paramlist*, `matrix`} to refer to matrix parameters with names in *paramlist* and with equation name *eqname*.

For example, the specification

```
{eqname:Sigma1,m} {eqname:Sigma2,m} {Sigma3,m} {Sigma4,m}
```

is the same as the specification

```
{eqname:Sigma1 Sigma2,m} {Sigma3 Sigma4,m}
```

You cannot refer to both scalar and matrix parameters in one *paramref* specification.

For referring to model parameters in postestimation commands, see *Different ways of specifying model parameters* in [BAYES] **bayesmh postestimation**.

Specifying arguments of likelihood models and prior distributions

As previously mentioned, likelihood distributions (or more precisely, likelihood models), *modelspec*, are specified in the `likelihood(`*modelspec*`)` option and prior distributions *priordist* are specified following the comma in the `prior(`*paramref*`,` *priordist*`)` option. For a list of supported models and distributions, see the corresponding tables in the syntax diagram.

In a likelihood model, mean and location parameters are determined by the specified regression function and thus need not be specified in the likelihood distributions. For example, for a normal linear regression, we use `likelihood(normal(`*var*`))`, where we specify only the variance parameter—the mean is already parameterized as a linear combination of the specified independent variables. In the prior distributions, we must specify all parameters of the distribution. For example, for a normal prior specification, we use `prior(`*paramref*`, normal(`*mu*`,` *var*`))`, where we must specify both mean *mu* and variance *var*. In addition, all multivariate prior distributions require that you specify the dimension *d* as the first argument.

Scalar arguments of the distributions may be specified as a number or as a scalar expression *exp*. Matrix arguments of the distributions may be specified as a matrix or as a matrix expression *exp*. Both types of arguments may be specified as a parameter (see *Declaring model parameters*) or as a substitutable expression, *subexp* (see *Substitutable expressions*). All distribution arguments, except the dimension *d* of multivariate prior distributions, support the above specifications. For likelihood models, arguments of the distributions may also contain variable names.

For example, in a normal linear regression, we can specify the variance as a known value of 25,

```
. bayesmh y x, likelihood(normal(25)) ...
```

or as a squared standard deviation of 5 (scalar expression),

```
. bayesmh y x, likelihood(normal(5^2)) ...
```

or as an unknown variance parameter `{var}`,

```
. bayesmh y x, likelihood(normal({var})) ...
```

or as a function of an unknown standard-deviation parameter `{sd}` (substitutable expression),

```
. bayesmh y x, likelihood(normal({sd}^2)) ...
```

In a multivariate normal linear regression, we can specify the variance–covariance matrix as a known matrix `S`,

```
. bayesmh y1 y2 = x, likelihood(mvnormal(S)) ...
```

or as a matrix function `S = R*R'` using its Cholesky decomposition,

```
. bayesmh y1 y2 = x, likelihood(mvnormal(R*R')) ...
```

or as an unknown matrix parameter `{Sigma,m}`,

```
. bayesmh y1 y2 = x, likelihood(mvnormal({Sigma,m})) ...
```

or as a function of an unknown variance parameter `{var}` (substitutable expression),

```
. bayesmh y1 y2 = x, likelihood(mvnormal({var}*S)) ...
```

Substitutable expressions

You may use substitutable expressions in `bayesmh` to define nonlinear expressions *subexpr*, arguments of outcome distributions in option `likelihood()`, observation-level log likelihood in option `llf()`, arguments of prior distributions in option `prior()`, and generic prior distributions in `prior()`'s suboptions `density()` and `logdensity()`. Substitutable expressions are just like any other mathematical expression in Stata, except that they may include model parameters.

To specify a substitutable expression in your `bayesmh` model, you must comply with the following rules:

1. Model parameters are bound in braces: `{mu}`, `{var:sigma2}`, `{Sigma, matrix}`, and `{Cov:Sigma, matrix}`.
2. Linear combinations can be specified using the notation `{`*eqname*`:`*varlist*`}`. For example,

   `{xb:mpg price weight}` is equivalent to

   ```
   {xb_mpg}*mpg + {xb_price}*price + {xb_weight}*weight
   ```

3. There is a small caveat with using the `{`*eqname*`:`*name*`}` specification 2 when *name* corresponds to both one of the variables in the dataset and a parameter in the model. The linear-combination specification takes precedence in this case. For example, `{eq:var}` will be expanded to `{eq_var}*var`, where `var` is the variable in a dataset and `eq_var` is the coefficient corresponding to this variable. To refer directly to the coefficient, you must use `{eq_var}`.
4. Initial values are given by including an equal sign and the initial value inside the braces, for example, `{b1=1.267}`, `{gamma=3}`, etc. If you do not specify an initial value, that parameter is initialized to one for positive scalar parameters and to zero for other scalar parameters, or it is initialized to its MLE, if available. The `initial()` option overrides initial values provided in substitutable expressions. Initial values for matrices must be specified in the `initial()` option. By default, matrix parameters are initialized with identity matrices.

Specifying linear combinations. We can use substitutable expressions to specify linear combinations.

For example, a normal linear regression,

```
. bayesmh y x1 x2, likelihood(normal(1)) noconstant prior({y:x1 x2}, normal(0,100))
```

may be equivalently (but less efficiently) fit using a nonlinear regression,

```
. bayesmh y = ({y:x1 x2}), likelihood(normal(1)) prior({y:x1 x2}, normal(0,100))
```

The above nonlinear specification is essentially,

```
. bayesmh y = ({y_x1}*x1+{y_x2}*x2), likelihood(normal(1))
> prior({y:x1 x2}, normal(0,100))
```

Notice that the specification `{y:x1 x2}` in the `prior()` option is not a substitutable expression, but it is one way of referring to model parameters described in *Referring to model parameters*. Substitutable expressions are not allowed in the parameter specification *paramref* of `prior(`*paramref*`, ...)`.

Specifying nonstandard densities. We can use substitutable expressions to define nonstandard or not-supported probability distributions.

For example, suppose we want to specify a Cauchy distribution with location a and scale b. We can specify the expression for the observation-level likelihood function in the `llf()` option within `likelihood()`.

```
. bayesmh y, likelihood(llf(ln({b})-ln({b}^2+(y-{a})^2)-ln(_pi))) noconstant ...
```

You can also use substitutable expressions to define nonstandard or not-supported prior distributions. For example, as suggested by Gelman et al. (2014), we can specify a Cauchy prior with location $a = 0$ and scale $b = 2.5$ for logistic regression coefficients, where continuous covariate `x` is standardized to have mean 0 and standard deviation 0.5.

```
. bayesmh y x, likelihood(logit) prior({y:},
> logdensity(ln(2.5)-ln(2.5^2+(y)^2)-ln(_pi)))
```

Checking model specification

Specifying a Bayesian model may be a tedious task when there are many model parameters and possibly hyperparameters. It is thus essential to verify model specification before starting a potentially time-consuming estimation.

`bayesmh` displays the summary of the specified model as a part of its standard output. You can use the `dryrun` option to obtain the model summary without estimation or simulation. Once you are satisfied with the specified model, you can use the `nomodelsummary` option to suppress a potentially long model summary during estimation. Even if you specify `nomodelsummary` during estimation, you will still be able to see the model summary, if desired, by simply replaying the results:

```
. bayesmh
```

Specifying MCMC sampling procedure

Once you specify a correct posterior model, `bayesmh` uses an adaptive random-walk MH algorithm to obtain MCMC samples of model parameters from their posterior distribution.

Reproducing results

Because `bayesmh` uses MCMC simulation—a stochastic procedure for sampling from a complicated and possibly nontractable distribution—it will produce different results each time you run the command. If the MCMC algorithm converged, the results should not change drastically. To obtain reproducible results, you must specify the random-number seed.

To specify a random-number seed, you can use `set seed #` prior to calling `bayesmh` (see [R] **set seed**) or you can specify the seed in `bayesmh`'s option `rseed()`. For simplicity and consistency, we use `set seed 14` in all of our examples throughout the documentation.

If you forgot to specify the random-number seed before calling `bayesmh`, you can retrieve the random-number state used by the command from `e(rngstate)` and use it later with `set rngstate`.

Burn-in period and MCMC sample size

`bayesmh` has the default burn-in period of 2,500 iterations and the default MCMC sample size of 10,000 iterations. That is, the first 2,500 iterations of the MCMC sampler are discarded and the next 10,000 iterations are used to form the MCMC samples of values of model parameters. You can change these numbers by specifying options `burnin()` and `mcmcsize()`.

The burn-in period must be long enough for the algorithm to reach convergence or, in other words, for the Markov chain to reach its stationary distribution or the desired posterior distribution of model parameters. The sample size for the MCMC sample is typically determined based on the autocorrelation present in the MCMC sample. The higher the autocorrelation, the larger the MCMC sample should be to achieve the same precision of the parameter estimates as obtained from the chain with low or negligible autocorrelation. Because of the nature of the sampling algorithm, all MCMC exhibit some autocorrelation and thus MCMC samples tend to have large sizes.

The defaults provided by bayesmh may not be sufficient for all Bayesian models and data types. You will need to explore the convergence of the MCMC algorithm for your particular data problem and modify the settings, if needed.

After the burn-in period, bayesmh includes every iteration in the MCMC sample. You can specify the thinning(#) option to store results from a subset of iterations. This option is useful if you want to subsample the chain to decrease autocorrelation in the final MCMC sample. If you use this option, bayesmh will perform a total of thinning() × (mcmcsize() − 1) + 1 iterations, excluding burn-in iterations, to obtain MCMC sample of size mcmcsize().

Improving efficiency of the MH algorithm—blocking of parameters

Although the MH algorithm is very general and can be applied to any Bayesian model, it is not the most optimal sampler and may require tuning to achieve higher efficiency.

Efficiency describes mixing properties of the Markov chain. High efficiency means good mixing (low autocorrelation) in the MCMC sample, and low efficiency means bad mixing (high autocorrelation) in the MCMC sample.

An AR is the number of accepted proposals of model parameters relative to the total number of proposals. It should not be confused with sampling efficiency. High AR does not mean high efficiency.

An efficient MH sampler has an AR between 15% and 50% (Roberts and Rosenthal 2001) and low autocorrelation and thus relatively large effective sample size (ESS) for all model parameters.

One way to improve efficiency of the MH algorithm is by blocking of model parameters. Blocking of model parameters is an important functional aspect of the MH sampler. By default, all parameters are used as one block and their covariance matrix is used to adapt the proposal distribution. With many parameters, estimation of this covariance matrix becomes difficult and imprecise and may lead to the loss of efficiency of the MH algorithm. In many cases, this matrix has a block diagonal structure because of independence of some blocks or sets of model parameters and its estimation may be replaced with estimation of the corresponding blocks, which are typically of smaller dimension. This may improve the efficiency of the sampler. To achieve optimal blocking, you need to identify the sets of approximately independent (a posteriori) model parameters and specify them in separate blocks.

To achieve an optimal blocking, you need to know or have some idea about the dependence between the parameters as determined by the posterior distribution. To improve efficiency, follow this principle: correlated parameters should be specified together, while independent groups of parameters should be specified in separate blocks. Because the posterior is usually defined indirectly, the relationship between the parameters is generally unknown. Often, however, we have some knowledge, either deduced from the model specification or based on prior experience with the model, about which parameters are highly correlated. In the worst case, you may need to run some preliminary simulations and determine an optimal blocking by using trial and error.

An ideal case for the MH algorithm is when all model parameters are independent with respect to the posterior distribution and are thus placed in separate blocks and sampled independently. In practice, this is not a realistic or interesting case, but it gives us an idea that we should always try to parameterize the model in such a way that the correlation between model parameters is minimized.

With bayesmh, you can use options block() to perform blocking. You specify one block() option for each set of independent model parameters. Model parameters that are dependent with each other are specified in the same block() option.

To illustrate a typical case, consider the following simple linear regression model:

$$\mathtt{y} = \mathtt{\{a\}} + \mathtt{\{b\}} \times \mathtt{x} + \epsilon,\ \epsilon \sim N(0, \mathtt{\{var\}})$$

Even when {a} and {b} have independent prior specifications, the location parameters {a} and {b} are expected to be correlated a posteriori because of their common dependence on y. Alternatively, if the variance parameter {var} is independent of {a} and {b} a priori, it is generally less correlated with the location parameters a posteriori. A good blocking scheme is to use options block({a} {b}) and block({var}) with bayesmh. We can also reparameterize our model to reduce the correlation between {a} and {b} by recentering. To center the slope parameter, we replace {b} with {b} − #, where # is a constant close to the mean of {b}. Now {a} and {b} − # can also be placed in separate blocks. See, for example, Thompson (2014) for more discussion related to model parameterization.

Other options that control MCMC sampling efficiency are scale(), covariance(), and adaptation(); see *Adaptation of the MH algorithm* for details.

Gibbs and hybrid MH sampling

In *Improving efficiency of the MH algorithm—blocking of parameters*, we discussed blocking of model parameters as a way of improving efficiency of the MH algorithm. For certain Bayesian models, further improvement is possible by using Gibbs sampling for certain blocks of parameters. This leads to what we call a hybrid MH sampling with Gibbs updates.

Gibbs sampling is the most effective sampling procedure with the maximum possible AR of one and with often very high efficiency. Using Gibbs sampling for some blocks of parameters will typically lead to higher efficiency of the hybrid MH sampling compared with the simple MH sampling.

To apply Gibbs sampling to a set of parameters, we need to know the full conditional distribution for each parameter and be able to generate random samples from it. Usually, the full conditionals are known in various special cases but are not available for general posterior distributions. Thus, Gibbs sampling is not available for all likelihood and prior combinations. bayesmh provides Gibbs sampling for Bayesian models with conjugate, or more specifically, semiconjugate prior distributions. See *Gibbs sampling for some likelihood-prior and prior-hyperprior configurations* for a list of supported models.

For a supported semiconjugate model, you can request Gibbs sampling for a block of parameters by specifying the gibbs suboption within option block(). In some cases, the gibbs suboption may be used in all parameter blocks, in which case we will have full Gibbs sampling.

To use Gibbs sampling for a set of parameters, you must first place them in separate prior() statements and specify semiconjugate prior distributions and then place them in a separate block and include the gibbs suboption, block(..., gibbs).

Here is a standard application of a full Gibbs sampling to a normal mean-only model. Under the normal–inverse-gamma prior, the conditional posterior distributions of the mean parameter is normal and of the variance parameter is inverse gamma.

```
. bayesmh y, likelihood(normal({var}))
>            prior({y:_cons},  normal(1,10))
>            prior({var},      igamma(10,1))
>            block({y:_cons},  gibbs)
>            block({var},      gibbs)
```

Because {y:_cons} and {var} are approximately independent a posteriori, we specified them in separate blocks.

Gibbs sampling can be applied to hyperparameters, which are not directly involved in the likelihood specification of the model. For example, we can use Gibbs sampling for the covariance matrix of regression coefficients.

```
. bayesmh y x, likelihood(normal(var))
>              prior(var,         igamma(10,1))
>              prior({y:_cons x}, mvnormal(2,1,0,{Sigma,m}))
>              prior({Sigma,m},   iwishart(2,10,V))
>              block({Sigma,m},   gibbs)
```

In the next example, the matrix parameter `{Sigma,m}` specifies the covariance matrix in the multivariate normal prior for a pair of model parameters, `{y:1.cat}` and `{y:2.cat}`. `{Sigma,m}` is a hyperparameter—it is not a model parameter of the likelihood but a parameter of a prior distribution, and it has an inverse-Wishart hyperprior distribution, which is a semiconjugate prior with respect to the multivariate normal prior distribution. Therefore, we can request a Gibbs sampler for `{Sigma,m}`.

```
bayesmh y x i.cat, likelihood(probit)
>                  prior(y:x _cons,     normal(0, 1000))
>                  prior(y:1.cat 2.cat, mvnormal0(2,{Sigma,m}))
>                  prior({Sigma,m}, iwishart(2,10,V))
>                  block({Sigma,m}, gibbs)
```

In general, Gibbs sampling, when available, is useful for covariance matrices because MH sampling has low efficiency for sampling positive-definite symmetric matrices. In a multivariate normal regression, the inverse Wishart distribution is a conjugate prior for the covariance matrix and thus inverse Wishart is the most common prior specification for a covariance matrix parameter. If an inverse-Wishart prior (`iwishart()`) is used for a covariance matrix, you can specify Gibbs sampling for the covariance matrix. You can do so by placing the matrix in a separate block and specifying the `gibbs` suboption in that block, as we showed above. Using Gibbs sampling for the covariance matrix usually greatly improves the sampling efficiency.

Adaptation of the MH algorithm

The MH algorithm simulates Markov chains by generating small moves or jumps from the current parameter values (or current state) according to the proposal distribution. At each iteration of the algorithm, the proposed new state is accepted with a probability that is calculated based on the newly proposed state and the current state. The choice of a proposal distribution is crucial for the mixing properties of the Markov chain, that is, the rate at which the chain explores its stationary distribution. (In a Bayesian context, a Markov chain state is a vector of model parameters, and a stationary distribution is the target posterior distribution.) If the jumps are too small, almost all moves will be accepted. If the jumps are too large, almost all moves will be rejected. Either case will cause the chain to explore the entire posterior domain slowly and will thus lead to poor mixing. Adaptive MH algorithms try to tune the proposal distribution so that some optimal AR is achieved (Haario, Saksman, and Tamminen [2001]; Roberts and Rosenthal [2009]; Andrieu and Thoms [2008]).

In the random-walk MH algorithm, the proposal distribution is a Gaussian distribution with a zero mean and is completely determined by its covariance matrix. It is useful to represent the proposal covariance matrix as a product of a (scalar) scale factor and a positive-definite scale matrix. Gelman, Gilks, and Roberts (1997) show that the optimal scale matrix is the true covariance matrix of the target distribution, and the optimal scale factor is inversely proportional to the number of parameters. Therefore, in the ideal case when the true covariance matrix is available, it can be used as a proposal covariance and an MCMC adaptation can be avoided altogether. In practice, the true covariance is rarely known and the adaptation is thus unavoidable.

In the `bayesmh` command, the scale factor and the scale matrix that form the proposal covariance are constantly tuned during the adaptation phase of an MCMC so that the current AR approaches some predefined value.

You can use `scale()`, `covariance()`, and `adaptation()` options to control adaptation of the MH algorithm. The TAR is controlled by option `adaptation(tarate())`. The initial scale factor and scale

matrix can be modified using the scale() and covariance() options. In the presence of blocks of parameters, these options can be specified separately for each block within the block() option. At each adaptation step, a new scale matrix is formed as a mixture (a linear combination) of the previous scale matrix and the current empirical covariance matrix of model parameters. The mixture of the two matrices is controlled by option adaptation(beta()). A positive adaptation(beta()) is recommended to have a more stable scale matrix between adaptation periods. The adaptation lasts until the maximum number adaptation(every())×adaptation(maxiter()) of adaptive iterations is reached or until adaptation(tarate()) is reached within the adaptation(tolerance()) limit. The default for maxiter() depends on the specified burn-in and adaptation(every()) and is computed as $\max\{25, \texttt{floor(burnin()/adaptation(every()))}\}$. The default for adaptation(every()) is 100. If you change the default values of these parameters, you may want to increase the burnin() to be as long as the specified adaptation period so that adaptation is finished before the final simulated sample is obtained. (There are adaptation regimes in which adaptation is performed during the simulation phase as well, such as continuous adaptation.) Two additional adaptation options, adaptation(alpha()) and adaptation(gamma()) control the AR and the adaptation rate. For a detailed description of the adaptation process, see *Adaptive random-walk Metropolis–Hastings* in [BAYES] **intro** and *Adaptive MH algorithm* in *Methods and formulas*.

Specifying initial values

When exploring convergence of MCMC, it may be useful to try different initial values to verify that the convergence is unaffected by starting values.

There are two different ways to specify initial values of model parameters in bayesmh. First is by specifying an initial value when declaring a model parameter. Second is by specifying an initial value in the initial() option. Initial values for matrix model parameters may be specified only in the initial() option.

For example, below we initialize variance parameter {var} with value of 1 using two equivalent ways, as follows:

```
. bayesmh y x, likelihood(normal({var=1})) ...
```

or

```
. bayesmh y x, likelihood(normal({var})) initial({var} 1) ...
```

If both initial-value specifications are used, initial values specified in the initial() option override any initial values specified during parameter declaration for the corresponding parameters.

You can initialize multiple parameters with the same value by supplying a list of parameters by using any of the specifications described in *Referring to model parameters* to initial(). For example, to initialize all regression coefficients from equations y1 and y2 to zero, you can type

```
. bayesmh ..., initial({y1:} {y2:} 0) ...
```

By default, if no initial value is specified and option nomleinitial is not used, bayesmh uses MLEs, whenever available, as starting values for model parameters.

For example, for the previous regression model, bayesmh uses regression coefficients and mean squared error from linear regression regress y x as the respective starting values for the regression model parameters and variance parameter {var}.

If MLE is not available and an initial value is not provided, then a scalar model parameter is initialized with 1 for positive parameters and 0 for other parameters, and a matrix model parameter is initialized with an identity matrix. Note, however, that this default initialization is not guaranteed to correspond to the feasible state for the specified posterior model; that is, posterior probability of the

initial state can be 0. When initial values are not feasible, bayesmh makes 500 random attempts to find a feasible initial-value vector. An initial-value vector is feasible if it corresponds to a state with positive posterior probability. If feasible initial values are not found after 500 attempts, bayesmh will issue the following error:

```
could not find feasible initial state
r(498);
```

You may use the search() option to modify the default settings for finding feasible initial values.

Summarizing and reporting results

As we discussed in *Checking model specification*, it is useful to verify the details about your model specification before estimation. The dryrun model will display the model summary without estimation. Once you are satisfied with the model specification, you can use the nomodelsummary option during estimation to suppress a potentially long model summary from the final output.

In the presence of blocking, you may also display the information about specified blocks by using the blocksummary option.

Simulation may be time consuming for large datasets and for models with many parameters. You can specify one of dots or dots(#) option to display a dot every # iterations to see the simulation progress.

Posterior summaries and credible intervals

After simulation, bayesmh reports various summaries about the model parameters in the output table. The summaries include posterior mean and median estimates, estimates of posterior standard deviation and MCSE, and credible intervals. By default, 95% equal-tailed credible intervals are reported. You can use the hpd option to request HPD intervals instead. You can also use the clevel() option to change the default credible level.

bayesmh provides two estimators for MCSE: one using ESS and one using batch means. The ESS-based estimator is the default. You can request the batch-means estimator by specifying the batch() option. Options corrlag() and corrtol() affect how ESS is estimated when computing MCSE; see *Methods and formulas* in [BAYES] **bayesstats summary** for details.

Saving MCMC results

In addition to postestimation summaries, bayesmh saves simulation results containing MCMC samples for all model parameters to a temporary Stata dataset. You can use the saving() option to save simulation results to a permanent dataset. In fact, if you want to store your estimation results in memory or save them to a disk, you must specify the saving() option with bayesmh; see *Storing estimation results after bayesmh* in [BAYES] **bayesmh postestimation**. You can also specify the saving() option on replay.

```
. bayesmh, saving(...)
```

By default, all model parameters are saved in the dataset. If desired, you can exclude some of the parameters from the dataset by specifying the exclude() option. Beware that you will not be able to obtain posterior summaries for these parameters or use them in any way in your analysis, because no simulation results will be available for them.

Convergence of MCMC

As we discuss in *Convergence diagnostics of MCMC* in [BAYES] **intro**, checking convergence is an essential step of any MCMC simulation. Bayesian inference based on an MCMC sample is only valid if the Markov chain has converged and the sample is drawn from the desired posterior distribution. It is important to emphasize that we need to verify the convergence for all model parameters and not only for a subset of parameters of interest. Another difficulty in accessing convergence of MCMC is the lack of a single conclusive convergence criterion. The diagnostic usually involves checking for several necessary (but not necessarily sufficient) conditions for convergence. In general, the more aspects of the MCMC sample you inspect, the more reliable your results are.

An MCMC is said to have converged if it reached its stationary distribution. In the Bayesian context, the stationary distribution is the true posterior distribution of model parameters. Provided that the considered Bayesian model is well specified (that is, it defines a proper posterior distribution of model parameters), the convergence of MCMC is determined by the properties of its sampling algorithm.

The main component of the MH algorithm, or any MCMC algorithm, is the number of iterations it takes for the chain to approach its stationary distribution or for the MCMC sample to become representative of a sample from the true posterior distribution of model parameters. The period during which the chain is converging to its stationary distribution from its initial state is called the burn-in period. The iterations of the burn-in period are discarded from the MCMC sample used for analysis. Another complication is that adjacent observations from the MCMC sample tend to be positively correlated; that is, autocorrelation is typically present in MCMC samples. In theory, this should not be a problem provided that the MCMC sample size is sufficiently large. In practice, the autocorrelation in the MCMC sample may be so high that obtaining a sample of the necessary size becomes infeasible and finding ways to reduce autocorrelation becomes important.

Two aspects of the MH algorithm that affect the length of the burn-in (and convergence) are the starting values of model parameters or, in other words, a starting state and a proposal distribution. `bayesmh` has the default burn-in of 2,500 iterations, but you can change it by specifying the `burnin()` option. `bayesmh` uses a Gaussian normal distribution with a zero mean and a covariance matrix that is updated with current sample values during the adaptation period. You can control the proposal distribution by changing the initial scale factor in option `scale()` and an initial scale matrix in option `covariance()`; see *Adaptation of the MH algorithm*.

For the starting values, `bayesmh` uses MLEs whenever available, but you can specify your own initial values in option `initial()`; see *Specifying initial values*. Good initial values help to achieve fast convergence of MCMC and bad initial values may slow convergence down. A common approach for eliminating the dependence of the chain on the initial values is to discard an initial part of the simulated sample: a burn-in period. The burn-in period must be sufficiently large for a chain to "forget" its initial state and approach its stationary distribution or the desired posterior distribution.

There are some researchers (for example, Geyer [2011]) who advocate that any starting point in the posterior domain is equally good and there should be no burn-in. While this is a sensible approach for a fixed, nonadaptive MH algorithm, it may not be as sensible for an adaptive MH algorithm because the proposal distribution is changing (possibly drastically) during the adaptation period. Therefore, adaptive iterations are better discarded from the analysis MCMC sample and thus it is recommended that the burn-in period is at least as long as the adaptation period. (There are adaptive regimes such as continuous adaptation in which adaptation continues after the burn-in period as well.)

In addition to fast convergence, an "ideal" MCMC chain will also have good mixing (or low autocorrelation). A good mixing can be viewed as a rapid movement of the chain around the parameter space. High autocorrelation in MCMC and consequently low efficiencies are usually indications of bad mixing. To improve the mixing of the chain, you may need to improve the efficiency of the algorithm (see *Improving efficiency of the MH algorithm—blocking of parameters*) or sometimes reparameterize

your model. In the presence of high autocorrelation, you may also consider subsampling or thinning the chain, option `thinning()`, to reduce autocorrelation, but this may not always be the best approach.

Even when the chain appears to have converged and has good mixing, you may still have a case of pseudoconvergence, which is common for multimodal posterior distributions. Specifying different sets of initial values may help detect pseudoconvergence.

For more information about convergence of MCMC and its diagnostics, see *Convergence diagnostics of MCMC* in [BAYES] **intro**, [BAYES] **bayesgraph**, and [BAYES] **bayesstats ess**.

In what follows, we concentrate on demonstrating various specifications of `bayesmh`, which may not always correspond to the optimal Bayesian analysis for the considered problem. In addition, although we skip checking convergence for some of our models to keep the exposition short, it is important that you always check the convergence of all parameters in your model in your analysis before you make any inferential conclusions. If you are also interested in any functions of model parameters, you must check convergence of those functions as well.

Getting started examples

We will use the familiar `auto.dta` for our introductory examples. This dataset contains information about 74 automobiles, including their make and model, price, and mileage (variable `mpg`). In our examples, we are interested in estimating the average fuel efficiency as measured by the `mpg` variable and its relationship with other automobile characteristics such as `weight`.

```
. use http://www.stata-press.com/data/r14/auto
(1978 Automobile Data)
. describe mpg weight length
              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------
mpg             int     %8.0g                 Mileage (mpg)
weight          int     %8.0gc                Weight (lbs.)
length          int     %8.0g                 Length (in.)
```

Mean of a normal distribution with a known variance

We start with an example of estimating the mean of a normal distribution with known variance. This corresponds to a constant-only normal linear regression with an unknown constant (or intercept) and a known error variance.

Suppose we are interested in estimating the average fuel efficiency as measured by the `mpg` variable. For illustration purposes, let's assume that `mpg` is normally distributed. We are interested in estimating its mean. Let's also assume that we know the variance of `mpg` and it is 36.

▷ Example 1: Noninformative prior for the mean when variance is known

To fit a Bayesian model, we must specify the likelihood model and priors for all model parameters. We have only one parameter in this model—the constant (or the mean) of `mpg`. We first consider a noninformative prior for the constant: the prior distribution with a density equal to one.

To specify this model in `bayesmh`, we use the likelihood specification `mpg, likelihood(normal(36))` and the prior specification `prior({mpg:_cons}, flat)`, where suboption `flat` requests a flat prior distribution with the density equal to one. This prior is an improper prior

for the constant—the prior distribution does not integrate to one. {mpg:_cons}, the constant or the mean of mpg, is the only model parameter and is declared automatically by bayesmh as a part of the regression function. (For this reason, we also did not need to specify the mean of the normal() distribution in the likelihood specification.) All other simulation and reporting options are left at default.

Because bayesmh uses MCMC sampling, a stochastic procedure, to obtain results, we specify a random-number seed (for example, 14) for reproducibility of results.

```
. set seed 14
. bayesmh mpg, likelihood(normal(36)) prior({mpg:_cons}, flat)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},36)
Prior:
  {mpg:_cons} ~ 1 (flat)
------------------------------------------------------------------------------

Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .4161
Log marginal likelihood = -233.96144            Efficiency       =      .2292

------------------------------------------------------------------------------
             |                                                 Equal-tailed
         mpg |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
       _cons |  21.29812    .703431   .014693   21.28049    19.93155   22.69867
------------------------------------------------------------------------------
```

bayesmh first reports the summary of the model. The likelihood model specified for mpg is normal with mean {mpg:_cons} and fixed variance of 36. The prior for {mpg:_cons} is flat or completely noninformative.

Our model is very simple, so its summary is very short. For other models, the model summary may get very long. You can use the nomodelsummary option to suppress it from the output. It is useful, however, to review the model summary before estimation for models with many parameters and complicated specifications. You can use the dryrun option to see the model summary without estimation. Once you verified the correctness of your model specification, you can specify nomodelsummary during estimation.

Next, bayesmh reports the header including the title for the fitted model, the used MCMC algorithm, and various numerical summaries of the sampling procedure. bayesmh performed 12,500 MCMC iterations, of which 2,500 were discarded as burn-in iterations and the next 10,000 iterations were kept in the final MCMC sample. An overall AR is 0.42, meaning that 42% out of 10,000 proposal parameter values were excepted by the algorithm. This is a good AR for the MH algorithm. Values below 10% may be a cause for concern and may indicate problems with convergence of MCMC. Very low ARs may also mean high autocorrelation. The efficiency is 0.23 and is also considered good for the MH algorithm. Efficiencies below 1% should be investigated further and would require further tuning of the algorithm and possibly revisiting the considered model.

Finally, bayesmh reports an estimation table that includes the posterior mean, posterior standard deviation, MCMC standard error (MCSE), posterior median, and the 95% credible interval.

The estimated posterior mean for {mpg:_cons} is 21.298 with a posterior standard deviation of 0.70. The efficiency of the estimator of the posterior mean is about 23%, which is relatively high for the random-walk MH sampling. In general, you should expect to see lower efficiencies from this algorithm for models with more parameters. The MCSE, which is an approximation of the error in estimating the true posterior mean, is about 0.015. Therefore, provided that the MCMC simulation has converged, the posterior mean of the constant is accurate to 1 decimal position, 21.3. If you want an estimation precision of, say, 2 decimal positions, you may need to increase the MCMC sample size 10^1 times; that is, use `mcmcsize(100000)`.

The estimated posterior mean and medians are very close, suggesting that the posterior distribution of {mpg:_cons} may be symmetric. In fact, the posterior distribution of a mean in this model is known to be a normal distribution.

According to the reported 95% credible interval, the probability that the mean of `mpg` is between 19.9 and 22.7 is about 0.95. You can use the `clevel()` option to change the default credible level; also see [BAYES] **set clevel**.

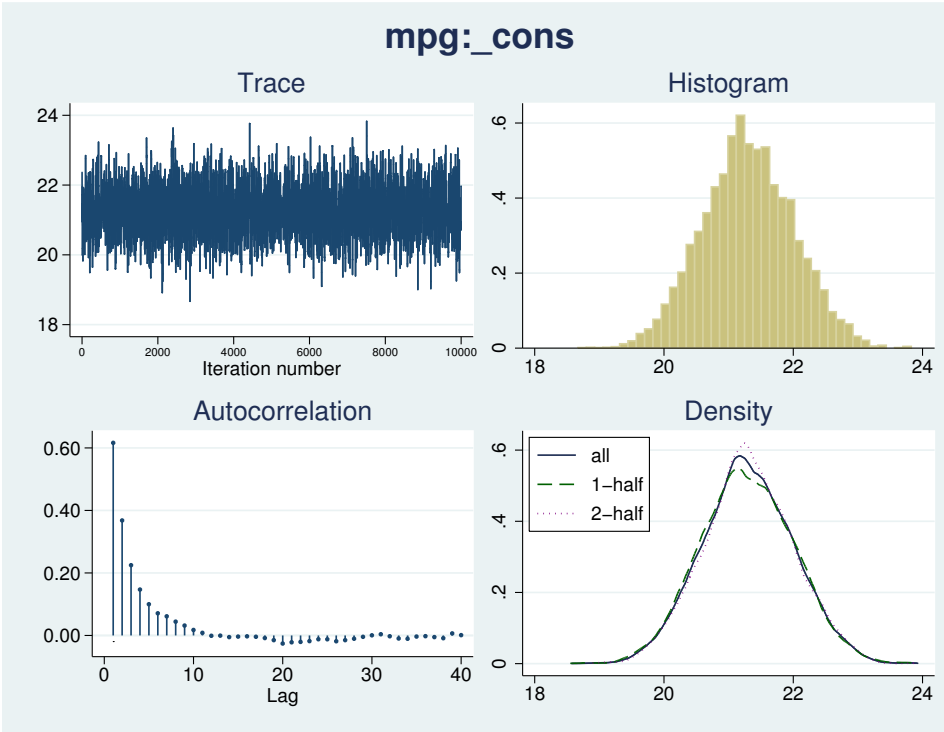
Because we used a completely noninformative prior, our results should be the same as frequentist results. In this Bayesian model, the posterior distribution of the constant parameter is known to be normal with a mean equal to the sample average. In the frequentist domain, the MLE of the constant is also the sample average, so the posterior mean estimate and the MLE should be the same in this model.

The sample average of `mpg` is 21.2973. Our posterior mean estimate is 21.298, which is very close. The reason it is not exactly the same is because we estimated the posterior mean of the constant based on an MCMC sample simulated from its posterior distribution instead of using the known formula. Closed-form expressions for posterior mean estimators are available only for some Bayesian models. In general, posterior distributions of parameters are unknown and posterior summaries may only be estimated from the MCMC samples of parameters.

In practice, we must verify the convergence of MCMC before making any inferential conclusions about the obtained results.

We start by looking at various graphical diagnostics as produced by `bayesgraph diagnostics`.

```
. bayesgraph diagnostics {mpg:_cons}
```



The trace plot represents a "perfect" trace plot. It does not exhibit any trends, and it traverses the distribution quickly. The chain is centered around 21.3, but also explores the portions of the distribution where the density is low, which is indicative of good mixing of the chain. The autocorrelation dies off very quickly. The posterior distribution looks normal. The kernel density estimates based on the first and second halves of the sample are very similar to each other and are close to the overall density estimate. We can see that MCMC converged and mixes well. See [BAYES] **bayesgraph** for details about this command.

See *Convergence diagnostics of MCMC* for more discussion about convergence of MCMC.

◁

▷ Example 2: Informative prior for the mean when variance is known

In example 1, we used a noninformative prior for `{mpg:_cons}`. Here, we consider a conjugate normal prior for `{mpg:_cons}`. A parameter is said to have a conjugate prior when the corresponding posterior belongs to the same family as the prior. In our example, if we assume a normal prior for the constant, its posterior is known to be normal too.

Suppose that based on previous studies, the distribution of the mean mileage was found to be normal with mean of 25 and variance of 10. We change the `flat` prior in `bayesmh`'s `prior()` option from example 1 with `normal(25,10)`.

```
. set seed 14
. bayesmh mpg, likelihood(normal(36)) prior({mpg:_cons}, normal(25,10))
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},36)
Prior:
  {mpg:_cons} ~ normal(25,10)
------------------------------------------------------------------------------

Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .4169
Log marginal likelihood = -236.71627            Efficiency       =      .2293

------------------------------------------------------------------------------
             |                                                 Equal-tailed
         mpg |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
       _cons |   21.47952   .6820238   .014243   21.47745   20.13141   22.82153
------------------------------------------------------------------------------
```

Compared with example 1, our results change only slightly: the estimates of posterior mean is 21.48 and of posterior standard deviation is 0.68. The 95% credible interval is $[20.1, 22.82]$.

The reason we obtained such similar results is that our specified prior is in close agreement with what we observed in this sample. The prior mean of 25 with a standard deviation of $\sqrt{10} = 3.16$ overlaps greatly with what we observe for `{mpg:_cons}` in the data.

If we place a very strong prior on the value for the mean by, for example, substantially decreasing the variance of the normal prior distribution,

```
. set seed 14
. bayesmh mpg, likelihood(normal(36)) prior({mpg:_cons}, normal(25,0.1))
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},36)
Prior:
  {mpg:_cons} ~ normal(25,0.1)
------------------------------------------------------------------------------

Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .4194
Log marginal likelihood =  -246.2939            Efficiency       =      .2352

------------------------------------------------------------------------------
             |                                                 Equal-tailed
         mpg |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
       _cons |   24.37211    .292777   .006037   24.36588   23.79701   24.94403
------------------------------------------------------------------------------
```

we obtain very different results. Now the posterior mean and standard deviation estimates are very close to their prior values, as one would expect with such strong prior information.

Which results are correct? The answer depends on how confident we are in our prior knowledge. If we previously observed many samples in which the average mileage for the considered population of cars was essentially 25, our last results are consistent with this and the information about the mean of {mpg:_cons} contained in the observed sample was not enough to counteract our belief. If, on the other hand, we had no prior information about the mean mileage, then we would use a noninformative or mildly informative prior in our Bayesian analysis. Also, if we believe that our observed data should have more weight in our analysis, we would not specify a very strong prior.

◁

▷ Example 3: Noninformative normal prior for the mean when variance is known

In example 1, we used a completely noninformative, flat prior for {mpg:_cons}. In example 2, we considered a conjugate normal prior for {mpg:_cons}. We also saw that by varying the variance of the normal prior distribution, we could control the "informativeness" of our prior. The larger the variance, the less informative the prior. In fact, if we let the variance approach infinity, we will arrive at the same posterior distribution of the constant as with the flat prior.

For example, if we specify a very large variance in the normal prior,

```
. set seed 14
. bayesmh mpg, likelihood(normal(36)) prior({mpg:_cons}, normal(0,1000000))
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},36)
Prior:
  {mpg:_cons} ~ normal(0,1000000)
------------------------------------------------------------------------------

Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .4161
Log marginal likelihood = -241.78836            Efficiency       =      .2292

------------------------------------------------------------------------------
             |                                                 Equal-tailed
         mpg |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
       _cons |  21.29812   .7034313   .014693   21.28049   19.93155   22.69868
------------------------------------------------------------------------------
```

we will obtain results that are very similar to the results from example 1 with the flat prior.

We do not need to use such an extreme value of the variance for the results to become less sensitive to the prior specification. As we saw in example 2, using the variance of 10 in that example resulted in very little impact of the prior on the results.

◁

Mean of a normal distribution with an unknown variance

Let's now consider the case where both mean and variance of the normal distribution are unknown.

▷ Example 4: Noninformative Jeffreys prior when mean and variance are unknown

A noninformative prior commonly used for the normal model with unknown mean and variance is the Jeffreys prior, under which the prior for the mean is flat and the prior for the variance is the reciprocal of the variance. We use the same `flat` prior for `{mpg:_cons}` as in example 1 and specify the `jeffreys` prior for `{var}` using a separate `prior()` statement.

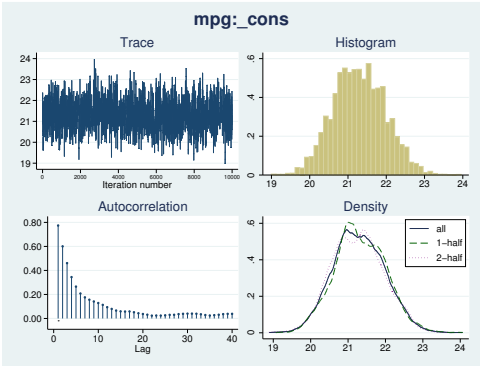
```
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, flat) prior({var}, jeffreys)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
Priors:
  {mpg:_cons} ~ 1 (flat)
        {var} ~ jeffreys
------------------------------------------------------------------------------
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2668
                                                Efficiency:  min =     .09718
                                                             avg =      .1021
Log marginal likelihood =   -234.645                         max =      .1071
------------------------------------------------------------------------------
             |                                                Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
       _cons |   21.29222   .6828864   .021906   21.27898   19.99152   22.61904
-------------+----------------------------------------------------------------
         var |   34.76572    5.91534   .180754   34.18391    24.9129   47.61286
------------------------------------------------------------------------------
```

Because we used a noninformative prior, our results should be similar to the frequentist results apart from simulation uncertainty. Compared with example 1, the average efficiency of the MH algorithm decreased to 10%, as is expected with more parameters, but is still considered a good efficiency for the MH algorithm.

The posterior mean estimate of `{mpg:_cons}` is close to the OLS estimate of 21.297, and the posterior standard deviation is close to the standard error of the OLS estimate 0.673. MCSE is slightly larger than in example 1 because we have lower efficiency. If we wanted to make MCSE smaller, we could increase our MCMC sample size. The posterior mean estimate of `{var}` agrees with the MLE of the variance 33.02, but we would not expect the two to be necessarily the same. We estimated the posterior mean of `{var}`, not the posterior mode, and because posterior distribution of `{var}` is not symmetric, the two estimates may not be the same.

Again, as with any MCMC analysis, we must verify the convergence of our MCMC sample before we can trust our results.

```
. bayesgraph diagnostics _all
```



Graphical diagnostic plots do not show any signs of nonconvergence for either of the parameters.

Recall that to access convergence of MCMC, we must explore convergence for all model parameters. ◁

▷ Example 5: Informative conjugate prior when mean and variance are unknown

For a normal distribution with unknown mean and variance, the informative conjugate prior is a normal prior for the mean and an inverse-gamma prior for the variance. Specifically, if $y \sim N(\mu, \sigma^2)$, then the informative conjugate prior for the parameters is

$$\mu|\sigma^2 \sim N(\mu_0, \sigma^2)$$
$$\sigma^2 \sim \text{InvGamma}(\nu_0/2, \nu_0\sigma_0^2/2)$$

where μ_0 is the prior mean of the normal distribution and ν_0 and σ_0^2 are the prior degrees of freedom and prior variance for the inverse-gamma distribution. Let's assume $\mu_0 = 25$, $\nu_0 = 10$, and $\sigma_0^2 = 30$.

Notice that in the specification of the prior for {mpg:_cons}, we specify the parameter {var} as the variance of the normal distribution. We use igamma(5,150) as the prior for the variance parameter {var}.

```
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, normal(25,{var}))
> prior({var}, igamma(5,150))
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
Priors:
  {mpg:_cons} ~ normal(25,{var})
        {var} ~ igamma(5,150)
------------------------------------------------------------------------------
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .1971
                                                Efficiency:  min =     .09822
                                                             avg =     .09923
Log marginal likelihood = -237.77006                         max =      .1002
------------------------------------------------------------------------------
             |                                                Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
       _cons |    21.314   .6639278    .02097   21.29516   20.08292   22.63049
-------------+----------------------------------------------------------------
         var |   33.54699   5.382861   .171756   32.77635   24.88107    46.0248
------------------------------------------------------------------------------
```

Compared with example 4, the variance is slightly smaller, but the results are still very similar.

Example 6: Noninformative inverse-gamma prior when mean and variance are unknown

The Jeffreys prior for the variance from example 4 can be viewed as a limiting case of an inverse-gamma distribution with the degrees of freedom approaching zero.

Indeed, if we replace the `jeffreys` prior in example 4 with an inverse-gamma distribution with very small degrees of freedom,

```
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, flat)
> prior({var}, igamma(0.0001,0.0001))
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
Priors:
  {mpg:_cons} ~ 1 (flat)
        {var} ~ igamma(0.0001,0.0001)
------------------------------------------------------------------------------
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2668
                                                Efficiency:  min =     .09718
                                                             avg =      .1021
Log marginal likelihood = -243.85656                         max =      .1071
------------------------------------------------------------------------------
             |                                                Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
       _cons |   21.29223   .6828811   .021905   21.27899   19.99154   22.61903
-------------+----------------------------------------------------------------
         var |   34.76569   5.915305   .180753   34.18389   24.91294   47.61275
------------------------------------------------------------------------------
```

we obtain results that are very close to the results from example 4.

◁

Simple linear regression

In this example, we consider a simple linear regression with one independent variable. We continue with `auto.dta`, but this time we regress `mpg` on a rescaled covariate `weight`.

```
. use http://www.stata-press.com/data/r14/auto
. replace weight = weight/100
variable weight was int now float
(74 real changes made)
```

We will have three model parameters: the slope and the intercept for the linear predictor and the variance parameter for the error term. Regression parameters, `{mpg:weight}` and `{mpg:_cons}`, will be declared implicitly by `bayesmh`, but we will need to explicitly specify the variance parameter `{var}`. We will also need to assign appropriate priors for all parameters.

▷ Example 7: Noninformative prior for regression coefficients and variance

As in our earlier examples, we start with a noninformative prior. For this model, a common noninformative prior for the parameters includes flat priors for {mpg:weight} and {mpg:_cons} and a Jeffreys prior for {var}.

```
. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, flat)  prior({var}, jeffreys)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:weight _cons} ~ 1 (flat)                                          (1)
               {var} ~ jeffreys
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_mpg.
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .1768
                                                Efficiency:  min =     .04557
                                                             avg =     .06624
Log marginal likelihood = -198.14389                         max =     .07961
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
mpg						
weight	-.6019838	.0512557	.001817	-.6018433	-.7015638	-.5021532
_cons	39.47227	1.589082	.058601	39.49735	36.26465	42.43594
var	12.22248	2.214665	.10374	11.92058	8.899955	17.47372

Our model summary shows the likelihood model for mpg, flat priors for the two regression coefficients, and a Jeffreys prior for the variance parameter. Now that we have a covariate in the model, the mean of the normal distribution is labeled as xb_mpg to emphasize that it is now a linear combination of independent variables. Regression coefficients involved in the linear predictor are marked with (1) on the right.

The results are again very similar to the frequentist results. Posterior mean estimates of the coefficients are very similar to the OLS estimates obtained by using regress below. Posterior standard deviations are similar to the standard errors from regress.

```
. regress mpg weight
```

Source	SS	df	MS	Number of obs	=	74
				F(1, 72)	=	134.62
Model	1591.99021	1	1591.99021	Prob > F	=	0.0000
Residual	851.469254	72	11.8259619	R-squared	=	0.6515
				Adj R-squared	=	0.6467
Total	2443.45946	73	33.4720474	Root MSE	=	3.4389

mpg	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
weight	-.6008687	.0517878	-11.60	0.000	-.7041058	-.4976315
_cons	39.44028	1.614003	24.44	0.000	36.22283	42.65774

◁

▷ Example 8: Conjugate prior for regression coefficients and variance

In this example, we use a conjugate prior for the parameters, which corresponds to normal priors for {mpg:weight} and {mpg:_cons} and an inverse-gamma prior for {var},

$$\begin{aligned}\beta_{\text{weight}}|\sigma^2 &\sim N(\mu_{\text{weight}}, \sigma^2)\\ \beta_{\text{cons}}|\sigma^2 &\sim N(\mu_{\text{cons}}, \sigma^2)\\ \sigma^2 &\sim \text{InvGamma}(\nu_0/2, \nu_0\sigma_0^2/2)\end{aligned}$$

where regression coefficients have different means but equal variances. μ_{weight} and μ_{cons} are the prior means of the normal distributions, and ν_0 and σ_0^2 are the prior degrees of freedom and prior variance for the inverse-gamma distribution. Let's assume $\mu_{\text{weight}} = -0.5$, $\mu_{\text{cons}} = 50$, $\nu_0 = 10$, and $\sigma_0^2 = 10$.

```
. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:weight}, normal(-0.5,{var}))
> prior({mpg:_cons},  normal(40,{var}))
> prior({var}, igamma(5,50))
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:weight} ~ normal(-0.5,{var})                                        (1)
   {mpg:_cons} ~ normal(40,{var})                                          (1)
         {var} ~ igamma(5,50)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_mpg.
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .1953
                                                Efficiency:  min =     .05953
                                                             avg =     .06394
Log marginal likelihood = -202.74075                         max =     .06932
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
mpg						
weight	-.6074375	.0480685	.001916	-.6078379	-.6991818	-.5119767
_cons	39.65274	1.499741	.05696	39.63501	36.59486	42.47547
var	11.696	1.929562	.079083	11.52554	8.570938	16.26954

For this mildly informative prior, our regression coefficients are still very similar to the results obtained using the noninformative prior in example 7, but the variance estimate is slightly smaller.

◁

▷ Example 9: Zellner's g prior for regression coefficients

In example 8, we assumed that `{mpg:weight}` and `{mpg:_cons}` are independent a priori. We can specify Zellner's g prior (Zellner 1986), often used for regression coefficients in a multiple regression, which allows correlation between the regression coefficients.

The prior for the coefficients can be written as

$$\boldsymbol{\beta}|\sigma^2 \sim \text{MVN}(\boldsymbol{\mu}_0, g\sigma^2(X'X)^{-1})$$

where $\boldsymbol{\beta}$ is a vector of coefficients, $\boldsymbol{\mu}_0$ is the vector of prior means, g is the prior degrees of freedom, and X is the design matrix. Let's, for example, use $g = 30$ and $\boldsymbol{\mu}_0 = (\mu_{\text{weight}}, \mu_{\text{cons}}) = (-0.5, 40)$. Zellner's g prior is not strictly a conventional Bayesian prior because it depends on the data.

In `bayesmh`, we can use prior `zellnersg()` to specify this prior. The first argument for this prior is the dimension (2), the second argument is the degrees of freedom (30), the next parameters are prior means (-0.5 and 40), and the last parameter is the name of the parameter corresponding to the variance term (`{var}`).

```
. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, zellnersg(2,30,-0.5,40,{var}))
> prior({var},  igamma(5,50))
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:weight _cons} ~ zellnersg(2,30,-0.5,40,{var})                    (1)
              {var} ~ igamma(5,50)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_mpg.
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2576
                                                Efficiency:  min =      .05636
                                                             avg =      .08661
Log marginal likelihood =  -201.1662                         max =       .1025
------------------------------------------------------------------------------
             |                                                Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
      weight |  -.6004123   .0510882   .001595  -.5998094  -.7040552  -.5058665
       _cons |   39.55017   1.590016   .050051   39.49377   36.56418   42.79701
-------------+----------------------------------------------------------------
         var |   12.18757   2.038488   .085865   11.90835   8.913695   16.88978
------------------------------------------------------------------------------
```

The results are now closer to the results using noninformative prior obtained in example 7, because we are introducing some information from the observed data by using $(X'X)^{-1}$.

◁

▷ Example 10: Specifying expressions as distributional arguments

We can actually reproduce what prior `zellnersg()` does in example 9 manually.

First, we need to create a matrix that contains $(X'X)^{-1}$, `S`.

```
. matrix accum xTx = weight
(obs=74)
. matrix S = invsym(xTx)
```

Then, we can use the multivariate normal prior `mvnormal()` with the variance specified as an expression `30*var*S`.

```
. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, mvnormal(2,-0.5,40,30*{var}*S))
> prior({var},  igamma(5,50))
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:weight _cons} ~ mvnormal(2,-0.5,40,30*{var}*S)                      (1)
                {var} ~ igamma(5,50)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_mpg.
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2576
                                                Efficiency:  min =     .05636
                                                             avg =     .08661
Log marginal likelihood =  -201.1662                         max =      .1025
------------------------------------------------------------------------------
             |                                                Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
      weight |  -.6004123   .0510882   .001595  -.5998094  -.7040552  -.5058665
       _cons |   39.55017   1.590016   .050051   39.49377   36.56418   42.79701
-------------+----------------------------------------------------------------
         var |   12.18757   2.038488   .085865   11.90835   8.913695   16.88978
------------------------------------------------------------------------------
```

We obtain results identical to those from example 9.

◁

Multiple linear regression

For a detailed example of a multiple linear regression, see *Overview example* in [BAYES] **bayes**.

Improving efficiency of MH sampling

In this section, we demonstrate how one can improve efficiency of the MH algorithm by using blocking of parameters and Gibbs sampling, whenever available. We continue with our simple linear regression of `mpg` on rescaled `weight` from *Simple linear regression*, but we use different values for the parameters of prior distributions. We also assume that regression coefficients and the variance parameter are independent a priori. We use the `blocksummary` option to include a summary about each block.

Example 11: First simulation run

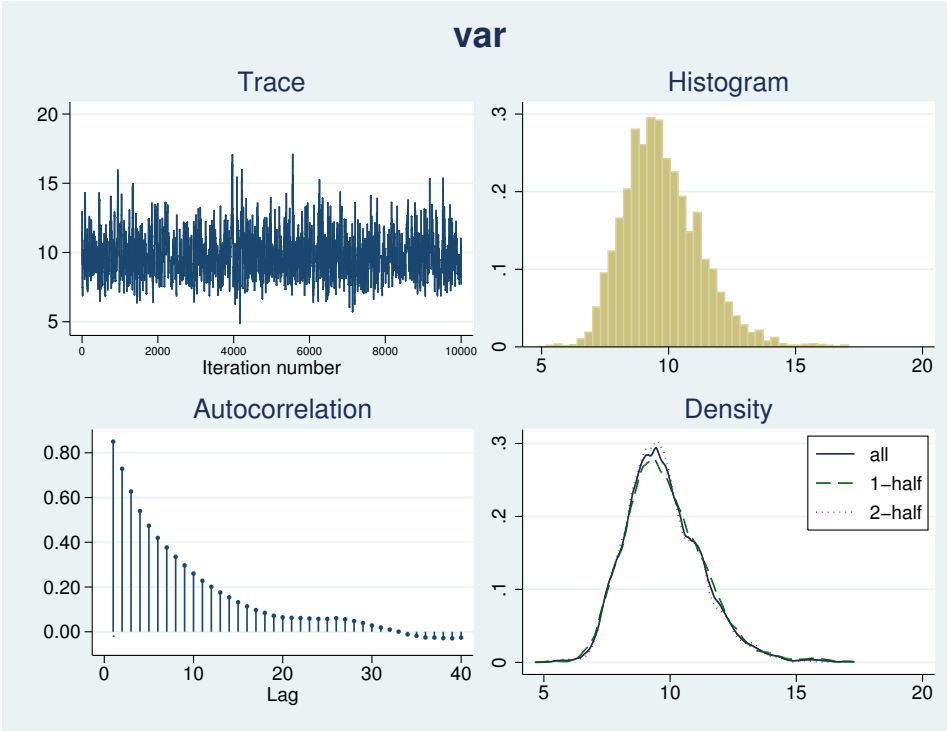
Our first simulation is performed using the default settings for the algorithm. Specifically, all three model parameters are placed in one simulation block and are updated simultaneously, as our block summary indicates.

```
. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, normal(0,100))
> prior({var}, igamma(10,10)) blocksummary
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:weight _cons} ~ normal(0,100)                                        (1)
                {var} ~ igamma(10,10)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_mpg.
Block summary
------------------------------------------------------------------------------
   1:  {mpg:weight _cons} {var}
------------------------------------------------------------------------------
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2432
                                                Efficiency:  min =     .06871
                                                             avg =     .08318
Log marginal likelihood = -226.63723                         max =     .09063
------------------------------------------------------------------------------
             |                                                 Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
      weight |  -.5759855   .0471288   .001569  -.5750919  -.6676517  -.4868595
       _cons |   38.65481   1.468605   .048784   38.70029   35.88062   41.49839
-------------+----------------------------------------------------------------
         var |   9.758003   1.514112   .057762   9.601339   7.302504   13.13189
------------------------------------------------------------------------------
```

The mean estimates based on the simulated sample are {mpg:weight}$= -0.58$, {mpg:_cons}$=$ 38.65, and {var}$=$ 9.8. The MH algorithm achieves an overall AR of 24% and an average efficiency of about 8%.

Our next step is to perform a visual inspection of the convergence of the chain.

```
. bayesgraph diagnostics {var}
```



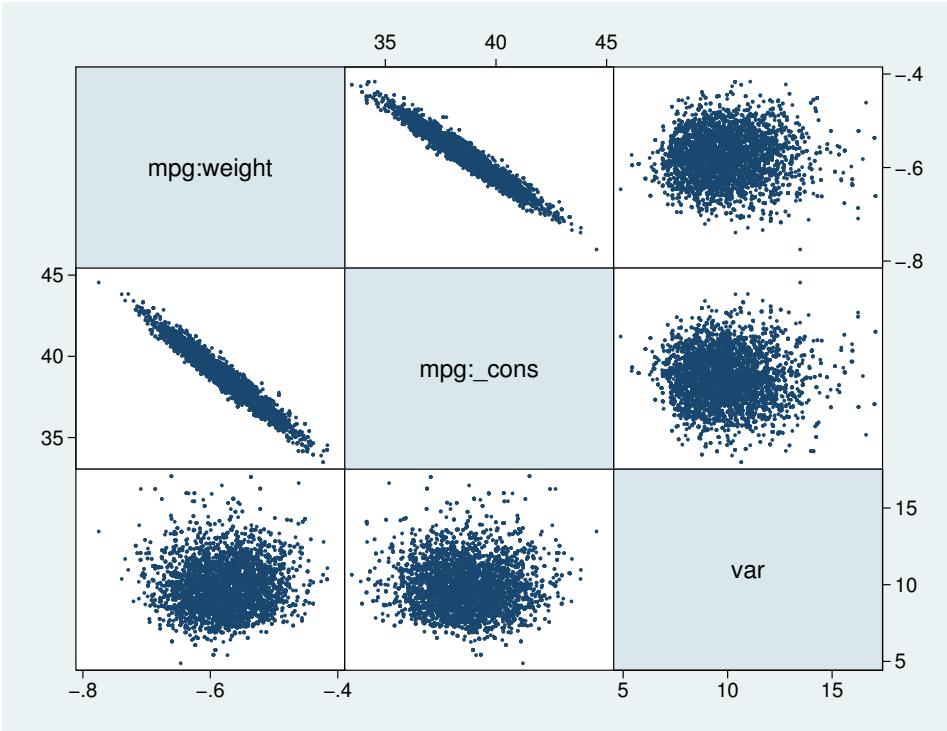
A graphical summary for the `{var}` parameter does not show any obvious problems. The trace plot reveals a good coverage of the domain of the marginal distribution, while the histogram and kernel density plots resemble the shape of an expected inverse-gamma distribution. The autocorrelation dies off after about lag 20.

◁

Example 12: Second simulation run—blocking of variance

Next, we show how to improve the mixing of the MCMC chain by using more careful blocking of model parameters. We can use the `bayesgraph matrix` command to view the scatterplots of the simulated values for {mpg:weight}, {mpg:_cons}, and {var}.

```
. bayesgraph matrix _all
```



The scatterplots reveal high correlation between {mpg:weight} and {mpg:_cons}. On the other hand, there is no significant correlation between {var} and the other two parameters.

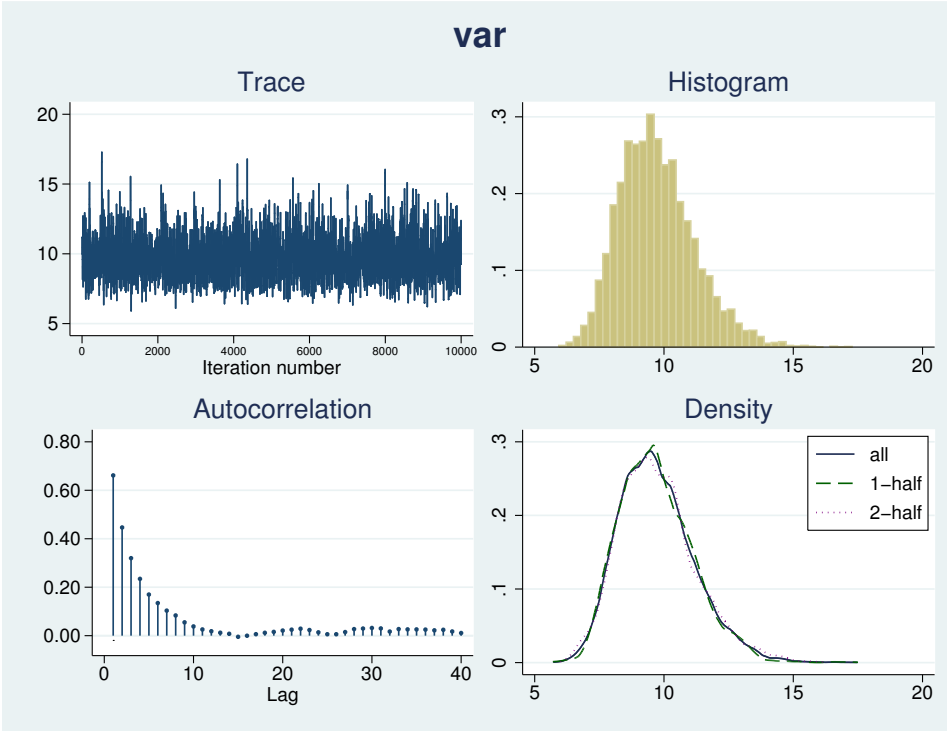
In cases like this, we can expect higher sampling efficiency if we place {var} in a separate block. We can do this by including the option `block({var})`. The other two parameters, {mpg:weight} and {mpg:_cons}, will be automatically considered as a second block.

```
. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, normal(0,100))
> prior({var}, igamma(10,10))
> block({var}) blocksummary
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:weight _cons} ~ normal(0,100)                                      (1)
               {var} ~ igamma(10,10)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_mpg.
Block summary
------------------------------------------------------------------------------
   1:  {var}
   2:  {mpg:weight _cons}
------------------------------------------------------------------------------
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .3309
                                                Efficiency:  min =     .09023
                                                             avg =      .1202
Log marginal likelihood = -226.73992                         max =      .1784

------------------------------------------------------------------------------
             |                                                 Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
      weight |  -.5744536   .0450094   .001484   -.576579   -.663291  -.4853636
       _cons |   38.59206   1.397983    .04654   38.63252   35.80229   41.32773
-------------+----------------------------------------------------------------
         var |   9.721684   1.454193   .034432   9.570546   7.303129   12.95105
------------------------------------------------------------------------------
```

In this second run, we achieve higher simulation efficiency, about 12% on average. The MCSE for {var} is 0.034 and is about half the value of 0.058 from example 11, which leads to twice as much accuracy in the estimation of the posterior mean of {var}.

Again, we can verify the convergence of the MCMC run for `{var}` by inspecting the `bayesgraph diagnostics` plot.

```
. bayesgraph diagnostics {var}
```



The improved sampling efficiency for `{var}` is evident by observing that the autocorrelation becomes negligible after about lag 10. The trace plot reveals more rapid traversing of the marginal posterior domain as well.

◁

▷ Example 13: Third simulation run—Gibbs update of variance

Further improvement of the mixing can be achieved by requesting a Gibbs sampling for the variance parameter. This is possible because `{var}` has an inverse-gamma prior, which is independent of the mean and is a semiconjugate prior in this model.

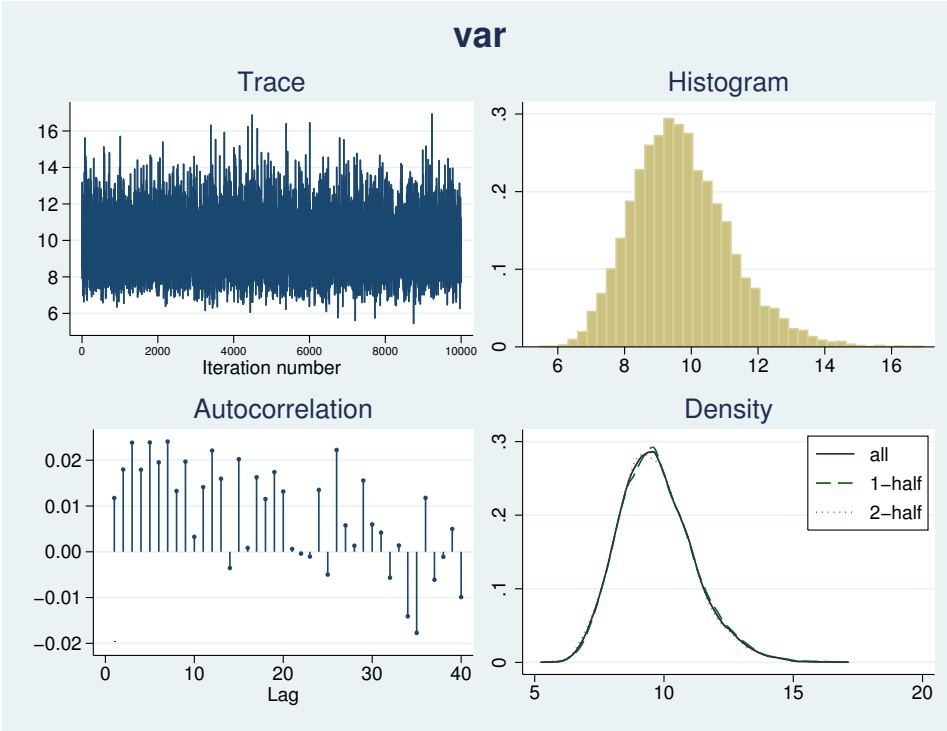
To request Gibbs sampling, we specify suboption `gibbs` within option `block()`.

```
. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, normal(0,100))
> prior({var}, igamma(10,10))
> block({var}, gibbs) blocksummary
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:weight _cons} ~ normal(0,100)                                      (1)
              {var} ~ igamma(10,10)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_mpg.
Block summary
------------------------------------------------------------------------------
   1:  {var}                                                        (Gibbs)
   2:  {mpg:weight _cons}
------------------------------------------------------------------------------
Bayesian normal regression                      MCMC iterations  =     12,500
Metropolis-Hastings and Gibbs sampling          Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .6285
                                                Efficiency:  min =      .1141
                                                             avg =      .3259
Log marginal likelihood = -226.72192                         max =      .7441
------------------------------------------------------------------------------
             |                                                 Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
      weight |  -.5764752   .0457856   .001324  -.5764938  -.6654439   -.486788
       _cons |   38.64148   1.438705    .04259    38.6177   35.82136   41.38734
-------------+----------------------------------------------------------------
         var |   9.711499   1.454721   .016865   9.585728   7.236344   12.95503
------------------------------------------------------------------------------
```

The average efficiency is now 0.33 with the maximum of 0.74 corresponding to the variance parameter.

The diagnostics plot for {var} is an example of almost perfect mixing.

```
. bayesgraph diagnostics {var}
```



Example 14: Fourth simulation run—full Gibbs sampling

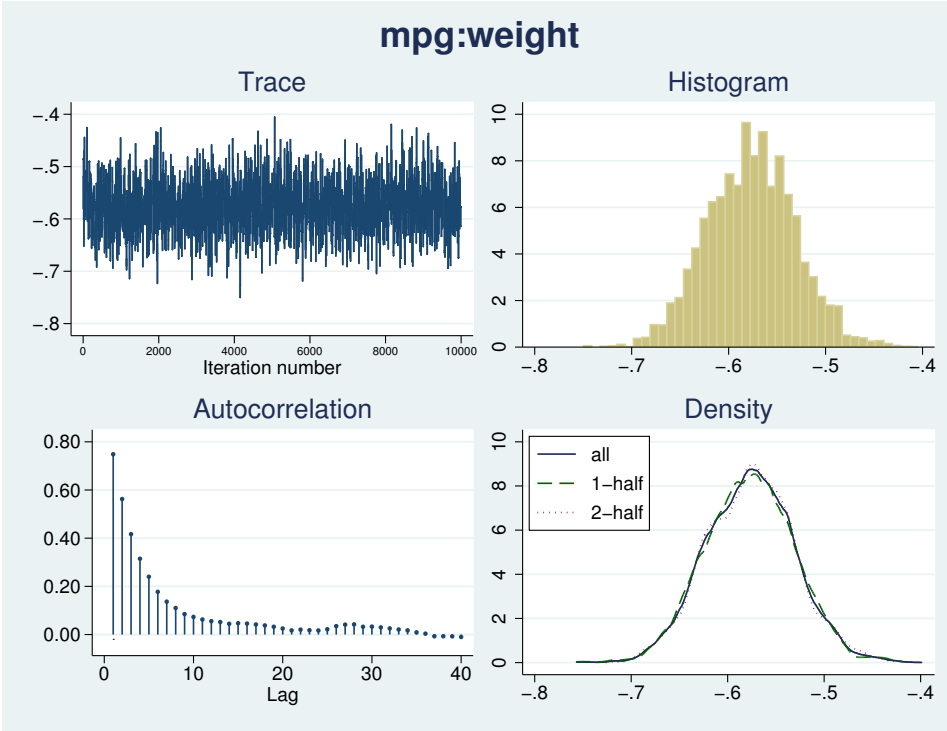
Continuing example 13, there is still room for improvement in our model in terms of sampling efficiency. The efficiency of the regression coefficients is now low relative to the variance efficiency.

```
. bayesstats ess
Efficiency summaries     MCMC sample size =     10,000
```

	ESS	Corr. time	Efficiency
mpg			
weight	1195.57	8.36	0.1196
_cons	1141.12	8.76	0.1141
var	7440.67	1.34	0.7441

For example, diagnostic plots for {weight:_cons} do not look as good as diagnostic plots for the variance parameter in example 13.

```
. bayesgraph diagnostics {mpg:weight}
```



Further improvement of the mixing can be achieved by requesting Gibbs sampling for the two blocks of parameters: regression coefficients and variance. Again, this is possible only because {mpg:weight}, {mpg:_cons}, and {var} have normal and an inverse-gamma priors, which are independent and are semiconjugate in this model.

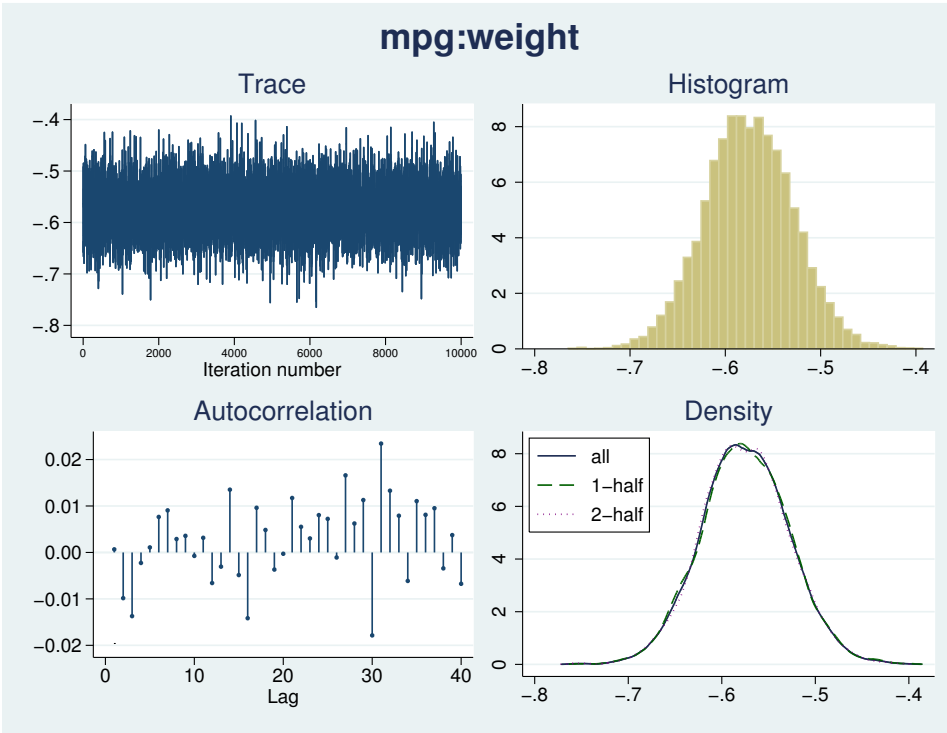
To request Gibbs sampling for the regression coefficients, we must place them in a separate block.

```
. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, normal(0,100))
> prior({var}, igamma(10,10))
> block({var}, gibbs)
> block({mpg:}, gibbs) blocksummary
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:weight _cons} ~ normal(0,100)                                      (1)
               {var} ~ igamma(10,10)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_mpg.
Block summary
------------------------------------------------------------------------------
   1:  {var}                                                          (Gibbs)
   2:  {mpg:weight _cons}                                             (Gibbs)
------------------------------------------------------------------------------
Bayesian normal regression                      MCMC iterations  =     12,500
Gibbs sampling                                  Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =          1
                                                Efficiency:  min =      .9423
                                                             avg =      .9808
Log marginal likelihood = -226.67227                         max =          1
------------------------------------------------------------------------------
             |                                                 Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
      weight |  -.5751071   .0467837   .000468  -.5757037  -.6659412  -.4823263
       _cons |   38.61033   1.459511   .014595   38.61058   35.79156   41.45336
-------------+----------------------------------------------------------------
         var |   9.703432   1.460435   .015045   9.564502   7.216982   12.96369
------------------------------------------------------------------------------
```

Now we have perfect sampling efficiency (with an average of 0.98) with essentially no autocorrelation. The estimators of posterior means have the lowest MCSEs among the four simulations.

For example, diagnostic plots for {mpg:weight} now look noticeably better.

```
. bayesgraph diagnostics {mpg:weight}
```



You can verify that the diagnostic plots of all parameters demonstrate almost perfect mixing as well.

```
. bayesgraph diagnostics _all
```
(*output omitted*)

Logistic regression model: a case of nonidentifiable parameters

We use the heart disease dataset from the UCI Machine Learning Repository (Lichman 2013) and, in particular, we consider a subset of the Switzerland data created by William Steinbrunn, M.D. of University Hospital in Zurich, Switzerland, and by Matthias Pfisterer, M.D. of University Hospital in Basel, Switzerland. The dataset is named `heartswitz.dta` and contains 6 variables, of which `num` is the predicted attribute that takes values from 0 (no heart disease) to 4. We dichotomized `num` to create a new binary variable `disease` as an indicator for the presence of a heart disease.

```
. use http://www.stata-press.com/data/r14/heartswitz
(Subset of Switzerland heart disease data from UCI Machine Learning Repository)
. describe
Contains data from http://www.stata-press.com/data/r14/heartswitz.dta
  obs:           123                          Subset of Switzerland heart
                                                disease data from UCI Machine
                                                Learning Repository
 vars:             6                          5 Feb 2015 16:55
 size:           738                          (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
age             byte    %9.0g                 Age (in years)
male            byte    %9.0g      malelab    1 = male, 0 = female
isfbs           byte    %9.0g      fbslab     Indicator for fasting blood sugar
                                                > 120 mg/dl: 0 = no, 1 = yes
restecg         byte    %28.0g     ecglab     Resting electrocardiographic
                                                results (3 categories)
num             byte    %9.0g                 Presence of heart disease: 0 =
                                                absent and 1,2,3,4 = present
disease         byte    %9.0g      dislab     Indicator for heart disease: 0 =
                                                absent, 1 = present (num>0)
-------------------------------------------------------------------------------
Sorted by:
```

Our goal is to investigate the relationship between the presence of a heart disease and covariates `restecg`, `isfbs`, `age`, and `male`.

First, we fit a standard logistic regression model using the `logit` command.

```
. logit disease restecg isfbs age male
note: restecg != 0 predicts success perfectly
      restecg dropped and 17 obs not used
note: isfbs != 0 predicts success perfectly
      isfbs dropped and 3 obs not used
note: male != 1 predicts success perfectly
      male dropped and 2 obs not used
Iteration 0:   log likelihood = -4.2386144
Iteration 1:   log likelihood = -4.2358116
Iteration 2:   log likelihood = -4.2358076
Iteration 3:   log likelihood = -4.2358076
Logistic regression                               Number of obs     =         26
                                                  LR chi2(1)        =       0.01
                                                  Prob > chi2       =     0.9403
Log likelihood = -4.2358076                       Pseudo R2         =     0.0007
------------------------------------------------------------------------------
     disease |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     restecg |          0  (omitted)
       isfbs |          0  (omitted)
         age |  -.0097846   .1313502    -0.07   0.941    -.2672263    .2476572
        male |          0  (omitted)
       _cons |   3.763893   7.423076     0.51   0.612    -10.78507    18.31285
------------------------------------------------------------------------------
```

We encounter collinearity and dropping of observations because of perfect prediction. As a result, the regression coefficients corresponding to `restecg`, `isfbs`, and `male` are essentially excluded from the model. The standard logistic analysis is limited because of the small size of the dataset.

Next we consider Bayesian analysis of the same data. We fit the same logistic regression model using `bayesmh` and apply fairly noninformative normal priors $N(0, 1e4)$ for all regression parameters.

```
. set seed 14

. bayesmh disease restecg isfbs age male, likelihood(logit)
> prior({disease:}, normal(0,10000))
Burn-in ...
Simulation ...

Model summary
------------------------------------------------------------------------------
Likelihood:
  disease ~ logit(xb_disease)

Prior:
  {disease:restecg isfbs age male _cons} ~ normal(0,10000)                (1)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_disease.

Bayesian logistic regression                    MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         48
                                                Acceptance rate  =      .2661
                                                Efficiency:  min =     .01685
                                                             avg =     .02389
Log marginal likelihood = -16.709588                         max =     .02966

------------------------------------------------------------------------------
             |                                                Equal-tailed
     disease |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
     restecg |   81.22007   63.87998   4.29587   68.31417   2.518447   237.8033
       isfbs |   81.65967   60.07603   4.03945   70.37466   2.035696   229.4291
         age |  -.0191681   .1777758   .013695  -.0154955  -.3833187   .3242438
        male |  -53.69173    42.4866   2.50654  -44.93144   -154.439   .7090207
       _cons |   59.39037    43.5938   2.53139   51.31836   .1225503   161.2943
------------------------------------------------------------------------------
```

The estimated posterior means of `{disease:restecg}`, `{disease:isfbs}`, `{disease:male}`, and `{disease:_cons}` are fairly large, roughly on the same scale as the prior standard deviation of 100.

Indeed, if we decrease the standard deviation of the priors to 10, we observe that the scale of the estimates decreases by the same order of magnitude.

```
. set seed 14
. bayesmh disease restecg isfbs age male, likelihood(logit)
> prior({disease:}, normal(0,100))
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  disease ~ logit(xb_disease)
Prior:
  {disease:restecg isfbs age male _cons} ~ normal(0,100)                (1)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_disease.
Bayesian logistic regression                    MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         48
                                                Acceptance rate  =      .3161
                                                Efficiency:  min =     .02287
                                                             avg =      .0331
Log marginal likelihood = -12.418273                         max =     .05204
------------------------------------------------------------------------------
             |                                                Equal-tailed
     disease |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
     restecg |   8.559131       6.71   .443681   7.447336   -.889714   23.93564
       isfbs |   6.322615   6.411998   .281084   5.504684   -3.85021   20.56641
         age |   .0526448   .1226056    .00718   .0468937  -.1734675   .3050607
        male |  -3.831954    5.31727   .279435  -3.048654  -15.77187   4.451594
       _cons |   5.624899   6.641158   .417961   5.181183  -6.408041    20.1234
------------------------------------------------------------------------------
```

We can, therefore, conclude that the regression parameters are highly sensitive to the choice of priors and their scale cannot be determined by the data alone; that is, it cannot be determined by the likelihood of the model. In other words, these model parameters are not identifiable from the likelihood alone. This conclusion is in agreement with the results of the `logit` command.

We may consider applying an informative prior. We can use information from other heart disease studies from Lichman (2013). For example, we use a subset of the Hungarian data created by Andras Janosi, M.D. of Hungarian Institute of Cardiology in Budapest, Hungary. `hearthungary.dta` contains the same attributes as in `heartswitz.dta` but from a Hungarian population.

We fit bayesmh with noninformative priors to hearthungary.dta and obtain the following posterior mean estimates for the regression parameters:

```
. use http://www.stata-press.com/data/r14/hearthungary
(Subset of Hungarian heart disease data from UCI Machine Learning Repository)
. set seed 14
. bayesmh disease restecg isfbs age male, likelihood(logit)
> prior({disease:}, normal(0,1000))
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  disease ~ logit(xb_disease)
Prior:
  {disease:restecg isfbs age male _cons} ~ normal(0,1000)                (1)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_disease.
Bayesian logistic regression                    MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =        285
                                                Acceptance rate  =      .2341
                                                Efficiency:  min =     .03088
                                                             avg =     .04524
Log marginal likelihood = -195.7454                          max =     .06362
------------------------------------------------------------------------------
                                                              Equal-tailed
     disease |      Mean   Std. Dev.      MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
     restecg |  -.1076298   .2931371   .013664  -.1036111  -.6753464   .4471483
       isfbs |   1.182073    .541182   .030797   1.169921   .2267485   2.268314
         age |    .042955   .0170492   .000676   .0432923   .0103757   .0763747
        male |   1.488844   .3612114   .018399   1.484816   .7847398   2.244648
       _cons |  -3.866674   .8904101   .041022  -3.869567  -5.658726  -2.112237
------------------------------------------------------------------------------
```

With this additional information, we can form more informative priors for the 5 parameters of interest—we center {restecg} and {age} at 0, {disease:isfbs} and {disease:male} at 1, and {disease:_cons} at −4, and we use a prior variance of 10 for all coefficients.

```
. use http://www.stata-press.com/data/r14/heartswitz
(Subset of Switzerland heart disease data from UCI Machine Learning Repository)
. set seed 14
. bayesmh disease restecg isfbs age male, likelihood(logit)
> prior({disease:restecg age}, normal( 0,10))
> prior({disease:isfbs male}, normal( 1,10))
> prior({disease:_cons}, normal(-4,10))
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  disease ~ logit(xb_disease)
Priors:
  {disease:restecg age} ~ normal(0,10)                                    (1)
   {disease:isfbs male} ~ normal(1,10)                                    (1)
        {disease:_cons} ~ normal(-4,10)                                   (1)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_disease.
Bayesian logistic regression                    MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         48
                                                Acceptance rate  =       .247
                                                Efficiency:  min =     .03691
                                                             avg =     .05447
Log marginal likelihood = -11.021903                         max =     .06737
```

disease	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
restecg	1.74292	2.21888	.097001	1.385537	-2.065912	6.584702
isfbs	1.885653	2.792842	.145375	1.595679	-2.976167	7.976913
age	.1221246	.0698409	.002691	.1174274	-.0078114	.2706446
male	.2631	2.201574	.089281	.2667496	-4.125275	4.646742
_cons	-2.304595	2.706482	.115472	-2.256248	-7.785531	3.098357

We now obtain more reasonable results that also agree with the Hungarian results. For the final analysis, we may consider other heart disease datasets to verify the reasonableness of our prior specifications and to check the sensitivity of the parameters to other prior specifications.

Ordered probit regression

Ordered probit and ordered logit regressions are appropriate for modeling ordinal response variables. You can perform Bayesian analysis of an ordinal outcome by specifying the `oprobit` or `ologit` likelihood function. In addition to regression coefficients in ordered models, `bayesmh` automatically introduces parameters representing the cutpoints for the linear predictor. The cutpoint parameters are declared as {*depname*:`_cut1`}, {*depname*:`_cut2`}, and so on, where *depname* is the name of the response variable.

In the next example, we consider the full auto dataset and model the ordinal variable `rep77`, the repair record, as a function of independent variables `foreign`, `length`, and `mpg`. The variable `rep77` has 5 levels, so the cutpoint parameters are {`rep77:_cut1`}, {`rep77:_cut2`}, {`rep77:_cut3`}, and {`rep77:_cut4`}. The independent variables are all positive, so it seems reasonable to use exponential prior for the cutpoint parameters. The exponential prior is controlled by a hyperparameter {`lambda`}. Based on the range of the independent predictors, we assign {`lambda`} a prior that is uniform in

the 10 to 40 range. We assign $N(0,1)$ prior for regression coefficients. To monitor the progress, we specify `dots` to request that `bayesmh` displays dots every 100 iterations and iteration numbers every 1,000 iterations.

```
. use http://www.stata-press.com/data/r14/fullauto
(Automobile Models)
. replace length = length/10
variable length was int now float
(74 real changes made)
. set seed 14
. bayesmh rep77 foreign length mpg, likelihood(oprobit)
> prior({rep77: foreign length mpg}, normal(0,1))
> prior({rep77:_cut1 _cut2 _cut3 _cut4}, exponential({lambda=30}))
> prior({lambda}, uniform(10,40)) block(lambda) dots
Burn-in 2500 aaaaaaaaa1000aaaaaaaaa2000aaaaa done
Simulation 10000 .........1000.........2000.........3000.........4000.........5
> 000.........6000.........7000.........8000.........9000.........10000 done
Model summary
------------------------------------------------------------------------------
Likelihood:
  rep77 ~ oprobit(xb_rep77,{rep77:_cut1 ... _cut4})
Priors:
  {rep77:foreign length mpg} ~ normal(0,1)                              (1)
     {rep77:_cut1 ... _cut4} ~ exponential({lambda})
Hyperprior:
  {lambda} ~ uniform(10,40)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_rep77.
Bayesian ordered probit regression              MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         66
                                                Acceptance rate  =      .3422
                                                Efficiency:  min =     .02171
                                                             avg =      .0355
Log marginal likelihood = -102.82883                         max =      .1136
------------------------------------------------------------------------------
             |                                                 Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
rep77        |
     foreign |   1.338071   .3750768   .022296   1.343838   .6331308   2.086062
      length |   .3479392   .1193329    .00787   .3447806   .1277292   .5844067
         mpg |   .1048089   .0356498   .002114   .1022382   .0373581   .1761636
       _cut1 |   7.204502   2.910222   .197522   7.223413    1.90771   13.07034
       _cut2 |   8.290923   2.926149   .197229   8.258871   2.983281   14.16535
       _cut3 |   9.584845   2.956191   .197144   9.497836    4.23589   15.52108
       _cut4 |   10.97314   3.003014   .192244   10.89227   5.544563   17.06189
-------------+----------------------------------------------------------------
      lambda |   18.52477   7.252342   .215137   16.40147   10.21155   36.44309
------------------------------------------------------------------------------
```

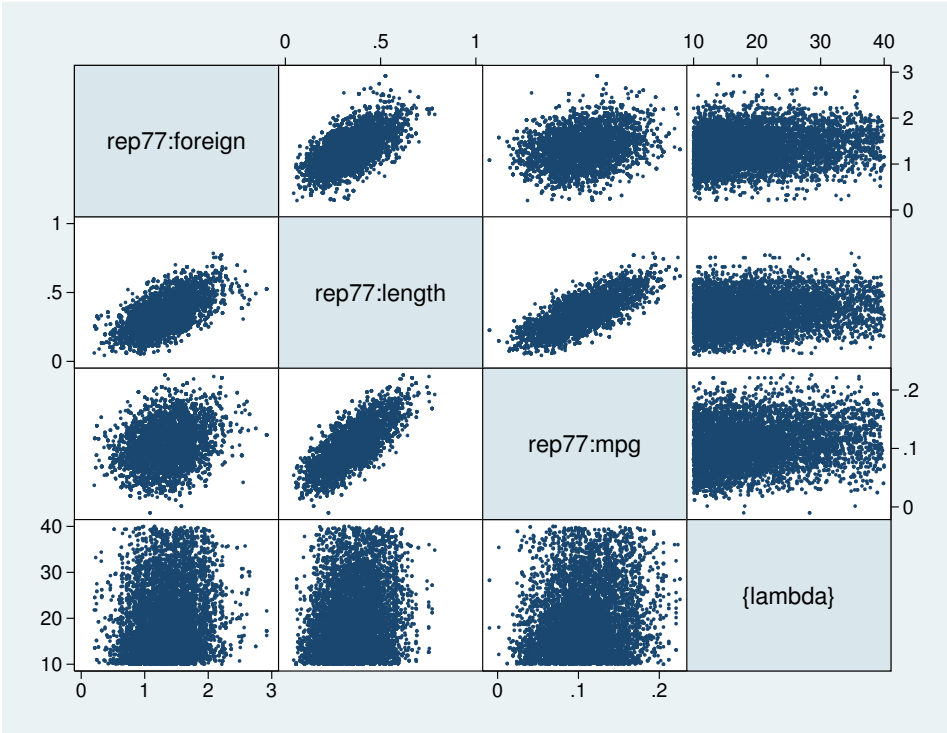
When we specify `dots` or `dots()`, `bayesmh` displays dots as simulation is performed. The burn-in and simulation iterations are displayed separately. During the adaptation period, iterations are displayed with a symbol `a` instead of a dot. This indicates the period during which the proposal distribution is still changing and thus may not be suitable for sampling from yet. Typically, adaptation is performed during the burn-in period, the iterations of which are discarded from the MCMC sample. You should pay closer attention to your results if you see adaptive iterations during the simulation period. This may happen, for example, if you increase `adaptation(maxiter())` without increasing `burnin()`

correspondingly. In this case, you may need to perform additional checks to verify that the part of the MCMC sample corresponding to the adaptation period is similar to the rest of the sample.

Posterior credible intervals suggest that `foreign`, `length`, and `mpg` are among the explanatory factors for `rep77`. Based on MCSEs, their posterior mean estimates are fairly precise. The posterior mean estimates of cutpoints, as expected, are not as precise. The estimated posterior mean for `{lambda}` is 18.52.

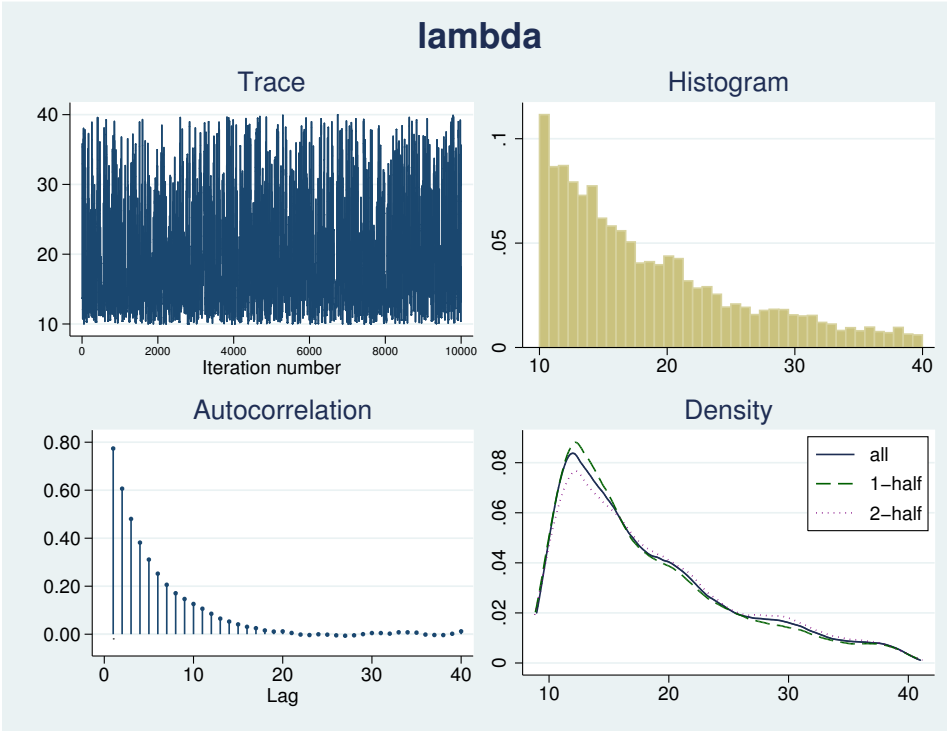
We placed the hyperparameter `{lambda}` in a separate block because we wanted to sample this nuisance parameter independently from the other model parameters. Based on the bivariate scatterplots, this parameter does appear to be independent of other model parameters a posteriori.

```
. bayesgraph matrix {rep77:foreign} {rep77:length} {rep77:mpg} {lambda}
```



As with any MCMC analysis, we should verify convergence of all of our parameters. Here we show diagnostic plots only for {lambda}.

```
. bayesgraph diagnostics {lambda}
```



The diagnostic plots for {lambda} do not cause any concern.

Beta-binomial model

bayesmh is a regression command, which models the mean of the outcome distribution as a function of predictors. There are cases when we do not have any predictors and want to model the outcome distribution directly. For example, we may want to fit a Poisson distribution or a binomial distribution to our outcome. We can do this by simply omitting covariates in the model specification of bayesmh and fitting a constant-only model. There is a caveat. Because bayesmh is a regression command, it will interpret our model as a regression model with one covariate containing values of one in all observations. For example, if we use likelihood(poisson) to fit a Poisson distribution to the outcome, bayesmh will fit a Poisson regression with the link function for the mean $\mu = \exp(\beta_0)$ as a function of parameter β_0, a constant, rather than modeling the mean parameter μ directly. We can specify noglmtransform within likelihood() to prevent the exponentiation in the Poisson model or, more generally, to request that no GLM-type transformation be used when fitting certain generalized linear or nonlinear outcome models. noglmtransform is supported with Poisson, exponential, and binomial likelihood models.

Let's revisit the example from *What is Bayesian analysis?* in [BAYES] **intro**, originally from Hoff (2009, 3), of estimating the prevalence of a rare infectious disease in a small city. The outcome

variable y is the number of infected subjects in a city of 20 subjects, and our data consist of only one observation, $y = 0$. We assume a binomial distribution for the outcome y, Binom(20,θ), where the infection probability θ is a parameter of interest. Based on some previous studies, the model parameter θ is assigned a Beta$(2, 20)$ prior. For this model, the posterior distribution of θ is known to be Beta$(2, 40)$.

To fit a binomial distribution to y using bayesmh, we specify a constant-only regression model and use option likelihood(binlogit(20), noglmtransform). The infection probability θ is represented by {y:_cons}.

```
. set obs 1
number of observations (_N) was 0, now 1
. generate y = 0
. set seed 14
. bayesmh y, likelihood(binlogit(20), noglmtransform)
> prior({y:_cons}, beta(2,20)) initial({y:_cons} 0.01)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  y ~ binomial({y:_cons},20)
Prior:
  {y:_cons} ~ beta(2,20)
------------------------------------------------------------------------------

Bayesian binomial regression                    MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =          1
                                                Acceptance rate  =      .4527
Log marginal likelihood = -1.1658052            Efficiency       =      .1549

------------------------------------------------------------------------------
           |                                                 Equal-tailed
         y |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-----------+------------------------------------------------------------------
       _cons |  .0467973   .0317862   .000808   .039931   .0051255   .1277823
------------------------------------------------------------------------------
```

The estimated posterior mean for {y:_cons} is 0.0468, which is close to the theoretical value of $2/(2 + 40) = 0.0476$ and is within the range of the MCSE of 0.0008.

Multivariate regression

We consider a simple multivariate normal regression model without covariates. We use auto.dta, and we fit a multivariate normal distribution to variables mpg, weight, and length.

We rescale these variables to have approximately equal ranges. Equalizing the range of model variables is always recommended, because this makes the model computationally more stable.

```
. use http://www.stata-press.com/data/r14/auto, clear
(1978 Automobile Data)
. quietly replace weight = weight/1000
. quietly replace length = length/100
. quietly replace mpg = mpg/10
```

▷ Example 15: Default MH sampling with inverse-Wishart prior for the covariance

For a multivariate normal distribution, an inverse-Wishart prior is commonly used as a prior for the covariance matrix. Let's fit our multivariate model using `bayesmh`.

We specify the multivariate normal likelihood `likelihood(mvnormal({Sigma,m}))` for the three variables `mpg`, `weight`, and `length`, where `{Sigma,m}` is a matrix parameter for the covariance matrix. We use vague normal priors `normal(0,100)` for all three means of the variables. For a covariance matrix `{Sigma,m}`, which is of dimension three, we specify an inverse-Wishart prior with the identity scale matrix. We also specify the mean parameters and the covariance parameter in two separate blocks. To monitor the simulation process, we specify `dots`.

```
. set seed 14
. bayesmh (mpg) (weight) (length), likelihood(mvnormal({Sigma,m}))
> prior({mpg:_cons} {weight:_cons} {length:_cons}, normal(0,100))
> prior({Sigma,m}, iwishart(3,100,I(3)))
> block({mpg:_cons} {weight:_cons} {length:_cons})
> block({Sigma,m}) dots
Burn-in 2500 aaaaaaaaa1000aaaaaaaaa2000aaaaa done
Simulation 10000 .........1000.........2000.........3000.........4000.........5
> 000.........6000.........7000.........8000.........9000.........10000 done
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg weight length ~ mvnormal(3,{mpg:},{weight:},{length:},{Sigma,m})
Priors:
     {mpg:_cons} ~ normal(0,100)
  {weight:_cons} ~ normal(0,100)
  {length:_cons} ~ normal(0,100)
       {Sigma,m} ~ iwishart(3,100,I(3))
------------------------------------------------------------------------------
Bayesian multivariate normal regression          MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling         Burn-in          =      2,500
                                                 MCMC sample size =     10,000
                                                 Number of obs    =         74
                                                 Acceptance rate  =      .3255
                                                 Efficiency:  min =    .001396
                                                              avg =     .04166
Log marginal likelihood = -254.88899                          max =      .1111
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
mpg						
_cons	2.13089	.0455363	.001763	2.129007	2.04435	2.223358
weight						
_cons	3.018691	.0671399	.00212	3.020777	2.880051	3.149828
length						
_cons	1.879233	.0210167	.00063	1.879951	1.837007	1.920619
Sigma_1_1	.1571554	.0038157	.000183	.1570586	.1499028	.1648159
Sigma_2_1	-.1864936	.0024051	.000343	-.1864259	-.1912537	-.18194
Sigma_3_1	-.0533863	.0033667	.000199	-.053342	-.0601722	-.0468986
Sigma_2_2	.3293518	.0044948	.001203	.329703	.3193904	.3366703
Sigma_3_2	.0894404	.0040487	.000471	.0894156	.0816045	.0976702
Sigma_3_3	.0329253	.002521	.00024	.0328027	.0285211	.0383005

```
Note: There is a high autocorrelation after 500 lags.
```

In this first run, we do not achieve good mixing of the MCMC chain. bayesmh issues a note about significant autocorrelation of the simulated parameters.

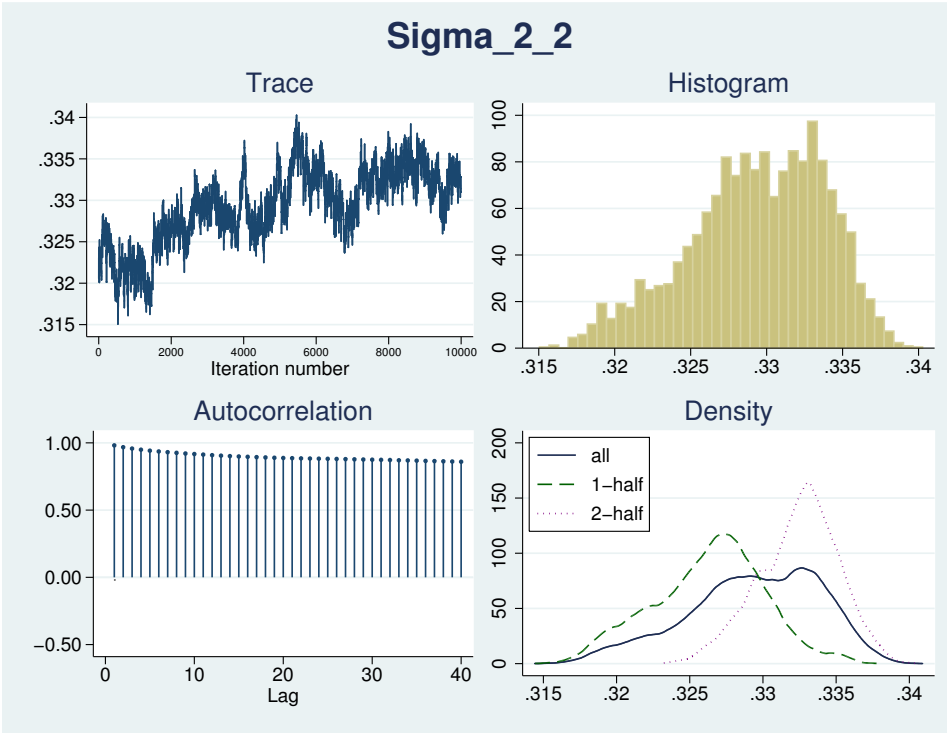
A closer inspection of the ESS table reveals very low sampling efficiencies for the elements of the covariance matrix {Sigma}.

```
. bayesstats ess
Efficiency summaries     MCMC sample size =     10,000
```

	ESS	Corr. time	Efficiency
mpg			
_cons	667.48	14.98	0.0667
weight			
_cons	1002.92	9.97	0.1003
length			
_cons	1111.14	9.00	0.1111
Sigma_1_1	433.25	23.08	0.0433
Sigma_2_1	49.03	203.96	0.0049
Sigma_3_1	287.03	34.84	0.0287
Sigma_2_2	13.96	716.45	0.0014
Sigma_3_2	73.76	135.57	0.0074
Sigma_3_3	110.41	90.58	0.0110

For example, the diagnostic plots for {Sigma_2_2} provide visual confirmation of the convergence issues—very poorly mixing trace plot, high autocorrelation, and a bimodal posterior distribution.

```
. bayesgraph diagnostics Sigma_2_2
```



What we see here is a general problem associated with the simulation of covariance matrices. Random-walk MH algorithm is not well suited for sampling positive-definite matrices. This is why even an adaptive version of the MH algorithm, as implemented in bayesmh, may not achieve good mixing.

◁

▷ Example 16: Adaptation of MH sampling with inverse-Wishart prior for the covariance

Continuing example 15, we can specify longer adaptation and burn-in periods to improve convergence.

```
. set seed 14
. bayesmh (mpg) (weight) (length), likelihood(mvnormal({Sigma,m}))
> prior({mpg:_cons} {weight:_cons} {length:_cons}, normal(0,100))
> prior({Sigma,m},                                 iwishart(3,100,I(3)))
> block({mpg:_cons} {weight:_cons} {length:_cons})
> block({Sigma,m}) dots burnin(5000) adaptation(maxiter(50))
Burn-in 5000 aaaaaaaaa1000aaaaaaaaa2000aaaaaaaaa3000aaaa.....4000.........5000
> done
Simulation 10000 .........1000.........2000.........3000.........4000.........5
> 000.........6000.........7000.........8000.........9000.........10000 done
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg weight length ~ mvnormal(3,{mpg:},{weight:},{length:},{Sigma,m})
Priors:
     {mpg:_cons} ~ normal(0,100)
  {weight:_cons} ~ normal(0,100)
  {length:_cons} ~ normal(0,100)
       {Sigma,m} ~ iwishart(3,100,I(3))
------------------------------------------------------------------------------
Bayesian multivariate normal regression          MCMC iterations  =     15,000
Random-walk Metropolis-Hastings sampling         Burn-in          =      5,000
                                                 MCMC sample size =     10,000
                                                 Number of obs    =         74
                                                 Acceptance rate  =      .2382
                                                 Efficiency:  min =     .02927
                                                              avg =     .05053
Log marginal likelihood = -245.83844                          max =     .07178
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
mpg						
_cons	2.13051	.0475691	.001809	2.13263	2.038676	2.220953
weight						
_cons	3.017943	.0626848	.00234	3.016794	2.898445	3.143252
length						
_cons	1.878912	.019905	.000769	1.878518	1.840311	1.918476
Sigma_1_1	.1711394	.0089943	.000419	.1706437	.1548036	.1898535
Sigma_2_1	-.1852432	.002432	.000126	-.1852973	-.1898398	-.1803992
Sigma_3_1	-.0517404	.0035831	.000201	-.051688	-.058747	-.0449874
Sigma_2_2	.3054418	.0144859	.000551	.3055426	.2783409	.3340654
Sigma_3_2	.0809091	.0057474	.000314	.080709	.0698331	.0924053
Sigma_3_3	.030056	.002622	.000153	.0299169	.0251627	.0355171

There is no note about high autocorrelation, and the average efficiency increases slightly from 4% to 5%.

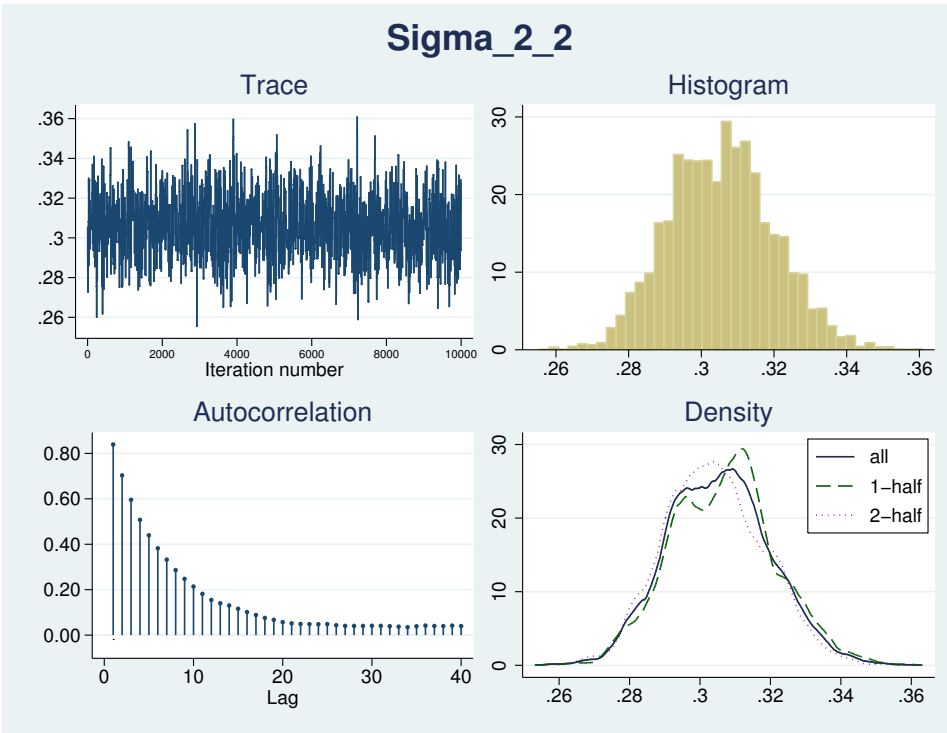
Sampling efficiencies of the elements of the covariance matrix improved substantially.

```
. bayesstats ess
Efficiency summaries     MCMC sample size =     10,000
```

	ESS	Corr. time	Efficiency
mpg			
_cons	691.54	14.46	0.0692
weight			
_cons	717.82	13.93	0.0718
length			
_cons	670.63	14.91	0.0671
Sigma_1_1	459.78	21.75	0.0460
Sigma_2_1	370.45	26.99	0.0370
Sigma_3_1	318.91	31.36	0.0319
Sigma_2_2	692.06	14.45	0.0692
Sigma_3_2	334.08	29.93	0.0334
Sigma_3_3	292.70	34.16	0.0293

The diagnostic plots for {Sigma_2_2} look much better.

```
. bayesgraph diagnostics Sigma_2_2
```



▷ Example 17: Gibbs sampling of a covariance matrix

Continuing example 15, the convergence of the chain can be greatly improved if we use Gibbs sampling for the covariance matrix parameter. For a multivariate normal model, inverse Wishart is a conjugate prior, or more precisely semiconjugate prior, for the covariance matrix and thus Gibbs sampling is available. To request Gibbs sampling, we only need to add the `gibbs` suboption to the block specification of `{Sigma,m}`. The mean parameters are still updated by the random-walk MH algorithm.

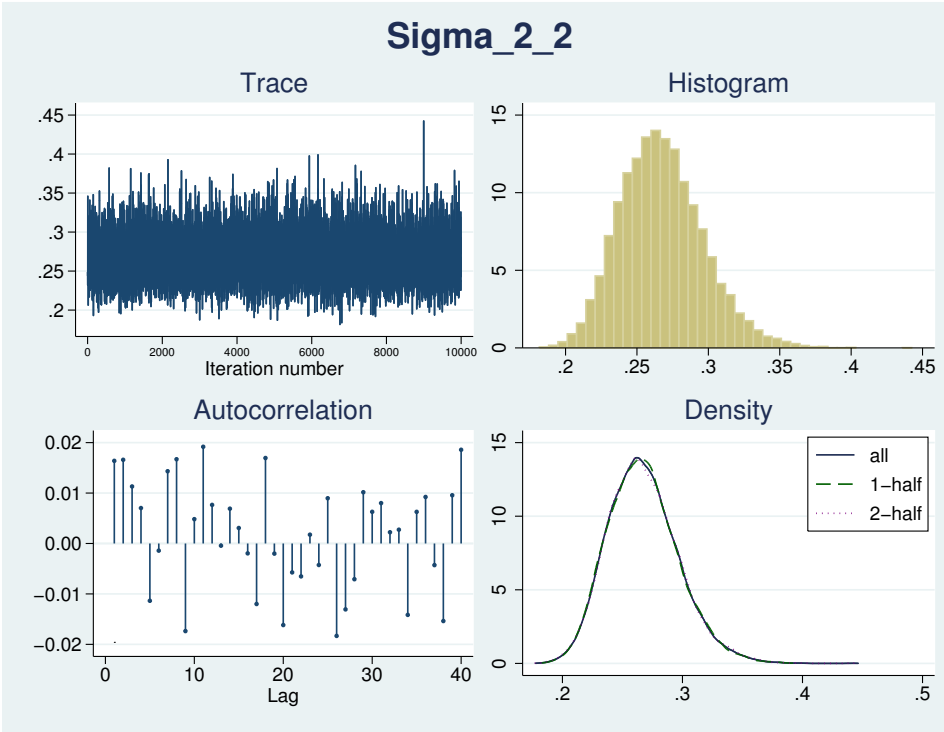
```
. set seed 14
. bayesmh (mpg) (weight) (length), likelihood(mvnormal({Sigma,m}))
> prior({mpg:_cons} {weight:_cons} {length:_cons}, normal(0,100))
> prior({Sigma,m}, iwishart(3,100,I(3)))
> block({mpg:_cons} {weight:_cons} {length:_cons})
> block({Sigma,m}, gibbs) dots
Burn-in 2500 aaaaaaaaa1000aaaaaaaaa2000aaa.. done
Simulation 10000 .........1000.........2000.........3000.........4000.........5
> 000.........6000.........7000.........8000.........9000.........10000 done
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg weight length ~ mvnormal(3,{mpg:},{weight:},{length:},{Sigma,m})
Priors:
     {mpg:_cons} ~ normal(0,100)
  {weight:_cons} ~ normal(0,100)
  {length:_cons} ~ normal(0,100)
       {Sigma,m} ~ iwishart(3,100,I(3))
------------------------------------------------------------------------------
Bayesian multivariate normal regression          MCMC iterations  =     12,500
Metropolis-Hastings and Gibbs sampling           Burn-in          =      2,500
                                                 MCMC sample size =     10,000
                                                 Number of obs    =         74
                                                 Acceptance rate  =      .5926
                                                 Efficiency:  min =     .05922
                                                              avg =      .6581
Log marginal likelihood = -240.48949                          max =      .9737
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	2.12953	.0462658	.001722	2.129648	2.038988	2.215482
weight						
_cons	3.018778	.0610962	.002511	3.019004	2.903026	3.141158
length						
_cons	1.880089	.0200601	.000795	1.880318	1.840313	1.920597
Sigma_1_1	.150745	.0164684	.000167	.1495192	.1219407	.1865202
Sigma_2_1	-.1571837	.0197078	.000202	-.1559806	-.199528	-.1223894
Sigma_3_1	-.0443814	.0060322	.000062	-.0439561	-.0572446	-.0338437
Sigma_2_2	.2673595	.0292434	.000305	.265501	.2159522	.3304341
Sigma_3_2	.0708204	.0085562	.000087	.0702694	.0557287	.089472
Sigma_3_3	.0273568	.0029966	.000031	.0271421	.0220828	.0337088

Compared with example 15, the results improved substantially. Compared with example 16, the minimum efficiency increases from about 3% to 6% and the average efficiency from 5% to 66%. MCSEs of posterior mean estimates, particularly for elements of `{Sigma}`, are lower.

The diagnostic plots, for example, for `Sigma_2_2` also indicate a very good convergence.

```
. bayesgraph diagnostics Sigma_2_2
```



▷ Example 18: Gibbs sampling of a covariance matrix with the Jeffreys prior

In this example, we perform a sensitivity analysis of the model by replacing the inverse-Wishart prior for the covariance matrix with a Jeffreys prior.

```
. set seed 14
. bayesmh (mpg) (weight) (length), likelihood(mvnormal({Sigma,m}))
> prior({mpg:} {weight:} {length:}, normal(0,100))
> prior({Sigma,m}, jeffreys(3))
> block({mpg:} {weight:} {length:})
> block({Sigma,m}, gibbs) dots
Burn-in 2500 aaaaaaaaa1000aaaaaaaaa2000aaaaa done
Simulation 10000 .........1000.........2000.........3000.........4000.........5
> 000.........6000.........7000.........8000.........9000.........10000 done
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg weight length ~ mvnormal(3,{mpg:},{weight:},{length:},{Sigma,m})
Priors:
     {mpg:_cons} ~ normal(0,100)
  {weight:_cons} ~ normal(0,100)
  {length:_cons} ~ normal(0,100)
       {Sigma,m} ~ jeffreys(3)
------------------------------------------------------------------------------
Bayesian multivariate normal regression          MCMC iterations  =     12,500
Metropolis-Hastings and Gibbs sampling           Burn-in          =      2,500
                                                 MCMC sample size =     10,000
                                                 Number of obs    =         74
                                                 Acceptance rate  =      .6223
                                                 Efficiency:  min =     .08573
                                                              avg =      .6886
Log marginal likelihood = -42.728723                          max =          1
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
mpg						
_cons	2.130704	.0709095	.002185	2.129449	1.989191	2.267987
weight						
_cons	3.019323	.0950116	.003245	3.019384	2.834254	3.208017
length						
_cons	1.879658	.0271562	.000892	1.879859	1.827791	1.933834
Sigma_1_1	.3596673	.0628489	.000628	.3526325	.2575809	.5028854
Sigma_2_1	-.3905511	.0772356	.000772	-.3824458	-.5668251	-.2654059
Sigma_3_1	-.1103824	.0220164	.000223	-.1077659	-.1611913	-.0751177
Sigma_2_2	.6503219	.1141333	.001141	.6378476	.466738	.9140429
Sigma_3_2	.1763159	.0318394	.000323	.1725042	.1248434	.2507866
Sigma_3_3	.0533981	.0093631	.000095	.0522228	.0382405	.0748096

```
Note: Adaptation tolerance is not met in at least one of the blocks.
```

Compared with example 17, the estimates of the means of the multivariate distribution do not change much, but the estimates of the elements of the covariance matrix do change. The estimates for `{Sigma,m}` obtained using the Jeffreys prior are approximately twice as big as the estimates obtained using the inverse-Wishart prior. If we compute correlation matrices corresponding to `{Sigma,m}` from the two models, they will be similar. This can be explained by the fact that both the Jeffreys prior and the inverse-Wishart prior with identity scale matrix are not informative for the correlation structure

because they only depend on the determinant and the trace of {Sigma,m} whereas the correlation structure is determined by the data alone.

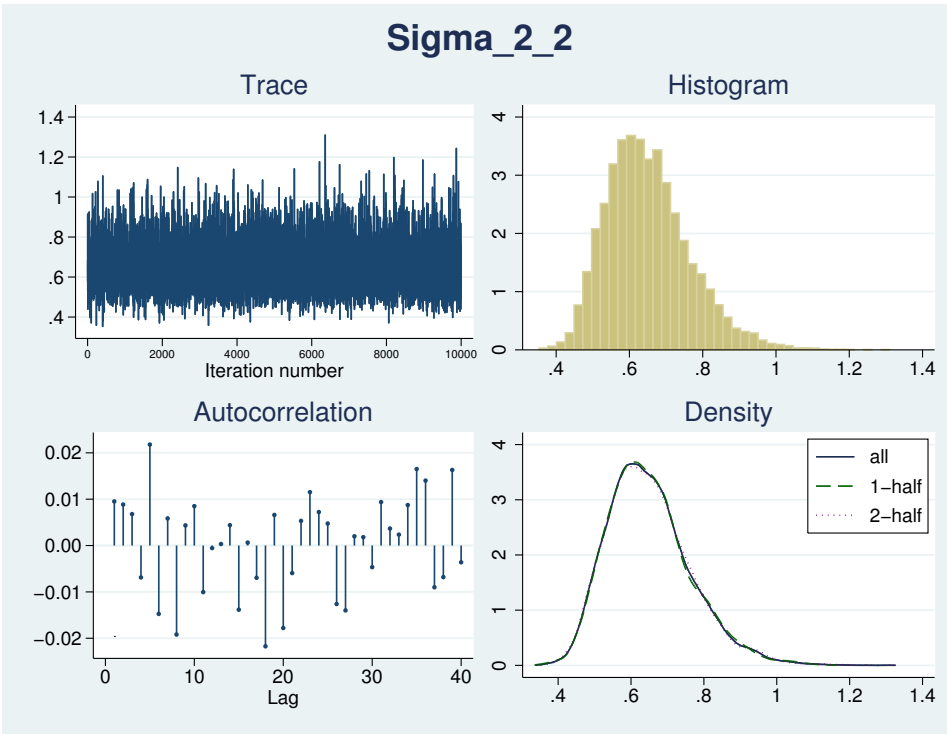
❑ Technical note: Adaptation tolerance is not met

At the bottom of the table in the previous output, the note about the adaptation tolerance not being met in one of the blocks is displayed. Adaptation is part of MH sampling, so the note refers to the block of regression coefficients. This note does not necessarily indicate a problem. It simply notifies you that the default target acceptance rate as specified in adaptation(tarate()) has not been reached within the tolerance specified in adaptation(tolerance()). The used default for the target acceptance rate corresponds to the theoretical asymptotically optimal acceptance rate of 0.44 for a block with one parameter and 0.234 for a block with multiple parameters. The rate is derived for a specific class of models and does not necessarily represent the optimal rate for all models. If your MCMC converged, you can safely ignore this note. Otherwise, you need to investigate your model further. One remedy is to increase the burn-in period, which automatically increases the adaptation period, or more specifically, the number of adaptive iterations as controlled by adaptation(maxiter()). For example, if we increase burn-in to 3,000 by specifying option burnin(3000) in the above example, we will meet the adaptation tolerance.

❑

The diagnostic plots of Sigma_2_2 demonstrate excellent mixing properties.

```
. bayesgraph diagnostics Sigma_2_2
```



◁

Panel-data and multilevel models

Although the MH algorithm underlying bayesmh is not optimal for fitting Bayesian multilevel models, you can use it to fit some multilevel models that do not have too many random effects. Below we consider two-level random-intercept and random-coefficients models. A two-level random-effects model is also known as a panel-data model.

Two-level random-intercept model or panel-data model

Ruppert, Wand, and Carroll (2003) and Diggle et al. (2002) analyzed a longitudinal dataset consisting of weight measurements of 48 pigs on 9 successive weeks. Pigs were identified by the group variable id.

The following two-level model was considered:

$$\texttt{weight}_{ij} = \beta_0 + \beta_1 \texttt{week}_{ij} + u_j + \epsilon_{ij}$$

where u_j is the random effect for pig j, $j = 1, \ldots, 48$, and the counter $i = 1, \ldots, 9$ identifies the weeks.

We first use mixed to fit this model by using maximum likelihood for comparison purposes; see [ME] **mixed**.

```
. use http://www.stata-press.com/data/r14/pig
(Longitudinal analysis of pig weights)
. mixed weight week || id:
Performing EM optimization:
Performing gradient-based optimization:
Iteration 0:   log likelihood = -1014.9268
Iteration 1:   log likelihood = -1014.9268
Computing standard errors:
Mixed-effects ML regression                     Number of obs     =        432
Group variable: id                              Number of groups  =         48
                                                Obs per group:
                                                              min =          9
                                                              avg =        9.0
                                                              max =          9
                                                Wald chi2(1)      =   25337.49
Log likelihood = -1014.9268                     Prob > chi2       =     0.0000
```

weight	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
week	6.209896	.0390124	159.18	0.000	6.133433	6.286359
_cons	19.35561	.5974059	32.40	0.000	18.18472	20.52651

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
id: Identity				
var(_cons)	14.81751	3.124226	9.801716	22.40002
var(Residual)	4.383264	.3163348	3.805112	5.04926

LR test vs. linear model: chibar2(01) = 472.65 Prob >= chibar2 = 0.0000

Consider the following Bayesian model for these data:

$$\texttt{weight}_{ij} = \beta_0 + \beta_1\texttt{week}_{ij} + u_j + \epsilon_{ij} = \beta_1\texttt{week}_{ij} + \tau_j + \epsilon_{ij},$$

$$\begin{aligned}
\epsilon_{ij} &\sim \text{i.i.d. } N(0, \sigma_0^2)\\
\tau_j &\sim \text{i.i.d. } N(\beta_0, \sigma_{\text{id}}^2)\\
\beta_0 &\sim N(0, 100)\\
\beta_1 &\sim N(0, 100)\\
\sigma_0^2 &\sim \text{InvGamma}(0.001, 0.001)\\
\sigma_{\text{id}}^2 &\sim \text{InvGamma}(0.001, 0.001)
\end{aligned}$$

The model has four main parameters of interest: regression coefficients β_0 and β_1 and variance components σ_0^2 and σ_{id}^2. β_0 is actually a hyperparameter in this example, because it is the mean parameter of the prior distribution for random effects τ_j. The pig random effects τ_j are considered nuisance parameters. We use normal priors for the regression coefficients and group levels identified by the `id` variable and inverse-gamma priors for the variance parameters. The chosen priors are fairly noninformative, so we would expect results to be similar to the frequentist results.

To fit this model using `bayesmh`, we need to include random effects for pig in our regression model. This can be done by adding factor levels of the `id` variable to the regression by using the factor-variable specification `i.id`. This specification, by default, will omit one of the `id` categories as a base category. In our Bayesian model, we need to keep all categories of `id`, so we use `fvset` to declare no base for the `id` variable.

```
. fvset base none id
```

In addition to two regression coefficients and two variance components, we have 48 random-effects parameters. As for other models, `bayesmh` will automatically create parameters of the regression function: `{weight:week}` for the regression coefficient of `week` and `{weight:1.id}`, `{weight:2.id}`, ..., `{weight:48.id}` for random effects. We do not include a constant in our regression function because it is modeled as a mean of random effects in their prior. So, we need to define the three remaining model parameters manually; we will use `{weight:_cons}` for the mean of random effects, `{var_id}` for the variance of random effects, and `{var_0}` for the error variance.

We will perform three simulations for the specified Bayesian model to illustrate some common difficulties in applying MH MCMC to multilevel models.

▷ Example 19: First simulation—default MH settings

In the first simulation, we use default simulation settings of the MH algorithm. We have many parameters in our model, so the simulation will take a few moments. For exploration purposes and to expedite results, here we use a smaller MCMC size of 5,000 instead of the default of 10,000. To monitor the progress of the simulation, we also specify `dots`.

```
. set seed 14
. bayesmh weight week i.id, likelihood(normal({var_0})) noconstant
>         prior({weight:i.id},  normal({weight:_cons},{var_id}))
>         prior({weight:_cons}, normal(0, 100))
>         prior({weight:week},  normal(0, 100))
>         prior({var_0},        igamma(0.001, 0.001))
>         prior({var_id},       igamma(0.001, 0.001))
>         mcmcsize(5000) dots
Burn-in 2500 aaaaaaaa.1000.....2000..... done
Simulation 5000 .....1000.....2000.....3000.....4000.....50
> 00 done
Model summary
------------------------------------------------------------------------------
Likelihood:
  weight ~ normal(xb_weight,{var_0})
Priors:
   {weight:i.id} ~ normal({weight:_cons},{var_id})                        (1)
   {weight:week} ~ normal(0,100)                                          (1)
         {var_0} ~ igamma(0.001,0.001)
  {weight:_cons} ~ normal(0,100)
Hyperprior:
  {var_id} ~ igamma(0.001,0.001)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_weight.
Bayesian normal regression                      MCMC iterations  =      7,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =      5,000
                                                Number of obs    =        432
                                                Acceptance rate  =      .2382
                                                Efficiency:  min =     .00136
                                                             avg =    .004915
Log marginal likelihood = -1483.9819                         max =     .03084
------------------------------------------------------------------------------
             |                                                      Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
weight       |
        week |  6.263434   .0264724   .002955   6.262433   6.214032    6.31423
             |
          id |
           1 |  16.24666   .2357628   .058097    16.2599   15.78635   16.67799
           2 |  24.06862   .3243331    .06509   24.07464   23.37339   24.67859
  (output omitted)
          47 |  29.73823   .3734104    .07144   29.71473   29.04301   30.48604
          48 |  20.82722   .4258745   .160651   20.78619   20.13018   21.71069
-------------+----------------------------------------------------------------
      var_0  |  9.218097   .5679745   .174024   9.181747   8.218479   10.38655
-------------+----------------------------------------------------------------
weight       |
       _cons |  13.59053   .3519081   .028341   13.62244   12.88323   14.25594
-------------+----------------------------------------------------------------
      var_id |  12.49858   .3116721   .050076   12.50611    11.9335   13.12018
------------------------------------------------------------------------------
Note: There is a high autocorrelation after 500 lags.
```

`bayesmh` reports the presence of a high correlation after 500 lags. This and the low average efficiency of 0.005 may indicate problems with MCMC convergence for some of the parameters.

For convenience, we use `bayesstats summary` to show posterior summaries for parameters of interest only. Alternatively, you can specify the `noshow(i.id)` option with `bayesmh` to suppress the summaries for factor levels.

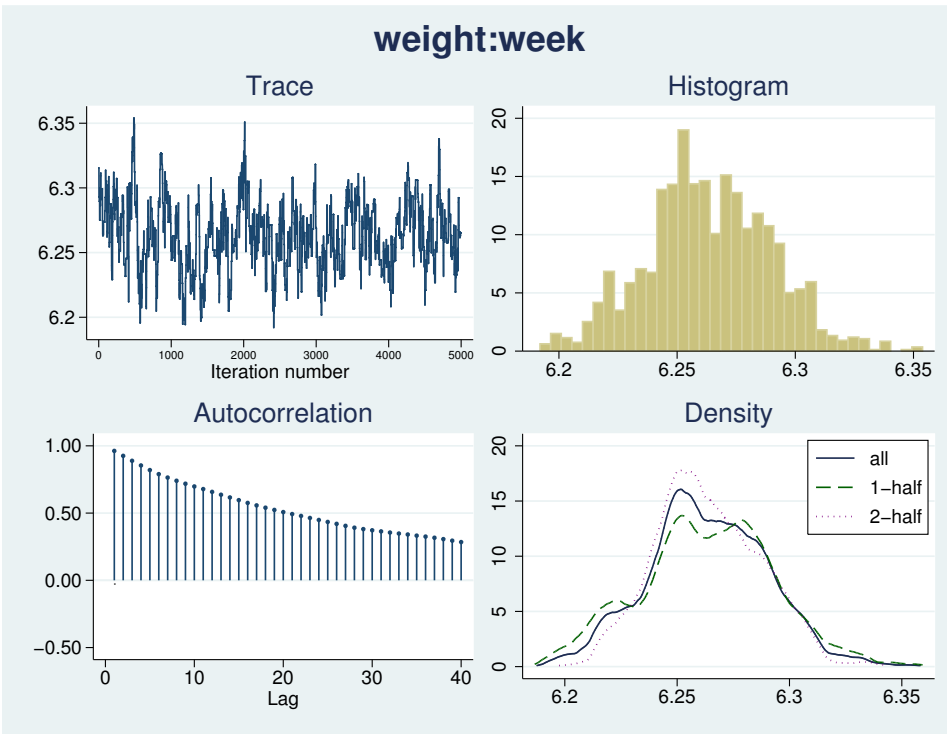
```
. bayesstats summary {weight:week _cons} {var_0} {var_id}
Posterior summary statistics                         MCMC sample size =     5,000
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
weight						
week	6.263434	.0264724	.002955	6.262433	6.214032	6.31423
_cons	13.59053	.3519081	.028341	13.62244	12.88323	14.25594
var_0	9.218097	.5679745	.174024	9.181747	8.218479	10.38655
var_id	12.49858	.3116721	.050076	12.50611	11.9335	13.12018

The posterior mean estimates for {weight:week} and {weight:_cons} are 6.26 and 13.59, respectively. The estimate for the residual variance {var_0} is 9.22 with the standard deviation of 0.57, and the estimate of the group-effect variance {var_id} is 12.5 with the standard deviation of 0.31.

Because of the low efficiencies, we should be suspicious of these results. If we look at diagnostic plots for, for example, {weight:week},

```
. bayesgraph diagnostics {weight:week}
```



we see that the trace plot exhibits some trend and does not show good mixing and that the autocorrelation is relatively high after at least lag 40. Our MCMC does not seem to converge and thus we cannot trust the obtained results.

◁

▷ Example 20: Second simulation—blocking of parameters

Continuing example 19, we can improve efficiency of the MH algorithm by separating model parameters into blocks to be sampled independently. We consider a separate block for each model parameter with random-effects parameters sharing the same block. We also specify `nomodelsummary` to suppress the model summary and `notable` to suppress the table output of `bayesmh`.

```
. set seed 14
. bayesmh weight week i.id, likelihood(normal({var_0})) noconstant
> prior({weight:i.id}, normal({weight:_cons},{var_id}))
> prior({weight:_cons},normal(0, 100))
> prior({weight:week}, normal(0, 100))
> prior({var_0}, igamma(0.001, 0.001))
> prior({var_id}, igamma(0.001, 0.001))
> block({var_0})
> block({var_id})
> block({weight:i.id})
> block({weight:week})
> block({weight:_cons})
> burnin(3000) mcmcsize(5000) dots notable nomodelsummary
Burn-in 3000 aaaaaaaaa1000aaaaaaaaa2000aaaaaaaaa3000 done
Simulation 5000 .........1000.........2000.........3000.........4000.........50
> 00 done
Bayesian normal regression                      MCMC iterations  =      8,000
Random-walk Metropolis-Hastings sampling        Burn-in          =      3,000
                                                MCMC sample size =      5,000
                                                Number of obs    =        432
                                                Acceptance rate  =      .4194
                                                Efficiency:  min =    .001727
                                                             avg =     .01731
Log marginal likelihood = -1204.9586                         max =      .2403
Note: There is a high autocorrelation after 500 lags.
```

Blocking certainly improved efficiencies: the average efficiency is now 0.017, but we still have a note about high autocorrelation.

We use `bayesstats summary` below to report summaries of only model parameters of interest.

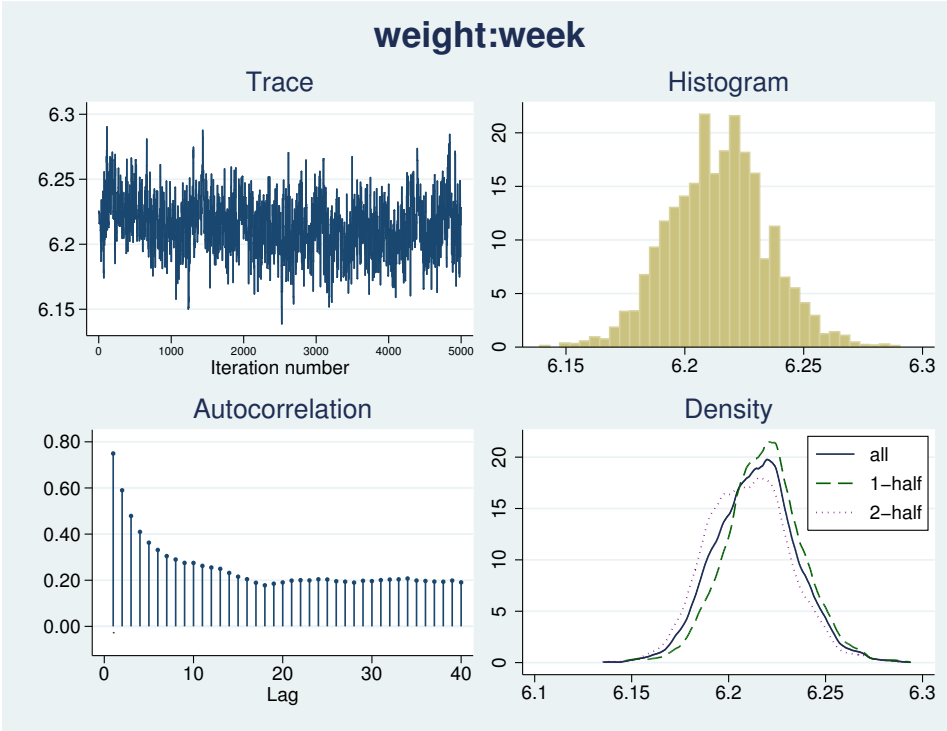
```
. bayesstats summary {weight:week _cons} {var_0} {var_id}
Posterior summary statistics                     MCMC sample size =     5,000
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
weight						
week	6.214099	.020815	.002059	6.214429	6.174678	6.255888
_cons	19.28371	.552023	.015925	19.28177	18.2078	20.35016
var_0	4.183143	.2908152	.009833	4.167876	3.669035	4.828092
var_id	15.53468	3.251813	.112054	15.16295	10.46451	23.19296

Here our estimates of variance components change noticeably: `{var_0}` is 4.18 and `{var_id}` is 15.53.

The diagnostic plots for {weight:week} are much better, but the mixing of MCMC is still not great.

```
. bayesgraph diagnostics {weight:week}
```



Example 21: Third simulation—Gibbs sampling

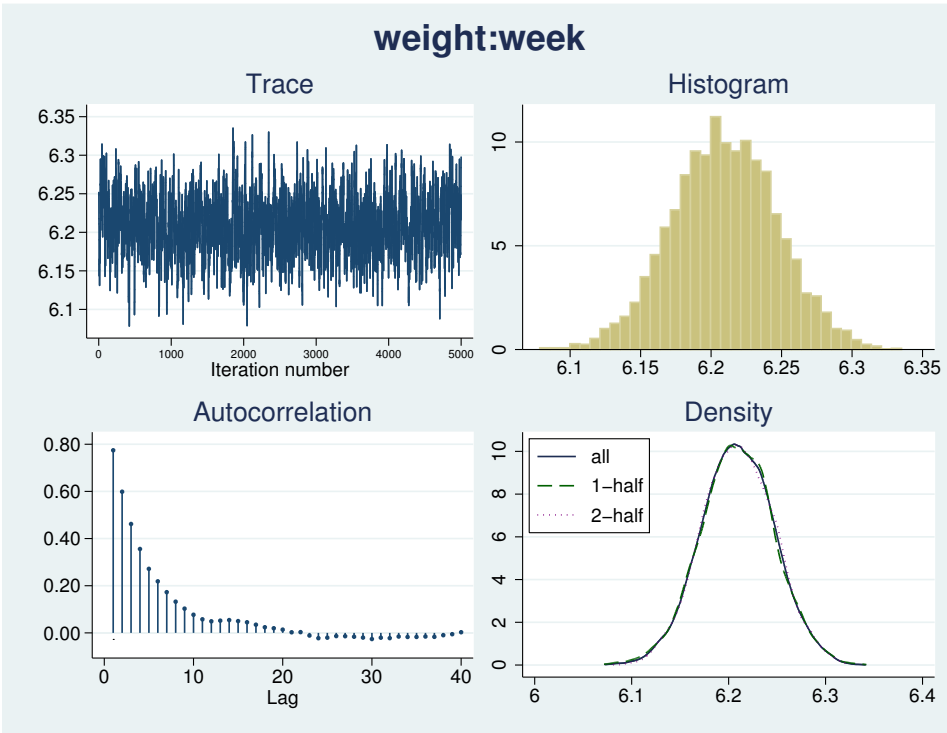
The most efficient MCMC procedure for our Bayesian model is Gibbs sampling, which can be set up as follows. To request a Gibbs sampling for a block of model parameters, we must first define them in a separate prior() statement and then put them in a separate block() with the gibbs suboption.

```
. set seed 14

. bayesmh weight week i.id, likelihood(normal({var_0})) noconstant
> prior({weight:i.id}, normal({weight:_cons},{var_id}))
> prior({weight:_cons},normal(0, 100))
> prior({weight:week}, normal(0, 100))
> prior({var_0}, igamma(0.001, 0.001))
> prior({var_id}, igamma(0.001, 0.001))
> block({var_0}, gibbs) block({var_id}, gibbs)
> block({weight:i.id}, gibbs) block({weight:week}, gibbs)
> block({weight:_cons},gibbs) mcmcsize(5000) dots notable nomodelsummary
Burn-in 2500 aaaaaaaaa1000aaaaaaaaa2000aaaaa done
Simulation 5000 .........1000.........2000.........3000.........4000.........50
> 00 done

Bayesian normal regression                      MCMC iterations  =      7,500
Gibbs sampling                                  Burn-in          =      2,500
                                                MCMC sample size =      5,000
                                                Number of obs    =        432
                                                Acceptance rate  =          1
                                                Efficiency:  min =       .123
                                                             avg =      .6764
Log marginal likelihood = -1051.4228                         max =       .857
```

There is no note about high autocorrelation in this run. The average efficiency increased dramatically to 0.68. It appears that our MCMC has now converged.

If we again inspect the diagnostic plots of, for example, `{weight:week}`, we will now see a very good mixing.

```
. bayesgraph diagnostics {weight:week}
```



We again use bayesstats summary to see posterior summaries of the model parameters of interest.

```
. bayesstats summary {weight:week _cons} {var_0} {var_id}
Posterior summary statistics                     MCMC sample size =     5,000
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
weight						
week	6.209425	.0373593	.001507	6.209439	6.135128	6.282676
_cons	19.29971	.6097913	.012916	19.2999	18.11953	20.47267
var_0	4.414173	.3194018	.004992	4.396302	3.828712	5.099535
var_id	15.85026	3.45786	.052824	15.44261	10.34387	23.6678

With Gibbs sampling, our estimates change only slightly. For example, the estimates of variance components are 4.41 for {var_0:_cons} and 15.85 for {var_id}.

All estimates are very close to the MLEs obtained earlier with the mixed command.

◁

Linear growth curve model—a random-coefficient model

Continuing our pig data example from *Two-level random-intercept model or panel-data model*, we extend the random-intercept model to include random coefficients for week by using

$$\texttt{weight}_{ij} = \beta_0 + \beta_1 \texttt{week}_{ij} + u_{0j} + u_{1j}\texttt{week}_{ij} + \epsilon_{ij}$$

where u_{0j} is the random effect for pig and u_{1j} is the pig-specific random coefficient on week for $j = 1, \ldots, 48$ and $i = 1, \ldots, 9$.

We again use mixed to fit this model by using maximum likelihood.

```
. use http://www.stata-press.com/data/r14/pig
(Longitudinal analysis of pig weights)
. mixed weight week || id: week
Performing EM optimization:
Performing gradient-based optimization:
Iteration 0:   log likelihood = -869.03825
Iteration 1:   log likelihood = -869.03825
Computing standard errors:
Mixed-effects ML regression                     Number of obs      =       432
Group variable: id                              Number of groups   =        48
                                                Obs per group:
                                                              min =         9
                                                              avg =       9.0
                                                              max =         9
                                                Wald chi2(1)       =   4689.51
Log likelihood = -869.03825                     Prob > chi2        =    0.0000
```

weight	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
week	6.209896	.0906819	68.48	0.000	6.032163	6.387629
_cons	19.35561	.3979159	48.64	0.000	18.57571	20.13551

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
id: Independent				
var(week)	.3680668	.0801181	.2402389	.5639103
var(_cons)	6.756364	1.543503	4.317721	10.57235
var(Residual)	1.598811	.1233988	1.374358	1.85992

LR test vs. linear model: chi2(2) = 764.42 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

Consider the following Bayesian model for these data:

$$\texttt{weight}_{ij} = \beta_0 + \beta_1\texttt{week}_{ij} + u_{0j} + u_{1j}\texttt{week}_{ij} + \epsilon_{ij} = \tau_{0j} + \tau_{1j}\texttt{week}_{ij} + \epsilon_{ij},$$

$$\begin{aligned}
\epsilon_{ij} &\sim \text{i.i.d. } N(0, \sigma_0^2)\\
\tau_{0j} &\sim \text{i.i.d. } N(\beta_0, \sigma_{\text{id}}^2)\\
\tau_{1j} &\sim \text{i.i.d. } N(\beta_1, \sigma_{\text{week}}^2)\\
\beta_0 &\sim N(0, 100)\\
\beta_1 &\sim N(0, 100)\\
\sigma_0^2 &\sim \text{InvGamma}(0.001, 0.001)\\
\sigma_{\text{id}}^2 &\sim \text{InvGamma}(0.001, 0.001)\\
\sigma_{\text{week}}^2 &\sim \text{InvGamma}(0.001, 0.001)
\end{aligned}$$

The model has five main parameters of interest: regression coefficients β_0 and β_1 and variance components σ_0^2, σ_{id}^2, and σ_{week}^2. β_0 and β_1 are hyperparameters because they are specified as mean parameters of the prior distributions for random effects τ_{0j} and τ_{1j}, respectively. Random effects τ_{0j} and τ_{1j} are considered nuisance parameters. We again use normal priors for the regression coefficients and group levels identified by the `id` variable and their interactions with `week` and inverse-gamma priors for the variance parameters. We specify fairly noninformative priors.

To fit this model using `bayesmh`, we need to include random effects for pig in our regression model. This can be done by adding factor levels of the `id` variable to the regression by using the factor-variable specification `i.id`. This specification, by default, will omit one of the `id` categories as a base category. In our Bayesian model, we need to keep all categories of `id`:

```
. fvset base none id
```

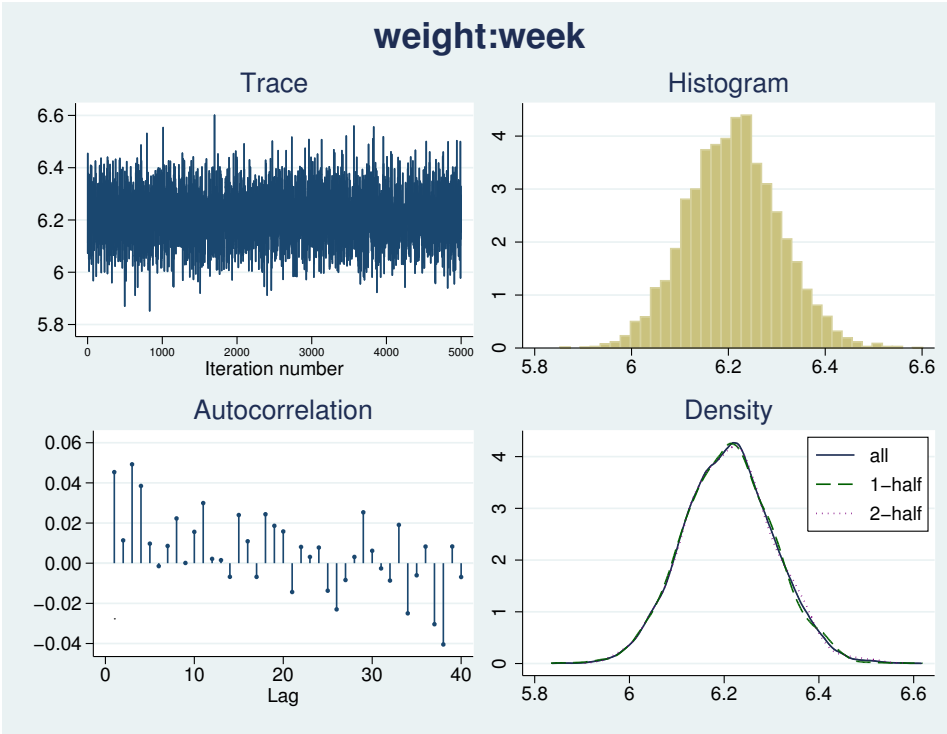
We fit our model using `bayesmh`. Following example 21, we perform blocking of parameters and use Gibbs sampling for the blocks.

```
. set seed 14
. bayesmh weight i.id i.id#c.week, likelihood(normal({var_0})) noconstant
> prior({weight:i.id}, normal({weight:_cons},{var_id}))
> prior({weight:i.id#c.week}, normal({weight:week},{var_week}))
> prior({weight:_cons}, normal(0, 100))
> prior({weight:week}, normal(0, 100))
> prior({var_0}, igamma(0.001, 0.001))
> prior({var_id}, igamma(0.001, 0.001))
> prior({var_week}, igamma(0.001, 0.001))
> block({var_0}, gibbs)
> block({var_id}, gibbs)
> block({var_week}, gibbs)
> block({weight:i.id}, gibbs)
> block({weight:i.id#c.week}, gibbs)
> block({weight:week}, gibbs)
> block({weight:_cons}, gibbs)
> mcmcsize(5000) dots notable
Burn-in 2500 aaaaaaaaa1000aaaaaaaaa2000aaaaa done
Simulation 5000 .........1000.........2000.........3000.........4000.........50
> 00 done
Model summary
------------------------------------------------------------------------------
Likelihood:
  weight ~ normal(xb_weight,{var_0})
Priors:
         {weight:i.id} ~ normal({weight:_cons},{var_id})                   (1)
  {weight:i.id#c.week} ~ normal({weight:week},{var_week})                  (1)
              {var_0} ~ igamma(0.001,0.001)
   {weight:_cons week} ~ normal(0,100)
Hyperprior:
  {var_id var_week} ~ igamma(0.001,0.001)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_weight.
Bayesian normal regression                      MCMC iterations  =      7,500
Gibbs sampling                                  Burn-in          =      2,500
                                                MCMC sample size =      5,000
                                                Number of obs    =        432
                                                Acceptance rate  =          1
                                                Efficiency:  min =     .08386
                                                             avg =      .1582
Log marginal likelihood = -929.94517                         max =      .7758
```

Our AR is good and efficiencies are high. We do not have a reason to suspect nonconvergence. Nevertheless, it is important to perform graphical convergence diagnostics to confirm this.

Let's look at diagnostic plots. We show only diagnostic plots for the mean of random coefficients on **week**, but convergence should be established for all parameters before any inference can be made. We leave it to you to verify convergence of the remaining parameters.

```
. bayesgraph diagnostics {weight:week}
```



The diagnostic plots look good.

Our posterior mean estimates of the main model parameters are in agreement with maximum likelihood results from `mixed`, as is expected with noninformative priors.

```
. bayesstats summary {weight:week _cons} {var_0} {var_id} {var_week}
Posterior summary statistics                       MCMC sample size =     5,000
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
weight						
week	6.210054	.0948751	.001523	6.210372	6.029255	6.398015
_cons	19.32719	.4096827	.007805	19.32701	18.53177	20.14601
var_0	1.607193	.1224062	.002371	1.600899	1.384723	1.863646
var_id	7.253204	1.705803	.038343	7.034003	4.566251	11.32263
var_week	.3940417	.0886511	.001723	.3822614	.2545719	.607737

Bayesian analysis of change-point problem

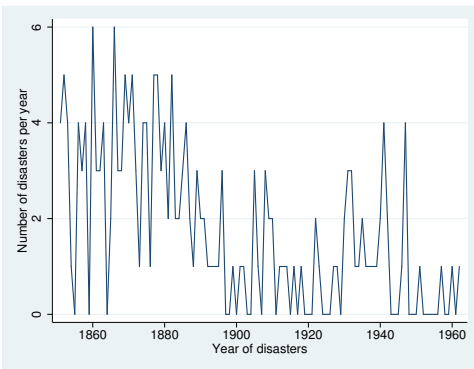
Change-point problems deal with stochastic data, usually time-series data, which undergoes some abrupt change at some time point. It is of interest to localize the point of change and estimate the properties of the stochastic process before and after the change.

Here we analyze the British coal mining disaster data for the years 1851 to 1962 as given in Table 5 in Carlin, Gelfand, and Smith (1992). The data are originally from Maguire, Pearson, and Wynn (1952) with updates from Jarrett (1979).

`coal.dta` contains 112 observations, and it includes the variables `id`, which records observation identifiers; `count`, which records the number of coal mining disasters involving 10 or more deaths; and `year`, which records the years corresponding to the disasters.

```
. use http://www.stata-press.com/data/r14/coal
(British coal-mining disaster data, 1851-1962)
. describe
Contains data from http://www.stata-press.com/data/r14/coal.dta
  obs:           112                          British coal-mining disaster
                                                data, 1851-1962
 vars:             3                          5 Feb 2015 18:03
 size:           560                          (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
id              int     %9.0g                 Observation identifier
year            int     %9.0g                 Year of disasters
count           byte    %9.0g                 Number of disasters per year
-------------------------------------------------------------------------------
Sorted by:
```

The figures below suggest a fairly abrupt decrease in the rate of disasters around the 1887–1895 period, possibly because of the decline in labor productivity in coal mining (Raftery and Akman 1986). The line plot of `count` versus `year` is shown in the left pane and its smoothed version in the right pane.



To find the change-point parameter (cp) in the rate of disasters, we apply the following Bayesian model with noninformative priors for the parameters (accounting for the restricted range of cp):

$$\begin{aligned}
\texttt{counts}_i &\sim \text{Poisson}(\mu_1),\ \texttt{if year}_i < cp\\
\texttt{counts}_i &\sim \text{Poisson}(\mu_2),\ \texttt{if year}_i \geq cp\\
\mu_1 &\sim 1\\
\mu_2 &\sim 1\\
cp &\sim \text{Uniform}(1851, 1962)
\end{aligned}$$

The model has three parameters: μ_1, μ_2, and cp, which we will declare as `{mu1}`, `{mu2}`, and `{cp}` with `bayesmh`. One interesting feature of this model is the specification of a mixture distribution for `count`. To accommodate this, we use the nonlinear specification of `bayesmh` and specify the substitutable expression

```
({mu1}*sign(year<{cp})+{mu2}*sign(year>={cp}))
```

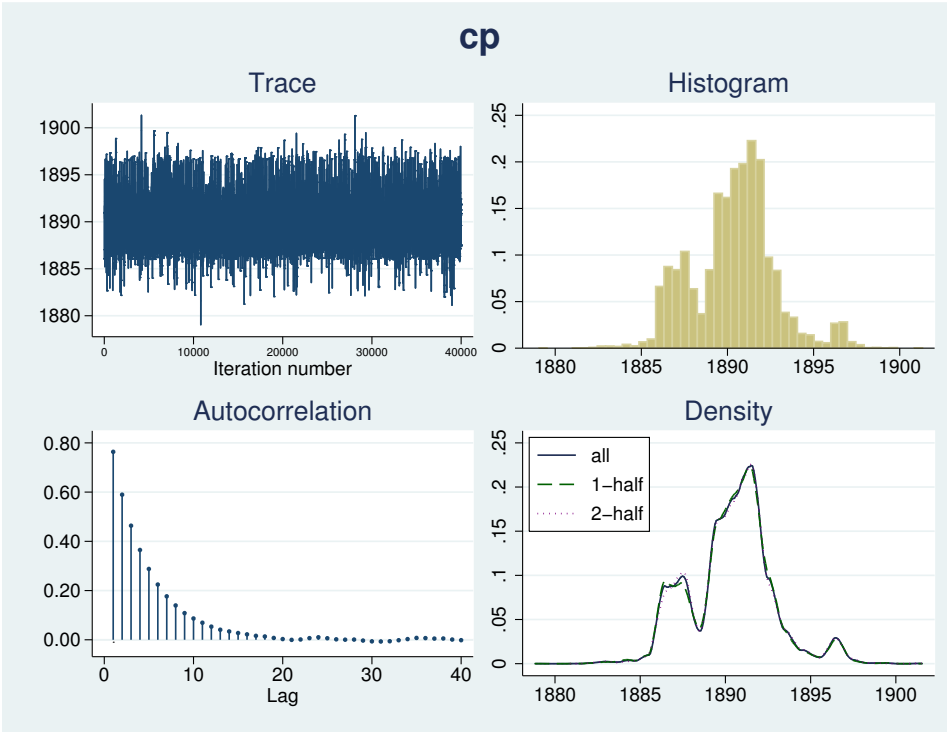
as the mean of a Poisson distribution. To ensure the feasibility of the initial state, we specify the desired initial values in option `initial()`. Because of high autocorrelation in the MCMC chain, we increase the MCMC size to achieve higher precision of our estimates. We change the default title to the title specific to our analysis. To monitor the progress of simulation, we request that `bayesmh` displays a dot every 500 iterations and an iteration number every 5,000 iterations.

```
. set seed 14
. bayesmh count = ({mu1}*sign(year<{cp})+{mu2}*sign(year>={cp})),
> likelihood(poisson, noglmtransform)
> prior({mu1} {mu2}, flat)
> prior({cp}, uniform(1851,1962))
> initial({mu1} 1 {mu2} 1 {cp} 1906)
> mcmcsize(40000) title(Change-point analysis) dots(500, every(5000))
Burn-in 2500 aa... done
Simulation 40000 .........5000.........10000.........15000.........20000.......
> ..25000.........30000.........35000.........40000 done
Model summary
------------------------------------------------------------------------------
Likelihood:
  count ~ poisson({mu1}*sign(year<{cp})+{mu2}*sign(year>={cp}))
Priors:
  {mu1 mu2} ~ 1 (flat)
       {cp} ~ uniform(1851,1962)
------------------------------------------------------------------------------
Change-point analysis                           MCMC iterations  =     42,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     40,000
                                                Number of obs    =        112
                                                Acceptance rate  =      .2243
                                                Efficiency:  min =     .03456
                                                             avg =      .0678
Log marginal likelihood = -173.33996                         max =      .1256
------------------------------------------------------------------------------
             |                                                     Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
         mu1 |  3.136251   .2942003   .007913   3.131459   2.599068   3.731112
          cp |  1890.358   2.424871   .034217   1890.554    1886.07   1896.303
         mu2 |  .9410287   .1199134   .002882   .9370863   .7219138   1.189728
------------------------------------------------------------------------------
```

According to our results, the change occurred in the first half of 1890. The drop of the disaster rate was significant, from an estimated average of 3.136 to 0.94.

The diagnostic plots, for example, for {cp} do not indicate any convergence problems. (This is also true for other parameters.)

```
. bayesgraph diagnostics {cp}
```



The simulated marginal density of {cp} shown in the right bottom corner provides more details. Apart from the main peak, there are two smaller bumps around the years 1886 and 1896, which correspond to local peaks in the number of disasters at these years: 4 in 1886 and 3 in 1896.

We may be interested in estimating the ratio between the two means. We can use bayesstats summary to estimate this ratio.

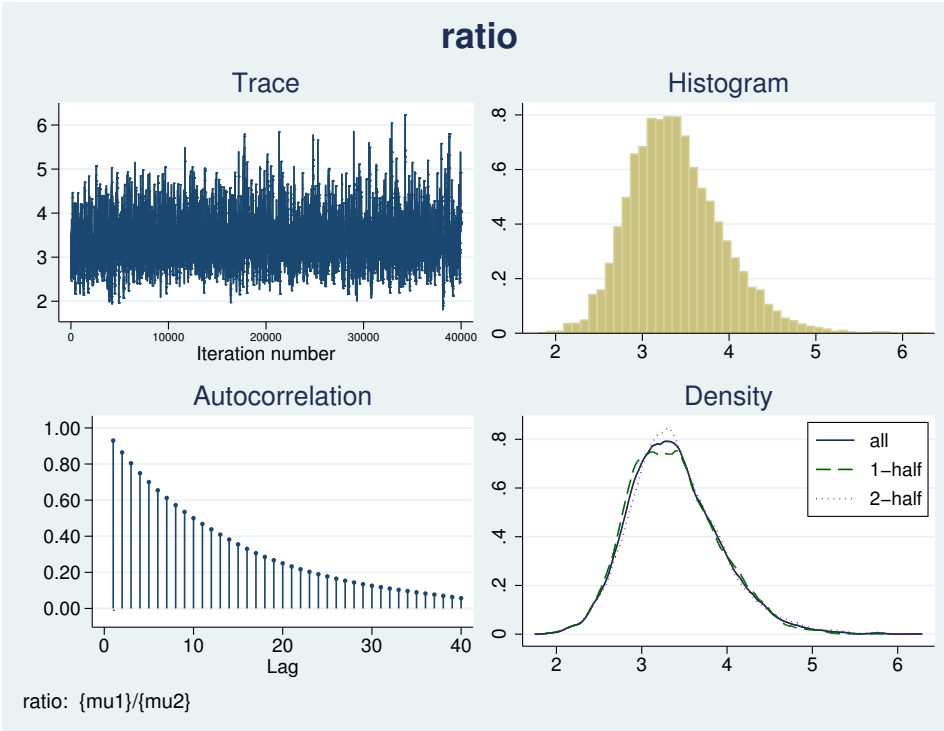
```
. bayesstats summary (ratio:{mu1}/{mu2})
Posterior summary statistics                         MCMC sample size =    40,000
       ratio : {mu1}/{mu2}
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
ratio	3.386058	.532557	.014112	3.336782	2.471534	4.553885

The posterior mean estimate of the ratio and its 95% credible intervals confirm the change between the two means. After 1890, the mean number of disasters decreased by a factor of about 3.4 with a 95% credible range of [2.47, 4.55].

Remember that convergence must be verified not only for all model parameters but also for the functions of interest. The diagnostic plots for `ratio` look good.

```
. bayesgraph diagnostics (ratio:{mu1}/{mu2})
```



Bioequivalence in a crossover trial

Balanced crossover designs are widely used in the pharmaceutical industry for testing the efficacy of new drugs. Gelfand et al. (1990) analyzed a two-treatment, two-period crossover trial comparing two Carbamazepine tablets. The data consist of log-concentration measurements and are originally described in Maas et al. (1987).

A random-effect two-treatment, two-period crossover design is given by

$$y_{i(jk)} = \mu + (-1)^{j-1}\frac{\phi}{2} + (-1)^{k-1}\frac{\pi}{2} + d_i + \epsilon_{i(jk)} = \mu_{i(jk)} + \epsilon_{i(jk)}$$

$$\epsilon_{i(jk)} \sim \text{i.i.d.} N(0, \sigma^2)$$
$$d_i \sim \text{i.i.d.} N(0, \tau^2)$$

where $i = 1, \ldots, n$ is the subject index, $j = 1, 2$ is the treatment group, and $k = 1, 2$ is the period.

`bioequiv.dta` has four main variables: subject identifier `id` from 1 to 10, treatment identifier `treat` containing values 1 or 2, period identifier `period` containing values 1 or 2, and outcome `y` measuring log concentration for the two tablets.

```
. use http://www.stata-press.com/data/r14/bioequiv
(Bioequivalent study of Carbamazepine tablets)
. describe
Contains data from http://www.stata-press.com/data/r14/bioequiv.dta
  obs:            20                          Bioequivalent study of
                                                Carbamazepine tablets
 vars:             5                          5 Feb 2015 23:45
 size:           160                          (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
obsid           byte    %9.0g                 Observation identifier
id              byte    %9.0g                 Subject identifier
treat           byte    %9.0g                 Assigned treatment
period          byte    %9.0g                 Period identifier
y               float   %9.0g                 Log-concentration measurement
-------------------------------------------------------------------------------
Sorted by: id  period
```

Before fitting `bayesmh`, we request no base category for the `id` variable.

```
. fvset base none id
```

The outcome is assumed to be normally distributed with mean $\mu_{i(jk)}$ and variance σ^2. To accommodate the specific structure of the regression function, we use a nonlinear specification of `bayesmh`. We specify the expression for the mean function $\mu_{i(jk)}$ as a nonlinear expression following the outcome `y`. We use noninformative priors for parameters and separate parameters in blocks. To improve convergence, we increase our adaptation and burn-in periods. (The command may take some time to produce results, so we specify the `dots()` option.)

```
. set seed 14
. bayesmh y = ({mu}+(-1)^(treat-1)*{phi}/2+(-1)^(period-1)*{pi}/2+{y:i.id}),
> likelihood(normal({var}))
> prior({y:i.id}, normal(0,{tau}))
> prior({tau}, igamma(0.001,0.001))
> prior({var}, igamma(0.001,0.001))
> prior({mu} {phi} {pi}, normal(0,1e6))
> block({y:i.id}, split)
> block({tau}, gibbs) block({var}, gibbs)
> adaptation(every(200) maxiter(50)) burnin(10000) dots(250, every(2500))
Burn-in 10000 aaaaaaaaa2500aaaaaaaaa5000aaaaaaaaa7500aaaaaaaaa10000 done
Simulation 10000 .........2500.........5000.........7500.........10000 done
Model summary
------------------------------------------------------------------------------
Likelihood:
  y ~ normal(<expr1>,{var})
Priors:
        {var} ~ igamma(0.001,0.001)
     {y:i.id} ~ normal(0,{tau})
  {mu phi pi} ~ normal(0,1e6)
Hyperprior:
  {tau} ~ igamma(0.001,0.001)
Expression:
  expr1 : {mu}+(-1)^(treat-1)*{phi}/2+(-1)^(period-1)*{pi}/2+({y:1bn.id}*1bn.i
          d+{y:2.id}*2.id+{y:3.id}*3.id+{y:4.id}*4.id+{y:5.id}*5.id+{y:6.id}*6
          .id+{y:7.id}*7.id+{y:8.id}*8.id+{y:9.id}*9.id+{y:10.id}*10.id)
------------------------------------------------------------------------------
```

```
Bayesian normal regression                      MCMC iterations  =     20,000
Metropolis-Hastings and Gibbs sampling          Burn-in          =     10,000
                                                MCMC sample size =     10,000
                                                Number of obs    =         20
                                                Acceptance rate  =      .5131
                                                Efficiency:  min =     .01345
                                                             avg =     .02821
Log marginal likelihood = -25.692825                         max =     .04365
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mu	1.43231	.0579197	.004993	1.434814	1.305574	1.545945
phi	-.0093502	.050824	.00257	-.0104379	-.1039488	.1010855
pi	-.1815055	.0542115	.003107	-.1821367	-.2963565	-.0702212
y						
id						
1	.0668345	.0834954	.005428	.0645855	-.0879197	.2407731
2	.1217473	.0895501	.005941	.1190309	-.037415	.308847
3	.0561551	.0812912	.005154	.0525818	-.0971676	.2344846
4	.0619807	.0827296	.005294	.0564789	-.0923602	.2365587
5	.1701813	.09874	.006345	.1685315	-.0149722	.3676389
6	-.1640241	.0917804	.005572	-.1690176	-.3443967	.0135562
7	-.1191101	.0864379	.005291	-.1168358	-.2894083	.0400566
8	-.0590061	.0803792	.004595	-.0572132	-.2217439	.0908653
9	-.0779055	.0814977	.00481	-.0769495	-.2428321	.0816219
10	-.014813	.0788845	.00452	-.0138628	-.1750312	.1463467
var	.0134664	.0087676	.000482	.0109334	.0042003	.0370388
tau	.0228884	.020285	.000971	.0182243	.0015547	.0725889

Sampling efficiencies look reasonable considering the number of model parameters. The diagnostic plots of the main model parameters (not shown here) look reasonable except there is a high autocorrelation in the MCMC for `{mu}`, so you may consider increasing the MCMC size or using thinning.

Parameter $\theta = \exp(\phi)$ is commonly used as a measure of bioequivalence. Bioequivalence is declared whenever θ lies in the interval $(0.8, 1.2)$ with a high posterior probability.

We use `bayesstats summary` to calculate this probability and to also display other main parameters.

```
. bayesstats summary {mu} {phi} {pi} {tau} {var}
> (theta:exp({phi})) (equiv:exp({phi})>0.8 & exp({phi})<1.2)
Posterior summary statistics                    MCMC sample size =     10,000
       theta : exp({phi})
       equiv : exp({phi})>0.8 & exp({phi})<1.2
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mu	1.43231	.0579197	.004993	1.434814	1.305574	1.545945
phi	-.0093502	.050824	.00257	-.0104379	-.1039488	.1010855
pi	-.1815055	.0542115	.003107	-.1821367	-.2963565	-.0702212
tau	.0228884	.020285	.000971	.0182243	.0015547	.0725889
var	.0134664	.0087676	.000482	.0109334	.0042003	.0370388
theta	.9919787	.0507755	.002569	.9896164	.9012714	1.106371
equiv	.9982	.0423903	.000892	1	1	1

We obtain an estimate of 0.998 for the posterior probability of bioequivalence specified as an expression `equiv`. So we would conclude bioequivalence between the two tablets.

Random-effects meta-analysis of clinical trials

In meta-analysis of clinical trials, one considers several distinct studies estimating an effect of interest. It is convenient to consider the true effect as varying randomly between the studies. A detailed description of the random-effects meta-analysis can be found in, for example, Carlin (1992).

We illustrate Bayesian random-effects meta-analysis of 2×2 tables for the beta-blockers dataset analyzed in Carlin (1992). These data are also analyzed in Yusuf, Simon, and Ellenberg (1987). The data summarize the results of 22 clinical trials of beta-blockers used as postmyocardial infarction treatment.

▷ Example 22: Normal–normal analysis

Here we follow the approach of Carlin (1992) for the normal–normal analysis of the beta-blockers data.

For our normal–normal analysis, we consider data in wide form and concentrate on modeling estimates of log odds-ratios from 22 studies.

```
. use http://www.stata-press.com/data/r14/betablockers_wide
(Beta-blockers data in wide form)
. describe
Contains data from http://www.stata-press.com/data/r14/betablockers_wide.dta
  obs:            22                          Beta-blockers data in wide form
 vars:             7                          5 Feb 2015 19:02
 size:           550                          (_dta has notes)
```

variable name	storage type	display format	value label	variable label
study	byte	%9.0g		Study identifier
deaths0	int	%9.0g		Number of deaths in the control group
total0	int	%9.0g		Number of subjects in the control group
deaths1	int	%9.0g		Number of deaths in the treatment group
total1	int	%9.0g		Number of subjects in the treatment group
D	double	%10.0g		Log odds-ratio (based on empirical logits)
var	double	%10.0g		Squared standard error of log odds-ratio

```
Sorted by:
```

The estimates of log odds-ratios and their squared standard errors are recorded in variables `D` and `var`, respectively. They are computed from variables `deaths0`, `total0`, `deaths1`, and `total1` based on empirical logits; see Carlin (1992, eq. (3) and (4)). The `study` variable records study identifiers.

In a normal–normal model, we assume a random-effects model for estimates of log odds-ratios with normally distributed errors and normally distributed random effects. Specifically,

$$\mathtt{D}_i = d + u_i + \epsilon_i = d_i + \epsilon_i$$

where $\epsilon_i \sim N(0, \mathtt{var}_i)$ and $d_i \sim N(d, \sigma^2)$. Errors ϵ_is represent uncertainty about estimates of log odds-ratios in each study i and are assumed to have known study-specific variances, $\mathtt{var}_i$s. Random effects d_is represent differences in estimates of log odds-ratios from study to study. The estimates of

their mean and variance are of interest in meta-analysis: d estimates a true effect and σ^2 estimates variation in estimating this effect across studies. Small values of σ^2 imply that the estimates of a true effect agree among studies.

In Bayesian analysis, we additionally specify prior distributions for d and σ^2. Following Carlin (1992), we use noninformative priors for these parameters: normal with large variance for d and inverse gamma with very small degrees of freedom for σ^2.

$$d \sim N(0, 1000)$$
$$\sigma^2 \sim \text{InvGamma}(0.001, 0.001)$$

In our data, random effects d_i is represented by a factor variable `i.study`. We use all levels of `study` in our analysis, so we use `fvset` to request no base level for this variable.

```
. fvset base none study
```

We specify `normal()` likelihood with `bayesmh` and request observation-specific variances by specifying variable `var` as `normal()`'s variance argument. We follow the above model formulation for specifying prior distributions. To improve efficiency, we request that all parameters be placed in separate blocks and use Gibbs sampling for the mean parameter `{d}` and the variance parameter `{sig2}`. We also increase the burn-in period to 3,000 iterations and request more frequent adaptation by specifying the `adaptation(every(10))` option. The command will take a little longer to run, so we request that a dot be displayed every 500 iterations and an iteration number be displayed every 2,500 iterations to monitor the progress of the simulation.

```
. set seed 14
. bayesmh D i.study, likelihood(normal(var)) noconstant
> prior({D:i.study}, normal({d},{sig2}))
> prior({d}, normal(0,1000))
> prior({sig2}, igamma(0.001,0.001))
> block({D:i.study}, split)
> block({sig2}, gibbs)
> block({d}, gibbs)
> burnin(3000) adaptation(every(10)) dots(500, every(2500))
Burn-in 3000 aaaa2500a done
Simulation 10000 ....2500....5000....7500....10000 done
Model summary
------------------------------------------------------------------------------
Likelihood:
  D ~ normal(xb_D,var)
Prior:
  {D:i.study} ~ normal({d},{sig2})                                      (1)
Hyperpriors:
     {d} ~ normal(0,1000)
  {sig2} ~ igamma(0.001,0.001)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_D.
```

```
Bayesian normal regression                      MCMC iterations  =     13,000
Metropolis-Hastings and Gibbs sampling          Burn-in          =      3,000
                                                MCMC sample size =     10,000
                                                Number of obs    =         22
                                                Acceptance rate  =      .5315
                                                Efficiency:  min =     .01845
                                                             avg =     .04462
Log marginal likelihood =   14.38145                         max =     .06842
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
D						
study						
1	-.2357346	.1380931	.005394	-.2396019	-.5018659	.0564967
2	-.2701697	.135307	.006741	-.2585033	-.5760455	-.0174336
3	-.2538771	.1376569	.005263	-.2495234	-.5436489	.0222503
4	-.246526	.08904	.003506	-.2483908	-.4212739	-.0643877
5	-.1969971	.12748	.006635	-.2072718	-.4149274	.1014951
6	-.2527047	.1339466	.00647	-.2526702	-.5224128	.0229356
7	-.3377723	.1100308	.006646	-.3283355	-.5829385	-.1548902
8	-.2054826	.1130796	.005594	-.2121369	-.4051584	.0546629
9	-.2666327	.1215781	.005263	-.2630645	-.5206763	-.0297599
10	-.2803866	.0841634	.003593	-.2771339	-.4590086	-.1252279
11	-.2354098	.1049351	.004449	-.237795	-.4360951	-.0191799
12	-.202938	.1178808	.005967	-.209884	-.4105608	.0725293
13	-.2714193	.1288598	.006394	-.263365	-.564746	-.023963
14	-.1273999	.1468804	.009997	-.1553146	-.3495763	.2172828
15	-.2518538	.1249082	.005184	-.2502685	-.5090334	-.0021013
16	-.2245814	.1210757	.004998	-.231592	-.4488306	.0415657
17	-.2043954	.1357651	.007347	-.2164064	-.4321717	.1044344
18	-.2153688	.1423256	.006983	-.222428	-.4718119	.0991941
19	-.2242526	.1360964	.006098	-.2300817	-.4938685	.075416
20	-.2428998	.1151988	.005403	-.2424417	-.4723024	-.0126589
21	-.2972177	.1281401	.006041	-.2862546	-.5946982	-.0770212
22	-.2979427	.1266137	.00575	-.2885006	-.5953839	-.0816952
d	-.2429052	.0611413	.004501	-.2426092	-.3623229	-.1261924
sig2	.0166923	.020771	.001488	.0095773	.0007359	.0753652

Our posterior mean estimates `d` and `sig2` of mean d and variance σ^2 are -0.24 and 0.017, respectively, with posterior standard deviations of 0.06 and 0.02. The estimates are close to those reported by Carlin (1992). Considering the number of parameters, the AR and efficiency summaries look good.

We can obtain the efficiencies for the main parameters by using `bayesstats ess`.

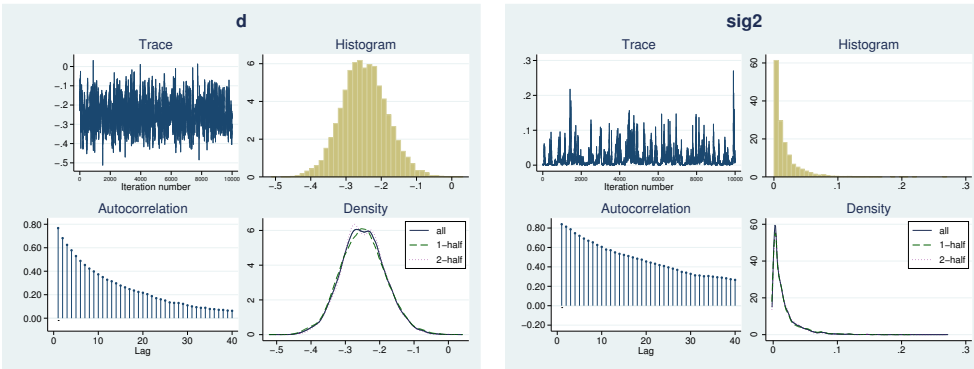
```
. bayesstats ess {d} {sig2}
Efficiency summaries    MCMC sample size =    10,000
```

	ESS	Corr. time	Efficiency
d	184.49	54.20	0.0184
sig2	194.88	51.31	0.0195

The efficiencies are acceptable, but based on the correlation times, the autocorrelation becomes small only after lag 50 or so. The precision of the mean and variance estimates is comparable to those based on 184 independent observations for the mean and 195 independent observations for the variance.

We explore convergence visually.

```
. bayesgraph diagnostics {d} {sig2}
```



The diagnostic plots look reasonable for both parameters, but autocorrelation is high. You may consider increasing the default MCMC size to obtain more precise estimates of posterior means.

◁

▷ Example 23: Binomial-normal model

There is an alternative but equivalent way of formulating the meta-analysis model from example 23 as a binomial-normal model. Instead of modeling estimates of log odds-ratios directly, one can model probabilities of success (an event of interest) in each group.

Let p_i^T and p_i^C be the probabilities of success for the treatment and control groups in the ith trial. The random-effects meta-analysis model can be given as

$$\text{logit}(p_i^C) = \mu_i$$
$$\text{logit}(p_i^T) = \mu_i + d_i$$

where μ_i is log odds of success in the control group in study i and $\mu_i + d_i$ is log odds of success in the treatment group. d_is are viewed as random effects and are assumed to be normally distributed as

$$d_i \sim \text{i.i.d.}N(d, \sigma^2)$$

where d is the population effect and σ^2 is its variability across trials.

Suppose that we observe y_i^C successes out of n_i^C events in the control group and y_i^T successes out of n_i^T events in the treatment group from the ith trial. Then,

$$y_i^C \sim \text{Binomial}(p_i^C, n_i^C)$$
$$y_i^T \sim \text{Binomial}(p_i^T, n_i^T)$$

The random effects are usually assumed to be normally distributed as

$$d_i \sim \text{i.i.d.}N(d, \sigma^2)$$

where d is the population effect and is the main parameter of interest in the model, and σ^2 is its variability across trials.

We can rewrite the model above assuming the data are in long form as

$$\begin{aligned}\text{logit}(p_i) &= \mu_i + (T_i == 1) \times d_i\\ y_i &\sim \text{Binomial}(p_i, n_i)\\ d_i &\sim \text{i.i.d.}N(d, \sigma^2)\end{aligned}$$

where T_i is a binary treatment with $T_i = 0$ for the control group and $T_i = 1$ for the treatment group.

In Bayesian analysis, we additionally specify prior distributions for μ_i, d, and σ^2. We use noninformative priors.

$$\begin{aligned}\mu_i &\sim 1\\ d &\sim N(0, 1000)\\ \sigma^2 &\sim \text{InvGamma}(0.001, 0.001)\end{aligned}$$

We continue our analysis of beta-blockers data. The analysis of these data using a binomial-normal model is also provided as an example in OpenBUGS (Thomas et al. 2006).

For this analysis, we use the beta-blockers data in long form.

```
. use http://www.stata-press.com/data/r14/betablockers_long
(Beta-blockers data in long form)
. describe
Contains data from http://www.stata-press.com/data/r14/betablockers_long.dta
  obs:            44                          Beta-blockers data in long form
 vars:             4                          5 Feb 2015 19:02
 size:           264                          (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
study           byte    %9.0g                 Study identifier
treat           byte    %9.0g      treatlab   Treatment group: 0 - control, 1 -
                                                treatment
deaths          int     %9.0g                 Number of deaths in each group
total           int     %9.0g                 Number of subjects in each group
-------------------------------------------------------------------------------
Sorted by: study  treat
```

Variable `treat` records the binary treatment: `treat==0` identifies the control group, and `treat==1` identifies the treatment group.

To relate to the notation of our model, we create variable `mu` to contain identifiers for each study. We also request that no base is set for our factor variables `mu` and `study`.

```
. generate mu = study
. fvset base none mu study
```

We use a `binlogit()` likelihood model for the number of `deaths`. We split all parameters into separate blocks and request Gibbs sampling for `sig2` to improve efficiency of the algorithm. We also specify `burnin(3000)` and perform more frequent adaptation using `adaptation(every(10))`.

```
. set seed 14
. bayesmh deaths i.mu 1.treat#i.study, likelihood(binlogit(total)) noconstant
> prior({deaths:i.mu}, flat)
> prior({deaths:1.treat#i.study}, normal({d},{sig2}))
> prior({d},normal(0,1000)) prior({sig2}, igamma(0.001,0.001))
> block({deaths:1.treat#i.study}, split)
> block({deaths:i.mu}, split) block({d}, gibbs)
> block({sig2}, gibbs)
> burnin(3000) adaptation(every(10)) dots(500, every(2500))
Burn-in 3000 aaaa2500a done
Simulation 10000 ....2500....5000....7500....10000 done
Model summary
------------------------------------------------------------------------------
Likelihood:
  deaths ~ binlogit(xb_deaths,total)
Priors:
             {deaths:i.mu} ~ 1 (flat)                                    (1)
  {deaths:i.treat#i.study} ~ normal({d},{sig2})                          (1)
Hyperpriors:
     {d} ~ normal(0,1000)
  {sig2} ~ igamma(0.001,0.001)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_deaths.
Bayesian binomial regression                    MCMC iterations  =     13,000
Metropolis-Hastings and Gibbs sampling          Burn-in          =      3,000
                                                MCMC sample size =     10,000
                                                Number of obs    =         44
                                                Acceptance rate  =      .5136
                                                Efficiency:  min =     .01331
                                                             avg =     .08388
Log marginal likelihood = -131.25444                         max =      .2121
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
deaths						
mu						
1	-2.434128	.445865	.009682	-2.426842	-3.332509	-1.634211
2	-2.186034	.2348462	.005222	-2.1798	-2.654988	-1.741019
3	-2.121815	.2711186	.006175	-2.111274	-2.680885	-1.617358
4	-2.395562	.0809699	.002675	-2.396577	-2.549786	-2.237609
5	-2.401359	.1540556	.004839	-2.397723	-2.697435	-2.099078
6	-2.221807	.3487384	.009421	-2.19026	-2.966567	-1.591299
7	-1.71437	.0784958	.002737	-1.714861	-1.86485	-1.558509
8	-2.110073	.1178488	.003906	-2.107124	-2.342254	-1.88807
9	-1.959062	.1492379	.004604	-1.958407	-2.264319	-1.682202
10	-2.241497	.0699547	.002446	-2.240693	-2.38241	-2.107086
11	-2.308927	.1095127	.003416	-2.310487	-2.527959	-2.095512
12	-1.458926	.1263283	.003392	-1.457141	-1.709941	-1.207061
13	-2.993073	.2129428	.004776	-2.985876	-3.43956	-2.606033
14	-2.722014	.1239681	.004773	-2.718786	-2.973852	-2.490443
15	-1.355571	.1596962	.004102	-1.354465	-1.676543	-1.04002
16	-1.489021	.1416432	.004123	-1.483373	-1.764335	-1.223957
17	-1.993007	.1853341	.005607	-1.98668	-2.378721	-1.646472
18	-2.964669	.2847685	.006746	-2.947758	-3.571054	-2.455845
19	-3.433652	.3440502	.007869	-3.421849	-4.142976	-2.799305
20	-1.486827	.1357797	.003528	-1.486625	-1.756737	-1.218571
21	-2.141426	.1384291	.00422	-2.140922	-2.410842	-1.870514
22	-2.923959	.1412969	.004275	-2.925278	-3.19683	-2.656737

treat#study						
1	1	-.2412583	.1366465	.005294	-.246471	-.5222947	.0360048
1	2	-.2805666	.1338706	.005662	-.2698089	-.5741051	-.0336747
1	3	-.2627764	.1328548	.008139	-.2548607	-.560451	-.0111309
1	4	-.2518226	.0939223	.004196	-.2503843	-.4532776	-.0673868
1	5	-.2017774	.1277379	.006441	-.2137416	-.4256577	.0927942
1	6	-.2586228	.1381994	.008395	-.2507483	-.5628148	.0127226
1	7	-.3472471	.1015792	.006818	-.3376469	-.5653234	-.1742163
1	8	-.2142745	.1108317	.005637	-.2186595	-.4201235	.0252865
1	9	-.278724	.1237785	.007732	-.2705361	-.5565986	-.048334
1	10	-.2895344	.0855712	.003741	-.2834565	-.4695875	-.1309819
1	11	-.2455467	.105304	.004622	-.2461274	-.4571545	-.0309278
1	12	-.2094773	.1127281	.005102	-.2184074	-.4059582	.0326186
1	13	-.2762859	.1352985	.007211	-.2669069	-.5767289	-.0217408
1	14	-.1279066	.1427634	.009247	-.1505654	-.3554016	.2047083
1	15	-.2617291	.1192822	.005606	-.2592285	-.5019967	-.0192021
1	16	-.2303032	.1178814	.005088	-.2340642	-.4559166	.0227396
1	17	-.2135575	.1312599	.006438	-.2233056	-.4489128	.0833568
1	18	-.2219846	.1455447	.006833	-.2345571	-.4844894	.1041897
1	19	-.2283609	.143887	.006233	-.2362389	-.4981321	.0853338
1	20	-.2433477	.116537	.00486	-.2461491	-.4661368	.0000666
1	21	-.3065246	.1182271	.007766	-.2933875	-.5769479	-.0992575
1	22	-.3038501	.1276486	.007902	-.2917561	-.6014336	-.0757054
d		-.249726	.060338	.004671	-.2481786	-.3694177	-.1323805
sig2		.0167392	.0191965	.001664	.0103045	.0007443	.0674249

This model has 22 more parameters than the model in example 22. The posterior mean estimates `d` and `sig2` of mean d and variance σ^2 are -0.25 and 0.017, respectively, with posterior standard deviations of 0.06 and 0.02. The estimates of the mean and variance are again close to the ones reported by Carlin (1992).

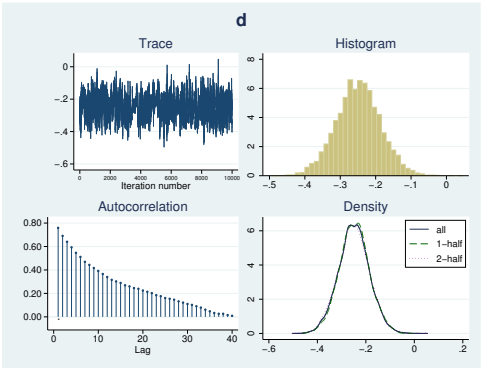
Compared with example 22, the efficiencies and other statistics for the main parameters are similar.

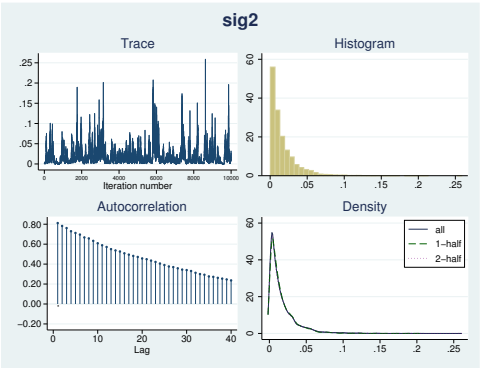
```
. bayesstats ess {d} {sig2}
Efficiency summaries     MCMC sample size =     10,000
```

	ESS	Corr. time	Efficiency
d	166.90	59.92	0.0167
sig2	133.14	75.11	0.0133

The diagnostic plots look similar to those shown in example 22.

```
. bayesgraph diagnostics {d} {sig2}
```





Stored results

`bayesmh` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(k)`	number of parameters
`e(k_sc)`	number of scalar parameters
`e(k_mat)`	number of matrix parameters
`e(n_eq)`	number of equations
`e(mcmcsize)`	MCMC sample size
`e(burnin)`	number of burn-in iterations
`e(mcmciter)`	total number of MCMC iterations
`e(thinning)`	thinning interval
`e(arate)`	overall AR
`e(eff_min)`	minimum efficiency
`e(eff_avg)`	average efficiency
`e(eff_max)`	maximum efficiency
`e(clevel)`	credible interval level
`e(hpd)`	`1` if `hpd` is specified; `0` otherwise
`e(batch)`	batch length for batch-means calculations
`e(corrlag)`	maximum autocorrelation lag
`e(corrtol)`	autocorrelation tolerance
`e(dic)`	deviation information criterion
`e(lml_lm)`	log marginal-likelihood using Laplace–Metropolis method
`e(scale)`	initial multiplier for scale factor; `scale()`
`e(block#_gibbs)`	`1` if Gibbs sampling is used in #th block; `0` otherwise
`e(block#_scale)`	#th block initial multiplier for scale factor
`e(block#_tarate)`	#th block target adaptation rate
`e(block#_arate_last)`	#th block AR from the last adaptive iteration
`e(block#_tolerance)`	#th block adaptation tolerance
`e(adapt_every)`	adaptation iterations `adaptation(every())`
`e(adapt_maxiter)`	maximum number of adaptive iterations `adaptation(maxiter())`
`e(adapt_miniter)`	minimum number of adaptive iterations `adaptation(miniter())`
`e(adapt_alpha)`	adaptation parameter `adaptation(alpha())`
`e(adapt_beta)`	adaptation parameter `adaptation(beta())`
`e(adapt_gamma)`	adaptation parameter `adaptation(gamma())`
`e(adapt_tolerance)`	adaptation tolerance `adaptation(tolerance())`
`e(repeat)`	number of attempts used to find feasible initial values

Macros

e(cmd)	bayesmh
e(cmdline)	command as typed
e(method)	sampling method
e(depvars)	names of dependent variables
e(eqnames)	names of equations
e(likelihood)	likelihood distribution (one equation)
e(likelihood#)	likelihood distribution for #th equation
e(prior)	prior distribution
e(prior#)	prior distribution, if more than one prior() is specified
e(priorparams)	parameter specification in prior()
e(priorparams#)	parameter specification from #th prior(), if more than one prior() is specified
e(parnames)	names of model parameters except exclude()
e(postvars)	variable names corresponding to model parameters in e(parnames)
e(subexpr)	substitutable expression
e(subexpr#)	substitutable expression, if more than one
e(wtype)	weight type (one equation)
e(wtype#)	weight type for #th equation
e(wexp)	weight expression (one equation)
e(wexp#)	weight expression for #th equation
e(block#_names)	parameter names from #th block
e(exclude)	names of excluded parameters
e(filename)	name of the file with simulation results
e(scparams)	scalar model parameters
e(matparams)	matrix model parameters
e(pareqmap)	model parameters in display order
e(title)	title in estimation output
e(rngstate)	random-number state at the time of simulation
e(search)	on, repeat(), or off

Matrices

e(mean)	posterior means
e(sd)	posterior standard deviations
e(mcse)	MCSE
e(median)	posterior medians
e(cri)	credible intervals
e(Cov)	variance–covariance matrix of parameters
e(ess)	effective sample sizes
e(init)	initial values vector

Functions

e(sample)	mark estimation sample

Methods and formulas

Methods and formulas are presented under the following headings:

Adaptive MH algorithm

The `bayesmh` command implements an adaptive random-walk Metropolis–Hastings algorithm with optional blocking of parameters. Providing an efficient MH procedure for simulating from a general posterior distribution is a difficult task, and various adaptive methods have been proposed (Haario, Saksman, and Tamminen 2001; Giordani and Kohn 2010; Roberts and Rosenthal 2009; Andrieu and Thoms 2008). The essence of the problem is in choosing an optimal proposal covariance matrix and a scale for parameter updates. Below we describe the implemented adaptation algorithm, assuming one block of parameters. In the presence of multiple blocks, the adaptation is applied to each block independently. The `adaptation()` option of `bayesmh` controls all the tuning parameters for the adaptation algorithm.

Let $\boldsymbol{\theta}$ be a vector of d scalar model parameters. Let T_0 be the length of a burn-in period (iterations that are discarded) as specified in `burnin()` and T be the size of the MCMC sample (iterations that are retained) as specified in `mcmcsize()`. The total number of MCMC iterations is then $T_{\text{total}} = T_0 + (T-1) \times$ `thinning()` $+ 1$. Also, let ALEN be the length of the adaptation interval (option `adaptation(every())`) and AMAX be the maximum number of adaptations (option `adaptation(maxiter())`).

The steps of the adaptive MH algorithm are the following. At $t = 0$, we initialize $\boldsymbol{\theta}_t = \boldsymbol{\theta}_0^f$, where $\boldsymbol{\theta}_0^f$ is the initial feasible state, and we set adaptation counter $k = 1$ and initialize $\rho_0 = 2.38/\sqrt{d}$, where d is the number of considered parameters. Σ_0 is the identity matrix. For $t = 1, \ldots, T_{\text{total}}$, do the following:

1. Generate proposal parameters: $\boldsymbol{\theta}_* = \boldsymbol{\theta}_{t-1} + \mathbf{e}$, $\mathbf{e} \sim N(\mathbf{0}, \rho_k^2 \Sigma_k)$, where ρ_k and Σ_k are current values of the proposal scale and covariance for adaptation iteration k.
2. Calculate the acceptance probability using
$$r = \min\left\{\frac{p(\boldsymbol{\theta}_*|\mathbf{y})}{p(\boldsymbol{\theta}_{t-1}|\mathbf{y})}, 1\right\}$$
where $p(\boldsymbol{\theta}|\mathbf{y}) = f(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})$ is the posterior distribution of $\boldsymbol{\theta}$ corresponding to the likelihood function $f(\mathbf{y}|\boldsymbol{\theta})$ and prior $p(\boldsymbol{\theta})$.
3. Draw $u \sim \text{Uniform}(0, 1)$ and set $\boldsymbol{\theta}_t = \boldsymbol{\theta}_*$ if $u < r$ or $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1}$, otherwise.
4. Perform adaptive iteration k. This step is performed only if $k \leq$ AMAX and t mod ALEN $= 0$. Update ρ_k according to (1), update Σ_k according to (2), and set $k = k + 1$.
5. Repeat steps 1–4. Note that the adaptation in step 4 is not performed at every MCMC iteration.

The output is the MCMC sequence $\{\boldsymbol{\theta}_t\}_{t=T_0+1}^{T_{\text{total}}}$ or $\boldsymbol{\theta}_1, \boldsymbol{\theta}_{1+l}, \boldsymbol{\theta}_{1+2l}, \ldots$, where l is the thinning interval as specified in the `thinning()` option.

If the parameter vector $\boldsymbol{\theta}$ is split into B blocks $\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \ldots, \boldsymbol{\theta}^B$, then steps 1 through 3 are repeated for each $\boldsymbol{\theta}^b$, $b = 1, \ldots, B$ sequentially. The adaptation in step 4 is then applied sequentially to each block $b = 1, 2, \ldots, B$. See *Blocking of parameters* in [BAYES] **intro** for details about blocking.

Initialization. We recommend that you carefully choose starting values for model parameters, $\boldsymbol{\theta}_0$, to be within the domain of the posterior distribution; see the `initial()` option. By default, MLEs are used as initial values, whenever available. If MLEs are not available, parameters with positive support are initialized with one, and the remaining parameters are initialized with zeros. Matrix parameters are initialized as identity matrices. If specified initial values $\boldsymbol{\theta}_0$ are within the domain of the posterior, then $\boldsymbol{\theta}_0^f = \boldsymbol{\theta}_0$. Otherwise, `bayesmh` performs 500 attempts (or as specified in `search(repeat())`) to find a feasible state $\boldsymbol{\theta}_0^f$, which is used as the initial state in the algorithm. If the command cannot find feasible values, it exits with an error.

Adaptation. The adaptation step is performed as follows. At each adaptive iteration k of the tth MCMC iteration, the proposal covariance Σ_k and scale ρ_k are tuned to achieve an optimal AR. Some asymptotic results (for example, Gelman, Gilks, and Roberts [1997]) show that the optimal AR, hereafter referred to as a TAR, for a single model parameter is 0.44 and is 0.234 for a block of multiple parameters.

Adaptation is performed periodically after a constant number of iterations as specified by the `adaptation(every())` option. At least `adaptation(miniter())` adaptive iterations are performed not to exceed `adaptation(maxiter())`. `bayesmh` does not perform adaptation if the absolute difference between the current AR and TAR is within the tolerance given by `adaptation(tolerance())`.

The `bayesmh` command allows you to control the calculation of AR through the `adaptation(alpha())` option with the default of 0.75, as follows,

$$\text{AR}_k = (1-\alpha)\text{AR}_{k-1} + \alpha\widehat{\text{AR}}_k,$$

where $\widehat{\text{AR}}_k$ is the expected acceptance probability, which is computed as the average of the acceptance probabilities, r, since the last adaptive iteration (for example, Andrieu and Thoms [2008]), and AR_0 is defined as described in the `adaptation(tarate())` option. Choosing $\alpha \in (0,1)$ allows for smoother change in the current AR between adaptive iterations.

The tuning of the proposal scale ρ is based on results in Gelman, Gilks, and Roberts (1997), Roberts and Rosenthal (2001), and Andrieu and Thoms (2008). The initial ρ_0 is set to $2.38/\sqrt{d}$, where d is the number of parameters in the considered block. Then, ρ_k is updated according to

$$\rho_k = \rho_{k-1} e^{\beta_k\left\{\Phi^{-1}(\text{AR}_k/2) - \Phi^{-1}(\text{TAR}/2)\right\}} \tag{1}$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function and β_k is defined below.

The adaptation of the covariance matrix is performed when multiple parameters are in the block and is based on Andrieu and Thoms (2008). You may specify an initial proposal covariance matrix Σ_0 in `covariance()` or use the identity matrix by default. Then, at time of adaptation k, the proposal covariance Σ_k is recomputed according to the formula

$$\Sigma_k = (1-\beta_k)\Sigma_{k-1} + \beta_k\widehat{\Sigma}_k, \quad \beta_k = \frac{\beta_0}{k^\gamma} \tag{2}$$

where $\widehat{\Sigma}_k = (\Theta_{t_k} - \boldsymbol{\mu}_{k-1})(\Theta_{t_k} - \boldsymbol{\mu}_{k-1})'/(t_k - t_{k-1})$ is the empirical covariance of the recent MCMC sample $\Theta_{t_k} = \{\boldsymbol{\theta}_s\}_{s=t_{k-1}}^{t_k}$ and t_{k-1} is the MCMC iteration corresponding to the adaptive iteration $k-1$ or 0 if adaptation did not take place. $\boldsymbol{\mu}_k$ is defined as

$$\boldsymbol{\mu}_k = \boldsymbol{\mu}_{k-1} + \beta_k(\overline{\Theta}_{t_k} - \boldsymbol{\mu}_{k-1}),\ k > 1$$

and $\boldsymbol{\mu}_1 = \overline{\Theta}_{t_k}$, where $\overline{\Theta}_{t_k}$ is the sample mean of Θ_{t_k}.

The constants $\beta_0 \in [0,1]$ and $\gamma \in [0,1]$ in (2) are specified in the options `adaptation(beta())` and `adaptation(gamma())`, respectively. The default values are 0.8 and 0, respectively. When $\gamma > 0$, we have a diminishing adaptation regime, which means that Σ_k is not changing much from one adaptive iteration to another. Random-walk Metropolis–Hastings algorithms with diminishing adaptation are shown to preserve the ergodicity of the Markov chain (Roberts and Rosenthal 2007; Andrieu and Moulines 2006; Atchadé and Rosenthal 2005).

The above algorithm is also used for discrete parameters, but discretization is used to obtain samples of discrete values. The default initial scale factor ρ_0 is set to $2.38/d$ for a block of d discrete parameters. The default TAR for discrete parameters with priors `bernoulli()` and `index()` is $\max\{0.1353, 1/n_{\text{maxbins}}\}$, where n_{maxbins} is the maximum number of discrete values a parameter can take among all the parameters specified in the same block. Blocks containing a mixture of continuous and discrete parameters are not allowed.

Gibbs sampling for some likelihood-prior and prior-hyperprior configurations

In some cases, when a block of parameters $\boldsymbol{\theta}^b$ has a conjugate prior, or more appropriately, a semiconjugate prior, with respect to the respective likelihood distribution for this block, you can request Gibbs sampling instead of random-walk MH sampling. Then, steps 1 through 4 of the algorithm described in *Adaptive MH algorithm* are replaced with just one step of Gibbs sampling as follows:

1'. Simulate proposal parameters: $\boldsymbol{\theta}^b_* \sim F_b(\boldsymbol{\theta}^b|\boldsymbol{\theta}^1_*, \ldots, \boldsymbol{\theta}^{b-1}_*, \boldsymbol{\theta}^{b+1}_*, \ldots, \boldsymbol{\theta}^B_*, \mathbf{y})$

Here $F_b(\cdot|\cdot)$ is the full conditional distribution of $\boldsymbol{\theta}^b$ with respect to the rest of the parameters.

Below we list the full conditional distributions for the likelihood-prior specifications for which `bayesmh` provides Gibbs sampling. All priors except Jeffreys priors are semiconjugate, meaning that full conditional distributions belong to the same family as the specified prior distributions for the chosen data model. This contrasts with a concept of conjugacy under which the posterior distribution of all parameters belongs to the same family as the joint prior distribution. All the combinations below assume prior independence; that is, all parameters are independent a priori. Thus their joint prior distribution is simply the product of the individual prior distributions.

Likelihood-prior configurations

Let $\mathbf{y} = (y_1, y_2, \ldots, y_n)'$ be a data sample of size n. For multivariate data, $Y = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)' = \{y_{ij}\}_{i,j=1}^{n,d}$ is an $n \times d$ data matrix.

1. **Normal–normal model**: θ^b is a mean of a normal distribution of y_is with a variance σ^2; mean and variance are independent a priori,

$$\begin{aligned} y_i|\theta^b, \sigma^2 &\sim N(\theta^b, \sigma^2),\ i = 1, 2, \ldots, n \\ \theta^b &\sim N(\mu_0, \tau_0^2) \\ \theta^b|\sigma^2, \mathbf{y} &\sim F_b = N(\mu_n, \tau_n^2) \end{aligned}$$

where μ_0 and τ_0^2 are hyperparameters (prior mean and prior variance) of a normal prior distribution for θ^b and

$$\begin{aligned} \mu_n &= \left(\mu_0\tau_0^{-2} + \sum y_i\sigma^{-2}\right)\tau_n^2 \\ \tau_n^2 &= (\tau_0^{-2} + n\sigma^{-2})^{-1} \end{aligned}$$

2. **Normal–normal regression**: $\boldsymbol{\theta}^b$ is a $p_1 \times 1$ subvector of a $p \times 1$ vector of regression coefficients $\boldsymbol{\beta}$ from a normal linear regression model for $\mathbf{y}$ with an $n \times p$ design matrix $X = (\mathbf{x}_1', \mathbf{x}_2', \ldots, \mathbf{x}_n')'$ and with a variance σ^2; regression coefficients and variance are independent a priori,

$$\begin{aligned} y_i|\boldsymbol{\theta}^b, \sigma^2 &\sim N(\mathbf{x}_i'\boldsymbol{\beta}, \sigma^2),\ i = 1, 2, \ldots, n \\ \theta^b_k &\sim \text{i.i.d. } N(\beta_0, \tau_0^2),\ k = 1, 2, \ldots, p_1 \\ \boldsymbol{\theta}^b|\sigma^2, \mathbf{y} &\sim F_b = \text{MVN}(\boldsymbol{\mu}_n, \Lambda_n) \end{aligned}$$

where β_0 and τ_0^2 are hyperparameters (prior regression coefficient and prior variance) of normal prior distributions for θ^b_k and

$$\begin{aligned} \boldsymbol{\mu}_n &= (\boldsymbol{\beta}_0\tau_0^{-2} + X_b'\mathbf{y}\sigma^{-2})\Lambda_n \\ \Lambda_n &= (\tau_0^{-2}I_{p_1} + \sigma^{-2}X_b'X_b)^{-1} \end{aligned}$$

In the above, I_{p_1} is a $p_1 \times p_1$ identity matrix, and $X_b = (\mathbf{x}'_{1b}, \mathbf{x}'_{2b}, \ldots, \mathbf{x}'_{nb})'$ is an $n \times p_1$ submatrix of X corresponding to the regression coefficients $\boldsymbol{\theta}^b$.

3. **Normal–inverse-gamma model**: θ^b is a variance of a normal distribution of y_is with a mean μ; mean and variance are independent a priori,

$$\begin{aligned} y_i|\mu,\theta^b &\sim N(\mu,\theta^b),\ i=1,2,\ldots,n \\ \theta^b &\sim \text{InvGamma}(\alpha,\beta) \\ \theta^b|\mu,\mathbf{y} &\sim F_b = \text{InvGamma}(\alpha+n/2, \beta+\sum_{i=1}^{n}(y_i-\mu)^2/2) \end{aligned}$$

where α and β are hyperparameters (prior shape and prior scale) of an inverse-gamma prior distribution for θ^b.

4. **Multivariate-normal–multivariate-normal model**: $\boldsymbol{\theta}^b$ is a mean vector of a multivariate normal distribution of $\mathbf{y}$s with a $d \times d$ covariance matrix Σ; mean and covariance are independent a priori,

$$\begin{aligned} \mathbf{y}_i|\boldsymbol{\theta}^b,\Sigma &\sim \text{MVN}(\boldsymbol{\theta}^b,\Sigma),\ i=1,2,\ldots,n \\ \boldsymbol{\theta}^b &\sim \text{MVN}(\boldsymbol{\mu}_0,\Lambda_0) \\ \boldsymbol{\theta}^b|\Sigma,Y &\sim F_b = \text{MVN}(\boldsymbol{\mu}_n,\Lambda_n) \end{aligned}$$

where $\boldsymbol{\mu}_0$ and Λ_0 are hyperparameters (prior mean vector and prior covariance) of a multivariate normal prior distribution for $\boldsymbol{\theta}^b$ and

$$\begin{aligned} \boldsymbol{\mu}_n &= \Lambda_n\Lambda_0^{-1}\boldsymbol{\mu}_0 + \Lambda_n\Sigma^{-1}\left(\sum_{i=1}^{n}\mathbf{y}_i\right) \\ \Lambda_n &= (\Lambda_0^{-1}+n\Sigma^{-1})^{-1} \end{aligned}$$

5. **Multivariate-normal–inverse-Wishart model**: Θ^b is a $d \times d$ covariance matrix of a multivariate normal distribution of $\mathbf{y}$s with a mean vector $\boldsymbol{\mu}$; mean and covariance are independent a priori,

$$\begin{aligned} \mathbf{y}_i|\boldsymbol{\mu},\Theta^b &\sim \text{MVN}(\boldsymbol{\mu},\Theta^b),\ i=1,2,\ldots,n \\ \Theta^b &\sim \text{InvWishart}(\nu,\Psi) \\ \Theta^b|\boldsymbol{\mu},Y &\sim F_b = \text{InvWishart}(n+\nu, \Psi+\sum_{i=1}^{n}(\mathbf{y}_i-\boldsymbol{\mu})(\mathbf{y}_i-\boldsymbol{\mu})') \end{aligned}$$

where ν and Ψ are hyperparameters (prior degrees of freedom and prior scale matrix) of an inverse-Wishart prior distribution for Θ^b.

6. **Multivariate-normal–Jeffreys model**: Θ^b is a $d \times d$ covariance matrix of a multivariate normal distribution of $\mathbf{y}$s with a mean vector $\boldsymbol{\mu}$; mean and covariance are independent a priori,

$$\begin{aligned} \mathbf{y}_i|\boldsymbol{\mu},\Theta^b &\sim \text{MVN}(\boldsymbol{\mu},\Theta^b),\ i=1,2,\ldots,n \\ \Theta^b &\sim |\Theta^b|^{-\frac{d+1}{2}}\ \text{(multivariate Jeffreys)} \\ \Theta^b|\boldsymbol{\mu},Y &\sim F_b = \text{InvWishart}(n-1, \sum_{i=1}^{n}(\mathbf{y}_i-\boldsymbol{\mu})(\mathbf{y}_i-\boldsymbol{\mu})') \end{aligned}$$

where $|\cdot|$ denotes the determinant of a matrix.

Prior-hyperprior configurations

Suppose that a prior distribution of a parameter of interest $\boldsymbol{\theta}$ has hyperparameters $\boldsymbol{\theta}_h$ for which a prior distribution is specified. We refer to the former prior distribution as a hyperprior. You can also request Gibbs sampling for the following prior-hyperprior combinations.

We use θ_h^b and $\boldsymbol{\theta}_h^b$ to refer to the hyperparameters from the block b.

1. **Normal–normal model**: θ_h^b is a mean of a normal prior distribution of θ with a variance σ_h^2; mean and variance are independent a priori,

$$\begin{aligned}\theta|\theta_h^b, \sigma_h^2 &\sim N(\theta_h^b, \sigma_h^2)\\ \theta_h^b &\sim N(\mu_0, \tau_0^2)\\ \theta_h^b|\sigma_h^2, \theta &\sim F_b = N(\mu_1, \tau_1^2)\end{aligned}$$

   where μ_0 and τ_0^2 are the prior mean and prior variance of a normal hyperprior distribution for θ_h^b and

$$\begin{aligned}\mu_1 &= \left(\mu_0\tau_0^{-2} + \theta\sigma_h^{-2}\right)\tau_1^2\\ \tau_1^2 &= (\tau_0^{-2} + \sigma_h^{-2})^{-1}\end{aligned}$$

2. **Normal–inverse-gamma model**: θ_h^b is a variance of a normal prior distribution of θ with a mean μ_h; mean and variance are independent a priori,

$$\begin{aligned}\theta|\mu_h, \theta_h^b &\sim N(\mu_h, \theta_h^b)\\ \theta_h^b &\sim \text{InvGamma}(\alpha, \beta)\\ \theta_h^b|\mu_h, \theta &\sim F_b = \text{InvGamma}(\alpha + 0.5, \beta + (\theta - \mu)^2/2)\end{aligned}$$

   where α and β are the prior shape and prior scale, respectively, of an inverse-gamma hyperprior distribution for θ_h^b.

3. **Bernoulli–beta model**: θ_h^b is a probability of success of a Bernoulli prior distribution of θ,

$$\begin{aligned}\theta|\theta_h^b &\sim \text{Bernoulli}(\theta_h^b)\\ \theta_h^b &\sim \text{Beta}(\alpha, \beta)\\ \theta_h^b|\theta &\sim F_b = \text{Beta}(\alpha + \theta, \beta + 1 - \theta)\end{aligned}$$

   where α and β are the prior shape and prior scale, respectively, of a beta hyperprior distribution for θ_h^b.

4. **Poisson–gamma model**: θ_h^b is a mean of a Poisson prior distribution of θ,

$$\begin{aligned}\theta|\theta_h^b &\sim \text{Poisson}(\theta_h^b)\\ \theta_h^b &\sim \text{Gamma}(\alpha, \beta)\\ \theta_h^b|\theta &\sim F_b = \text{Gamma}(\alpha + \theta, \beta/(\beta + 1))\end{aligned}$$

   where α and β are the prior shape and prior scale, respectively, of a gamma hyperprior distribution for θ_h^b.

5. **Multivariate-normal–multivariate-normal model**: $\boldsymbol{\theta}_h^b$ is a mean vector of a multivariate normal prior distribution of $\boldsymbol{\theta}$ with a $d \times d$ covariance matrix Σ_h; mean and covariance are independent a priori,

$$\begin{aligned}\boldsymbol{\theta}|\boldsymbol{\theta}_h^b, \Sigma_h &\sim \text{MVN}(\boldsymbol{\theta}_h^b, \Sigma_h)\\ \boldsymbol{\theta}_h^b &\sim \text{MVN}(\boldsymbol{\mu}_0, \Lambda_0)\\ \boldsymbol{\theta}_h^b|\Sigma_h, \boldsymbol{\theta} &\sim F_b = \text{MVN}(\boldsymbol{\mu}_1, \Lambda_1)\end{aligned}$$

where $\boldsymbol{\mu}_0$ and Λ_0 are the prior mean vector and prior covariance of a multivariate normal hyperprior distribution for $\boldsymbol{\theta}_h^b$ and

$$\begin{aligned}\boldsymbol{\mu}_1 &= \Lambda_1\Lambda_0^{-1}\boldsymbol{\mu}_0 + \Lambda_1\Sigma_h^{-1}\boldsymbol{\theta}\\ \Lambda_1 &= (\Lambda_0^{-1} + \Sigma_h^{-1})^{-1}\end{aligned}$$

6. **Multivariate-normal–inverse-Wishart model**: Θ_h^b is a $d \times d$ covariance matrix of a multivariate normal prior distribution of $\boldsymbol{\theta}$ with a mean vector $\boldsymbol{\mu}_h$; mean and covariance are independent a priori,

$$\begin{aligned}\boldsymbol{\theta}|\boldsymbol{\mu}_h, \Theta_h^b &\sim \text{MVN}(\boldsymbol{\mu}_h, \Theta_h^b)\\ \Theta_h^b &\sim \text{InvWishart}(\nu, \Psi)\\ \Theta_h^b|\boldsymbol{\mu}_h, \boldsymbol{\theta} &\sim F_b = \text{InvWishart}(\nu+1, \Psi + (\boldsymbol{\theta}-\boldsymbol{\mu}_h)(\boldsymbol{\theta}-\boldsymbol{\mu}_h)')\end{aligned}$$

where ν and Ψ are the prior degrees of freedom and prior scale matrix of an inverse-Wishart hyperprior distribution for Θ_h^b.

Marginal likelihood

The marginal likelihood is defined as

$$m(\mathbf{y}) = \int p(\mathbf{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}$$

where $p(\mathbf{y}|\boldsymbol{\theta})$ is the probability density of data $\mathbf{y}$ given $\boldsymbol{\theta}$ and $\pi(\boldsymbol{\theta})$ is the density of the prior distribution for $\boldsymbol{\theta}$.

Marginal likelihood $m(\mathbf{y})$, being the denominator term in the posterior distribution, has a major role in Bayesian analysis. It is sometimes referred to as "model evidence", and it is used as a goodness-of-fit criterion. For example, marginal likelihoods are used in calculating Bayes factors for the purpose of model comparison; see *Methods and formulas* in [BAYES] **bayesstats ic**.

The simplest approximation to $m(\mathbf{y})$ is provided by the Monte Carlo integration,

$$\widehat{m}_p = \frac{1}{M}\sum_{s=1}^{M} p(\mathbf{y}|\boldsymbol{\theta}_s)$$

where $\{\boldsymbol{\theta}_s\}_{s=1}^M$ is an independent sample from the prior distribution $\pi(\boldsymbol{\theta})$. This estimation is very inefficient, however, because of the high variance of the likelihood function. MCMC samples are not independent and cannot be used directly for calculating $\widehat{m}_p$.

An improved estimation of the marginal likelihood can be obtained by using importance sampling. For a sample $\{\boldsymbol{\theta}_t\}_{t=1}^{T}$, not necessarily independent, from the posterior distribution, the harmonic mean of the likelihood values,

$$\widehat{m}_h = \left\{ \frac{1}{T} \sum_{t=1}^{T} p(\mathbf{y}|\boldsymbol{\theta}_t)^{-1} \right\}^{-1}$$

approximates $m(\mathbf{y})$ (Geweke 1989).

Another method for estimating $m(\mathbf{y})$ uses the Laplace approximation,

$$\widehat{m}_l = (2\pi)^{p/2} |-\widetilde{H}|^{-1/2} p(\mathbf{y}|\widetilde{\boldsymbol{\theta}})\pi(\widetilde{\boldsymbol{\theta}})$$

where p is the number of parameters (or dimension of $\boldsymbol{\theta}$), $\widetilde{\boldsymbol{\theta}}$ is the posterior mode, and $\widetilde{H}$ is the Hessian matrix of $l(\boldsymbol{\theta}) = p(\mathbf{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$ calculated at the mode $\widetilde{\boldsymbol{\theta}}$.

Using the fact that the posterior sample covariance matrix, which we denote as $\widehat{\Sigma}$, is asymptotically equal to $(-\widetilde{H})^{-1}$, Raftery (1996) proposed what he called the Laplace–Metropolis estimator (implemented by `bayesmh`):

$$\widehat{m}_{lm} = (2\pi)^{p/2} |\widehat{\Sigma}|^{1/2} p(\mathbf{y}|\widetilde{\boldsymbol{\theta}})\pi(\widetilde{\boldsymbol{\theta}})$$

Raftery (1996) recommends that a robust and consistent estimator be used for the posterior covariance matrix.

Nicholas Constantine Metropolis (1915–1999) was born in Chicago, where he received BSc and PhD degrees in physics at the University of Chicago. He oscillated through his career between posts there and at what later became the Los Alamos National Laboratory in New Mexico. Metropolis is best known for his contributions to Monte Carlo methods, algorithms based on repeated random sampling. He was the first author on an outstanding paper about a Monte Carlo algorithm (Metropolis et al. 1953), with Arianna W. Rosenbluth, Marshall N. Rosenbluth (1927–2003), Augusta H. Teller (1909–2000), and Edward Teller (1908–2003). However, the relative contributions of these authors have been much disputed, and general and specific credit for the method should also be given to others, including John von Neumann (1903–1957), Stanisław M. Ulam (1909–1984), and Enrico Fermi (1901–1954). According to Google Scholar, Metropolis et al. (1953) has been cited over 28,000 times.

W. Keith Hastings (1930–) was born in Toronto, Ontario. He received BA, MA, and PhD degrees in applied mathematics and statistics from the University of Toronto; his doctoral thesis was on invariant fiducial distributions. Hastings worked as a consultant in computer applications for a Toronto firm, at the University of Canterbury in New Zealand, and at Bell Labs in New Jersey before returning from 1966 to 1971 to his alma mater. In this period, he wrote a famous paper (Hastings 1970) generalizing the work of Metropolis et al. (1953) to produce what is now often called the Metropolis–Hastings algorithm. It is the most common Markov chain Monte Carlo method, widely used throughout statistical science to sample from high-dimensional distributions. According to Google Scholar, Hastings (1970) has been cited over 8,000 times. Hastings worked at the University of Victoria in British Columbia from 1971 to 1992, when he retired.

Harold Jeffreys (1891–1989) was born near Durham, England, and spent more than 75 years studying and working at the University of Cambridge, principally on theoretical and observational problems in geophysics, astronomy, mathematics, and statistics. He developed a systematic Bayesian approach to inference in his monograph *Theory of Probability*.

References

Andrieu, C., and É. Moulines. 2006. On the ergodicity properties of some adaptive MCMC algorithms. *Annals of Applied Probability* 16: 1462–1505.

Andrieu, C., and J. Thoms. 2008. A tutorial on adaptive MCMC. *Statistics and Computing* 18: 343–373.

Atchadé, Y. F., and J. S. Rosenthal. 2005. On adaptive Markov chain Monte Carlo algorithms. *Bernoulli* 11: 815–828.

Carlin, B. P., A. E. Gelfand, and A. F. M. Smith. 1992. Hierarchical Bayesian analysis of changepoint problems. *Journal of the Royal Statistical Society, Series C* 41: 389–405.

Carlin, J. B. 1992. Meta-analysis for 2×2 tables: A Bayesian approach. *Statistics in Medicine* 11: 141–158.

Diggle, P. J., P. J. Heagerty, K.-Y. Liang, and S. L. Zeger. 2002. *Analysis of Longitudinal Data*. 2nd ed. Oxford: Oxford University Press.

Gelfand, A. E., S. E. Hills, A. Racine-Poon, and A. F. M. Smith. 1990. Illustration of Bayesian inference in normal data models using Gibbs sampling. *Journal of the American Statistical Association* 85: 972–985.

Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. 2014. *Bayesian Data Analysis*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.

Gelman, A., W. R. Gilks, and G. O. Roberts. 1997. Weak convergence and optimal scaling of random walk Metropolis algorithms. *Annals of Applied Probability* 7: 110–120.

Geweke, J. 1989. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica* 57: 1317–1339.

Geyer, C. J. 2011. Introduction to Markov chain Monte Carlo. In *Handbook of Markov Chain Monte Carlo*, ed. S. Brooks, A. Gelman, G. L. Jones, and X.-L. Meng, 3–48. Boca Raton, FL: Chapman & Hall/CRC.

Giordani, P., and R. J. Kohn. 2010. Adaptive independent Metropolis–Hastings by fast estimation of mixtures of normals. *Journal of Computational and Graphical Statistics* 19: 243–259.

Haario, H., E. Saksman, and J. Tamminen. 2001. An adaptive Metropolis algorithm. *Bernoulli* 7: 223–242.

Hastings, W. K. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 57: 97–109.

Hoff, P. D. 2009. *A First Course in Bayesian Statistical Methods*. New York: Springer.

Jarrett, R. G. 1979. A note on the intervals between coal-mining disasters. *Biometrika* 66: 191–193.

Jeffreys, H. 1946. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London, Series A* 186: 453–461.

Lichman, M. 2013. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml.

Maas, B., W. R. Garnett, I. M. Pellock, and T. J. Comstock. 1987. A comparative bioavailability study of Carbamazepine tablets and chewable formulation. *Therapeutic Drug Monitoring* 9: 28–33.

Maguire, B. A., E. S. Pearson, and A. H. A. Wynn. 1952. The time intervals between industrial accidents. *Biometrika* 39: 168–180.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics* 21: 1087–1092.

Raftery, A. E. 1996. Hypothesis testing and model selection. In *Markov Chain Monte Carlo in Practice*, ed. W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, 163–187. Boca Raton, FL: Chapman and Hall.

Raftery, A. E., and V. E. Akman. 1986. Bayesian analysis of a Poisson process with a change-point. *Biometrika* 73: 85–89.

Roberts, G. O., and J. S. Rosenthal. 2001. Optimal scaling for various Metropolis–Hastings algorithms. *Statistical Science* 16: 351–367.

——. 2007. Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. *Journal of Applied Probability* 44: 458–475.

——. 2009. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics* 18: 349–367.

Ruppert, D., M. P. Wand, and R. J. Carroll. 2003. *Semiparametric Regression*. Cambridge: Cambridge University Press.

Thomas, A., B. O'Hara, U. Ligges, and S. Sturtz. 2006. Making BUGS Open. *R News* 6: 12–17.

Thompson, J. 2014. *Bayesian Analysis with Stata*. College Station, TX: Stata Press.

Yusuf, S., R. Simon, and S. S. Ellenberg. 1987. Proceedings of the workshop on methodological issues in overviews of randomized clinical trials, May 1986. In *Statistics in Medicine*, vol. 6.

Zellner, A. 1986. On assessing prior distributions and Bayesian regression analysis with g-prior distributions. In Vol. 6 of *Bayesian Inference and Decision Techniques: Essays in Honor of Bruno De Finetti (Studies in Bayesian Econometrics and Statistics)*, ed. P. K. Goel and A. Zellner, 233–343. Amsterdam: North-Holland.

Zellner, A., and N. S. Revankar. 1969. Generalized production functions. *Review of Economic Studies* 36: 241–250.

Also see

[BAYES] **bayesmh postestimation** — Postestimation tools for bayesmh

[BAYES] **bayesmh evaluators** — User-defined evaluators with bayesmh

[BAYES] **bayes** — Introduction to commands for Bayesian analysis

[BAYES] **intro** — Introduction to Bayesian analysis

[BAYES] **Glossary**

Title

bayesmh evaluators — User-defined evaluators with bayesmh

Description

`bayesmh` provides two options, `evaluator()` and `llevaluator()`, that facilitate user-defined evaluators for fitting general Bayesian regression models. `bayesmh, evaluator()` accommodates log-posterior evaluators. `bayesmh, llevaluator()` accommodates log-likelihood evaluators, which are combined with built-in prior distributions to form the desired posterior density. For a catalog of built-in likelihood models and prior distributions, see [BAYES] **bayesmh**.

Syntax

Single-equation models

User-defined log-posterior evaluator

`bayesmh` *depvar* [*indepvars*] [*if*] [*in*] [*weight*], `evaluator(`*evalspec*`)` [*options*]

User-defined log-likelihood evaluator

`bayesmh` *depvar* [*indepvars*] [*if*] [*in*] [*weight*], `llevaluator(`*evalspec*`)`
`prior(`*priorspec*`)` [*options*]

Multiple-equations models

User-defined log-posterior evaluator

`bayesmh (`*eqspecp*`)` [`(`*eqspecp*`)` [...]] [*if*] [*in*] [*weight*], `evaluator(`*evalspec*`)`
[*options*]

User-defined log-likelihood evaluator

`bayesmh (`*eqspecll*`)` [`(`*eqspecll*`)` [...]] [*if*] [*in*] [*weight*], `prior(`*priorspec*`)`
[*options*]

The syntax of *eqspecp* is

varspec [, noconstant]

The syntax of *eqspecll* for built-in likelihood models is

varspec, likelihood(*modelspec*) [noconstant]

The syntax of *eqspecll* for user-defined log-likelihood evaluators is

varspec, llevaluator(*evalspec*) [noconstant]

The syntax of *varspec* is one of the following:

for single outcome

[*eqname*:]*depvar* [*indepvars*]

for multiple outcomes with common regressors

depvars = [*indepvars*]

for multiple outcomes with outcome-specific regressors

([*eqname1*:]*depvar1* [*indepvars1*]) ([*eqname2*:]*depvar2* [*indepvars2*]) [...]

The syntax of *evalspec* is

progname, parameters(*paramlist*) [extravars(*varlist*) passthruopts(*string*)]

where *progname* is the name of a Stata program that you write to evaluate the log-posterior density or the log-likelihood function (see *Program evaluators*), and *paramlist* is a list of model parameters:

paramdef [*paramdef* [...]]

The syntax of *paramdef* is

{[*eqname*:]*param* [*param* [...]] [, matrix]}

where the parameter label *eqname* and parameter names *param* are valid Stata names. Model parameters are either scalars such as {var}, {mean}, and {shape:alpha} or matrices such as {Sigma, matrix} and {Scale:V, matrix}. For scalar parameters, you can use {param=#} in the above to specify an initial value. For example, you can specify {var=1}, {mean=1.267}, or {shape:alpha=3}. You can specify the multiple parameters with same equation as {eq:p1 p2 p3} or {eq: S1 S2, matrix}. Also see *Declaring model parameters* in [BAYES] **bayesmh**.

options	Description
*`evaluator(`*evalspec*`)`	specify log-posterior evaluator; may not be combined with `llevaluator()` and `prior()`
*`llevaluator(`*evalspec*`)`	specify log-likelihood evaluator; requires `prior()` and may not be combined with `evaluator()`
*`prior(`*priorspec*`)`	prior for model parameters; required with log-likelihood evaluator and may be repeated
`likelihood(`*modelspec*`)`	distribution for the likelihood model; allowed within an equation of a multiple-equations model only
`noconstant`	suppress constant term; not allowed with ordered models specified in `likelihood()` with multiple-equations models
bayesmhopts	any options of [BAYES] **bayesmh** except `likelihood()` and `prior()`

*Option `evaluator()` is required for log-posterior evaluators, and options `llevaluator()` and `prior()` are required for log-likelihood evaluators. With log-likelihood evaluators, `prior()` must be specified for all model parameters and can be repeated.

indepvars may contain factor variables; see [U] **11.4.3 Factor variables**.

Only `fweights` are allowed; see [U] **11.1.6 weight**.

Options

`evaluator(`*evalspec*`)` specifies the name and the attributes of the log-posterior evaluator; see *Program evaluators* for details. This option may not be combined with `llevaluator()` or `likelihood()`.

`llevaluator(`*evalspec*`)` specifies the name and the attributes of the log-likelihood evaluator; see *Program evaluators* for details. This option may not be combined with `evaluator()` or `likelihood()` and requires the `prior()` option.

`prior(`*priorspec*`)`; see [BAYES] **bayesmh**.

`likelihood(`*modelspec*`)`; see [BAYES] **bayesmh**. This option is allowed within an equation of a multiple-equations model only.

`noconstant`; see [BAYES] **bayesmh**.

bayesmhopts specify any *options* of [BAYES] **bayesmh**, except `likelihood()` and `prior()`.

Remarks and examples

Remarks are presented under the following headings:

Program evaluators

If your likelihood model or prior distributions are particularly complex and cannot be represented by one of the predefined sets of distributions or by substitutable expressions provided with bayesmh, you can program these functions by writing your own evaluator program.

Evaluator programs can be used for programming the full posterior density by specifying the evaluator() option or only the likelihood portion of your Bayesian model by specifying the llevaluator() option. For likelihood evaluators, prior() option(s) must be specified for all model parameters. Your program is expected to calculate and return an overall log-posterior or a log-likelihood density value.

It is allowed for the return values to match the log density up to an additive constant, in which case, however, some of the reported statistics such as DIC and log marginal-likelihood may not be applicable.

Your program evaluator *progname* must be a Stata program; see [U] 18 Programming Stata. The program must follow the style below.

```
program progname
    args lnden xb1 [xb2 ...] [modelparams]
    ... computations ...
    scalar 'lnden' = ...
end
```

Here lnden contains the name of a temporary scalar to be filled in with an overall log-posterior or log-likelihood value;

xb# contains the name of a temporary variable containing the linear predictor from the #th equation; and

modelparams is a list of names of scalars or matrices to contain the values of model parameters specified in suboption parameters() of evaluator() or llevaluator(). For matrix parameters, the specified names will contain the names of temporary matrices containing current values. For scalar parameters, these are the names of temporary scalars containing current values. The order in which names are listed should correspond to the order in which model parameters are specified in parameters().

Also see *Global macros* for a list of global macros available to the program evaluator.

After you write a program evaluator, you specify its name in the option evaluator() for log-posterior evaluators,

```
. bayesmh ..., evaluator(progname, evalopts)
```

or option llevaluator() for log-likelihood evaluators,

```
. bayesmh ..., llevaluator(progname, evalopts)
```

Evaluator options *evalopts* include parameters(), extravars(), and passthruopts().

parameters(*paramlist*) specifies model parameters. Model parameters can be scalars or matrices. Each parameter must be specified in curly braces {}. Multiple parameters with the same equation names may be specified within one set of {}.

For example,

```
parameters({mu} {var:sig2} {S,matrix} {cov:Sigma, matrix} {prob:p1 p2})
```

specifies a scalar parameter with name mu without an equation label, a scalar parameter with name sig2 and label var, a matrix parameter with name S, a matrix parameter with name Sigma and label cov, and two scalar parameters {prob:p1} and {prob:p2}.

`extravars(`*varlist*`)` specifies any variables in addition to dependent and independent variables that you may need in your program evaluator. Examples of such variables are offset variables, exposure variables for count-data models, and failure or censoring indicators for survival-time models. See *Cox proportional hazards regression* for an example.

`passthruopts(`*string*`)` specifies a list of options you may want to pass to your program evaluator. For example, these options may contain fixed values of model parameters and hyperparameters. See *Multivariate normal regression model* for an example.

`bayesmh` automatically creates parameters for regression coefficients: {*depname*:*varname*} for every *varname* in *indepvars*, and a constant parameter {*depname*:_cons} unless `noconstant` is specified. These parameters are used to form linear predictors used by the program evaluator. If you need to access values of the parameters in the evaluator, you can use `$MH_b`; see the log-posterior evaluator in *Cox proportional hazards regression* for an example. With multiple dependent variables, regression coefficients are defined for each dependent variable.

Simple linear regression model

Suppose that we want to fit a Bayesian normal regression where we program the posterior distribution ourselves. The `normaljeffreys` program below computes the log-posterior density for the normal linear regression with flat priors for the coefficients and the Jeffreys prior for the variance parameter.

```
. program normaljeffreys
  1.         version 14
  2.         args lnp xb var
  3.         /* compute log likelihood */
.         tempname sd
  4.         scalar `sd' = sqrt(`var')
  5.         tempvar lnfj
  6.         quietly generate double `lnfj'=lnnormalden($MH_y,`xb',`sd') if $MH
> _touse
  7.         quietly summarize `lnfj', meanonly
  8.         if r(N) < $MH_n {
  9.                 scalar `lnp' = .
 10.                 exit
 11.         }
 12.         tempname lnf
 13.         scalar `lnf' = r(sum)
 14.         /* compute log prior */
.         tempname lnprior
 15.         scalar `lnprior' = -2*ln(`sd')
 16.         /* compute log posterior */
.         scalar `lnp' = `lnf' + `lnprior'
 17. end
```

The program accepts three parameters: a temporary name `` `lnp' `` of a scalar to contain the log-posterior value, a temporary name `` `xb' `` of the variable that contains the linear predictor, and a temporary name `` `var' `` of a scalar that contains the values of the variance parameter.

The first part of the program calculates the overall log likelihood of the normal regression. The second part of the program calculates the log of prior distributions of the parameters. Because the coefficients have flat prior distributions with densities of 1, their log is 0 and does not contribute to the overall prior. The only contribution is from the Jeffreys prior $\ln(1/\sigma^2) = -2\ln(\sigma)$ for the variance σ^2. The third and final part of the program computes the values of the posterior density as the sum of the overall log likelihood and the log of the prior.

The substantial portion of this program is the computation of the overall log likelihood. The global macro `$MH_y` contains the name of the dependent variable, `$MH_touse` contains a temporary marker

variable identifying observations to be used in computations, and $MH_n contains the total number of observations in the sample identified by the $MH_touse variable.

We used the built-in function lnnormalden() to compute observation-specific log likelihood and used summarize to obtain the overall value. Whenever a temporary variable is needed for calculations, such as 'lnfj' in our program, it is important to create it of type double to ensure the highest precision of the results. It is also important to perform computations using only the relevant subset of observations as identified by the marker variable stored in $MH_touse. This variable contains the value of 1 for observations to be used in the computations and 0 for the remaining observations. Missing values in used variables, if, and in affect this variable. After we compute the log-likelihood value, we should verify that the number of nonmissing observation-specific contributions to the log likelihood equals $MH_n. If it does not, the log-posterior value (or log-likelihood value in a log-likelihood evaluator) must be set to missing.

We can now specify the normaljeffreys evaluator in the evaluator() option of bayesmh. In addition to the regression coefficients, we have one extra parameter, the variance of the normal distribution, which we must specify in the parameters() suboption of evaluator().

We use auto.dta to illustrate the command. We specify a simple regression of mpg on rescaled weight.

```
. use http://www.stata-press.com/data/r14/auto
(1978 Automobile Data)
. quietly replace weight = weight/100
. set seed 14
. bayesmh mpg weight, evaluator(normaljeffreys, parameters({var}))
Burn-in ...
note: invalid initial state
Simulation ...
Model summary
------------------------------------------------------------------------------
Posterior:
  mpg ~ normaljeffreys(xb_mpg,{var})
------------------------------------------------------------------------------

Bayesian regression                             MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .1433
                                                Efficiency:  min =     .06246
                                                             avg =     .06669
Log marginal likelihood =   -198.247                         max =     .07091

------------------------------------------------------------------------------
             |                                                 Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
      weight |  -.6052218    .053604   .002075  -.6062666  -.7121237  -.4992178
       _cons |   39.56782   1.658124   .066344   39.54211   36.35645   42.89876
-------------+----------------------------------------------------------------
         var |   12.19046   2.008871   .075442   12.03002   8.831172   17.07787
------------------------------------------------------------------------------
```

The output of bayesmh with user-written evaluators is the same as the output of bayesmh with built-in distributions, except the title and the model summary. The generic title Bayesian regression is used for all evaluators, but you can change it by specifying the title() option. The model summary provides the name of the posterior evaluator.

Following the command line, there is a note about invalid initial state. For program evaluators, `bayesmh` initializes all parameters with zeros, except for positive parameters used in prior specifications, which are initialized with ones. This may not be sensible for all parameters, such as the variance parameter in our example. We may consider using, for example, OLS estimates as initial values of the parameters.

```
. regress mpg weight
```

Source	SS	df	MS	Number of obs	=	74
				F(1, 72)	=	134.62
Model	1591.99021	1	1591.99021	Prob > F	=	0.0000
Residual	851.469254	72	11.8259619	R-squared	=	0.6515
				Adj R-squared	=	0.6467
Total	2443.45946	73	33.4720474	Root MSE	=	3.4389

mpg	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
weight	-.6008687	.0517878	-11.60	0.000	-.7041058	-.4976315
_cons	39.44028	1.614003	24.44	0.000	36.22283	42.65774

```
. display e(rmse)^2
11.825962
```

We specify initial values in the `initial()` option.

```
. set seed 14
. bayesmh mpg weight, evaluator(normaljeffreys, parameters({var}))
> initial({mpg:weight} -0.6 {mpg:_cons} 39 {var} 11.83)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Posterior:
  mpg ~ normaljeffreys(xb_mpg,{var})
------------------------------------------------------------------------------
Bayesian regression                             MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .1668
                                                Efficiency:  min =     .04114
                                                             avg =     .04811
Log marginal likelihood = -198.14302                         max =     .05938
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
mpg						
weight	-.6025616	.0540995	.002667	-.6038729	-.7115221	-.5005915
_cons	39.50491	1.677906	.080156	39.45537	36.2433	43.14319
var	12.26586	2.117858	.086915	12.05298	8.827655	17.10703

We can compare our results with `bayesmh` that uses a built-in normal likelihood and flat and Jeffreys priors. To match the results, we must use the same initial values, because `bayesmh` has a different initialization logic for built-in distributions.

```
. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, flat) prior({var}, jeffreys)
> initial({mpg:weight} -0.6 {mpg:_cons} 39 {var} 11.83)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:weight _cons} ~ 1 (flat)                                          (1)
               {var} ~ jeffreys
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_mpg.
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .1668
                                                Efficiency:  min =      .04114
                                                             avg =      .04811
Log marginal likelihood = -198.14302                         max =      .05938
------------------------------------------------------------------------------
             |                                                Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
      weight |  -.6025616   .0540995   .002667  -.6038729  -.7115221  -.5005915
       _cons |   39.50491   1.677906   .080156   39.45537   36.2433   43.14319
-------------+----------------------------------------------------------------
         var |   12.26586   2.117858   .086915   12.05298   8.827655   17.10703
------------------------------------------------------------------------------
```

If your Bayesian model uses prior distributions that are supported by `bayesmh` but the likelihood model is not supported, you can write only the likelihood evaluator and use built-in prior distributions.

For example, we extract the portion of the `normaljeffreys` program computing the overall log likelihood into a separate program and call it `normalreg`.

```
. program normalreg
  1.         version 14
  2.         args lnf xb var
  3.                                              /* compute log likelihood */
.         tempname sd
  4.         scalar `sd' = sqrt(`var')
  5.         tempvar lnfj
  6.         quietly generate double `lnfj' = lnnormalden($MH_y,`xb',`sd') ///
>                                         if $MH_touse
  7.         quietly summarize `lnfj', meanonly
  8.         if r(N) < $MH_n {
  9.             scalar `lnf' = .
 10.             exit
 11.         }
 12.         scalar `lnf' = r(sum)
 13. end
```

We can now specify this program in the `llevaluator()` option and use `prior()` options to specify built-in flat priors for the coefficients and the Jeffreys prior for the variance.

```
. set seed 14
. bayesmh mpg weight, llevaluator(normalreg, parameters({var}))
> prior({mpg:}, flat) prior({var}, jeffreys)
> initial({mpg:weight} -0.6 {mpg:_cons} 39 {var} 11.83)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normalreg(xb_mpg,{var})
Priors:
  {mpg:weight _cons} ~ 1 (flat)                                          (1)
                {var} ~ jeffreys
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_mpg.
Bayesian regression                             MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .1668
                                                Efficiency:  min =     .04114
                                                             avg =     .04811
Log marginal likelihood = -198.14302                         max =     .05938
------------------------------------------------------------------------------
             |                                                      Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
      weight |  -.6025616   .0540995   .002667  -.6038729  -.7115221  -.5005915
       _cons |   39.50491   1.677906   .080156   39.45537    36.2433   43.14319
-------------+----------------------------------------------------------------
         var |   12.26586   2.117858   .086915   12.05298   8.827655   17.10703
------------------------------------------------------------------------------
```

We obtain the same results as earlier.

Logistic regression model

Some models, such as logistic regression, do not have any additional parameters except regression coefficients. Here we show how to use a program evaluator for fitting a Bayesian logistic regression model.

We start by creating a program for computing the log likelihood.

```
. program logitll
  1.         version 14
  2.         args lnf xb
  3.         tempvar lnfj
  4.         quietly generate `lnfj' = ln(invlogit( `xb')) if $MH_y == 1 & $MH_
> touse
  5.         quietly replace `lnfj'  = ln(invlogit(-`xb')) if $MH_y == 0 & $MH_
> touse
  6.         quietly summarize `lnfj', meanonly
  7.         if r(N) < $MH_n {
  8.             scalar `lnf' = .
  9.             exit
 10.         }
 11.         scalar `lnf' = r(sum)
 12. end
```

The structure of our log-likelihood evaluator is similar to the one described in *Simple linear regression model*, except we have no extra parameters.

We continue with `auto.dta` and regress `foreign` on `mpg`. For simplicity, we assume a flat prior for the coefficients and use `bayesmh, llevaluator()` to fit this model.

```
. use http://www.stata-press.com/data/r14/auto
(1978 Automobile Data)
. set seed 14
. bayesmh foreign mpg, llevaluator(logitll) prior({foreign:}, flat)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  foreign ~ logitll(xb_foreign)
Prior:
  {foreign:mpg _cons} ~ 1 (flat)                                         (1)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_foreign.
Bayesian regression                             MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2216
                                                Efficiency:  min =     .09293
                                                             avg =     .09989
Log marginal likelihood = -41.626028                         max =      .1068
```

foreign	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
mpg	.16716	.0545771	.00167	.1644019	.0669937	.2790017
_cons	-4.560637	1.261675	.041387	-4.503921	-7.107851	-2.207665

The results from the program-evaluator version match the results from `bayesmh` with a built-in logistic model.

```
. set seed 14
. bayesmh foreign mpg, likelihood(logit) prior({foreign:}, flat)
> initial({foreign:} 0)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  foreign ~ logit(xb_foreign)
Prior:
  {foreign:mpg _cons} ~ 1 (flat)                                          (1)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_foreign.
Bayesian logistic regression                    MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2216
                                                Efficiency:  min =     .09293
                                                             avg =     .09989
Log marginal likelihood = -41.626029                         max =      .1068
------------------------------------------------------------------------------
                                                                Equal-tailed
     foreign |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
         mpg |     .16716   .0545771    .00167   .1644019    .0669937   .2790017
       _cons |  -4.560636   1.261675   .041387  -4.503921   -7.10785  -2.207665
------------------------------------------------------------------------------
```

Because we assumed a flat prior with the density of 1, the log prior is 0, so the log-posterior evaluator for this model is the same as the log-likelihood evaluator.

```
. set seed 14
. bayesmh foreign mpg, evaluator(logitll)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Posterior:
  foreign ~ logitll(xb_foreign)
------------------------------------------------------------------------------
Bayesian regression                             MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2216
                                                Efficiency:  min =     .09293
                                                             avg =     .09989
Log marginal likelihood = -41.626028                         max =      .1068
------------------------------------------------------------------------------
                                                                Equal-tailed
     foreign |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
         mpg |     .16716   .0545771    .00167   .1644019    .0669937   .2790017
       _cons |  -4.560637   1.261675   .041387  -4.503921  -7.107851  -2.207665
------------------------------------------------------------------------------
```

Multivariate normal regression model

Here we demonstrate how to write a program evaluator for a multivariate response. We consider a bivariate normal regression, and we again start with a log-likelihood evaluator. In this example, we also use Mata to speed up our computations.

```
. program mvnregll
  1.         version 14
  2.         args lnf xb1 xb2
  3.         tempvar diff1 diff2
  4.         quietly generate double `diff1' = $MH_y1 - `xb1' if $MH_touse
  5.         quietly generate double `diff2' = $MH_y2 - `xb2' if $MH_touse
  6.         local d $MH_yn
  7.         local n $MH_n
  8.         mata: st_numscalar("`lnf'", mvnll_mata(`d',`n',"`diff1'","`diff2'"))
  9. end
.
. mata:
------------------------------------------------- mata (type end to exit) -----
: real scalar mvnll_mata(real scalar d, n, string scalar sdiff1, sdiff2)
> {
>         real matrix Diff
>         real scalar trace, lnf
>         real matrix Sigma
>
>         Sigma = st_matrix(st_global("MH_m1"))
>         st_view(Diff=.,.,(sdiff1,sdiff2),st_global("MH_touse"))
>
>         /* compute log likelihood */
>         trace = trace(cross(cross(Diff',invsym(Sigma))',Diff'))
>         lnf = -0.5*n*(d*ln(2*pi())+ln(det(Sigma)))-0.5*trace
>
>         return(lnf)
> }
: end
-------------------------------------------------------------------------------
```

The `mvnregll` program has three arguments: a scalar to store the log-likelihood values and two temporary variables containing linear predictors corresponding to each of the two dependent variables. It creates deviations `` `diff1' `` and `` `diff2' `` and passes them, along with other parameters, to the Mata function `mvnll_mata()` to compute the bivariate normal log-likelihood value.

The extra parameter in this model is a covariance matrix of a bivariate response. In *Simple linear regression model*, we specified an extra parameter, variance, which was a scalar, as an additional argument of the evaluator. This is not allowed with matrix parameters. They should be accessed via globals `$MH_m1`, `$MH_m2`, and so on for each matrix model parameters in the order they are specified in option `parameters()`. In our example, we have only one matrix and we access it via `$MH_m1`. `$MH_m1` contains the temporary name of a matrix containing the current value of the covariance matrix parameter.

To demonstrate, we again use `auto.dta`. We rescale the variables to be used in our example to stabilize the results.

```
. use http://www.stata-press.com/data/r14/auto
(1978 Automobile Data)
. replace weight = weight/100
variable weight was int now float
(74 real changes made)
. replace length = length/10
variable length was int now float
(74 real changes made)
```

We fit a bivariate normal regression of `mpg` and `weight` on `length`. We specify the extra covariance parameter as a matrix model parameter `{Sigma,m}` in suboption `parameters()` of `llevaluator()`. We specify flat priors for the coefficients and an inverse-Wishart prior for the covariance matrix.

```
. set seed 14
. bayesmh mpg weight = length, llevaluator(mvnregll, parameters({Sigma,m}))
> prior({mpg:} {weight:}, flat)
> prior({Sigma,m}, iwishart(2,12,I(2))) mcmcsize(1000)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg weight ~ mvnregll(xb_mpg,xb_weight,{Sigma,m})
Priors:
     {mpg:length _cons} ~ 1 (flat)                                       (1)
  {weight:length _cons} ~ 1 (flat)                                       (2)
              {Sigma,m} ~ iwishart(2,12,I(2))
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_mpg.
(2) Parameters are elements of the linear form xb_weight.
Bayesian regression                             MCMC iterations  =      3,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =      1,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .1728
                                                Efficiency:  min =     .02882
                                                             avg =     .05012
Log marginal likelihood = -415.01504                         max =      .1275
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
mpg						
length	-2.040162	.2009062	.037423	-2.045437	-2.369287	-1.676332
_cons	59.6706	3.816341	.705609	59.63619	52.54652	65.84583
weight						
length	3.31773	.1461644	.026319	3.316183	3.008416	3.598753
_cons	-32.19877	2.79005	.484962	-32.4154	-37.72904	-26.09976
Sigma_1_1	11.49666	1.682975	.149035	11.3523	8.691888	14.92026
Sigma_2_1	-2.33596	1.046729	.153957	-2.238129	-4.414118	-.6414916
Sigma_2_2	5.830413	.9051206	.121931	5.630011	4.383648	8.000739

To reduce computation time, we used a smaller MCMC sample size of 1,000 in our example. In your analysis, you should always verify whether a smaller MCMC sample size results in precise enough estimates before using it for final results.

We can check our results against `bayesmh` using the built-in multivariate normal regression after adjusting the initial values.

```
. set seed 14
. bayesmh mpg weight = length, likelihood(mvnormal({Sigma,m}))
> prior({mpg:} {weight:}, flat)
> prior({Sigma,m}, iwishart(2,12,I(2)))
> mcmcsize(1000) initial({mpg:} {weight:} 0)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg weight ~ mvnormal(2,xb_mpg,xb_weight,{Sigma,m})
Priors:
     {mpg:length _cons} ~ 1 (flat)                                        (1)
  {weight:length _cons} ~ 1 (flat)                                        (2)
              {Sigma,m} ~ iwishart(2,12,I(2))
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_mpg.
(2) Parameters are elements of the linear form xb_weight.
Bayesian multivariate normal regression          MCMC iterations  =      3,500
Random-walk Metropolis-Hastings sampling         Burn-in          =      2,500
                                                 MCMC sample size =      1,000
                                                 Number of obs    =         74
                                                 Acceptance rate  =      .1728
                                                 Efficiency:  min =     .02882
                                                              avg =     .05012
Log marginal likelihood = -415.01504                          max =      .1275
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
length	-2.040162	.2009062	.037423	-2.045437	-2.369287	-1.676332
_cons	59.6706	3.816341	.705609	59.63619	52.54652	65.84583
weight						
length	3.31773	.1461644	.026319	3.316183	3.008416	3.598753
_cons	-32.19877	2.79005	.484962	-32.4154	-37.72904	-26.09976
Sigma_1_1	11.49666	1.682975	.149035	11.3523	8.691888	14.92026
Sigma_2_1	-2.33596	1.046729	.153957	-2.238129	-4.414118	-.6414916
Sigma_2_2	5.830413	.9051206	.121931	5.630011	4.383648	8.000739

We obtain the same results.

Similarly, we can define the log-posterior evaluator. We already have the log-likelihood evaluator, which we can reuse in our log-posterior evaluator. The only additional portion is to compute the log of the inverse-Wishart prior density for the covariance parameter.

```
. program mvniWishart
  1.         version 14
  2.         args lnp xb1 xb2
  3.         tempvar diff1 diff2
  4.         quietly generate double `diff1' = $MH_y1 - `xb1' if $MH_touse
  5.         quietly generate double `diff2' = $MH_y2 - `xb2' if $MH_touse
  6.         local d $MH_yn
  7.         local n $MH_n
  8.         mata: ///
>             st_numscalar("`lnp'", mvniWish_mata(`d',`n',"`diff1'","`diff2'"))
  9. end
.
. mata:
------------------------------------------------- mata (type end to exit) -----
: real scalar mvniWish_mata(real scalar d, n, string scalar sdiff1, sdiff2)
> {
>         real scalar lnf, lnprior
>         real matrix Sigma
>
>         /* compute log likelihood */
>         lnf = mvnll_mata(d,n,sdiff1,sdiff2)
>         /* compute log of inverse-Wishart prior for Sigma */
>         Sigma = st_matrix(st_global("MH_m1"))
>         lnprior = lniwishartden(12,I(2),Sigma)
>         return(lnf + lnprior)
> }
: end
-------------------------------------------------------------------------------
```

The results of the log-posterior evaluator match our earlier results.

```
. set seed 14
. bayesmh mpg weight = length, evaluator(mvniWishart, parameters({Sigma,m}))
> mcmcsize(1000)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Posterior:
  mpg weight ~ mvniWishart(xb_mpg,xb_weight,{Sigma,m})
------------------------------------------------------------------------------

Bayesian regression                             MCMC iterations  =      3,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =      1,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .1728
                                                Efficiency:  min =     .02882
                                                             avg =     .05012
Log marginal likelihood = -415.01504                         max =      .1275

------------------------------------------------------------------------------
             |                                                 Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
      length |  -2.040162   .2009062   .037423  -2.045437  -2.369287  -1.676332
       _cons |   59.6706   3.816341   .705609   59.63619   52.54652   65.84583
-------------+----------------------------------------------------------------
weight       |
      length |   3.31773   .1461644   .026319   3.316183   3.008416   3.598753
       _cons |  -32.19877    2.79005   .484962   -32.4154  -37.72904  -26.09976
-------------+----------------------------------------------------------------
   Sigma_1_1 |   11.49666   1.682975   .149035    11.3523   8.691888   14.92026
   Sigma_2_1 |  -2.33596   1.046729   .153957  -2.238129  -4.414118  -.6414916
   Sigma_2_2 |   5.830413   .9051206   .121931   5.630011   4.383648   8.000739
------------------------------------------------------------------------------
```

Sometimes, it may be useful to be able to pass options to our evaluators. For example, we used the identity `I(2)` matrix as a scale matrix of the inverse Wishart distribution. Suppose that we want to check the sensitivity of our results to other choices of the scale matrix. We can pass the name of a matrix we want to use in an option. In our example, we use the `vmatrix()` option to pass the name of the scale matrix. We later specify this option within suboption `passthruopts()` of the `evaluator()` option. The options passed this way are stored in the `$MH_passthruopts` global macro.

```
. program mvniWishartV
  1.         version 14
  2.         args lnp xb1 xb2
  3.         tempvar diff1 diff2
  4.         quietly generate double `diff1' = $MH_y1 - `xb1' if $MH_touse
  5.         quietly generate double `diff2' = $MH_y2 - `xb2' if $MH_touse
  6.         local d $MH_yn
  7.         local n $MH_n
  8.         local 0 , $MH_passthruopts
  9.         syntax, vmatrix(string)
 10.         mata: st_numscalar("`lnp'", ///
>               mvniWishV_mata(`d',`n',"`diff1'","`diff2'","`vmatrix'"))
 11. end
```

```
. mata:
------------------------------------------------- mata (type end to exit) -----
: real scalar mvniWishV_mata(real scalar d, n, string scalar sdiff1, sdiff2,
> vmat)
> {
>         real scalar lnf, lnprior
>         real matrix Sigma
>
>         /* compute log likelihood */
>         lnf = mvnll_mata(d,n,sdiff1,sdiff2)
>         /* compute log of inverse-Wishart prior for Sigma */
>         Sigma = st_matrix(st_global("MH_m1"))
>         lnprior = lniwishartden(12,st_matrix(vmat),Sigma)
>         return(lnf + lnprior)
> }
: end
-------------------------------------------------------------------------------
```

We now define the scale matrix `V` (as the identity matrix to match our previous results) and specify `vmatrix(V)` in suboption `passthruopts()` of `evaluator()`.

```
. set seed 14
. matrix V = I(2)
. bayesmh mpg weight = length,
> evaluator(mvniWishartV, parameters({Sigma,m}) passthruopts(vmatrix(V)))
> mcmcsize(1000)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Posterior:
  mpg weight ~ mvniWishartV(xb_mpg,xb_weight,{Sigma,m})
------------------------------------------------------------------------------

Bayesian regression                             MCMC iterations  =      3,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =      1,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .1728
                                                Efficiency:  min =     .02882
                                                             avg =     .05012
Log marginal likelihood = -415.01504                         max =      .1275

------------------------------------------------------------------------------
             |                                                 Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
      length |  -2.040162   .2009062   .037423  -2.045437  -2.369287  -1.676332
       _cons |    59.6706   3.816341   .705609   59.63619   52.54652   65.84583
-------------+----------------------------------------------------------------
weight       |
      length |   3.31773   .1461644   .026319   3.316183   3.008416   3.598753
       _cons |  -32.19877    2.79005   .484962   -32.4154  -37.72904  -26.09976
-------------+----------------------------------------------------------------
   Sigma_1_1 |   11.49666   1.682975   .149035    11.3523   8.691888   14.92026
   Sigma_2_1 |   -2.33596   1.046729   .153957  -2.238129  -4.414118  -.6414916
   Sigma_2_2 |   5.830413   .9051206   .121931   5.630011   4.383648   8.000739
------------------------------------------------------------------------------
```

The results are the same as before.

Cox proportional hazards regression

Some evaluators may require additional variables, apart from the dependent and independent variables, for computation. For example, in a Cox proportional hazards model such variable is a failure or censoring indicator. The `coxphll` program below computes partial log likelihood for the Cox proportional hazards regression. The failure indicator will be passed to the evaluator as an extra variable in suboption `extravars()` of option `llevaluator()` or option `evaluator()` and can be accessed from the global macro `$MH_extravars`.

```
. program coxphll
  1.         version 14
  2.         args lnf xb
  3.         tempvar negt
  4.         quietly generate double `negt' = -$MH_y1
  5.         local d "$MH_extravars"
  6.         sort $MH_touse `negt' `d'
  7.         tempvar B A sumd last L
  8.         local byby "by $MH_touse `negt' `d'"
  9.         quietly {
 10.                gen double `B' = sum(exp(`xb')) if $MH_touse
 11.                `byby': gen double `A' = cond(_n==_N, sum(`xb'), .) ///
>                                         if `d'==1 & $MH_touse
 12.                `byby': gen `sumd' = cond(_n==_N, sum(`d'), .) if $MH_touse
 13.                `byby': gen byte `last' = (_n==_N & `d' == 1) if $MH_touse
 14.                gen double `L' = `A' - `sumd'*ln(`B') if `last' & $MH_touse
 15.                quietly count if $MH_touse & `last'
 16.                local n = r(N)
 17.                summarize `L' if `last' & $MH_touse, meanonly
 18.         }
 19.         if r(N) < `n' {
 20.             scalar `lnf' = .
 21.             exit
 22.         }
 23.         scalar `lnf' = r(sum)
 24. end
```

We demonstrate the command using the survival-time `cancer` dataset. The survival-time variable is `studytime` and the failure indicator is `died`. The regressor of interest in this model is `age`. We use a fairly noninformative normal prior with a zero mean and a variance of 100 for the regression coefficient of `age`. (The constant in the Cox proportional hazards model is not likelihood-identifiable, so we omit it from this model with a noninformative prior.)

```
. use http://www.stata-press.com/data/r14/cancer
(Patient Survival in Drug Trial)

. gsort -studytime died

. set seed 14

. bayesmh studytime age, llevaluator(coxphll, extravars(died))
> prior({studytime:}, normal(0,100)) noconstant mcmcsize(1000)
Burn-in ...
Simulation ...

Model summary
------------------------------------------------------------------------------
Likelihood:
  studytime ~ coxphll(xb_studytime)

Prior:
  {studytime:age} ~ normal(0,100)                                         (1)
------------------------------------------------------------------------------
(1) Parameter is an element of the linear form xb_studytime.
```

```
Bayesian regression                             MCMC iterations  =      3,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =      1,000
                                                Number of obs    =         48
                                                Acceptance rate  =      .4066
Log marginal likelihood = -103.04797            Efficiency       =      .3568

------------------------------------------------------------------------------
             |                                                 Equal-tailed
   studytime |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
         age |    .076705   .0330669   .001751   .077936   .0099328   .1454275
------------------------------------------------------------------------------
```

We specified the failure indicator `died` in suboption `extravars()` of `llevaluator()`. We again used a smaller value for the MCMC sample size only to reduce computation time.

For the log-posterior evaluator, we add the log of the normal prior of the `age` coefficient to the log-likelihood value to obtain the final log-posterior value. We did not need to specify the loop in the log-prior computation in this example, but we did this to be general, in case more than one regressor is included in the model.

```
. program coxphnormal
  1.         version 14
  2.         args lnp xb
.         /* compute log likelihood */
.         tempname lnf
  3.         scalar `lnf' = .
  4.         quietly coxphll `lnf' `xb'
.         /* compute log priors of regression coefficients */
.         tempname lnprior
  5.         scalar `lnprior' = 0
  6.         forvalues i = 1/$MH_bn {
  7.             scalar `lnprior' = `lnprior' + lnnormalden($MH_b[1,`i'], 10)
  8.         }
  9.         /* compute log posterior */
.         scalar `lnp' = `lnf' + `lnprior'
 10. end
```

As expected, we obtain the same results as previously.

```
. set seed 14
. bayesmh studytime age, evaluator(coxphnormal, extravars(died))
> noconstant mcmcsize(1000)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Posterior:
  studytime ~ coxphnormal(xb_studytime)
------------------------------------------------------------------------------

Bayesian regression                             MCMC iterations  =      3,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =      1,000
                                                Number of obs    =         48
                                                Acceptance rate  =      .4066
Log marginal likelihood = -103.04797            Efficiency       =      .3568

------------------------------------------------------------------------------
             |                                                 Equal-tailed
   studytime |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
         age |    .076705   .0330669   .001751   .077936   .0099328   .1454275
------------------------------------------------------------------------------
```

Global macros

Global macros	Description
`$MH_n`	number of observations
`$MH_yn`	number of dependent variables
`$MH_touse`	variable containing 1 for the observations to be used; 0 otherwise
`$MH_w`	variable containing weight associated with the observations
`$MH_extravars`	*varlist* specified in `extravars()`
`$MH_passthruopts`	options specified in `passthruopts()`
One outcome	
`$MH_y1`	name of the dependent variable
`$MH_x1`	name of the first independent variable
`$MH_x2`	name of the second independent variable
...	
`$MH_xn`	number of independent variables
`$MH_xb`	name of a temporary variable containing the linear combination
Multiple outcomes	
`$MH_y1`	name of the first dependent variable
`$MH_y2`	name of the second dependent variable
...	
`$MH_y1x1`	name of the first independent variable modeling `y1`
`$MH_y1x2`	name of the second independent variable modeling `y1`
...	
`$MH_y1xn`	number of independent variables modeling `y1`
`$MH_y1xb`	name of a temporary variable containing the linear combination modeling `y1`
`$MH_y2x1`	name of the first independent variable modeling `y2`
`$MH_y2x2`	name of the second independent variable modeling `y2`
...	
`$MH_y2xn`	number of independent variables modeling `y2`
`$MH_y2xb`	name of a temporary variable containing the linear combination modeling `y2`
...	
Scalar and matrix parameters	
`$MH_b`	name of a temporary vector of coefficients; stripes are properly named after the name of the coefficients
`$MH_bn`	number of coefficients
`$MH_p`	name of a temporary vector of additional scalar model parameters, if any; stripes are properly named
`$MH_pn`	number of additional scalar model parameters
`$MH_m1`	name of a temporary matrix of the first matrix parameter, if any
`$MH_m2`	name of a temporary matrix of the second matrix parameter, if any
...	
`$MH_mn`	number of matrix model parameters

Stored results

In addition to the results stored by `bayesmh`, `bayesmh, evaluator()` and `bayesmh, llevaluator()` store the following in `e()`:

Macros

`e(evaluator)`	program evaluator (one equation)
`e(evaluator#)`	program evaluator for the #th equation
`e(evalparams)`	evaluator parameters (one equation)
`e(evalparams#)`	evaluator parameters for the #th equation
`e(extravars)`	extra variables (one equation)
`e(extravars#)`	extra variables for the #th equation
`e(passthruopts)`	pass-through options (one equation)
`e(passthruopts#)`	pass-through options for the #th equation

Also see

[BAYES] **bayesmh** — Bayesian regression using Metropolis–Hastings algorithm

[BAYES] **bayesmh postestimation** — Postestimation tools for bayesmh

[BAYES] **intro** — Introduction to Bayesian analysis

[BAYES] **Glossary**

Title

bayesmh postestimation — Postestimation tools for bayesmh

Postestimation commands Remarks and examples Also see

Postestimation commands

The following postestimation commands are available after `bayesmh`:

Command	Description
`bayesgraph`	graphical summaries and convergence diagnostics
`bayesstats ess`	effective sample sizes and related statistics
`bayesstats summary`	Bayesian summary statistics for model parameters and their functions
`bayesstats ic`	Bayesian information criteria and Bayes factors
`bayestest model`	hypothesis testing using model posterior probabilities
`bayestest interval`	interval hypothesis testing
* `estimates`	cataloging estimation results

* `estimates table` and `estimates stats` are not appropriate with `bayesmh` estimation results.

Remarks and examples

Remarks are presented under the following headings:

Different ways of specifying model parameters
Specifying functions of model parameters
Storing estimation results after bayesmh

After estimation, you can use `bayesgraph` to check convergence of MCMC visually. Once convergence is established, you can use `bayesstats summary` to obtain Bayesian summaries such as posterior means and standard deviations of model parameters and functions of model parameters; `bayesstats ess` to compute effective sample sizes and related statistics for model parameters and functions of model parameters; and `bayesstats ic` to compute Bayesian information criteria and Bayes factors for model parameters and their functions. You can use `bayestest model` to test hypotheses by comparing posterior probabilities of models. You can also use `bayestest interval` to test interval hypotheses about parameters and functions of parameters.

For an overview example of postestimation commands, see *Overview example* in [BAYES] **bayes**.

Different ways of specifying model parameters

Many `bayesmh` postestimation commands such as `bayesstats summary` and `bayesgraph` allow you to specify model parameters for which you want to see the results. To see results for all parameters, simply type a postestimation command without arguments after `bayesmh` estimation, for example,

```
. bayesstats summary
```

or you could type

```
. bayesstats summary _all
```

To manually list all model parameters, type

```
. bayesstats summary {param1} {param2} ...
```

or

```
. bayesstats summary {param1 param2} ...
```

The only exception is the `bayesgraph` command when there is more than one model parameter. In that case, `bayesgraph` requires that you either specify `_all` to request all model parameters or specify the model parameters of interest.

You can refer to a single model parameter in the same way you define parameters in the `bayesmh` command. For example, for a parameter with name `param` and no equation name, you can use `{param}`. For a parameter with name `param` and equation name `eqname`, you can use its full name `{eqname:name}`, where the equation name and the parameter name are separated with a colon. With postestimation commands, you can also omit the equation name when referring to the parameter with an equation name.

In the presence of more than one model parameter, you have several ways for referring to multiple parameters at once. If parameters have the same equation name, you can refer to all the parameters with that equation name as follows.

Suppose that you have three parameters with the same equation name `eqname`. Then the specification

```
. bayesstats summary {eqname:param1} {eqname:param2} {eqname:param3}
```

is the same as the specification

```
. bayesstats summary {eqname:}
```

or the specification

```
. bayesstats summary {eqname:param1 param2 param3}
```

The above specification is useful if we want to refer to a subset of parameters with the same equation name. For example, in the above, if we wanted to use only `param1` and `param2`, we could type

```
. bayesstats summary {eqname:param1 param2}
```

There is also a convenient way to refer to the parameters with the same name but different equation names. For example, typing

```
. bayesstats summary {eqname1:param} {eqname2:param}
```

is the same as simply typing

```
. bayesstats summary {param}
```

You can mix and match all the specifications above in one call to a postestimation command. You can also specify expressions of model parameters; see *Specifying functions of model parameters* for details.

Note that if `param` refers to a matrix model parameter, then the results will be provided for all elements of the matrix. For example, if `param` is the name of a 2×2 matrix, then typing

```
. bayesstats summary {param}
```

implies the following:

```
. bayesstats summary {param_1_1} {param_1_2} {param_2_1} {param_2_2}
```

Specifying functions of model parameters

You can use bayesmh postestimation commands to obtain results for functions or expressions of model parameters. Each expression must be specified in parentheses. An expression can be any Stata expression, but it may not include matrix model parameters. However, you may include individual elements of matrix model parameters. You may provide labels for your expressions.

For example, we can obtain results for the exponentiated parameter {param} as follows:

```
. bayesstats summary (exp({param}))
```

Note that we specified the expression in parentheses.

We can include a label, say, myexp, in the above by typing

```
. bayesstats summary (myexp: exp({param}))
```

We can specify multiple expressions by typing

```
. bayesstats summary (myexp: exp({param}) (sd: sqrt({var})))
```

If param is a matrix, we can specify expressions, including its elements, but not the matrix itself in the following:

```
. bayesstats summary (exp({param_1_1})) (exp({param_1_2})) ...
```

Storing estimation results after bayesmh

The bayesmh command stores various e() results such as scalars, macros, and matrices in memory like any other estimation command. Unlike other estimation commands, bayesmh also saves the resulting simulation dataset containing MCMC samples of parameters to disk. Many bayesmh postestimation commands such as bayesstats summary and bayesstats ess require access to this file. If you do not specify the saving() option with bayesmh, the command saves simulation results in a temporary Stata dataset. This file is being replaced with the new simulation results each time bayesmh is run. To save your simulation results, you must specify the saving() option with bayesmh, in which case your simulation results are saved to the specified file in the specified location and will not be overridden by the next call to bayesmh.

You can specify the saving() option during estimation by typing

```
. bayesmh ..., likelihood() prior() ... saving()
```

or on replay by typing

```
. bayesmh, saving()
```

As you can with other estimation commands, you can use estimates store to store bayesmh estimation results in memory and estimates save to save them to disk, but you must first use the saving() option with bayesmh to save simulation data in a permanent dataset. For example, type

```
. bayesmh ..., likelihood() prior() ... saving(bmh_simdata)
. estimates store model1
```

or, after bayesmh estimation, type

```
. bayesmh, saving(bmh_simdata)
. estimates store model1
```

Once you create a permanent dataset, it is your responsibility to erase it after it is no longer needed. `estimates drop` and `estimates clear` will drop estimation results only from memory; they will not erase the simulation files you saved.

```
. estimates drop model1
. erase bmh_simdata.dta
```

See [R] **estimates** for more information about commands managing estimation results. `estimates table` and `estimates stats` are not appropriate after `bayesmh`.

Also see

[BAYES] **bayesmh** — Bayesian regression using Metropolis–Hastings algorithm

[BAYES] **bayesmh evaluators** — User-defined evaluators with bayesmh

[BAYES] **bayes** — Introduction to commands for Bayesian analysis

[BAYES] **intro** — Introduction to Bayesian analysis

[BAYES] **Glossary**

[U] **20 Estimation and postestimation commands**

Title

bayesgraph — Graphical summaries and convergence diagnostics

Description

`bayesgraph` provides graphical summaries and convergence diagnostics for simulated posterior distributions (MCMC samples) of model parameters and functions of model parameters obtained from the `bayesmh` command. Graphical summaries include trace plots, autocorrelation plots, and various distributional plots.

Quick start

Trace plot, histogram, autocorrelation plot, and density plot for parameter {p}

```
bayesgraph diagnostics {p}
```

Add plots for parameter {y:x1}

```
bayesgraph diagnostics {p} {y:x1}
```

As above, but for all model parameters

```
bayesgraph diagnostics _all
```

As above, but for a function of model parameters {y:x1} and {p}

```
bayesgraph diagnostics ({y:x1}/{p})
```

Specify a blue trace plot line for all plots

```
bayesgraph diagnostics {p} {y:x1} {y:x2}, traceopts(lcolor(blue))
```

Specify a blue trace plot line only for the second trace plot

```
bayesgraph diagnostics {p} {y:x1} {y:x2}, trace2opts(lcolor(blue))
```

Trace plots for all parameters in a single graph

```
bayesgraph trace _all, byparm
```

Cumulative sum plot for parameter {p}

```
bayesgraph cusum {p}
```

Scatterplot matrix for parameters {p} and {y:x1}

```
bayesgraph matrix {p} {y:x1}
```

Autocorrelation plots for elements 1,1 and 2,1 of matrix parameter {S}

```
bayesgraph ac {S_1_1} {S_2_1}
```

Diagnostic plots for all parameters in the model and pause at least 3 seconds before displaying the next graph

```
bayesgraph diagnostics _all, sleep(3)
```

As above, but pause until the user presses any key

```
bayesgraph diagnostics _all, wait
```

As above, but close the current Graph window when the next graph is displayed

```
bayesgraph diagnostics _all, close
```

Menu

Statistics > Bayesian analysis > Graphical summaries

Syntax

Graphical summaries and convergence diagnostics for a single parameter

```
bayesgraph graph scalar_param [ , singleopts ]
```

Graphical summaries and convergence diagnostics for multiple parameters

```
bayesgraph graph spec [spec ...] [ , multiopts ]
```

```
bayesgraph matrix spec spec [spec ...] [ , singleopts ]
```

Graphical summaries and convergence diagnostics for all parameters

```
bayesgraph graph _all [ , multiopts ]
```

graph	Description
diagnostics	multiple diagnostics in compact form
trace	trace plots
ac	autocorrelation plots
histogram	histograms
kdensity	density plots
cusum	cumulative sum plots
matrix	scatterplot matrix

`bayesgraph matrix` requires at least two parameters.

scalar_param is a scalar model parameter specified as `{param}` or `{eqname:param}` or an expression *exprspec* of scalar model parameters. Matrix model parameters are not allowed, but you may refer to their individual elements.

exprspec is an optionally labeled expression of model parameters specified in parentheses:

([*exprlabel* :] *expr*)

exprlabel is a valid Stata name, and *expr* is a scalar expression that may not contain matrix model parameters. See *Specifying functions of model parameters* in [BAYES] **bayesmh postestimation** for examples.

spec is either *scalar_param* or *exprspec*.

singleopts	Description
Options	
`skip(#)`	skip every # observations from the MCMC sample; default is `skip(0)`
`name(`*name*`, ...)`	specify name of graph
`saving(`*filename*`, ...)`	save graph in file
graphopts	graph-specific options

multiopts	Description
Options	
`byparm[ (`*grbyparmopts*`) ]`	specify the display of plots on one graph; default is separate graph for each plot; not allowed with graphs `diagnostics` and `matrix` or with option `combine()`
`combine[ (`*grcombineopts*`) ]`	specify the display of plots on one graph; recommended when the number of parameters is large; not allowed with graphs `diagnostics` and `matrix` or with option `byparm()`
`sleep(#)`	pause for # seconds between multiple graphs; default is `sleep(0)`
`wait`	pause until the —`more`— condition is cleared
`[no]close`	(do not) close Graph windows when the next graph is displayed with multiple graphs; default is `noclose`
`skip(#)`	skip every # observations from the MCMC sample; default is `skip(0)`
`name(`*namespec*`, ...)`	specify names of graphs
`saving(`*filespec*`, ...)`	save graphs in files
`graphopts(`*graphopts*`)`	control the look of all graphs; not allowed with `byparm()`
`graph[#]opts(`*graphopts*`)`	control the look of #th graph; not allowed with `byparm()`
graphopts	equivalent to `graphopts(`*graphopts*`)`; only one may be specified

graphopts	Description
diagnosticsopts	options for `bayesgraph diagnostics`
tslineopts	options for `bayesgraph trace` and `bayesgraph cusum`
acopts	options for `bayesgraph ac`
histopts	options for `bayesgraph histogram`
kdensityopts	options for `bayesgraph kdensity`
grmatrixopts	options for `bayesgraph matrix`

diagnosticsopts	Description
`traceopts(`*tslineopts*`)`	affect rendition of all trace plots
`trace[#]opts(`*tslineopts*`)`	affect rendition of #th trace plot
`acopts(`*acopts*`)`	affect rendition of all autocorrelation plots
`ac[#]opts(`*acopts*`)`	affect rendition of #th autocorrelation plot
`histopts(`*histopts*`)`	affect rendition of all histogram plots
`hist[#]opts(`*histopts*`)`	affect rendition of #th histogram plot
`kdensopts(`*kdensityopts*`)`	affect rendition of all density plots
`kdens[#]opts(`*kdensityopts*`)`	affect rendition of #th density plot
grcombineopts	any option documented in [G-2] **graph combine**

acopts	Description
`ci`	plot autocorrelations with confidence intervals; not allowed with `byparm()`
acopts	any options other than `generate()` documented for the `ac` command in [TS] **corrgram**

kdensityopts	Description
kdensopts	options for the overall kernel density plot
`show(`*showspec*`)`	show first-half density, second-half density, or both; default is `both`
`kdensfirst(`*kdens1opts*`)`	affect rendition of the first-half density plot
`kdenssecond(`*kdens2opts*`)`	affect rendition of the second-half density plot

Options

Options

`byparm[(`*grbyparmopts*`)]` specifies the display of all plots of parameters as subgraphs on one graph. By default, a separate graph is produced for each plot when multiple parameters are specified. This option is not allowed with `bayesgraph diagnostics` or `bayesgraph matrix` and may not be combined with option `combine()`. When many parameters or expressions are specified, this option may fail because of memory constraints. In that case, you may use option `combine()` instead.

grbyparmopts is any of the suboptions of `by()` documented in [G-3] ***by_option***.

`byparm()` allows y scales to differ for all graph types and forces x scales to be the same only for `bayesgraph trace` and `bayesgraph cusum`. Use `noyrescale` within `byparm()` to specify a common y axis, and use `xrescale` or `noxrescale` to change the default behavior for the x axis.

`byparm()` with `bayesgraph trace` and `bayesgraph cusum` defaults to displaying multiple plots in one column to accommodate the x axis with many iterations. Use `norowcoldefault` within `byparm()` to switch back to the default behavior of options `rows()` and `cols()` of the [G-3] ***by_option***.

`combine[(`*grcombineopts*`)]` specifies the display of all plots of parameters as subgraphs on one graph and is an alternative to `byparm()` with a large number of parameters. By default, a separate

graph is produced for each plot when multiple parameters are specified. This option is not allowed with `bayesgraph diagnostics` or `bayesgraph matrix` and may not be combined with option `byparm()`. It can be used in cases where a large number of parameters or expressions are specified and the `byparm()` option would cause an error because of memory constraints.

grcombineopts is any of the options documented in [G-2] **graph combine**.

`sleep(#)` specifies pausing for # seconds before producing the next graph. This option is allowed only when multiple parameters are specified. This option may not be combined with `wait`, `combine()`, or `byparm()`.

`wait` causes `bayesgraph` to display —`more`— and pause until any key is pressed before producing the next graph. This option is allowed when multiple parameters are specified. This option may not be combined with `sleep()`, `combine()`, or `byparm()`. `wait` temporarily ignores the global setting that is specified using `set more off`.

[`no`]`close` specifies that, for multiple graphs, the Graph window be closed when the next graph is displayed. The default is `noclose` or to not close any Graph windows.

`skip(#)` specifies that every # observations from the MCMC sample not be used for computation. The default is `skip(0)` or to use all observations in the MCMC sample. Option `skip()` can be used to subsample or thin the chain. `skip(#)` is equivalent to a thinning interval of #+1. For example, if you specify `skip(1)`, corresponding to the thinning interval of 2, the command will skip every other observation in the sample and will use only observations 1, 3, 5, and so on in the computation. If you specify `skip(2)`, corresponding to the thinning interval of 3, the command will skip every 2 observations in the sample and will use only observations 1, 4, 7, and so on in the computation. `skip()` does not thin the chain in the sense of physically removing observations from the sample, as is done by `bayesmh`'s `thinning()` option. It only discards selected observations from the computation and leaves the original sample unmodified.

`name(`*namespec*[`, replace`]`)` specifies the name of the graph or multiple graphs. See [G-3] ***name_option*** for a single graph. If multiple graphs are produced, then the argument of `name()` is either a list of names or a *stub*, in which case graphs are named *stub*`1`, *stub*`2`, and so on. With multiple graphs, if `name()` is not specified and neither `sleep()` nor `wait` is specified, `name(Graph__#, replace)` is assumed, and thus the produced graphs may be replaced by subsequent `bayesgraph` commands.

The `replace` suboption causes existing graphs with the specified name or names to be replaced.

`saving(`*filespec*[`, replace`]`)` specifies the filename or filenames to use to save the graph or multiple graphs to disk. See [G-3] ***saving_option*** for a single graph. If multiple graphs are produced, then the argument of `saving()` is either a list of filenames or a *stub*, in which case graphs are saved with filenames *stub*`1`, *stub*`2`, and so on.

The `replace` suboption specifies that the file (or files) may be replaced if it already exists.

`graphopts(`*graphopts*`)` and `graph`[`#`]`opts(`*graphopts*`)` affect the rendition of graphs. `graphopts()` affects the rendition of all graphs but may be overridden for specific graphs by using the `graph#opts()` option. The options specified within `graph#opts()` are specific for each type of graph.

The two specifications

```
bayesgraph ..., graphopts(graphopts)
```

and

```
bayesgraph ..., graphopts
```

are equivalent, but you may specify one or the other.

These options are not allowed with `byparm()` and when only one parameter is specified.

graphopts specifies options specific to each graph type.

diagnosticsopts specifies options for use with `bayesgraph diagnostics`. See the corresponding table in the syntax diagram for a list of options.

tslineopts specifies options for use with `bayesgraph trace` and `bayesgraph cusum`. See the options of [TS] **tsline** except `by()`.

acopts specifies options for use with `bayesgraph ac`.

`ci` requests that the graph of autocorrelations with confidence intervals be plotted. By default, confidence intervals are not plotted. This option is not allowed with `byparm()`.

acopts specifies any options except `generate()` of the `ac` command in [TS] **corrgram**.

histopts specifies options for use with `bayesgraph histogram`. See options of [R] **histogram** except `by()`.

kdensityopts specifies options for use with `bayesgraph kdensity`.

kdensopts specifies options for the overall kernel density plot. See the options documented in [R] **kdensity** except `generate()` and `at()`.

`show(`*showspec*`)` specifies which kernel density curves to plot. *showspec* is one of `both`, `first`, `second`, or `none`. `show(both)`, the default, overlays both the first-half density curve and the second-half density curve with the overall kernel density curve. If `show(first)` is specified, only the first-half density curve, obtained from the first half of an MCMC sample, is plotted. If `show(second)` is specified, only the second-half density curve, obtained from the second half of an MCMC sample, is plotted. If `show(none)` is specified, only the overall kernel density curve is shown.

`kdensityfirst(`*kdens1opts*`)` specifies options of [G-2] **graph twoway kdensity** except `by()` to affect rendition of the first-half kernel density plot.

`kdensitysecond(`*kdens2opts*`)` specifies options of [G-2] **graph twoway kdensity** except `by()` to affect rendition of the second-half kernel density plot.

grmatrixopts specifies options for use with `bayesgraph matrix`. See the options of [G-2] **graph matrix** except `by()`.

Remarks and examples

Remarks are presented under the following headings:

Using bayesgraph

bayesgraph requires specifying at least one parameter with all graph types except matrix, which requires at least two parameters. To request graphs for all parameters, use _all.

When multiple graphs are produced, they are automatically stored in memory with names Graph__# and will all appear on the screen. After you are done reviewing the graphs, you can type

```
. graph close Graph__*
```

to close these graphs or type

```
. graph drop Graph__*
```

to close the graphs and drop them from memory.

If you would like to see only one graph at a time, you can specify option close to close the Graph window when the next graph is displayed. You can also use option sleep() or option wait to pause between the subsequent graphs. The sleep(#) option causes each graph to pause for # seconds. The wait option causes bayesgraph to wait until a key is pressed before producing the next graph.

You can combine separate graphs into one by specifying one of byparm() or combine(). These options are not allowed with diagnostics or matrix graphs. The byparm() option produces more compact graphs, but it may not be feasible with many parameters or expressions and large sizes of MCMC samples.

With multiple graphs, you can control the look of each individual graph with graph#opts(). Options common to all graphs may be specified in graphopts() or passed directly to the command as with single graphs.

Examples

We demonstrate the bayesgraph command using an example of Bayesian normal linear regression applied to auto.dta. We model the mpg variable using a normal distribution with unknown mean and variance. Our Bayesian model thus has two parameters, {mpg:_cons} and {var}, for which we need to specify prior distributions. We consider fairly noninformative prior distributions for these parameters: $N(0, 1000)$ for the constant and inverse gamma with shape and scale of 0.1 for the variance. Because the specified prior distributions are independent and semiconjugate relative to the normal data distribution, we can use Gibbs sampling for both parameters instead of the default MH sampling. To illustrate, we will use Gibbs sampling for the variance and MH sampling (default) for the mean.

We use bayesmh to fit our model.

```
. use http://www.stata-press.com/data/r14/auto
(1978 Automobile Data)
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, normal(0,1000))
> prior({var}, igamma(0.1,0.1)) block({var}, gibbs)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
Priors:
  {mpg:_cons} ~ normal(0,1000)
        {var} ~ igamma(0.1,0.1)
------------------------------------------------------------------------------
Bayesian normal regression                      MCMC iterations  =     12,500
Metropolis-Hastings and Gibbs sampling          Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .7133
                                                Efficiency:  min =      .2331
                                                             avg =      .6166
Log marginal likelihood = -242.1155                          max =          1
------------------------------------------------------------------------------
                                                              Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
       _cons |  21.29231   .6648867   .013771   21.29419   19.94367   22.56746
-------------+----------------------------------------------------------------
         var |  34.2805    5.844213   .058442    33.6464    24.65882   47.5822
------------------------------------------------------------------------------
```

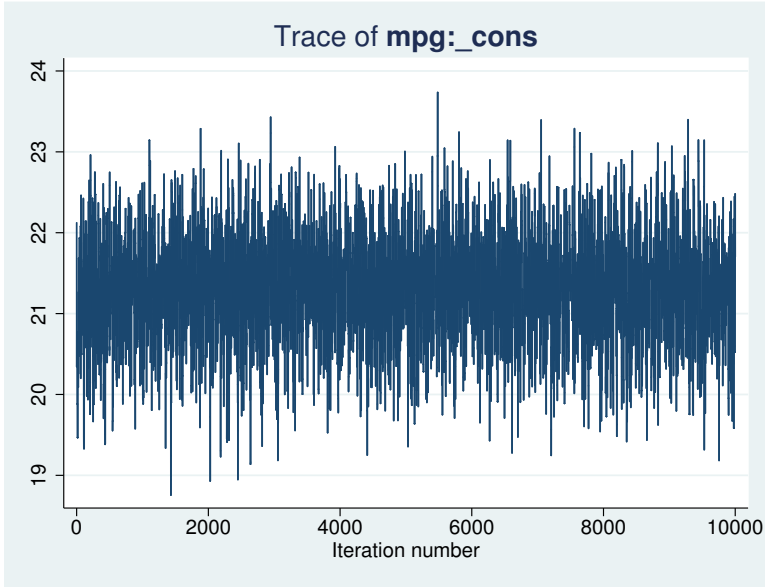
The MCMC simulation has a fairly high efficiency for the MH algorithm of 23% for the mean and an efficiency of 1 for the variance because of the Gibbs sampling. The output suggests no convergence problems. However, it is important to verify this and to also inspect various other graphical summaries of the parameters. This example demonstrates graphical summaries for a well-mixing MCMC chain that has converged and that generates samples from the posterior distribution of the model. For examples of poor-mixing MCMC chains, see *Convergence diagnostics in* MCMC in [BAYES] **intro**.

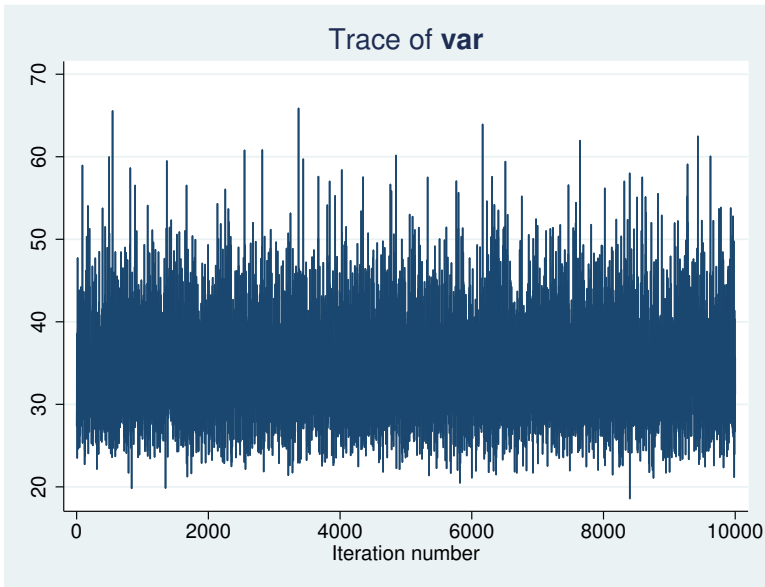
Trace plots

We start with trace plots, which plot the values of the simulated parameters against the iteration number and connect consecutive values with a line. For a well-mixing parameter, the range of the parameter is traversed rapidly by the MCMC chain, which makes the drawn lines look almost vertical and dense. Sparseness and trends in the trace plot of a parameter suggest convergence problems.

Let's use `bayesgraph trace` to obtain trace plots for `{mpg:_cons}` and `{var}`. We specify `_all` to request both plots at once.

```
. bayesgraph trace _all
```

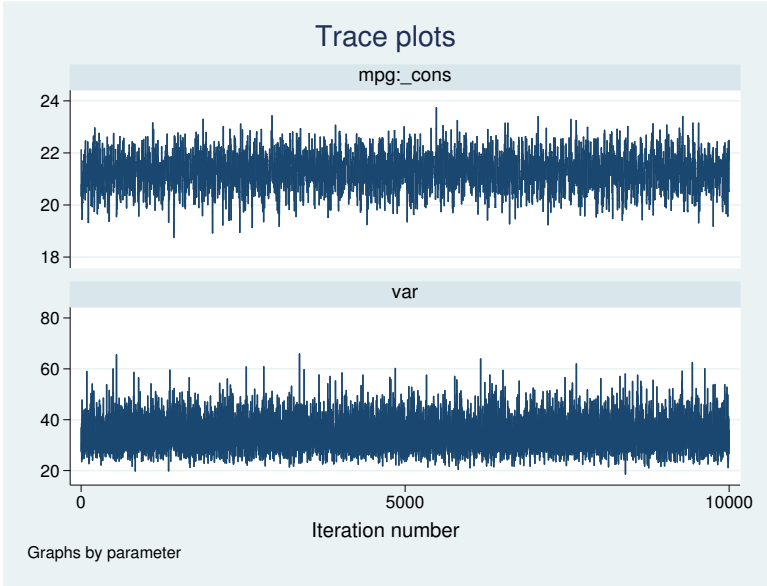




The mean parameter mixes very well and the variance parameter mixes perfectly.

Alternatively, we can use the `byparm()` option to plot results on one graph.

```
. bayesgraph trace _all, byparm
```



`bayesgraph trace` (as well as `bayesgraph cusum`) with option `byparm()` displays multiple plots in one column to accommodate an x axis with many iterations. You can specify `byparm(norowcoldefault)` to switch to the default behavior of options `rows()` and `cols()` documented in [G-3] ***by_option***.

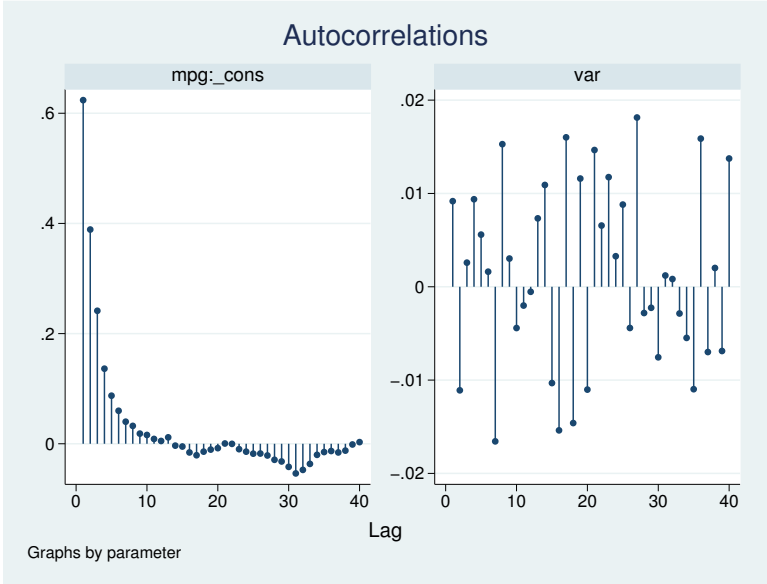
Autocorrelation plots

The second graphical summary we demonstrate is an autocorrelation plot. This plot shows the degree of autocorrelation in an MCMC sample for a range of lags, starting from lag 0. At lag 0, the plotted value corresponds to the sample variance of MCMC.

Autocorrelation is usually present in any MCMC sample. Typically, autocorrelation starts from some positive value for lag 0 and decreases toward 0 as the lag index increases. For a well-mixing MCMC chain, autocorrelation dies off fairly rapidly.

For example, autocorrelation for {mpg:_cons} becomes negligible after about lag 8 and is basically nonexistent for {var}.

```
. bayesgraph ac _all, byparm
```



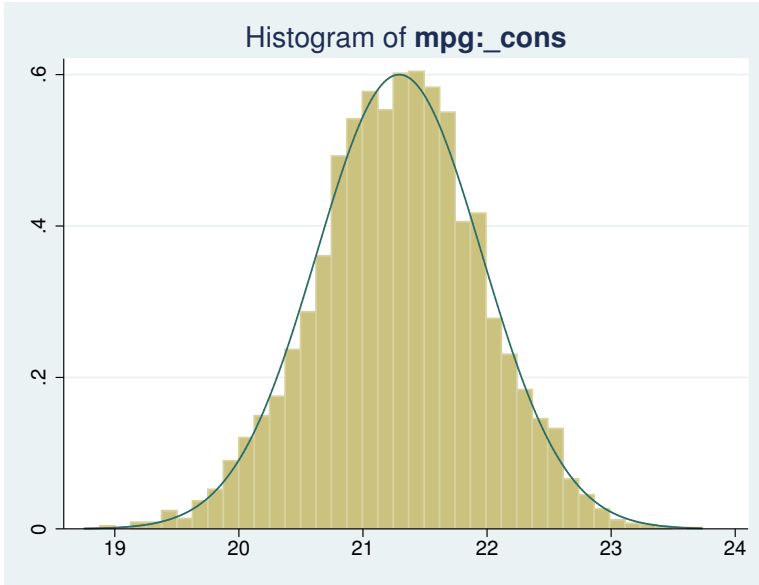
Autocorrelation lags are approximated by correlation times of parameters as reported by the bayesstats ess command; see [BAYES] **bayesstats ess** for details. Autocorrelation lags are also used to determine the batch size for the batch-means estimator of the MCMC standard errors; see [BAYES] **bayesstats summary**.

Histogram plots

Graphical posterior summaries such as histograms and kernel density estimates provide useful additions to the various numerical statistics (see [BAYES] **bayesstats summary**) for summarizing MCMC output. It is always a good practice to inspect the histogram and kernel density estimates of the marginal posterior distributions of parameters to ensure that these empirical distributions behave as expected. These plots can be used to compare the empirical posterior and the specified prior distributions to visualize the impact of the data.

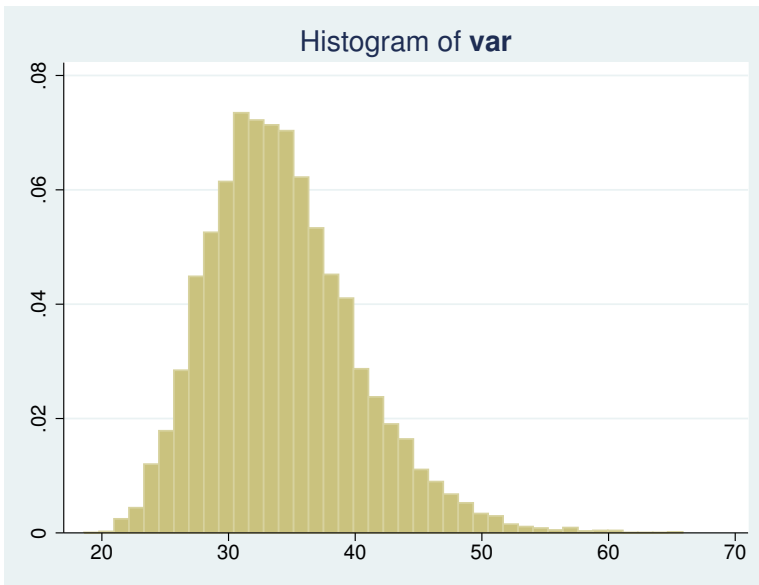
A histogram depicts the general shape of the marginal posterior distribution of a model parameter. Let's look at histograms of our parameters.

```
. bayesgraph histogram {mpg:_cons}, normal
```



The distribution of {mpg:_cons} is in good agreement with the normal distribution. This is not surprising, because the specified conjugate normal prior implies that the marginal posterior for {mpg:_cons} is a normal distribution. The unimodal histogram is also another confirmation that we have obtained a good simulation of the marginal posterior distribution of {mpg:_cons}.

```
. bayesgraph histogram {var}
```



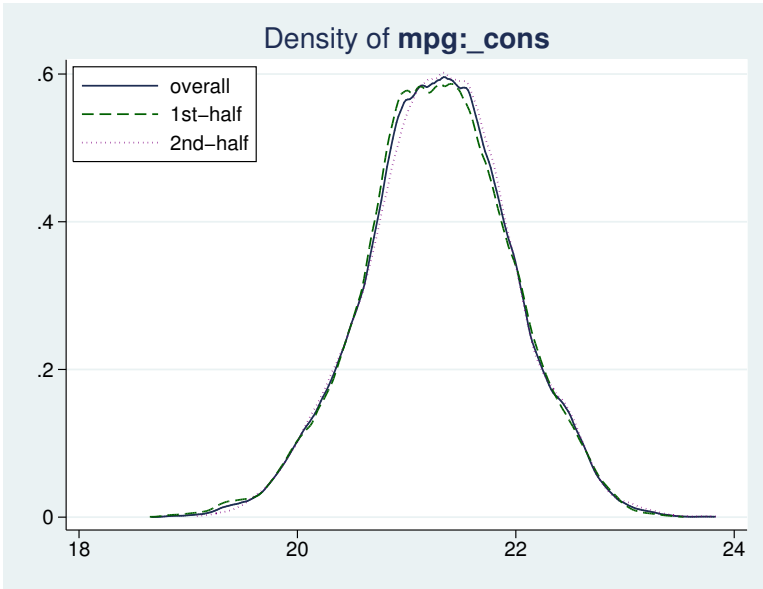
The histogram for {var} is also unimodal but is slightly skewed to the right. This is also in agreement with the specified prior because the marginal posterior for the variance is inverse gamma for the specified model.

Kernel density plots

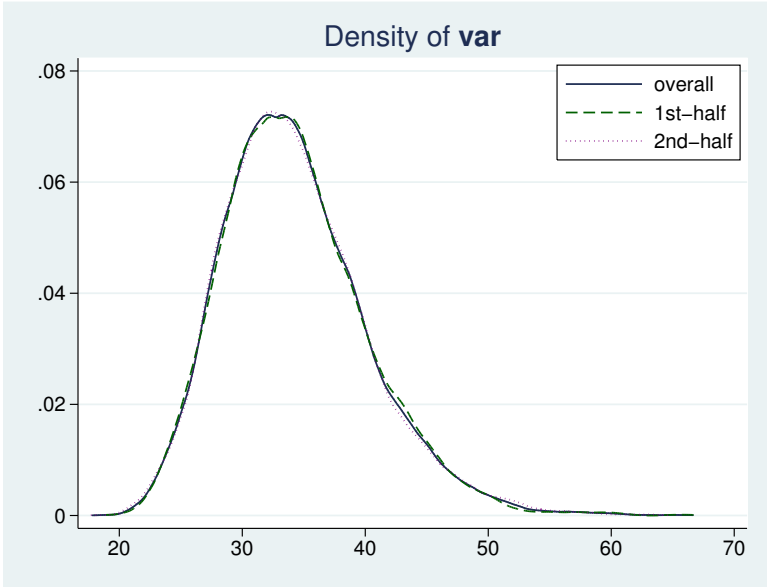
Kernel density plots provide alternative visualizations of the simulated marginal posterior distributions. They may be viewed as smoothed histograms. By default, the bayesgraph kdensity command shows three density curves: an overall density of the entire MCMC sample, the first-half density obtained using the first half of the MCMC sample, and the second-half density obtained using the second half of the MCMC sample. If the chain has converged and mixes well, we expect these three density curves to be close to each other. Large discrepancies between the first-half curve and the second-half curve suggest convergence problems.

Let's look at kernel density plots for our two parameters.

```
. bayesgraph kdensity {mpg:_cons}
```



```
. bayesgraph kdensity {var}
```



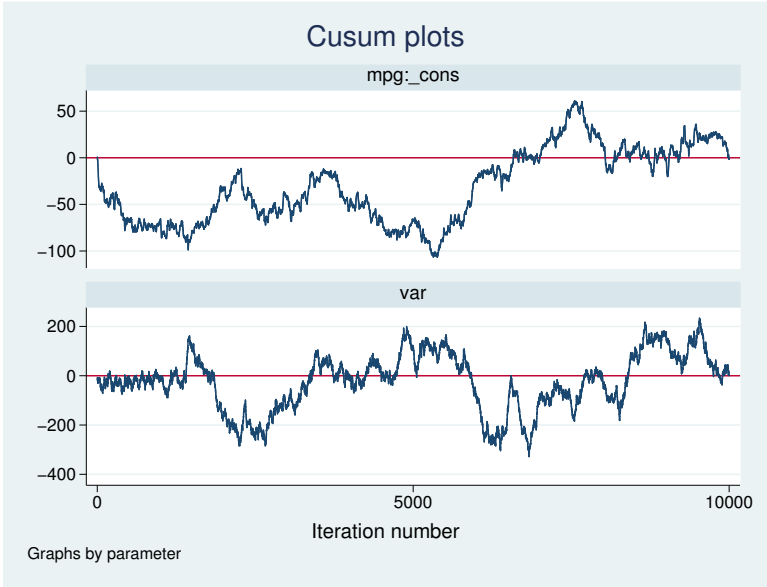
Kernel density plots for `{mpg:_cons}` and `{var}` are similar in shape to the histograms' plots from the previous section. All three density curves are close to each other for both parameters.

Cumulative sum plots

Cumulative sum (cusum) plots are useful graphical summaries for detecting persistent trends in MCMC chains. All cusum plots start and end at 0 and may or may not cross the x axis. There is great variability in the looks of cusum plots, which make them difficult to interpret sometimes. Typically, if the cusum line never crosses the x axis, this may indicate a problem. See, for example, *Convergence diagnostics of* MCMC in [BAYES] **intro** for a cusum plot demonstrating convergence problems.

By inspecting a cusum plot, we may detect an early drift in the simulated sample because of an insufficient burn-in period. In cases of pronounced persistent trends, the cusum curve may stay either in the positive or in the negative y plane. For a well-mixing parameter, the cusum curve typically crosses the x axis several times. This is the case for the cusum plots of `{mpg:_cons}` and `{var}`.

```
. bayesgraph cusum _all, byparm
```



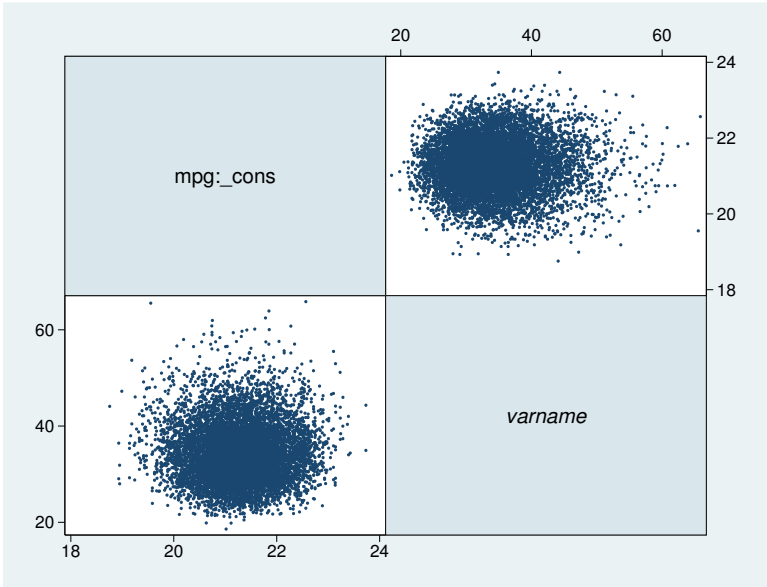
Bivariate scatterplots

The `bayesgraph matrix` command draws bivariate scatterplots of model parameters based on MCMC samples. A bivariate scatterplot represents a joint sample posterior distribution for pairs of parameters. It may reveal correlation between parameters and characterize a general shape of a multivariate posterior distribution. For example, bivariate scatterplots are useful for detecting multimodal posterior distributions.

Typically, scatterplots depict clouds of points. Sparseness and irregularities in the scatterplots can be strong indications of nonconvergence of an MCMC. For a well-mixing chain, the scatterplots have an ellipsoidal form with an increasing concentration around the posterior mode.

This scatterplot of {mpg:_cons} and {var} is an example of a well-behaved scatterplot.

```
. bayesgraph matrix {mpg:_cons} {var}
```

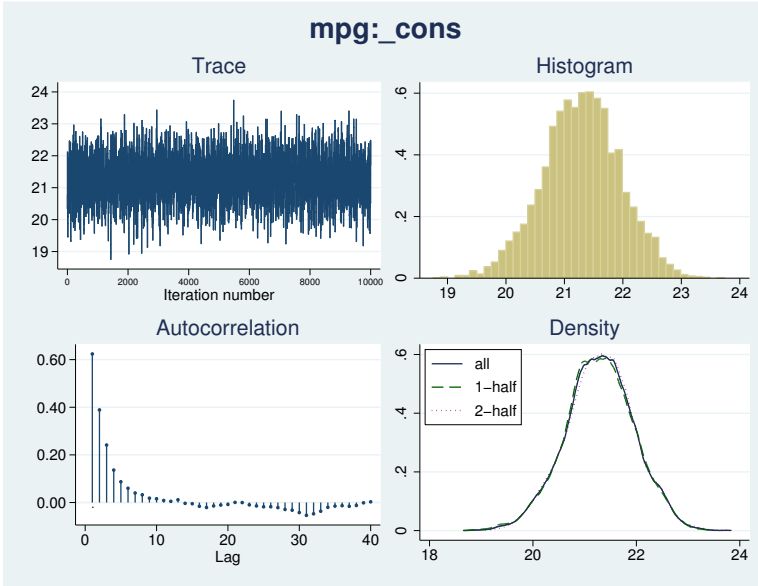


Diagnostic plots

Finally, we demonstrate the bayesgraph diagnostics command, which combines the trace, histogram, autocorrelation, and kernel density plots compactly on one graph. We already discussed the individual plots in the previous sections. Diagnostic plots are convenient for inspecting the overall behavior of a particular model parameter. We recommend that diagnostic plots for all parameters be inspected routinely as a part of the convergence-checking process.

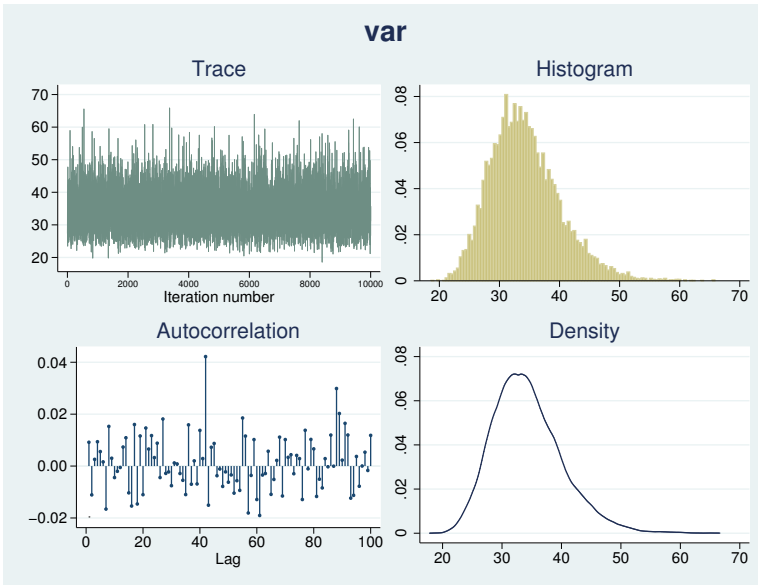
Let's obtain the diagnostic plot for {mpg:_cons}.

```
. bayesgraph diagnostics {mpg:_cons}
```



In the diagnostics plot for {var}, let's also demonstrate the use of several options of the depicted plots.

```
. bayesgraph diagnostics {var}, traceopts(lwidth(0.2) lcolor(teal))
> acopts(lag(100)) histopts(bin(100)) kdensopts(show(none))
```

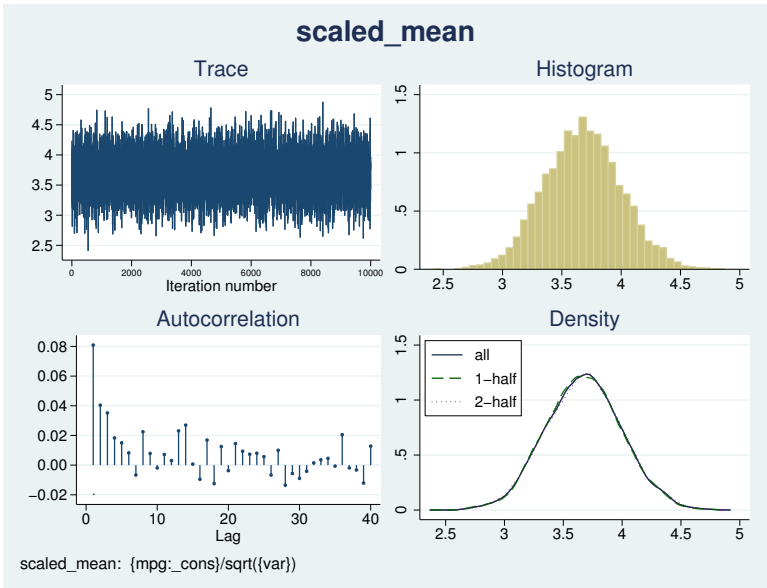


In the above, we changed the width and color of the trace line, the maximum lag for calculating the autocorrelation, the number of bins for the histogram, and requested that the two subsample kernel densities not be shown on the kernel density plot.

Functions of model parameters

All `bayesgraph` subcommands can provide graphical summaries of functions of model parameters. Below we apply `bayesgraph diagnostics` to the expression `{mpg:_cons}/sqrt({var})`, which we label as `scaled_mean`.

```
. bayesgraph diagnostics (scaled_mean: {mpg:_cons}/sqrt({var}))
```



If you detect convergence problems in a function of parameters, you must inspect every parameter used in the expression individually. In fact, we recommend that you inspect all model parameters before you proceed with any postestimation analysis.

Methods and formulas

Let θ be a scalar model parameter and $\{\theta_t\}_{t=1}^{T}$ be an MCMC sample of size T drawn from the marginal posterior distribution of θ.

The trace plot of θ plots θ_t against t with connecting lines for $t = 1, \ldots, T$.

The autocorrelation plot of θ shows the autocorrelation in the $\{\theta_t\}_{t=1}^{T}$ sample for lags from 0 to the `lag(#)` option of the `ac` command.

The histogram and kernel density plots of θ are drawn using the `histogram` and `kdensity` commands.

Yu and Mykland (1998) proposed a graphical procedure for assessing the convergence of individual parameters based on cumulative sums, also known as a cusum plot. The cusum plot for θ plots S_t against t for $t = 1, \ldots, T$ and connects the successive points. S_t is the cumulative sum at time t:

$$S_t = \sum_{k=1}^{t}(\theta_k - \widehat{\theta}), \quad \widehat{\theta} = \frac{1}{T}\sum_{k=1}^{T}\theta_k$$

and $S_0 = 0$.

The scatterplot of two model parameters θ^1 and θ^2 plots points (θ_t^1, θ_t^2) for $t = 1, \ldots, T$.

Reference

Yu, B., and P. Mykland. 1998. Looking at Markov samplers through cusum path plots: A simple diagnostic idea. *Statistics and Computing* 8: 275–286.

Also see

[BAYES] **bayesmh** — Bayesian regression using Metropolis–Hastings algorithm

[BAYES] **bayesmh postestimation** — Postestimation tools for bayesmh

[BAYES] **bayesstats ess** — Effective sample sizes and related statistics

[BAYES] **bayesstats summary** — Bayesian summary statistics

[G-2] **graph matrix** — Matrix graphs

[G-2] **graph twoway kdensity** — Kernel density plots

[R] **histogram** — Histograms for continuous and categorical variables

[R] **kdensity** — Univariate kernel density estimation

[TS] **corrgram** — Tabulate and graph autocorrelations

[TS] **tsline** — Plot time-series data

Title

bayesstats — Bayesian statistics after bayesmh

Description Also see

Description

The following subcommands are available with `bayesstats` after `bayesmh`:

Command	Description
`bayesstats ess`	effective sample sizes and related statistics
`bayesstats summary`	Bayesian summary statistics for model parameters and their functions
`bayesstats ic`	Bayesian information criteria and Bayes factors

Also see

[BAYES] **bayesmh postestimation** — Postestimation tools for bayesmh

[BAYES] **bayesstats ess** — Effective sample sizes and related statistics

[BAYES] **bayesstats summary** — Bayesian summary statistics

[BAYES] **bayesstats ic** — Bayesian information criteria and Bayes factors

Title

bayesstats ess — Effective sample sizes and related statistics

Description

`bayesstats ess` calculates effective sample sizes (ESS), correlation times, and efficiencies for model parameters and functions of model parameters using current estimation results from the `bayesmh` command.

Quick start

Effective sample sizes for all model parameters after a Bayesian regression model

```
bayesstats ess
```

As above, but only for model parameters `{y:x1}` and `{var}`

```
bayesstats ess {y:x1} {var}
```

As above, but skip every 5 observations from the full MCMC sample

```
bayesstats ess  {y:x1} {var}, skip(5)
```

Effective sample sizes for functions of scalar model parameters

```
bayesstats ess ({y:x1}-{y:_cons}) (sd:sqrt({var}))
```

As above, and include `{y:x1}` and `{var}`

```
bayesstats ess {y:x1} {var} ({y:x1}-{y:_cons}) (sd:sqrt({var}))
```

Menu

Statistics > Bayesian analysis > Effective sample sizes

Syntax

Statistics for all model parameters

```
bayesstats ess [ , options ]

bayesstats ess _all [ , options ]
```

Statistics for selected model parameters

```
bayesstats ess paramspec [ , options ]
```

Statistics for functions of model parameters

```
bayesstats ess exprspec [ , options ]
```

Full syntax

```
bayesstats ess spec [ spec ... ] [ , options ]
```

paramspec can be one of the following:

{*eqname*:*param*} refers to a parameter *param* with equation name *eqname*;

{*eqname*:} refers to all model parameters with equation name *eqname*;

{*eqname*:*paramlist*} refers to parameters with names in *paramlist* and with equation name *eqname*; or

{*param*} refers to all parameters named *param* from all equations.

In the above, *param* can refer to a matrix name, in which case it will imply all elements of this matrix. See *Different ways of specifying model parameters* in [BAYES] **bayesmh postestimation** for examples.

exprspec is an optionally labeled expression of model parameters specified in parentheses:

([*exprlabel*:] *expr*)

exprlabel is a valid Stata name, and *expr* is a scalar expression that may not contain matrix model parameters. See *Specifying functions of model parameters* in [BAYES] **bayesmh postestimation** for examples.

spec is one of *paramspec* or *exprspec*.

options	Description
Main	
`skip(#)`	skip every # observations from the MCMC sample; default is `skip(0)`
`nolegend`	suppress table legend
display_options	control spacing, line width, and base and empty cells
Advanced	
`corrlag(#)`	specify maximum autocorrelation lag; default varies
`corrtol(#)`	specify autocorrelation tolerance; default is `corrtol(0.01)`

Options

Main

`skip(#)` specifies that every # observations from the MCMC sample not be used for computation. The default is `skip(0)` or to use all observations in the MCMC sample. Option `skip()` can be used to subsample or thin the chain. `skip(#)` is equivalent to a thinning interval of #+1. For example, if you specify `skip(1)`, corresponding to the thinning interval of 2, the command will skip every other observation in the sample and will use only observations 1, 3, 5, and so on in the computation. If you specify `skip(2)`, corresponding to the thinning interval of 3, the command will skip every 2 observations in the sample and will use only observations 1, 4, 7, and so on in the computation. `skip()` does not thin the chain in the sense of physically removing observations from the sample, as is done by `bayesmh`'s `thinning()` option. It only discards selected observations from the computation and leaves the original sample unmodified.

`nolegend` suppresses the display of the table legend. The table legend identifies the rows of the table with the expressions they represent.

display_options: `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fvwrap(#)`, `fvwrapon(`*style*`)`, and `nolstretch`; see [R] **estimation options**.

Advanced

`corrlag(#)` specifies the maximum autocorrelation lag used for calculating effective sample sizes. The default is $\min\{500, \texttt{mcmcsize()}/2\}$. The total autocorrelation is computed as the sum of all lag-k autocorrelation values for k from 0 to either `corrlag()` or the index at which the autocorrelation becomes less than `corrtol()` if the latter is less than `corrlag()`.

`corrtol(#)` specifies the autocorrelation tolerance used for calculating effective sample sizes. The default is `corrtol(0.01)`. For a given model parameter, if the absolute value of the lag-k autocorrelation is less than `corrtol()`, then all autocorrelation lags beyond the kth lag are discarded.

Remarks and examples

Remarks are presented under the following headings:

Effective sample size and MCMC sampling efficiency
Using bayesstats ess

Effective sample size and MCMC sampling efficiency

It is well known that for a random sample of T independent subjects, the standard error of the sample mean estimator is proportional to $1/\sqrt{T}$. In Bayesian inference, it is of interest to estimate the standard error of the posterior mean estimator. The posterior mean of a parameter of interest is typically estimated as a sample mean from an MCMC sample obtained from the marginal posterior distribution of the parameter of interest. Observations from an MCMC sample are not independent and are usually positively correlated, which must be taken into account when computing the standard error. Thus the standard error of the posterior mean estimator is proportional to $1/\sqrt{\text{ESS}}$, where ESS is the effective sample size for the parameter of interest. Typically, ESS is less than T, the total number of observations in the MCMC sample. We can thus interpret the posterior mean estimate as a sample mean estimate from an independent sample of size ESS. In other words, the effective sample size is an estimate of the number of independent observations that the MCMC chain represents. We say that MCMC samples with higher ESS are more efficient.

Effective sample size is directly related to the convergence properties of an MCMC sample—very low ESS relative to T suggests nonconvergence. In the extreme case of a perfectly correlated MCMC observation, ESS is 1. It is thus a standard practice to assess the quality of an MCMC sample by inspecting ESS values for all involved model parameters. Note, however, that high ESS values are not generally sufficient for declaring convergence of MCMC because pseudoconvergence, which may occur when MCMC does not explore the entire distribution, may also lead to high ESS values.

Using bayesstats ess

`bayesstats ess` reports effective sample sizes, correlation times, and efficiencies for model parameters and their functions using the current estimation results from the `bayesmh` command. When typed without arguments, the command displays results for all model parameters. Alternatively, you can specify a subset of model parameters following the command name; see *Different ways of specifying model parameters* in [BAYES] **bayesmh postestimation**. You can also obtain results for scalar functions of model parameters; see *Specifying functions of model parameters* in [BAYES] **bayesmh postestimation**.

Consider our analysis of `auto.dta` from example 4 in [BAYES] **bayesmh** using the mean-only normal model for `mpg` with a noninformative prior.

```
. use http://www.stata-press.com/data/r14/auto
(1978 Automobile Data)
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, flat) prior({var}, jeffreys)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
Priors:
  {mpg:_cons} ~ 1 (flat)
        {var} ~ jeffreys
------------------------------------------------------------------------------
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2668
                                                Efficiency:  min =     .09718
                                                             avg =      .1021
Log marginal likelihood =   -234.645                         max =      .1071
------------------------------------------------------------------------------
             |                                                Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
       _cons |   21.29222   .6828864   .021906   21.27898   19.99152   22.61904
-------------+----------------------------------------------------------------
         var |   34.76572    5.91534   .180754   34.18391    24.9129   47.61286
------------------------------------------------------------------------------
```

▷ Example 1: Effective sample sizes for all parameters

To compute effective sample sizes and other related statistics for all model parameters, we type `bayesstats ess` without arguments after the `bayesmh` command.

```
. bayesstats ess
Efficiency summaries     MCMC sample size =     10,000
```

	ESS	Corr. time	Efficiency
mpg			
_cons	971.82	10.29	0.0972
var	1070.99	9.34	0.1071

The closer the ESS estimates are to the MCMC sample size, the better. Also, the lower the correlation times are and the higher the efficiencies are, the better. ESS estimates can be interpreted as follows. In a sample of 10,000 MCMC observations, we have only about 972 independent observations to obtain estimates for `{mpg:_cons}` and only about 1,071 independent observations to obtain estimates for `{var}`. Correlation times are reciprocal of efficiencies. You can interpret them as an estimated lag after which autocorrelation in an MCMC sample is small. In our example, the estimated lag is roughly 10 for both parameters. In general, efficiencies above 10% are considered good for the MH algorithm. In our example, they are about 12% for both parameters.

Alternatively, we could have listed all parameters manually:

```
. bayesstats ess {mpg:_cons} {var}
  (output omitted)
```

◁

▷ Example 2: Effective sample sizes for functions of model parameters

Similarly to other `bayesmh` postestimation commands, `bayesstats ess` accepts expressions to compute results for functions of model parameters. For example, we can use expression `(sd:sqrt({var}))` with a label, `sd`, to compute effective sample sizes for the standard deviation of `mpg` in addition to the variance.

```
. bayesstats ess (sd:sqrt({var})) {var}
Efficiency summaries     MCMC sample size =     10,000
          sd : sqrt({var})
```

	ESS	Corr. time	Efficiency
sd	1093.85	9.14	0.1094
var	1070.99	9.34	0.1071

ESS and efficiency are higher for the standard deviation than for the variance, which means that we need slightly more iterations to estimate `{var}` with the same precision as `sd`.

If we wanted, we could have suppressed the `sd` legend in the output above by specifying the `nolegend` option.

◁

Stored results

`bayesstats ess` stores the following in `r()`:

Scalars

`r(skip)`	number of MCMC observations to skip in the computation; every `r(skip)` observations are skipped
`r(corrlag)`	maximum autocorrelation lag
`r(corrtol)`	autocorrelation tolerance

Macros

`r(expr_#)`	#th expression
`r(names)`	names of model parameters and expressions
`r(exprnames)`	expression labels

Matrices

`r(ess)`	matrix with effective sample sizes, correlation times, and efficiencies for parameters in `r(names)`

Methods and formulas

Let θ be a scalar model parameter and $\{\theta_t\}_{t=1}^{T}$ be an MCMC sample of size T drawn from the marginal posterior distribution of θ. The effective sample size of the MCMC sample of θ is given by

$$\text{ESS} = T/(1 + 2 \sum_{k=1}^{\max\_\text{lags}} \rho_k)$$

where $\rho_k = \gamma_k/\gamma_0$ is the lag-k autocorrelation of the MCMC sample, and $\max\_\text{lags}$ is the maximum number less than or equal to ρ_{lag} such that for all $k = 1, \ldots, \max\_\text{lags}$, $|\rho_k| > \rho_{\text{tol}}$, where ρ_{lag} and ρ_{tol} are specified in options `corrlag()` and `corrtol()` with the respective default values of 500 and 0.01.

The lag-k autocorrelation is $\rho_k = \gamma_k/\gamma_0$, where

$$\gamma_k = \frac{1}{T} \sum_{t=1}^{T-k} (\theta_t - \widehat{\theta})(\theta_{t+k} - \widehat{\theta})$$

is the empirical autocovariance of lag k, and γ_0 simplifies to the sample variance. $\widehat{\theta}$ is the posterior mean estimator.

Correlation time is defined as T/ESS, and efficiency is defined as the reciprocal of the correlation time, ESS/T. Because ESS is between 0 and T, inclusively, the efficiency is always between 0 and 1.

Also see

[BAYES] **bayesmh** — Bayesian regression using Metropolis–Hastings algorithm

[BAYES] **bayesmh postestimation** — Postestimation tools for bayesmh

[BAYES] **bayesstats summary** — Bayesian summary statistics

Title

bayesstats ic — Bayesian information criteria and Bayes factors

Description

`bayesstats ic` calculates and reports model-selection statistics, including the deviance information criterion (DIC), log marginal-likelihood, and Bayes factors (BFs), using current estimation results from the `bayesmh` command. BFs can be displayed in the original metric or in the log metric. The command also provides two different methods to approximate marginal likelihood.

Quick start

Information criteria for previously saved estimation results `A` and `B` with `A` used as the base model by default

```
bayesstats ic A B
```

As above, but use `B` as the base model instead of `A`

```
bayesstats ic A B, basemodel(B)
```

Report BFs instead of the default log BFs

```
bayesstats ic A B, bayesfactor
```

Menu

Statistics > Bayesian analysis > Information criteria

Syntax

```
bayesstats ic [namelist] [, options]
```

namelist is a name, a list of names, `_all`, or `*`. A name may be `.`, meaning the current (active) estimates. `_all` and `*` mean the same thing.

options	Description
Main	
`basemodel(`*name*`)`	specify a base or reference model; default is the first-listed model
`bayesfactor`	report BFs instead of the default log BFs
Advanced	
`marglmethod(`*method*`)`	specify marginal-likelihood approximation method; default is to use Laplace–Metropolis approximation, `lmetropolis`; rarely used

method	Description
`lmetropolis`	Laplace–Metropolis approximation; the default
`hmean`	harmonic-mean approximation

Options

Main

`basemodel(`*name*`)` specifies the name of the model to be used as a base or reference model when computing BFs. By default, the first-listed model is used as a base model.

`bayesfactor` specifies that BFs be reported instead of the default log BFs.

Advanced

`marglmethod(`*method*`)` specifies a method for approximating the marginal likelihood. *method* is either `lmetropolis`, the default, for Laplace–Metropolis approximation or `hmean` for harmonic-mean approximation. This option is rarely used.

Remarks and examples

Remarks are presented under the following headings:

Bayesian information criteria
Bayes factors
Using bayesstats ic

Bayesian information criteria

Bayesian information criteria are used for selecting a model among a set of candidate models that best fits the data. Likelihood-based inference is known to be prone to overfitting the data. Indeed, it is often possible to increase the likelihood by simply including more parameters in a model. Bayesian information criteria address this problem by applying a penalty proportional to the complexity of the models to the likelihood.

Consider a finite set of Bayesian models $M_1, \ldots, M_r$, which we want to compare with a base model M_b. All models M_js are fit to the same dataset but may differ in their likelihood or prior specification.

Three commonly used information criteria are Akaike information criterion (AIC), Bayesian information criterion (BIC), and DIC. All three criteria are likelihood based and include a goodness-of-fit term proportional to the negative likelihood of the model and a penalty term proportional to the number of parameters in the model. Models with smaller values of these criteria are preferable.

The BIC, originally derived for the exponential family of distributions, is based on the assumption that the model has a flat, noninformative prior. In frequentist statistics, BIC is widely used as a variable-selection criterion, particularly in linear regression. In BIC, the penalty term is a product of the number of parameters in the model and the log of the sample size. The penalty of BIC thus increases not only with the number of parameters but also with the sample size. In the AIC, the penalty term is two times the number of parameters and does not depend on the sample size. As a result, BIC is more conservative than AIC and prefers simpler models. DIC is similar to AIC, but its penalty term is based on a complexity term that measures the difference between the expected log likelihood and the log likelihood at the posterior mean point. DIC is designed specifically for Bayesian estimation that involves MCMC simulations.

The limitation of all three criteria is that they either ignore prior distributions or assume that prior distributions are noninformative. They are thus not well suited for Bayesian sensitivity analysis, when models with the same parameters but different priors are being compared.

The `bayesstats ic` command reports DIC. See [R] **estat ic** after the corresponding maximum likelihood estimation command for values of AIC and BIC.

Bayes factors

In Bayesian inference, BFs are preferred to model-selection criteria because, unlike BIC, AIC, and DIC, they incorporate the information about model priors. Taking into account prior information is essential for Bayesian sensitivity analysis, when models with the same parameters but different priors are being compared.

The BF of two models is just the ratio of their marginal likelihoods calculated using the same dataset. Unlike BIC, AIC, and DIC, BFs include all information about the specified Bayesian model. Thus BFs are not applicable to models with improper priors, whereas BIC, AIC, and DIC are still applicable because they ignore prior information. BFs, however, are often difficult to compute reliably because of the difficulty in computing marginal likelihoods.

BFs also require that posterior distributions be completely specified, including the normalizing constants. The latter is especially important in Bayesian estimation using MCMC simulations, when the normalizing constants are often omitted from the specification of a posterior distribution. The `bayesmh` command always simulates from a complete posterior distribution when you select one of the supported Bayesian models, but you need to make sure to include all normalizing constants with your posterior distribution when you are programming your own Bayesian model (see [BAYES] **bayesmh evaluators**) and would like to use BFs during postestimation.

Let BF_{jb}, $j = 1, \ldots, r$, be the BF of model M_j with respect to the base model M_b. All models M_j are fit to the same dataset; otherwise, BFs are meaningless. The `bayesstats ic` command calculates BF_{jb}'s and reports them in absolute metric or in log metric when the `bayesfactor` option is specified.

Jeffreys (1961) proposes the following interpretation of the values of BF_{jb} based on half-units of the log metric:

$\log_{10}(\text{BF}_{jb})$	BF_{jb}	Evidence against M_b
0 to 1/2	1 to 3.2	Bare mention
1/2 to 1	3.2 to 10	Substantial
1 to 2	10 to 100	Strong
>2	>100	Decisive

Kass and Raftery (1995) suggest using twice the natural logarithm of the BF to make it have the same scale as the DIC and likelihood-ratio test statistic. They suggest the following interpretation table:

$2\log_e(\text{BF}_{jb})$	BF_{jb}	Evidence against M_b
0 to 2	1 to 3	Bare mention
2 to 6	3 to 20	Positive
6 to 10	20 to 150	Strong
>10	>150	Very strong

Typically, the worst-fitting model is chosen as a base model. If the base model happens to be better than the comparison model, the corresponding BF will be negative. In this case, you can apply results above to the absolute value of the BF.

BFs compute relative probabilities of how well each model fits the data compared with the base model. Being relative quantities, BFs cannot be used to measure goodness of fit of a particular model unless one assumes that the base model fits the data well. Some researchers view this as a limitation of BFs (Gelman et al. 2014). Kass and Raftery (1995), on the other hand, show that BFs can be viewed as differences between predictive scores and thus can be used to measure success of different models at predicting the data.

BFs have several advantages over the more traditional, frequentist testing methods. For example, they do not have the limitation of the p-value approach to systematically reject the null hypothesis in large samples. BFs are also suitable for comparing both nonnested and nested models. Also see *Comparing Bayesian models* in [BAYES] **intro** for more information about Bayesian model comparison.

A key element in computing BFs is calculating the marginal likelihood. Except for some rare cases, marginal likelihood does not have a closed form and needs to be approximated. A detailed review of different approximation methods is given by Kass and Raftery (1995). The default method implemented in `bayesstats ic` (and `bayesmh`) is the Laplace–Metropolis approximation (Lewis and Raftery 1997). The harmonic-mean approximation of the marginal likelihood is also available via the `marglmethod(hmean)` option, but we recommend that you use the default method. See *Methods and formulas* in [BAYES] **bayesmh** for technical details.

Using bayesstats ic

▷ Example 1

The `bayesstats ic` command provides several model-selection statistics that can be used to compare models. To illustrate the use of `bayesstats ic`, we consider `auto.dta`. We model the fuel-efficiency variable `mpg` using a normal distribution with fixed variance but unknown, random mean. There is only one random parameter in this model—`{mpg:_cons}`. We compare the models with three different prior distributions to find the best one among them. We fit the three models using `bayesmh` and save the corresponding estimation results as `uniform1`, `uniform2`, and `normal`.

First, for comparison purposes, let's obtain the maximum likelihood estimate (MLE) of the mean of `mpg`, which is simply the sample mean in our example:

```
. use http://www.stata-press.com/data/r14/auto
(1978 Automobile Data)
. summarize mpg
```

Variable	Obs	Mean	Std. Dev.	Min	Max
mpg	74	21.2973	5.785503	12	41

The sample mean of `mpg` is roughly 21.3.

Next, we use `bayesmh` to fit our first model of interest. We fix the variance of the normal distribution to 30, which is close to the estimated variance of `mpg` of $5.79^2 = 33.52$.

```
. set seed 14
. bayesmh mpg, likelihood(normal(30))
> prior({mpg:_cons}, uniform(-10, 10))
> initial({mpg:_cons} 2) saving(uniform1_simdata)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},30)
Prior:
  {mpg:_cons} ~ uniform(-10,10)
------------------------------------------------------------------------------
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .4102
Log marginal likelihood = -397.42978            Efficiency       =     .08018
------------------------------------------------------------------------------
                                                              Equal-tailed
         mpg |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
       _cons |  9.965511   .0342812   .001211   9.975729   9.871825   9.998796
------------------------------------------------------------------------------
file uniform1_simdata.dta saved
. estimates store uniform1
```

In the first model, we deliberately chose a prior for `{mpg:_cons}`, `uniform(-10,10)`, that does not include the value of the sample mean. We thus expect this model to fit poorly. Because of the restricted domain of the specified uniform prior, we also needed to specify an initial value for `{mpg:_cons}` for MCMC to start from a point of positive posterior probability.

We also specified the `saving()` option to save the MCMC simulation dataset so that we could use `estimates store` to store our estimation results for future use. See *Storing estimation results after bayesmh* in [BAYES] **bayesmh postestimation** for details.

```
. set seed 14
. bayesmh mpg, likelihood(normal(30))
> prior({mpg:_cons}, uniform(10, 30))
> initial({mpg:_cons} 20) saving(uniform2_simdata)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},30)
Prior:
  {mpg:_cons} ~ uniform(10,30)
------------------------------------------------------------------------------

Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .4272
Log marginal likelihood = -237.08583            Efficiency       =      .2414

------------------------------------------------------------------------------
             |                                                 Equal-tailed
         mpg |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
       _cons |  21.31085   .6447073   .013123   21.31485   20.06381   22.57936
------------------------------------------------------------------------------
file uniform2_simdata.dta saved
. estimates store uniform2
```

In the second model, we used a uniform prior that included the value of the sample mean in its domain.

```
. set seed 14
. bayesmh mpg, likelihood(normal(30))
> prior({mpg:_cons}, normal(30)) saving(normal_simdata)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},30)
Prior:
  {mpg:_cons} ~ normal(30)
------------------------------------------------------------------------------

Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .4295
Log marginal likelihood = -244.16624            Efficiency       =      .2319

------------------------------------------------------------------------------
             |                                                 Equal-tailed
         mpg |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
       _cons |   21.01901   .6461194   .013417   21.01596   19.76637    22.3019
------------------------------------------------------------------------------

file normal_simdata.dta saved
. estimates store normal
```

In the third model, we used a normal prior with a variance fixed at 30. Note that we did not need to specify an initial value for `{mpg:_cons}` in this model, because the domain of the normal distribution is the whole real line.

Both the `uniform2` and `normal` models yield estimates close to the MLE of 21.3. According to their credible intervals, the domain of the posterior distribution of `{mpg:_cons}` is concentrated around MLE. For example, the 95% credible interval for the `uniform2` model is [20.06, 22.60].

Now, let's use `bayesstats ic` to compare the three models. We list all the models following the command name and use the `normal` model as a reference model.

```
. bayesstats ic uniform1 uniform2 normal, basemodel(normal)
Bayesian information criteria
------------------------------------------------------------------------------
             |       DIC     log(ML)      log(BF)
-------------+----------------------------------------------------------------
    uniform1 |  785.8891  -397.4298  -153.2635
    uniform2 |  471.1909  -237.0858   7.080404
      normal |  471.3905  -244.1662          .
------------------------------------------------------------------------------
Note: Marginal likelihood (ML) is computed
      using Laplace-Metropolis approximation.
```

The `uniform1` model performs worse than the other two models according to the log marginal-likelihood, `log(ML)`, and DIC—the `DIC` value is much larger, and the `log(ML)` value is much smaller for the `uniform1` model. The other two models have only slightly different values for `DIC` and `log(ML)`, according to which the `uniform2` model is preferable.

Although the `uniform2` and `normal` models have different prior distributions, they have almost identical posterior domain, that is, the range of values of `{mpg:_cons}` where the posterior is strictly positive. As such, they will have the same values for AIC and BIC, and we will not be able to discriminate between the two models based on these information criteria.

The most decisive factor between the `uniform2` and `normal` models is the BF. The value of log BF, `log(BF)`, is 7.08, which provides very strong evidence in favor of the `uniform2` model.

We thus conclude that `uniform2` is the best model among the three considered models. This may be explained by the fact that the specified `uniform(10,30)` prior is in more agreement with the likelihood of the data than the specified `normal(0,30)` prior.

After your analysis, remember to erase the saved simulation datasets you no longer need. For example, we erase all of them by typing

```
. erase uniform1_simdata.dta
. erase uniform2_simdata.dta
. erase normal_simdata.dta
```

◁

Stored results

`bayesstats ic` stores the following in `r()`:

Scalars

`r(bayesfactor)`	`1` if `bayesfactor` is specified; `0` otherwise

Macros

`r(names)`	names of estimation results used
`r(basemodel)`	name of the base or reference model
`r(marglmethod)`	method for approximating marginal likelihood: `lmetropolis` or `hmean`

Matrices

`r(ic)`	matrix reporting `DIC`, `log(ML)`, and `log(BF)` or `BF` if `bayesfactor` is used

Methods and formulas

DIC was introduced by Spiegelhalter et al. (2002) for Bayesian model selection using MCMC simulations. DIC is based on the deviance statistics

$$D(\boldsymbol{\theta}) = -2\left\{\log f(\mathbf{y};\boldsymbol{\theta}) - \log f^*(\mathbf{y};\boldsymbol{\theta}^*)\right\}$$

where $f(\cdot;\cdot)$ is the likelihood function of the model and $f^*(\mathbf{y};\boldsymbol{\theta}^*)$ is the likelihood of the full model that fits data perfectly. Because $f^*(\mathbf{y};\boldsymbol{\theta}^*)$ is constant across models fit to the same data, it is ignored in the actual calculation of DIC. Given an MCMC sample $\{\boldsymbol{\theta}_t\}_{t=1}^T$, the expected deviance can be estimated by the sample average $\overline{D}(\boldsymbol{\theta}) = 1/T\sum_{t=1}^T D(\boldsymbol{\theta}_t)$. Similarly to AIC and BIC, DIC is a sum of two components: the goodness-of-fit term $\overline{D}(\boldsymbol{\theta})$ and the model complexity term p_D: DIC $= \overline{D}(\boldsymbol{\theta}) + p_D$. The complexity is defined as the difference between the expected deviance and the deviance at the sample posterior mean: $p_D = \overline{D}(\boldsymbol{\theta}) - D(\overline{\boldsymbol{\theta}})$. We thus have

$$\text{DIC} = D(\overline{\boldsymbol{\theta}}) + 2p_D$$

Models with smaller values of DIC are preferred to models with larger values of DIC.

BFs were introduced by Jeffreys (1961). The BF of two models, M_1 and M_2, is given by

$$\text{BF}_{12} = \frac{P(\mathbf{y}|M_1)}{P(\mathbf{y}|M_2)} = \frac{m_1(\mathbf{y})}{m_2(\mathbf{y})}$$

where $m_1(\cdot)$ and $m_2(\cdot)$ are the corresponding marginal likelihoods associated with models M_1 and M_2. (See *Methods and formulas* in [BAYES] **bayesmh** for details about computing marginal likelihood.) BFs are defined only for proper marginal densities. Comparing models with improper priors is allowed as long as the resulting marginal densities are proper. The methodological importance of BFs comes from the fact that the so-called posterior odds is a product of prior odds and BF:

$$\frac{P(M_1|\mathbf{y})}{P(M_2|\mathbf{y})} = \frac{P(M_1)}{P(M_2)} \times \text{BF}_{12}$$

Therefore, if we assume that M_1 and M_2 are equally probable a priori, the posterior odds will be equal to the BF. We thus prefer model M_1 if $\text{BF}_{12} > 1$ and model M_2 otherwise. In practice, because of the higher numerical stability, we often calculate BFs in the (natural) log metric and compare its value against 0.

$$\log\text{BF}_{12} = \log m_1(\mathbf{y}) - \log m_2(\mathbf{y})$$

References

Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. 2014. *Bayesian Data Analysis*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.

Jeffreys, H. 1961. *Theory of Probability*. 3rd ed. Oxford: Oxford University Press.

Kass, R. E., and A. E. Raftery. 1995. Bayes factors. *Journal of the American Statistical Association* 90: 773–795.

Lewis, S. M., and A. E. Raftery. 1997. Estimating Bayes factors via posterior simulation with the Laplace–Metropolis estimator. *Journal of the American Statistical Association* 92: 648–655.

Spiegelhalter, D. J., N. G. Best, B. P. Carlin, and A. Van Der Linde. 2002. Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society, Series B* 64: 583–639.

Also see

[BAYES] **bayesmh** — Bayesian regression using Metropolis–Hastings algorithm

[BAYES] **bayesmh postestimation** — Postestimation tools for bayesmh

[BAYES] **bayestest model** — Hypothesis testing using model posterior probabilities

[R] **estimates** — Save and manipulate estimation results

Title

bayesstats summary — Bayesian summary statistics

Description

`bayesstats summary` calculates and reports posterior summary statistics for model parameters and functions of model parameters using current estimation results from the `bayesmh` command. Posterior summary statistics include posterior means, posterior standard deviations, MCMC standard errors (MCSE), posterior medians, and equal-tailed credible intervals or highest posterior density (HPD) credible intervals.

Quick start

Posterior summaries for all model parameters after a Bayesian regression model

```
bayesstats summary
```

As above, but only for parameters `{y:x1}` and `{y:x2}`

```
bayesstats summary {y:x1} {y:x2}
```

Same as above

```
bayesstats summary {y:x1 x2}
```

Posterior summaries for elements 1,1 and 2,1 of matrix parameter `{S}`

```
bayesstats summary {S_1_1 S_2_1}
```

Posterior summaries for all elements of matrix parameter `{S}`

```
bayesstats summary {S}
```

Posterior summaries with HPD instead of equal-tailed credible intervals and with credible level of 90%

```
bayesstats summary, hpd clevel(90)
```

Posterior summaries with MCSE calculated using batch means

```
bayesstats summary, batch(100)
```

Posterior summaries for functions of scalar model parameters

```
bayesstats summary ({y:x1}-{y:_cons}) (sd:sqrt({var}))
```

Posterior summaries for the log-likelihood and log-posterior functions

```
bayesstats summary _loglikelihood _logposterior
```

Posterior summaries for selected model parameters and functions of model parameters and for log-likelihood and log-posterior functions using abbreviated syntax

```
bayesstats summary {var} ({y:x1}-{y:_cons}) _ll _lp
```

Menu

Statistics > Bayesian analysis > Summary statistics

Syntax

Summary statistics for all model parameters

```
bayesstats summary [ , options ]

bayesstats summary _all [ , options ]
```

Summary statistics for selected model parameters

```
bayesstats summary paramspec [ , options ]
```

Summary statistics for functions of model parameters

```
bayesstats summary exprspec [ , options ]
```

Summary statistics of log-likelihood or log-posterior functions

```
bayesstats summary _loglikelihood | _logposterior [ , options ]
```

Full syntax

```
bayesstats summary spec [ spec ... ] [ , options ]
```

paramspec can be one of the following:

{*eqname*:*param*} refers to a parameter *param* with equation name *eqname*;

{*eqname*:} refers to all model parameters with equation name *eqname*;

{*eqname*:*paramlist*} refers to parameters with names in *paramlist* and with equation name *eqname*; or

{*param*} refers to all parameters named *param* from all equations.

In the above, *param* can refer to a matrix name, in which case it will imply all elements of this matrix. See *Different ways of specifying model parameters* in [BAYES] **bayesmh postestimation** for examples.

exprspec is an optionally labeled expression of model parameters specified in parentheses:

([*exprlabel*:] *expr*)

exprlabel is a valid Stata name, and *expr* is a scalar expression that may not contain matrix model parameters. See *Specifying functions of model parameters* in [BAYES] **bayesmh postestimation** for examples.

`_loglikelihood` and `_logposterior` also have respective synonyms `_ll` and `_lp`.

spec is one of *paramspec*, *exprspec*, `_loglikelihood` (or `_ll`), or `_logposterior` (or `_lp`).

options	Description
Main	
clevel(#)	set credible interval level; default is clevel(95)
hpd	display HPD credible intervals instead of the default equal-tailed credible intervals
batch(#)	specify length of block for batch-means calculations; default is batch(0)
skip(#)	skip every # observations from the MCMC sample; default is skip(0)
nolegend	suppress table legend
display_options	control spacing, line width, and base and empty cells
Advanced	
corrlag(#)	specify maximum autocorrelation lag; default varies
corrtol(#)	specify autocorrelation tolerance; default is corrtol(0.01)

Options

Main

clevel(#) specifies the credible level, as a percentage, for equal-tailed and HPD credible intervals. The default is clevel(95) or as set by [BAYES] **set clevel**.

hpd specifies the display of HPD credible intervals instead of the default equal-tailed credible intervals.

batch(#) specifies the length of the block for calculating batch means, batch standard deviation, and MCSE using batch means. The default is batch(0), which means no batch calculations. When batch() is not specified, MCSE is computed using effective sample sizes instead of batch means. Option batch() may not be combined with corrlag() or corrtol().

skip(#) specifies that every # observations from the MCMC sample not be used for computation. The default is skip(0) or to use all observations in the MCMC sample. Option skip() can be used to subsample or thin the chain. skip(#) is equivalent to a thinning interval of #+1. For example, if you specify skip(1), corresponding to the thinning interval of 2, the command will skip every other observation in the sample and will use only observations 1, 3, 5, and so on in the computation. If you specify skip(2), corresponding to the thinning interval of 3, the command will skip every 2 observations in the sample and will use only observations 1, 4, 7, and so on in the computation. skip() does not thin the chain in the sense of physically removing observations from the sample, as is done by bayesmh's thinning() option. It only discards selected observations from the computation and leaves the original sample unmodified.

nolegend suppresses the display of the table legend. The table legend identifies the rows of the table with the expressions they represent.

display_options: vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), and nolstretch; see [R] **estimation options**.

Advanced

corrlag(#) specifies the maximum autocorrelation lag used for calculating effective sample sizes. The default is $\min\{500, \texttt{mcmcsize()}/2\}$. The total autocorrelation is computed as the sum of all lag-k autocorrelation values for k from 0 to either corrlag() or the index at which the autocorrelation becomes less than corrtol() if the latter is less than corrlag(). Options corrlag() and batch() may not be combined.

corrtol(#) specifies the autocorrelation tolerance used for calculating effective sample sizes. The default is corrtol(0.01). For a given model parameter, if the absolute value of the lag-k autocorrelation is less than corrtol(), then all autocorrelation lags beyond the kth lag are discarded. Options corrtol() and batch() may not be combined.

Remarks and examples

Remarks are presented under the following headings:

Introduction

bayesstats summary reports posterior summary statistics for model parameters and their functions using the current estimation results from the bayesmh command. When typed without arguments, the command displays results for all model parameters. Alternatively, you can specify a subset of model parameters following the command name; see *Different ways of specifying model parameters* in [BAYES] **bayesmh postestimation**. You can also obtain results for scalar functions of model parameters; see *Specifying functions of model parameters* in [BAYES] **bayesmh postestimation**.

Sometimes, it may be useful to obtain posterior summaries of log-likelihood and log-posterior functions. This can be done by specifying _loglikelihood and _logposterior (or the respective synonyms _ll and _lp) following the command name.

bayesstats summary reports the following posterior summary statistics: posterior mean, posterior standard deviation, MCMC standard error, posterior median, and equal-tailed credible intervals or, if the hpd option is specified, HPD credible intervals. The default credible level is set to 95%, but you can change this by specifying the clevel() option. Equal-tailed and HPD intervals may produce very different results for asymmetric or highly skewed marginal posterior distributions. The HPD intervals are preferable in this situation.

You should not confuse the term "HPD interval" with the term "HPD region". A $\{100\times(1-\alpha)\}\%$ HPD interval is defined such that it contains $\{100\times(1-\alpha)\}\%$ of the posterior density. A $\{100\times(1-\alpha)\}\%$ HPD region also satisfies the condition that the density inside the region is never lower than that outside the region. For multimodal univariate marginal posterior distributions, the HPD regions may include unions of nonintersecting HPD intervals. For unimodal univariate marginal posterior distributions, HPD regions are indeed simply HPD intervals. The bayesstats summary command thus calculates HPD intervals assuming unimodal marginal posterior distributions (Chen and Shao 1999).

Some authors use the term "posterior intervals" instead of "credible intervals" and the term "central posterior intervals" instead of "equal-tailed credible intervals" (for example, Gelman et al. [2014]).

Bayesian summaries for an auto data example

Recall our analysis of auto.dta from example 4 in [BAYES] **bayesmh** using the mean-only normal model for mpg with a noninformative prior.

```
. use http://www.stata-press.com/data/r14/auto
(1978 Automobile Data)
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, flat) prior({var}, jeffreys)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
Priors:
  {mpg:_cons} ~ 1 (flat)
        {var} ~ jeffreys
------------------------------------------------------------------------------

Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2668
                                                Efficiency:  min =     .09718
                                                             avg =      .1021
Log marginal likelihood =   -234.645                         max =      .1071

------------------------------------------------------------------------------
             |                                                 Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
       _cons |   21.29222   .6828864   .021906   21.27898   19.99152   22.61904
-------------+----------------------------------------------------------------
         var |   34.76572    5.91534   .180754   34.18391    24.9129   47.61286
------------------------------------------------------------------------------
```

▷ Example 1: Summaries for all parameters

If we type `bayesstats summary` without arguments after the `bayesmh` command, we will obtain the same summary table as reported by `bayesmh`.

```
. bayesstats summary
Posterior summary statistics                    MCMC sample size =     10,000

------------------------------------------------------------------------------
             |                                                 Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
       _cons |   21.29222   .6828864   .021906   21.27898   19.99152   22.61904
-------------+----------------------------------------------------------------
         var |   34.76572    5.91534   .180754   34.18391    24.9129   47.61286
------------------------------------------------------------------------------
```

The posterior mean of `{mpg:_cons}` is 21.29 and of `{var}` is 34.8. They are close to their respective frequentist analogs (the sample mean of `mpg` is 21.297, and the sample variance is 33.47), because we used a noninformative prior. Posterior standard deviations are 0.68 for `{mpg:_cons}` and 5.92 for `{var}`, and they are comparable to frequentist standard errors under this noninformative prior. The standard error estimates of the posterior means, MCSEs, are low. For example, MCSE is 0.022 for `{mpg:_cons}`. This means that the precision of our estimate is, up to one decimal point, 21.3 provided that MCMC converged. The posterior means and medians of `{mpg:_cons}` are close, which suggests that the posterior distribution for `{mpg:_cons}` may be symmetric. According to the credible

intervals, we are 95% certain that the posterior mean of {mpg:_cons} is roughly between 20 and 23 and that the posterior mean of {var} is roughly between 25 and 48. We can infer from this that {mpg:_cons} is greater than, say, 15, and that {var} is greater than, say, 20, with a very high probability. (We can use [BAYES] **bayestest interval** to compute the actual probabilities.)

The above is also equivalent to typing

```
. bayesstats summary {mpg:_cons} {var}
(output omitted)
```

◁

▷ Example 2: Credible intervals

By default, bayesstats summary reports 95% equal-tailed credible intervals. We can change the default credible level by specifying the clevel() option.

```
. bayesstats summary, clevel(90)
Posterior summary statistics                     MCMC sample size =    10,000
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [90% Cred. Interval]	
mpg						
_cons	21.29222	.6828864	.021906	21.27898	20.18807	22.44172
var	34.76572	5.91534	.180754	34.18391	26.28517	44.81732

As expected, 90% credible intervals are more narrow.

To calculate and report HPD intervals, we specify the hpd option.

```
. bayesstats summary, hpd
Posterior summary statistics                     MCMC sample size =    10,000
```

	Mean	Std. Dev.	MCSE	Median	HPD [95% Cred. Interval]	
mpg						
_cons	21.29222	.6828864	.021906	21.27898	19.94985	22.54917
var	34.76572	5.91534	.180754	34.18391	24.34876	46.12339

The posterior distribution of {mpg:_cons} is symmetric about the posterior mean; thus there is little difference between the 95% equal-tailed credible interval from example 1 and this 95% HPD credible interval for {mpg:_cons}. The 95% HPD interval for {var} has a smaller width than the corresponding equal-tailed interval in example 1.

◁

▷ Example 3: Batch-means estimator

bayesstats summary provides two estimators for MCSE: effective-sample-size and batch-means. Estimation using effective sample sizes is the default. You can use the batch(#) option to request the batch-means estimator, where # is the batch size. The optimal batch size depends on the autocorrelation in the MCMC sample. For example, if we observe that the autocorrelation for the parameters of interest is negligible after lag 100, we can specify batch(100) to estimate MCSE.

In our example, autocorrelation dies out after about lag 10 (see, for example, *Autocorrelation plots* in [BAYES] **bayesgraph** and example 1 in [BAYES] **bayesstats ess**), so we use 10 as our batch size:

```
. bayesstats summary, batch(10)
Posterior summary statistics                     MCMC sample size =    10,000
                                                 Batch size       =        10
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	21.29222	.4842889	.015315	21.27898	19.99152	22.61904
var	34.76572	4.278417	.135295	34.18391	24.9129	47.61286

```
Note: Mean, Std. Dev., and MCSE are estimated using batch means.
```

The batch-means MCSE estimates are somewhat smaller than those obtained by default using effective sample sizes.

Use caution when choosing the batch size for the batch-means method. For example, if you use the batch size of 1, you will obtain MCSE estimates under the assumption that the draws in the MCMC sample are independent, which is not true.

◁

▷ Example 4: Subsampling or thinning the chain

You can reduce correlation between MCMC draws by thinning or subsampling the MCMC chain. You can use the `skip(#)` option to skip every # observations from the MCMC sample, which is equivalent to a thinning interval of $\# + 1$. For example, if you specify `skip(1)`, corresponding to the thinning interval of 2, `bayesstats summary` will skip every other observation in the sample and will use only observations 1, 3, 5, and so on in the computation. If you specify `skip(2)`, corresponding to the thinning interval of 3, `bayesstats summary` will skip every two observations in the sample and will use only observations 1, 4, 7, and so on in the computation. By default, no observations are skipped—`skip(0)`. Note that `skip()` does not thin the chain in the sense of physically removing observations from the sample, as is done by `bayesmh`'s `thinning()` option. It discards only selected observations from the computation and leaves the original sample unmodified.

```
. bayesstats summary, skip(9)
note: skipping every 9 sample observations; using observations 1,11,21,...
Posterior summary statistics                     MCMC sample size =     1,000
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	21.29554	.6813796	.029517	21.27907	19.98813	22.58582
var	34.7396	5.897313	.206269	33.91782	24.9554	48.11452

We selected to skip every 9 observations, which led to a significant reduction of the MCMC sample size and thus increased our standard deviations. In some cases, with larger MCMC sample sizes, subsampling may decrease standard deviations because of the decreased autocorrelation in the reduced MCMC sample.

◁

▷ Example 5: Summaries for functions of model parameters

`bayesstats summary` accepts expressions to provide summaries of functions of model parameters. For example, we can use expression `(sd:sqrt({var}))` with a label, `sd`, to summarize the standard deviation of `mpg` in addition to the variance.

```
. bayesstats summary (sd:sqrt({var})) {var}
Posterior summary statistics                       MCMC sample size =    10,000
          sd : sqrt({var})
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
sd	5.87542	.4951654	.014972	5.846701	4.991282	6.900207
var	34.76572	5.91534	.180754	34.18391	24.9129	47.61286

Expressions can also be used for calculating posterior probabilities, although this can be more easily done using `bayestest interval` (see [BAYES] **bayestest interval**). For illustration, let's verify the probability that `{var}` is within the endpoints of the reported credible interval, indeed 0.95.

```
. bayesstats summary (prob:{var}>24.913 & {var}<47.613)
Posterior summary statistics                       MCMC sample size =    10,000
        prob : {var}>24.913 & {var}<47.613
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
prob	.9502	.2175424	.005301	1	0	1

◁

▷ Example 6: Summaries for log likelihood and log posterior

We can use reserved names `_loglikelihood` (or the synonym `_ll`) and `_logposterior` (or the synonym `_lp`) to obtain summaries of the log likelihood and log posterior for the simulated MCMC sample.

```
. bayesstats summary _ll _lp
Posterior summary statistics                       MCMC sample size =    10,000
         _ll : _loglikelihood
         _lp : _logposterior
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
_ll	-235.4162	.990654	.032232	-235.1379	-238.1236	-234.4345
_lp	-238.9507	1.037785	.034535	-238.6508	-241.7889	-237.9187

◁

Stored results

`bayesstats summary` stores the following in `r()`:

Scalars

`r(clevel)`	credible interval level
`r(hpd)`	`1` if `hpd` is specified; `0` otherwise
`r(batch)`	batch length for batch-means calculations
`r(skip)`	number of MCMC observations to skip in the computation; every `r(skip)` observations are skipped
`r(corrlag)`	maximum autocorrelation lag
`r(corrtol)`	autocorrelation tolerance

Macros

`r(expr_#)`	#th expression
`r(names)`	names of model parameters and expressions
`r(exprnames)`	expression labels

Matrices

`r(summary)`	matrix with posterior summaries statistics for parameters in `r(names)`

Methods and formulas

Methods and formulas are presented under the following headings:

Point estimates
Credible intervals

Most of the summary statistics employed in Bayesian analysis are based on the marginal posterior distributions of individual model parameters or functions of model parameters.

Let θ be a scalar model parameter and $\{\theta_t\}_{t=1}^T$ be an MCMC chain of size T drawn from the marginal posterior distribution of θ. For a function $g(\theta)$, substitute $\{\theta_t\}_{t=1}^T$ with $\{g(\theta_t)\}_{t=1}^T$ in the formulas below. If θ is a covariance matrix model parameter, the formulas below are applied to each element of the lower-diagonal portion of θ.

Point estimates

Marginal posterior moments are approximated using the Monte Carlo integration applied to the simulated samples $\{\theta_t\}_{t=1}^T$.

Sample posterior mean and sample standard deviation are defined as follows,

$$\widehat{\theta} = \frac{1}{T}\sum_{t=1}^{T}\theta_t,\ \ \widehat{s}^2 = \frac{1}{T-1}\sum_{t=1}^{T}(\theta_t - \widehat{\theta})^2$$

where $\widehat{\theta}$ and $\widehat{s}^2$ are sample estimators of the population posterior mean $E(\theta_t)$ and posterior variance $\mathrm{Var}(\theta_t)$.

The precision of the sample posterior mean is evaluated by its standard error, also known as the Monte Carlo standard error (MCSE). Note that MCSE cannot be estimated using the classical formula for the standard error, $\widehat{s}/\sqrt{T}$, because of the dependence between θ_t's.

Let

$$\sigma^2 = \mathrm{Var}(\theta_t) + 2\sum_{k=1}^{\infty}\mathrm{Cov}(\theta_t, \theta_{t+k})$$

Then, $\sqrt{T}\times$MCSE approaches σ^2 asymptotically in T.

`bayesstats summary` provides two different approaches for estimating MCSE. Both approaches try to adjust for the existing autocorrelation in the MCMC sample. The first one uses the so-called effective sample size (ESS), and the second one uses batch means (Roberts 1996; Jones et al. 2006).

The ESS-based estimator for MCSE, the default in `bayesstats summary`, is given by

$$\text{MCSE}(\widehat{\theta}) = \widehat{s}/\sqrt{\text{ESS}}$$

ESS is defined as

$$\text{ESS} = T/(1 + 2\sum_{k=1}^{\text{max\_lags}} \rho_k)$$

where ρ_k is the lag-k autocorrelation, and $\max\_\text{lags}$ is the maximum number less than or equal to ρ_{lag} such that for all $k = 1, \ldots, \max\_\text{lags}$, $|\rho_k| > \rho_{\text{tol}}$, where ρ_{lag} and ρ_{tol} are specified in options `corrlag()` and `corrtol()` with the respective default values of 500 and 0.01. ρ_k is estimated as γ_k/γ_0, where

$$\gamma_k = \frac{1}{T}\sum_{t=1}^{T-k}(\theta_t - \widehat{\theta})(\theta_{t+k} - \widehat{\theta})$$

is the lag-k empirical autocovariance.

The batch-means estimator of MCSE is obtained as follows. For a given batch of length b, the initial MCMC chain is split into m batches of size b,

$$\{\theta_{j'+1}, \ldots, \theta_{j'+b}\}\ \{\theta_{j'+b+1}, \ldots, \theta_{j'+2b}\}\ \ldots\ \{\theta_{T-b+1}, \ldots, \theta_T\}$$

where $j' = T - m \times b$ and m batch means $\widehat{\mu}_1, \ldots, \widehat{\mu}_m$ are calculated as sample means of each batch. m is chosen as the maximum number such that $m \times b \leq T$. If m is not a divisor of T, the first $T - m \times b$ observations of the sample are not used in the batch-means computation. The batch-means estimator of the posterior variance, $\widehat{s}^2_{\text{batch}}$, is based on the assumption that $\widehat{\mu}_j$s are much less correlated than the original sample draws.

The batch-means estimators of the posterior mean and posterior variance are

$$\widehat{\theta}_{\text{batch}} = \frac{1}{m}\sum_{j=1}^{m}\widehat{\mu}_j,\ \ \widehat{s}^2_{\text{batch}} = \frac{1}{m-1}\sum_{j=1}^{m}(\widehat{\mu}_j - \widehat{\theta}_{\text{batch}})^2$$

We have $\widehat{\theta}_{\text{batch}} = \widehat{\theta}$, whenever $m \times b = T$. Under the assumption that the batch means are uncorrelated, $\widehat{s}^2_{\text{batch}}$ can be used as an estimator of σ^2/b. This fact justifies the batch-means estimator of MCSE given by

$$\text{MCSE}_{\text{batch}}(\widehat{\theta}) = \frac{\widehat{s}_{\text{batch}}}{\sqrt{m}}$$

The accuracy of the batch-means estimator depends on the choice of the batch length b. The higher the autocorrelation in the original MCMC sample, the larger the batch length b should be, provided that the number of batches m does not become too small; $\sqrt{T}$ is typically used as the maximum value for b. The batch length is commonly determined by inspecting the autocorrelation plot for θ. Under certain assumptions, Flegal and Jones (2010) establish that an asymptotically optimal batch size is of order $T^{1/3}$.

Credible intervals

Let $\theta_{(1)}, \ldots, \theta_{(T)}$ be an MCMC sample ordered from smallest to largest. Let $(1-\alpha)$ be a credible level. Then, a $\{100 \times (1-\alpha)\}\%$ equal-tailed credible interval is

$$(\theta_{([T\alpha/2])}, \theta_{([T(1-\alpha/2)])})$$

where $[\,]$ in the above imply an integer number.

A $\{100 \times (1-\alpha)\}\%$ HPD interval is defined as the shortest interval among the $\{100 \times (1-\alpha)\}\%$ credible intervals $(\theta_{(j)}, \theta_{(j+[T(1-\alpha)])})$, $j = 1, \ldots, T - [T(1-\alpha)]$.

References

Chen, M.-H., and Q.-M. Shao. 1999. Monte Carlo estimation of Bayesian credible and HPD intervals. *Journal of Computational and Graphical Statistics* 8: 69–92.

Flegal, J. M., and G. L. Jones. 2010. Batch means and spectral variance estimators in Markov chain Monte Carlo. *Annals of Statistics* 38: 1034–1070.

Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. 2014. *Bayesian Data Analysis*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.

Jones, G. L., M. Haran, B. S. Caffo, and R. Neath. 2006. Fixed-width output analysis for Markov chain Monte Carlo. *Journal of the American Statistical Association* 101: 1537–1547.

Roberts, G. O. 1996. Markov chain concepts related to sampling algorithms. In *Markov Chain Monte Carlo in Practice*, ed. W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, 45–57. Boca Raton, FL: Chapman and Hall.

Also see

[BAYES] **bayesmh** — Bayesian regression using Metropolis–Hastings algorithm

[BAYES] **bayesmh postestimation** — Postestimation tools for bayesmh

[BAYES] **bayesgraph** — Graphical summaries and convergence diagnostics

[BAYES] **bayesstats ess** — Effective sample sizes and related statistics

[BAYES] **bayestest interval** — Interval hypothesis testing

Title

bayestest — Bayesian hypothesis testing

Description Remarks and examples Also see

Description

`bayestest` provides two types of Bayesian hypothesis testing, interval hypothesis testing and model hypothesis testing, using current estimation results from the `bayesmh` command.

`bayestest interval` performs interval hypothesis tests for model parameters and functions of model parameters; see [BAYES] **bayestest interval**.

`bayestest model` tests hypotheses about models by computing posterior probabilities of the models; see [BAYES] **bayestest model**.

Remarks and examples

Bayesian hypothesis testing is fundamentally different from the conventional frequentist hypothesis testing using p-values. Frequentist hypothesis testing is based on the deterministic decision of whether to reject a null hypothesis against an alternative hypothesis based on the obtained p-value. Bayesian hypothesis testing is built upon a probabilistic formulation for a parameter of interest. For example, it can provide a probabilistic summary of how likely that parameter of interest belongs to some prespecified set of values. Also, Bayesian testing can assign a probability to a hypothesis of interest or model of interest given the observed data. This cannot be done in the frequentist testing. The ability to assign a probability to a hypothesis often provides a more natural interpretation of the results. For example, Bayesian hypothesis testing provides a direct answer to the following questions. How likely is it that the mean height of males is larger than six feet? What is the probability that a person is guilty versus being innocent? How likely is one model over the other model? Frequentist hypothesis testing cannot be used to answer these questions.

We consider two forms of Bayesian hypothesis testing: interval hypothesis testing and what we call model hypothesis testing.

The goal of interval hypothesis testing is to estimate the probability that a model parameter lies in a certain interval; see [BAYES] **bayestest interval** for details.

The goal of model hypothesis testing is to test hypotheses about models by computing probabilities of the specified models given the observed data; see [BAYES] **bayestest model** for details.

Also see

[BAYES] **bayestest interval** — Interval hypothesis testing

[BAYES] **bayestest model** — Hypothesis testing using model posterior probabilities

[BAYES] **bayesmh postestimation** — Postestimation tools for bayesmh

[BAYES] **Glossary**

Title

bayestest interval — Interval hypothesis testing

Description

`bayestest interval` performs interval hypothesis tests for model parameters and functions of model parameters using current estimation results from the `bayesmh` command. `bayestest interval` reports mean estimates, standard deviations, and MCMC standard errors of posterior probabilities associated with an interval hypothesis.

Quick start

Posterior probability of the hypothesis that $45 <$ `{y:_cons}` < 50

```
bayestest interval {y:_cons}, lower(45) upper(50)
```

As above, but skip every 5 observations from the full MCMC sample

```
bayestest interval {y:_cons}, lower(45) upper(50) skip(5)
```

Posterior probability of a hypothesis about a function of model parameter `{y:x1}`

```
bayestest interval (OR:exp({y:x1})), lower(1.1) upper(1.5)
```

Posterior probability of hypotheses $45 <$ `{y:_cons}` < 50 and $0 <$ `{var}` < 10 tested independently

```
bayestest interval ({y:_cons}, lower(45) upper(50))  ///
     ({var}, lower(0) upper(10))
```

As above, but tested jointly

```
bayestest interval (({y:_cons}, lower(45) upper(50))  ///
     ({var}, lower(0) upper(10)), joint)
```

Posterior probability of the hypothesis `{mean}` $= 2$ for discrete parameter `{mean}`

```
bayestest interval ({mean}==2)
```

Posterior probability of the interval hypothesis $0 \leq$ `{mean}` ≤ 4

```
bayestest interval {mean}, lower(0, inclusive) upper(4, inclusive)
```

Menu

Statistics > Bayesian analysis > Interval hypothesis testing

Syntax

Test one interval hypothesis about continuous or discrete parameter

```
bayestest interval exspec [ , luspec options ]
```

Test one point hypothesis about discrete parameter

```
bayestest interval exspec==# [ , options ]
```

Test multiple hypotheses separately

```
bayestest interval (testspec) [ (testspec) ... ] [ , options ]
```

Test multiple hypotheses jointly

```
bayestest interval (jointspec) [ , options ]
```

Full syntax

```
bayestest interval (spec) [ (spec) ... ] [ , options ]
```

exspec is optionally labeled expression of model parameters, [*prlabel*:]*expr*, where *prlabel* is a valid Stata name (or `prob#` by default), and *expr* is a scalar model parameter or scalar expression (parentheses are optional) containing scalar model parameters. The expression *expr* may not contain variable names.

testspec is *exspec*[, *luspec*] or *exspec*`==#` for discrete parameters only.

jointspec is [*prlabel:*]`(`*testspec*`) (`*testspec*`) ... , joint.` The labels (if any) of *testspec* are ignored.

spec is one of *testspec* or *jointspec*.

luspec	Null hypothesis
`lower(#) [upper(.)]`	$\theta > \#$
`lower(#, inclusive) [upper(.)]`	$\theta \geq \#$
`[lower(.)] upper(#)`	$\theta < \#$
`[lower(.)] upper(#, inclusive)`	$\theta \leq \#$
`lower(#`$_l$`) upper(#`$_u$`)`	$\#_l < \theta < \#_u$
`lower(#`$_l$`) upper(#`$_u$`, inclusive)`	$\#_l < \theta \leq \#_u$
`lower(#`$_l$`, inclusive) upper(#`$_u$`)`	$\#_l \leq \theta < \#_u$
`lower(#`$_l$`, inclusive) upper(#`$_u$`, inclusive)`	$\#_l \leq \theta \leq \#_u$

`lower(`*intspec*`)` and `upper(`*intspec*`)` specify the lower- and upper-interval values, respectively.

intspec is # [, `inclusive`]

where # is the interval value, and suboption `inclusive` specifies that this value should be included in the interval, meaning a closed interval. Closed intervals make sense only for discrete parameters.

intspec may also contain a dot (`.`), meaning negative infinity for `lower()` and positive infinity for `upper()`. Either option `lower(.)` or option `upper(.)` must be specified.

options	Description
Main	
`skip(#)`	skip every # observations from the MCMC sample; default is `skip(0)`
`nolegend`	suppress table legend
Advanced	
`corrlag(#)`	specify maximum autocorrelation lag; default varies
`corrtol(#)`	specify autocorrelation tolerance; default is `corrtol(0.01)`

Options

Main

`skip(#)` specifies that every # observations from the MCMC sample not be used for computation. The default is `skip(0)` or to use all observations in the MCMC sample. Option `skip()` can be used to subsample or thin the chain. `skip(#)` is equivalent to a thinning interval of #+1. For example, if you specify `skip(1)`, corresponding to the thinning interval of 2, the command will skip every other observation in the sample and will use only observations 1, 3, 5, and so on in the computation. If you specify `skip(2)`, corresponding to the thinning interval of 3, the command will skip every 2 observations in the sample and will use only observations 1, 4, 7, and so on in the computation. `skip()` does not thin the chain in the sense of physically removing observations from the sample, as is done by `bayesmh`'s `thinning()` option. It only discards selected observations from the computation and leaves the original sample unmodified.

`nolegend` suppresses the display of the table legend. The table legend identifies the rows of the table with the expressions they represent.

Advanced

`corrlag(#)` specifies the maximum autocorrelation lag used for calculating effective sample sizes. The default is $\min\{500, \texttt{mcmcsize()}/2\}$. The total autocorrelation is computed as the sum of all lag-k autocorrelation values for k from 0 to either `corrlag()` or the index at which the autocorrelation becomes less than `corrtol()` if the latter is less than `corrlag()`.

`corrtol(#)` specifies the autocorrelation tolerance used for calculating effective sample sizes. The default is `corrtol(0.01)`. For a given model parameter, if the absolute value of the lag-k autocorrelation is less than `corrtol()`, then all autocorrelation lags beyond the kth lag are discarded.

Remarks and examples

Remarks are presented under the following headings:

Introduction

In this entry, we describe interval hypothesis testing, the goal of which is to estimate the probability that a model parameter lies in a certain interval. Interval hypothesis testing is inversely related to credible intervals. For example, if we have a 95% credible interval for θ with endpoints U and L, then the probability of a hypothesis H_0: $\theta \in [U, L]$ is 0.95. For hypothesis testing using model posterior probabilities, see [BAYES] **bayestest model**.

In frequentist hypothesis testing, we often consider a point hypothesis such as H_0: $\theta = \theta_0$ versus H_a: $\theta \neq \theta_0$. In Bayesian hypothesis testing, the probability $P(\theta = \theta_0)$ is 0 whenever θ has a continuous posterior distribution. A point hypothesis is relevant only to parameters with discrete posterior distributions. For continuous parameters, all hypotheses should be formulated as intervals. One possibility is to consider an interval hypothesis H_0: $\theta \in (\theta_0 - \epsilon, \theta_0 + \epsilon)$, where ϵ is some small value.

Note that Bayesian hypothesis testing does not really need a distinction between the null and alternative hypotheses, in the sense that they are defined in a frequentist statistic. There is no need to "protect" the null hypothesis: if $P\{H_0$: $\theta \in (a, b)\} = p$, then $P\{H_a$: $\theta \notin (a, b)\} = 1 - p$. In what follows, when we refer to H_0, we imply a hypothesis of interest H_0: $\theta \in \Theta$, and when we refer to H_a, we imply the complement hypothesis H_a: $\theta \in \Theta^c$, where Θ is a set of points from the domain of θ and Θ^c is its complement.

The `bayestest interval` command estimates the posterior probability of a null interval hypothesis H_0 using the simulated posterior distributions of model parameters produced by `bayesmh`. Essentially, `bayestest interval` reports posterior summaries for a dichotomous expression that represents H_0.

For example, suppose we would like to test the following hypothesis: H_0: $\theta \in (a, b)$. Then,

```
bayestest interval ({theta}, lower(a) upper(b))
```

is equivalent to

```
bayesstats summary ({theta} > a & {theta} < b)
```

`bayestest interval` reports the estimated posterior mean probability for H_0, which is not a p-value—as reported by classical frequentist tests—used to decide whether to reject H_0 in favor of the alternative H_a. The p-value interpretation is based on the dichotomous problem formulation of H_0 versus H_a, assuming that one of these two alternatives is actually true. The answer in the Bayesian context is a probability statement about θ that is free of any deterministic presumptions. For example, if you estimate $P(H_0)$ to be 0.15, you cannot ask whether this value is significant or whether you can reject the null hypothesis. Bayesian interpretation of this probability is that if you draw θ from the specified prior distribution and update your knowledge about θ based on the observed data, then there is a 15% chance that θ will belong to the interval (a, b). So the conclusion of Bayesian hypothesis testing is not an acceptance or rejection of the null hypothesis but an explicit probability statement about the tested hypothesis.

Interval tests for continuous parameters

Let's continue our analysis of `auto.dta` from example 4 in [BAYES] **bayesmh** using the mean-only normal model for `mpg` with a noninformative prior.

```
. use http://www.stata-press.com/data/r14/auto
(1978 Automobile Data)
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, flat) prior({var}, jeffreys)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
Priors:
  {mpg:_cons} ~ 1 (flat)
        {var} ~ jeffreys
------------------------------------------------------------------------------

Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2668
                                                Efficiency:  min =     .09718
                                                             avg =      .1021
Log marginal likelihood =   -234.645                         max =      .1071

------------------------------------------------------------------------------
             |                                                 Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
       _cons |   21.29222   .6828864   .021906   21.27898   19.99152   22.61904
-------------+----------------------------------------------------------------
         var |   34.76572    5.91534   .180754   34.18391    24.9129   47.61286
------------------------------------------------------------------------------
```

▷ Example 1: Interval hypothesis and credible intervals

In the introduction, we commented on the inverse relationship that exists between interval hypothesis tests and credible intervals. Let's verify this using `bayestest interval`. We are interested in a hypothesis H_0: `{mpg:_cons}` $\in (19.992, 22.619)$, where the specified numbers are the endpoints of the credible interval for `{mpg:_cons}` from the `bayesmh` output. To compute the posterior probability for this hypothesis, we specify the parameter following the command line and specify interval endpoints in `lower()` and `upper()`.

```
. bayestest interval {mpg:_cons}, lower(19.992) upper(22.619)
Interval tests     MCMC sample size =    10,000
       prob1 : 19.992 < {mpg:_cons} < 22.619
------------------------------------------------------------------------------
             |      Mean    Std. Dev.      MCSE
-------------+----------------------------------------------------------------
       prob1 |      .9496     0.21878   .0053652
------------------------------------------------------------------------------
```

The estimated posterior probability is close to 0.95, as we expected, because we used the endpoints of the 95% credible intervals for `{mpg:_cons}`.

By default, `bayestest interval` labels probabilities as `prob#` (`prob1` in our example). You can specify your own label as long as you enclose the parameter in parentheses:

```
. bayestest interval (mean:{mpg:_cons}), lower(19.992) upper(22.619)
Interval tests     MCMC sample size =    10,000
        mean : 19.992 < {mpg:_cons} < 22.619
```

	Mean	Std. Dev.	MCSE
mean	.9496	0.21878	.0053652

◁

▷ Example 2: Testing multiple hypotheses separately

Continuing example 1, we can verify that the probability associated with the credible interval for `{var}` is also close to 0.95.

We can specify multiple hypotheses with `bayestest interval`, but we must enclose them in parentheses.

```
. bayestest interval ({mpg:_cons}, lower(19.992) upper(22.619))
>                    ({var}, lower(24.913) upper(47.613))
Interval tests     MCMC sample size =    10,000
       prob1 : 19.992 < {mpg:_cons} < 22.619
       prob2 : 24.913 < {var} < 47.613
```

	Mean	Std. Dev.	MCSE
prob1	.9496	0.21878	.0053652
prob2	.9502	0.21754	.0053011

The estimated posterior probability `prob2` is also close to 0.95.

◁

▷ Example 3: Testing multiple hypotheses jointly

We can perform joint tests of multiple hypotheses by enclosing hypothesis to be tested jointly in parentheses and by specifying suboption `joint`. Notice that each individual hypothesis must also be enclosed in parentheses.

```
. bayestest interval (({mpg:_cons}, lower(19.992) upper(22.619))
>                    ({var}, lower(24.913) upper(47.613)), joint)
Interval tests     MCMC sample size =    10,000
       prob1 : 19.992 < {mpg:_cons} < 22.619,
               24.913 < {var} < 47.613
```

	Mean	Std. Dev.	MCSE
prob1	.9034	0.29543	.0076789

The joint posterior probability of both `{mpg:_cons}` and `{var}` belonging to their respective intervals is 0.9 with a posterior variance of 0.3 and MCSE of 0.008.

◁

Example 4: Full syntax

We can specify multiple separate hypotheses and hypotheses tested jointly in one call to `bayestest interval`.

```
. bayestest interval (({mpg:_cons}, lower(19.992) upper(22.619))
>                     ({var}, lower(24.913) upper(47.613)), joint)
>                    ({mpg:_cons}, lower(21))
>                    ({var}, upper(40))
Interval tests     MCMC sample size =    10,000
       prob1 : 19.992 < {mpg:_cons} < 22.619,
               24.913 < {var} < 47.613
       prob2 : {mpg:_cons} > 21
       prob3 : {var} < 40
```

	Mean	Std. Dev.	MCSE
prob1	.9034	0.29543	.0076789
prob2	.6505	0.47684	.015786
prob3	.8136	0.38945	.0110613

In addition to the joint hypothesis from the previous example, we specified two new separate interval hypotheses for testing {mpg:_cons} $>$ 21 and for testing {var} $<$ 40. The estimated posterior probabilities for these hypotheses are 0.65 and 0.81, respectively.

◁

Example 5: Point hypothesis for continuous parameters

As we discussed in *Introduction* above, point hypothesis for continuous parameters do not make sense, because the corresponding probability is 0:

```
. bayestest interval ({mpg:_cons}==21)
Interval tests     MCMC sample size =    10,000
       prob1 : {mpg:_cons}==21
```

	Mean	Std. Dev.	MCSE
prob1	0	0.00000	0

We can consider a small window around the value of interest and test an interval hypothesis instead:

```
. bayestest interval ({mpg:_cons}, lower(20.5) upper(21.5))
Interval tests     MCMC sample size =    10,000
       prob1 : 20.5 < {mpg:_cons} < 21.5
```

	Mean	Std. Dev.	MCSE
prob1	.4932	0.49998	.0138391

The probability that {mpg:_cons} is between 20.5 and 21.5 is about 50%.

Note that the probability of a continuous parameter belonging to a closed interval or semiclosed interval is the same as that for the open interval. Below we use suboption `inclusive` within `lower()` and `upper()` to request the closed interval.

```
. bayestest interval ({mpg:_cons}, lower(20.5,inclusive) upper(21.5,inclusive))
Interval tests     MCMC sample size =    10,000
       prob1 : 20.5 <= {mpg:_cons} <= 21.5
```

	Mean	Std. Dev.	MCSE
prob1	.4932	0.49998	.0138391

We obtain the same results as above for the corresponding open interval.

◁

▷ Example 6: Functions of parameters

We can test functions of model parameters. For example, let's compute the probability that the posterior standard deviation is greater than 6.

```
. bayestest interval (sd: sqrt({var}), lower(6))
Interval tests     MCMC sample size =    10,000
          sd : sqrt({var}) > 6
```

	Mean	Std. Dev.	MCSE
sd	.3793	0.48524	.0143883

The estimated probability is 0.38.

◁

Interval tests for discrete parameters

In this section, we demonstrate how to perform hypothesis testing for a discrete parameter.

First, we simulate data from the Poisson distribution with a mean of 2.

```
. clear
. set seed 12345
. set obs 20
number of observations (_N) was 0, now 20
. generate double y = rpoisson(2)
```

We fit a Bayesian Poisson model to the data and specify a discrete prior for the mean $P(\mu = k) = 0.25$ for $k = 1, 2, 3, 4$.

```
. set seed 14
. bayesmh y, likelihood(poisson, noglmtransform)
> prior({y:}, index(0.25,0.25,0.25,0.25)) initial({y:_cons} 2)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  y ~ poisson({y:_cons})
Prior:
  {y:_cons} ~ index(0.25,0.25,0.25,0.25)
------------------------------------------------------------------------------

Bayesian Poisson regression                     MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         20
                                                Acceptance rate  =      .2552
Log marginal likelihood =  -31.58903            Efficiency       =      .4428

------------------------------------------------------------------------------
             |                                                 Equal-tailed
           y |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
       _cons |      2.0014   .1039188   .001562          2           2           2
------------------------------------------------------------------------------
```

We specified the `noglmtransform` option with the Poisson likelihood to request that the GLM logit link function not be applied to the model constant, `{y:_cons}`, so that we could model the mean parameter directly. Without this option, `bayesmh, likelihood(poisson)` fits a Poisson regression with a logit link function between the mean and the linear predictor formed by the specified regressors.

▷ Example 7: Point hypotheses for discrete parameters

We can compute probabilities for each of the four discrete values of `{y:_cons}`.

```
. bayestest interval ({y:_cons}==1) ({y:_cons}==2) ({y:_cons}==3) ({y:_cons}==4)
Interval tests     MCMC sample size =    10,000
       prob1 : {y:_cons}==1
       prob2 : {y:_cons}==2
       prob3 : {y:_cons}==3
       prob4 : {y:_cons}==4
------------------------------------------------------------------------------
             |      Mean    Std. Dev.       MCSE
-------------+----------------------------------------------------------------
       prob1 |      .0047     0.06840   .0013918
       prob2 |      .9892     0.10337   .0027909
       prob3 |      .0061     0.07787   .0017691
       prob4 |          0     0.00000          0
------------------------------------------------------------------------------
```

The posterior probability that `{y}` equals 2 is 0.99.

◁

▷ Example 8: Interval hypotheses for discrete parameters

As we can with continuous parameters, we can test interval hypotheses for discrete parameters. For example, we can compute the probability of whether {y:_cons} is between 2 and 4.

```
. bayestest interval {y:_cons}, lower(2) upper(4)
Interval tests     MCMC sample size =    10,000
       prob1 : 2 < {y:_cons} < 4
------------------------------------------------------------
             |       Mean    Std. Dev.       MCSE
-------------+----------------------------------------------
       prob1 |      .0061     0.07787   .0017691
------------------------------------------------------------
```

The estimated probability is very small.

Note that unlike hypotheses for continuous parameters, hypotheses including open intervals and closed or semiclosed intervals for discrete parameters may have different probabilities.

```
. bayestest interval {y:_cons}, lower(2, inclusive) upper(4, inclusive)
Interval tests     MCMC sample size =    10,000
       prob1 : 2 <= {y:_cons} <= 4
------------------------------------------------------------
             |       Mean    Std. Dev.       MCSE
-------------+----------------------------------------------
       prob1 |      .9953     0.06840   .0013918
------------------------------------------------------------
```

The estimated posterior probability that {y:_cons} is between 2 and 4, inclusively, is drastically different compared with the results for the corresponding open interval.

◁

Stored results

bayestest interval stores the following in r():

Scalars

r(skip)	number of MCMC observations to skip in the computation; every r(skip) observations are skipped
r(corrlag)	maximum autocorrelation lag
r(corrtol)	autocorrelation tolerance

Macros

r(expr_#)	#th probability expression
r(names)	names of probability expressions

Matrices

r(summary)	test results for parameters in r(names)

Methods and formulas

Let θ be a model parameter and $\{\theta_t\}_{t=1}^{T}$ be an MCMC sample of size T drawn from the marginal posterior distribution of θ. It is often of interest to test how likely it is that θ belongs to a particular range of values. Note that testing a point null hypothesis such as H_0: $\theta = \theta_0$ is usually of no interest for parameters with continuous posterior distributions, because the posterior probability $P(H_0)$ is 0.

To perform an open-interval test of the form

$$H_0\text{: } \theta \in (a, b) \text{ versus } H_a\text{: } \theta \notin (a, b)$$

we estimate the posterior probability of H_0 from the given MCMC sample. The `bayestest interval` command calculates the probability $P(H_0)$ based on the simulated marginal posterior distribution of θ. The estimate is given by the frequency of inclusion of θ_ts in the test interval

$$\widehat{P}(H_0) = \frac{1}{T}\sum_{t=1}^{T} 1_{\{\theta_t \in (a,b)\}} \tag{1}$$

where $1_{\{A\}}$ is an indicator function and equals 1 if A is true and 0 otherwise.

When a model parameter θ is discrete, the following closed- and semiclosed-interval tests may be of interest in addition to open-interval tests:

$$H_0\colon\ \theta = a \text{ versus } H_a\colon\ \theta \neq a$$

$$H_0\colon\ \theta \in [a,b] \text{ versus } H_a\colon\ \theta \notin [a,b]$$

$$H_0\colon\ \theta \in [a,b) \text{ versus } H_a\colon\ \theta \notin [a,b)$$

$$H_0\colon\ \theta \in (a,b] \text{ versus } H_a\colon\ \theta \notin (a,b]$$

The corresponding probabilities are calculated as follows:

$$\widehat{P}(H_0) = \frac{1}{T}\sum_{t=1}^{T} 1_{\{\theta_t = a\}}$$

$$\widehat{P}(H_0) = \frac{1}{T}\sum_{t=1}^{T} 1_{\{\theta_t \in [a,b]\}}$$

$$\widehat{P}(H_0) = \frac{1}{T}\sum_{t=1}^{T} 1_{\{\theta_t \in [a,b)\}}$$

$$\widehat{P}(H_0) = \frac{1}{T}\sum_{t=1}^{T} 1_{\{\theta_t \in (a,b]\}}$$

The probability of an alternative hypothesis is always given by $P(H_a) = 1 - P(H_0)$.

The formulas above can be modified to accommodate joint hypotheses tests by multiplying the indicator functions of the individual hypothesis statements. For example, for a joint hypothesis $H_0\colon \theta_1 > a, \theta_2 < b$, we would replace the indicator function with $1_{\{\theta_{1t} > a\}} \times 1_{\{\theta_{2t} < b\}}$ in (1), where $\{\theta_{1t}\}_{t=1}^{T}$ and $\{\theta_{2t}\}_{t=1}^{T}$ are the corresponding MCMC samples for θ_1 and θ_2.

Also see

[BAYES] **bayesmh** — Bayesian regression using Metropolis–Hastings algorithm

[BAYES] **bayesmh postestimation** — Postestimation tools for bayesmh

[BAYES] **bayesstats summary** — Bayesian summary statistics

[BAYES] **bayestest model** — Hypothesis testing using model posterior probabilities

Title

bayestest model — Hypothesis testing using model posterior probabilities

Description

`bayestest model` computes posterior probabilities of Bayesian models fit using the `bayesmh` command. These posterior probabilities can be used to test hypotheses about model parameters. The command reports marginal likelihoods, prior probabilities, and posterior probabilities for all tested models.

Quick start

Compute posterior probabilities of models corresponding to previously saved estimation results `M1` and `M2`

```
bayestest model M1 M2
```

As above, but specify prior probabilities for models

```
bayestest model M1 M2, prior(0.3 0.7)
```

Menu

Statistics > Bayesian analysis > Hypothesis testing using model posterior probabilities

Syntax

```
bayestest model [namelist] [, options]
```

where *namelist* is a name, a list of names, `_all`, or `*`. A name may be `.`, meaning the current (active) estimates. `_all` and `*` mean the same thing.

options	Description
Main	
`prior(`*numlist*`)`	specify prior probabilities for tested models; default is all models are equally likely
Advanced	
`marglmethod(`*method*`)`	specify marginal-likelihood approximation method; default is to use Laplace–Metropolis approximation, `lmetropolis`; rarely used

method	Description
`lmetropolis`	Laplace–Metropolis approximation; default
`hmean`	harmonic-mean approximation

Options

Main

`prior(`*numlist*`)` specifies prior probabilities for models. By default, all models are assumed to be equally likely. You may specify probabilities for all tested models, in which case the probabilities must sum to one. Alternatively, you may specify probabilities for all but the last model, in which case the sum of the specified probabilities must be less than one, and the probability for the last model is computed as one minus this sum.

Advanced

`marglmethod(`*method*`)` specifies a method for approximating the marginal likelihood. *method* is either `lmetropolis`, the default, for Laplace–Metropolis approximation or `hmean` for harmonic-mean approximation. This option is rarely used.

Remarks and examples

Remarks are presented under the following headings:

Introduction

In this entry, we describe hypothesis testing by computing model posterior probabilities, probabilities of Bayesian models given observed data. For interval hypothesis testing, see [BAYES] **bayestest interval**.

The `bayestest model` command computes posterior probabilities for specified models. The computed probabilities can be used to compare which model is more likely among considered models given observed data. You can compare models that differ only in several covariates or models with completely different regression functions, such as linear and nonlinear models. You can compare models with different outcome distributions or with different prior distributions or both. The only requirements are that the considered models have proper posterior distributions and that the same data are used to fit the models. If MCMC is used to approximate posterior distributions, convergence of MCMC should also be verified before model comparison.

The results reported by `bayestest model` are related to Bayes factors; see [BAYES] **bayesstats ic** to compute Bayes factors.

To use `bayestest model`, you must store estimation results after each `bayesmh` model of interest. You can use `estimates store` (see [R] **estimates store**) to store estimation results after `bayesmh`, as you can with other estimation commands, provided you also saved simulation results from `bayesmh` using the `saving()` option. See *Storing estimation results after bayesmh* in [BAYES] **bayesmh postestimation** for details.

Testing nested hypotheses

Consider the following Bayesian regression model for `auto.dta`,

$$\texttt{mpg} = \beta_0 + \beta_1\texttt{weight1} + \beta_2\texttt{length1} + \epsilon$$

where `weight1` and `length1` are the original `weight` and `length` variables rescaled to have similar scale as `mpg`.

We assume that errors are normally distributed: $\epsilon \sim \text{normal}(0, \sigma^2)$. We also assume a noninformative Jeffreys prior for the parameters: $(\boldsymbol{\beta}, \sigma^2) \sim 1/\sigma^2$. Suppose that we are interested in testing whether there is a relationship between mileage and weight and length of cars. We will consider four models: the mean-only model, the model with weight only, the model with length only, and the full model with both covariates.

In a frequentist setting, the four models correspond to the following hypotheses: H_0: $\beta_1 = 0$, $\beta_2 = 0$, H_0: $\beta_1 = 0$, and H_0: $\beta_2 = 0$. In a Bayesian setting, we cannot formulate point hypotheses for parameters with continuous distributions; see [BAYES] **bayestest interval** for examples. However, we can compute probabilities of how likely each of the four models is given the observed data.

Let's load auto.dta and generate rescaled versions of weight and length.

```
. use http://www.stata-press.com/data/r14/auto
(1978 Automobile Data)
. generate weight1 = weight/100
. generate length1 = length/10
```

Next, we fit the four models using bayesmh. We use the saving() option to save the simulation datasets so that we can store estimation results of each model for later use with bayestest model.

The first model we fit is the mean-only model. We store its estimation results as meanonly.

```
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:}, flat) prior({var}, jeffreys)
> saving(meanonly_simdata) burnin(3500)
note: adaptation option maxiter() changed to 35
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
Priors:
  {mpg:_cons} ~ 1 (flat)
        {var} ~ jeffreys
------------------------------------------------------------------------------
Bayesian normal regression                      MCMC iterations  =     13,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      3,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2627
                                                Efficiency:  min =       .105
                                                             avg =      .1064
Log marginal likelihood = -234.64617                         max =      .1078
------------------------------------------------------------------------------
             |                                                Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
       _cons |   21.29355   .6768607   .020887   21.28059   20.00132   22.61904
-------------+----------------------------------------------------------------
         var |   34.80707   5.963995   .181615   34.23247    24.9129    47.6883
------------------------------------------------------------------------------
file meanonly_simdata.dta saved
. estimates store meanonly
```

To accommodate the Jeffreys prior for the parameters, we specify suboption flat within the prior() option for coefficients to request the flat prior with the density of 1 and suboption jeffreys within prior() for the variance parameter to request a Jeffreys prior. We also specify a longer burn-in period to improve convergence of MCMC samples for all examples. (Remember to use bayesgraph to check convergence of MCMC.)

We fit the second model containing only covariate `length1` and store its results as `length`:

```
. set seed 14
. bayesmh mpg length1, likelihood(normal({var}))
> prior({mpg:}, flat) prior({var}, jeffreys)
> saving(length_simdata) burnin(3500)
note: adaptation option maxiter() changed to 35
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:length1 _cons} ~ 1 (flat)                                        (1)
                {var} ~ jeffreys
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_mpg.
Bayesian normal regression                      MCMC iterations  =     13,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      3,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2865
                                                Efficiency:  min =      .0771
                                                             avg =      .07938
Log marginal likelihood =  -198.7678                         max =      .08286
------------------------------------------------------------------------------
             |                                                 Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
     length1 |  -2.069861   .1882345   .006539  -2.068094   -2.44718  -1.706264
       _cons |   60.20346   3.562119   .127411   60.20927   53.34306   67.22423
-------------+----------------------------------------------------------------
         var |   12.88852   2.273808   .081887   12.62042   9.169482   18.16685
------------------------------------------------------------------------------
file length_simdata.dta saved
. estimates store length
```

We fit the third model containing only covariate `weight1` and store its results as `weight`:

```
. set seed 14
. bayesmh mpg weight1, likelihood(normal({var}))
> prior({mpg:}, flat) prior({var}, jeffreys)
> saving(weight_simdata) burnin(3500)
note: adaptation option maxiter() changed to 35
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:weight1 _cons} ~ 1 (flat)                                        (1)
                {var} ~ jeffreys
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_mpg.
Bayesian normal regression                      MCMC iterations  =     13,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      3,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .1735
                                                Efficiency:  min =      .0463
                                                             avg =      .06694
Log marginal likelihood = -198.20751                         max =      .07989
------------------------------------------------------------------------------
             |                                                Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
     weight1 |  -.6014409   .0506121   .001791  -.6013071  -.6996976    -.50121
       _cons |   39.45934   1.574673   .057646   39.49735   36.31386   42.33547
-------------+----------------------------------------------------------------
         var |   12.13997   2.141741   .099534   11.87332   8.883221   17.14041
------------------------------------------------------------------------------
file weight_simdata.dta saved
. estimates store weight
```

Finally, we fit the last model containing both covariates and store its results as `full`:

```
. set seed 14
. bayesmh mpg weight1 length1, likelihood(normal({var}))
> prior({mpg:}, flat) prior({var}, jeffreys)
> saving(full_simdata) burnin(3500)
note: adaptation option maxiter() changed to 35
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:weight1 length1 _cons} ~ 1 (flat)                                  (1)
                        {var} ~ jeffreys
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_mpg.
Bayesian normal regression                      MCMC iterations  =     13,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      3,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2323
                                                Efficiency:  min =     .05455
                                                             avg =     .06647
Log marginal likelihood = -196.86195                         max =     .08085
------------------------------------------------------------------------------
                                                              Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
     weight1 |  -.3977027   .1580411   .005558  -.401646  -.6965175  -.0721332
     length1 |  -.7599159   .5546754   .021944  -.7502182  -1.907818   .3106868
       _cons |    47.5913   6.132597   .262563    47.5656   35.89593   60.18002
-------------+----------------------------------------------------------------
         var |   11.81753    1.96315    .07608   11.59273   8.729182   16.14065
------------------------------------------------------------------------------
file full_simdata.dta saved
. estimates store full
```

▷ Example 1: Computing posterior probabilities of models

We now use `bayestest model` to compute posterior probabilities of the four models.

```
. bayestest model meanonly length weight full
Bayesian model tests
------------------------------------------------
             |    log(ML)       P(M)     P(M|y)
-------------+----------------------------------
    meanonly | -234.6462     0.2500     0.0000
      length | -198.7678     0.2500     0.1055
      weight | -198.2075     0.2500     0.1848
        full | -196.8619     0.2500     0.7097
------------------------------------------------
Note: Marginal likelihood (ML) is computed using
      Laplace-Metropolis approximation.
```

The mean-only model is very unlikely compared with other models. The length and weight models are somewhat likely with the respective posterior probabilities of 0.11 and 0.18, and the full model has the highest posterior probability of 0.71.

◁

▷ Example 2: Specifying prior probabilities of models

If we have some prior knowledge about each of the models, we can use the `prior()` option to specify prior probabilities for each model. For example, suppose that we have prior knowledge that the weight model is much more likely than the full model so that the prior probabilities are 0.1 for the mean-only model and the length model, 0.6 for the weight model, and only 0.2 for the full model.

```
. bayestest model meanonly length weight full, prior(0.1 0.1 0.6 0.2)
Bayesian model tests
```

	log(ML)	P(M)	P(M|y)
meanonly	-234.6462	0.1000	0.0000
length	-198.7678	0.1000	0.0401
weight	-198.2075	0.6000	0.4210
full	-196.8619	0.2000	0.5389

```
Note: Marginal likelihood (ML) is computed using
      Laplace-Metropolis approximation.
```

Under the specified prior, posterior probabilities of the weight and full models are now more similar: 0.42 and 0.54, respectively, but the full model is still preferable.

The above is equivalent to the following prior specification:

```
. bayestest model meanonly length weight full, prior(0.1 0.1 0.6)
```
(*output omitted*)

◁

Using our results, we conclude that `mpg` is related to both `weight` and `length` and would proceed with the full model.

After your analysis, remember to erase the saved simulation datasets you no longer need. For example, we erase all of them by typing

```
. erase meanonly_simdata.dta
. erase weight_simdata.dta
. erase length_simdata.dta
. erase full_simdata.dta
```

Comparing models with different priors

In the previous section, we used `bayestest model` to compare nested hypotheses about which covariates to include in the regression function. We can use `bayestest model` to compare models with not only different covariates but also different outcome distributions and priors for parameters.

We continue our analysis of `auto.dta`, but for simplicity, we now consider the mean-only model for `mpg`. Let's compare models with two slightly different informative priors. We use an informative normal–inverse-gamma prior for both models,

$$(\beta_0|\sigma^2) \sim N(\mu_0, \sigma^2/n_0)$$
$$\sigma^2 \sim \text{InvGamma}(\nu_0/2, \nu_0\sigma_0^2/2)$$

with $\mu_0 = 25$, $n_0 = 10$, and $\sigma_0^2 = 30$, but we consider two different values for the degrees of freedom: $\nu_0 = 5$ and $\nu_0 = 1$.

We use bayesmh to fit our models. Following the formulas, we specify a normal() prior for the constant {mpg:_cons} (mean parameter) and an inverse-gamma prior igamma() for the variance parameter {var}. We specify an expression for the variance of the normal prior distribution in parentheses.

We fit the first model with $\nu_0 = 5$ and store its estimation results as informative1.

```
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:}, normal(25,{var}/10))
> prior({var}, igamma(2.5,75)) saving(inf1_simdata)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
Priors:
  {mpg:_cons} ~ normal(25,{var}/10)
        {var} ~ igamma(2.5,75)
------------------------------------------------------------------------------
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2548
                                                Efficiency:  min =     .09065
                                                             avg =      .1049
Log marginal likelihood = -238.55856                         max =      .1192
------------------------------------------------------------------------------
             |                                                Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
       _cons |   21.71853   .6592655   .019091   21.69554   20.44644   23.04896
-------------+----------------------------------------------------------------
         var |   35.47405   5.823372   .193417   34.72454   25.84419     48.228
------------------------------------------------------------------------------
file inf1_simdata.dta saved
. estimates store informative1
```

We fit the second model with $\nu_0 = 1$ and store its estimation results as `informative2`.

```
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:}, normal(25,{var}/10))
> prior({var}, igamma(0.5,15)) saving(inf2_simdata)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
Priors:
  {mpg:_cons} ~ normal(25,{var}/10)
        {var} ~ igamma(0.5,15)
------------------------------------------------------------------------------

Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2261
                                                Efficiency:  min =      .0941
                                                             avg =       .109
Log marginal likelihood = -239.4049                          max =      .1239

------------------------------------------------------------------------------
             |                                                 Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
       _cons |   21.7175   .6539814   .021319    21.7295   20.47311   23.02638
-------------+----------------------------------------------------------------
         var |   35.89504   6.288571   .178665   35.17056   25.86084   50.21624
------------------------------------------------------------------------------

file inf2_simdata.dta saved
. estimates store informative2
```

▷ Example 3: Comparing models with informative priors

We now use `bayestest model` to compare our models with two different informative priors.

```
. bayestest model informative1 informative2
Bayesian model tests

------------------------------------------------
             |   log(ML)       P(M)     P(M|y)
-------------+----------------------------------
informative1 | -238.5586     0.5000     0.6998
informative2 | -239.4049     0.5000     0.3002
------------------------------------------------
Note: Marginal likelihood (ML) is computed using
      Laplace-Metropolis approximation.
```

Assuming that both models are equally likely a priori, the posterior probability of the `informative1` stored results, 0.70, is much higher than the probability of the `informative2` stored results, 0.3.

◁

▷ Example 4: Comparing a model with noninformative prior

A note of caution regarding comparing models with informative and noninformative priors—models with noninformative priors will often win because they are typically in most agreement with the observed data. For models with noninformative priors, most of the information about parameters is contained in a likelihood. As such, any model with an informative prior that is not in perfect agreement with the data will not fit data as well as a model with a noninformative prior.

For example, let's fit our constant-only model using a noninformative Jeffreys prior for the parameters.

```
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:}, flat) prior({var}, jeffreys)
> saving(jeffreys_simdata)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
Priors:
  {mpg:_cons} ~ 1 (flat)
        {var} ~ jeffreys
------------------------------------------------------------------------------

Bayesian normal regression                       MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling         Burn-in          =      2,500
                                                 MCMC sample size =     10,000
                                                 Number of obs    =         74
                                                 Acceptance rate  =      .2668
                                                 Efficiency:  min =     .09718
                                                              avg =      .1021
Log marginal likelihood =   -234.645                          max =      .1071

------------------------------------------------------------------------------
             |                                                Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
       _cons |  21.29222   .6828864   .021906   21.27898   19.99152   22.61904
-------------+----------------------------------------------------------------
         var |  34.76572    5.91534   .180754   34.18391    24.9129   47.61286
------------------------------------------------------------------------------

file jeffreys_simdata.dta saved
. estimates store jeffreys
```

Let's now compare this model with our two informative models.

```
. bayestest model informative1 informative2 jeffreys
Bayesian model tests
----------------------------------------------
             |   log(ML)       P(M)    P(M|y)
-------------+--------------------------------
informative1 | -238.5586     0.3333    0.0194
informative2 | -239.4049     0.3333    0.0083
    jeffreys | -234.6450     0.3333    0.9723
----------------------------------------------
Note: Marginal likelihood (ML) is computed using
      Laplace-Metropolis approximation.
```

The posterior probability of the Jeffreys model is 0.97.

◁

Finally, at the end of our analysis, we erase all the simulation datasets we no longer need. We erase all of them by typing

```
. erase inf1_simdata.dta
. erase inf2_simdata.dta
. erase jeffreys_simdata.dta
```

Stored results

`bayestest model` stores the following in `r()`:

Macros

`r(names)`	names of estimation results used
`r(marglmethod)`	method for approximating marginal likelihood: `lmetropolis` or `hmean`

Matrices

`r(test)`	test results for parameters in `r(names)`

Methods and formulas

Suppose we have r models M_j for $j = 1, \ldots, r$ with prior probabilities $P(M_j)$ such that $\sum_{j=1}^{r} p(M_j) = 1$. Then, posterior probability for model J is

$$P(M_j|\mathbf{y}) = \frac{P(\mathbf{y}|M_j)P(M_j)}{P(\mathbf{y})}$$

where $P(\mathbf{y}|M_j) = m_j(y)$ is the marginal likelihood of M_j with respect to $\mathbf{y}$, and $P(\mathbf{y}) = \sum_{j=1}^{r} P(\mathbf{y}|M_j)P(M_j)$. See *Methods and formulas* in [BAYES] **bayesmh** for details about computing marginal likelihood.

Also see

[BAYES] **bayesmh** — Bayesian regression using Metropolis–Hastings algorithm

[BAYES] **bayesmh postestimation** — Postestimation tools for bayesmh

[BAYES] **bayesstats ic** — Bayesian information criteria and Bayes factors

[BAYES] **bayesstats summary** — Bayesian summary statistics

[BAYES] **bayestest interval** — Interval hypothesis testing

Title

set clevel — Set default credible level

Description Syntax Option Remarks and examples Also see

Description

`set clevel` specifies the default credible level for credible intervals for all Bayesian commands (see [BAYES] **bayes**) that report credible intervals. The initial value is 95, meaning 95% credible intervals.

Syntax

```
set clevel # [, permanently]
```

is any number between 10.00 and 99.99 and may be specified with at most two digits after the decimal point.

Option

`permanently` specifies that in addition to making the change right now, the `clevel` setting be remembered and become the default setting when you invoke Stata.

Remarks and examples

To change the level of credible intervals reported by a particular command, you need not reset the default credible level. All commands that report credible intervals have a `clevel(#)` option. When you do not specify the option, the credible intervals are calculated for the default level set by `set clevel` or for 95% if you have not reset `set clevel`.

Example 1

We use the bayesmh command to obtain the credible interval for the mean of mpg:

```
. use http://www.stata-press.com/data/r14/auto
(1978 Automobile Data)
. set seed 14
. bayesmh mpg, likelihood(normal(30)) prior({mpg:_cons}, flat)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},30)
Prior:
  {mpg:_cons} ~ 1 (flat)
------------------------------------------------------------------------------

Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .4195
Log marginal likelihood = -234.09275            Efficiency       =      .2378

------------------------------------------------------------------------------
             |                                                 Equal-tailed
         mpg |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
       _cons |   21.30364   .6429995   .013186   21.30381    20.03481    22.5555
------------------------------------------------------------------------------
```

To obtain 90% credible intervals, we would type

```
. bayesmh, clevel(90)
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},30)
Prior:
  {mpg:_cons} ~ 1 (flat)
------------------------------------------------------------------------------

Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .4195
Log marginal likelihood = -234.09275            Efficiency       =      .2378

------------------------------------------------------------------------------
             |                                                 Equal-tailed
         mpg |      Mean   Std. Dev.     MCSE     Median  [90% Cred. Interval]
-------------+----------------------------------------------------------------
       _cons |   21.30364   .6429995   .013186   21.30381    20.24172    22.35158
------------------------------------------------------------------------------
```

or we could type

```
. set clevel 90
. bayesmh
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},30)
Prior:
  {mpg:_cons} ~ 1 (flat)
------------------------------------------------------------------------------

Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .4195
Log marginal likelihood = -234.09275            Efficiency       =      .2378

------------------------------------------------------------------------------
             |                                                 Equal-tailed
         mpg |      Mean   Std. Dev.     MCSE     Median  [90% Cred. Interval]
-------------+----------------------------------------------------------------
       _cons |   21.30364   .6429995   .013186   21.30381   20.24172   22.35158
------------------------------------------------------------------------------
```

If we opt for the second alternative, the next time that we fit a model, 90% credible intervals will be reported. If we wanted 95% credible intervals, we could specify `clevel(95)` on the estimation command, or we could reset the default by typing `set clevel 95`.

The current setting of `clevel()` is stored as the c-class value `c(clevel)`; see [P] **creturn**.

◁

Also see

[BAYES] **bayesmh** — Bayesian regression using Metropolis–Hastings algorithm

[R] **query** — Display system parameters

[P] **creturn** — Return c-class values

Glossary

a posteriori. In the context of Bayesian analysis, we use a posteriori to mean "after the sample is observed". For example, a posteriori information is any information obtained after the data sample is observed. See *posterior distribution, posterior*.

a priori. In the context of Bayesian analysis, we use a priori to mean "before the sample is observed". For example, a priori information is any information obtained before the data sample is observed. In a Bayesian model, a priori information about model parameters is specified by prior distributions.

acceptance rate. In the context of the MH algorithm, acceptance rate is the fraction of the proposed samples that is accepted. The optimal acceptance rate depends on the properties of the target distribution and is not known in general. If the target distribution is normal, however, the optimal acceptance rate is known to be 0.44 for univariate distributions and 0.234 for multivariate distributions.

adaptation. In the context of the MH algorithm, adaptation refers to the process of tuning or adapting the proposal distribution to optimize the MCMC sampling. Typically, adaptation is performed periodically during the MCMC sampling. The `bayesmh` command performs adaptation every # of iterations as specified in option `adaptation(every(#))` for a maximum of `adaptation(maxiter())` iterations. In a continuous-adaptation regimes, the adaptation lasts during the entire process of the MCMC sampling. See [BAYES] **bayesmh**.

adaptation period. Adaptation period includes all MH adaptive iterations. It equals the length of the adaptation interval, as specified by `adaptation(every())`, times the maximum number of adaptations, `adaptation(maxiter())`.

adaptive iteration. In the adaptive MH algorithm, adaptive iterations are iterations during which adaptation is performed.

Akaike information criterion, AIC. Akaike information criterion (AIC) is an information-based model-selection criterion. It is given by the formula $-2 \times \log$ likelihood $+ 2k$, where k is the number of parameters. AIC favors simpler models by penalizing for the number of model parameters. It does not, however, account for the sample size. As a result, the AIC penalization diminishes as the sample size increases, as does its ability to guard against overparameterization.

batch means. Batch means are means obtained from batches of sample values of equal size. Batch means provide an alternative method for estimating MCMC standard errors (MCSE). The batch size is usually chosen to minimize the correlation between different batches of means.

Bayes factor. Bayes factor is given by the ratio of the marginal likelihoods of two models, M_1 and M_2. It is a widely used criterion for Bayesian model comparison. Bayes factor is used in calculating the posterior odds ratio of model M_1 versus M_2,

$$\frac{P(M_1|\mathbf{y})}{P(M_2|\mathbf{y})} = \frac{P(\mathbf{y}|M_1)}{P(\mathbf{y}|M_2)}\frac{P(M_1)}{P(M_2)}$$

where $P(M_i|\mathbf{y})$ is a posterior probability of model M_i, and $P(M_i)$ is a prior probability of model M_i. When the two models are equally likely, that is, when $P(M_1) = P(M_2)$, the Bayes factor equals the posterior odds ratio of the two models.

Bayes's rule. The Bayes's rule is a formal method for relating conditional probability statements. For two (random) events X and Y, the Bayes's rule states that

$$P(X|Y) \propto P(Y|X)P(X)$$

that is, the probability of X conditional on Y is proportional to the probability of X and the probability of Y conditional on X. In Bayesian analysis, the Bayes's rule is used for combining prior information about model parameters and evidence from the observed data to form the posterior distribution.

Bayesian analysis. Bayesian analysis is a statistical methodology that considers model parameters to be random quantities and estimates their posterior distribution by combining prior knowledge about parameters with the evidence from the observed data sample. Prior knowledge about parameters is described by prior distributions and evidence from the observed data is incorporated through a likelihood model. Using the Bayes's rule, the prior distribution and the likelihood model are combined to form the posterior distribution of model parameters. The posterior distribution is then used for parameter inference, hypothesis testing, and prediction.

Bayesian hypothesis testing. Bayesian hypothesis testing computes probabilities of hypotheses conditional on the observed data. In contrast to the frequentist hypothesis testing, the Bayesian hypothesis testing computes the actual probability of a hypothesis H by using the Bayes's rule,

$$P(H|\mathbf{y}) \propto P(\mathbf{y}|H)P(H)$$

where $\mathbf{y}$ is the observed data, $P(\mathbf{y}|H)$ is the marginal likelihood of $\mathbf{y}$ given H, and $P(H)$ is the prior probability of H. Two different hypotheses, H_1 and H_2, can be compared by simply comparing $P(H_1|\mathbf{y})$ to $P(H_2|\mathbf{y})$.

Bayesian information criterion, BIC. The Bayesian information criterion (BIC), also known as Schwarz criterion, is an information based criterion used for model selection in classical statistics. It is given by the formula $-0.5 \times \log$ likelihood $+ k \times \ln n$, where k is the number of parameters and n is the sample size. BIC favors simpler, in terms of complexity, models and it is more conservative than AIC.

blocking. In the context of the MH algorithm, blocking refers to the process of separating model parameters into different subsets or blocks to be sampled independently of each other. MH algorithm generates proposals and applies the acceptance–rejection rule sequentially for each block. It is recommended that correlated parameters are kept in one block. Separating less-correlated or independent model parameters in different blocks may improve the mixing of the MH algorithm.

burn-in period. The burn-in period is the number of iterations it takes for an MCMC sequence to reach stationarity.

central posterior interval. See *equal-tailed credible interval*.

conditional conjugacy. See *semiconjugate prior*.

conjugate prior. A prior distribution is conjugate for a family of likelihood distributions if the prior and posterior distributions belong to the same family of distributions. For example, the gamma distribution is a conjugate prior for the Poisson likelihood. Conjugacy may provide an efficient way of sampling from posterior distributions and is used in Gibbs sampling.

continuous parameters. Continuous parameters are parameters with continuous prior distributions.

credible interval. In Bayesian analysis, the credible interval of a scalar model parameter is an interval from the domain of the marginal posterior distribution of that parameter. Two types of credible intervals are typically used in practice: equal-tailed credible intervals and HPD credible intervals.

credible level. The credible level is a probability level between 0% and 100% used for calculating credible intervals in Bayesian analysis. For example, a 95% credible interval for a scalar parameter is an interval the parameter belongs to with the probability of 95%.

cusum plot, CUSUM plot. The cusum (CUSUM) plot of an MCMC sample is a plot of cumulative sums of the differences between sample values and their overall mean against the iteration number. Cusum plots are useful graphical summaries for detecting early drifts in MCMC samples.

deviance information criterion, DIC. The deviance information criterion (DIC) is an information based criterion used for Bayesian model selection. It is an analog of AIC and is given by the formula $D(\overline{\theta}) + 2 \times p_D$, where $D(\overline{\theta})$ is the deviance at the sample mean and p_D is the effective complexity, a quantity equivalent to the number of parameters in the model. Models with smaller DIC are preferred.

diminishing adaptation. Diminishing adaptation of the adaptive algorithm is the type of adaptation in which the amount of adaptation decreases with the size of the MCMC chain.

discrete parameters. Discrete parameters are parameters with discrete prior distributions.

effective sample size, ESS. Effective sample size (ESS) is the MCMC sample size T adjusted for the autocorrelation in the sample. It represents the number of independent observations in an MCMC sample. ESS is used instead of T in calculating MCSE. Small ESS relative to T indicates high autocorrelation and consequently poor mixing of the chain.

efficiency. In the context of MCMC, efficiency is a term used for assessing the mixing quality of an MCMC procedure. Efficient MCMC algorithms are able to explore posterior domains in less time (using fewer iterations). Efficiency is typically quantified by the sample autocorrelation and effective sample size. An MCMC procedure that generates samples with low autocorrelation and consequently high ESS is more efficient.

equal-tailed credible interval. An equal-tailed credible interval is a credible interval defined in such a way that both tails of the marginal posterior distribution have the same probability. A $\{100 \times (1-\alpha)\}\%$ equal-tailed credible interval is defined by the $\alpha/2$th and $\{(1-\alpha)/2\}$th quantiles of the marginal posterior distribution.

feasible initial value. An initial-value vector is feasible if it corresponds to a state with a positive posterior probability.

frequentist analysis. Frequentist analysis is a form of statistical analysis where model parameters are considered to be unknown but fixed constants and the observed data are viewed as a repeatable random sample. Inference is based on the sampling distribution of the data.

full conditionals. A full conditional is the probability distribution of a random variate conditioned on all other random variates in a joint probability model. Full conditional distributions are used in Gibbs sampling.

full Gibbs sampling. See *Gibbs sampling, Gibbs sampler*.

Gibbs sampling, Gibbs sampler. Gibbs sampling is an MCMC method, according to which each random variable from a joint probability model is sampled according to its full conditional distribution.

highest posterior density credible interval, HPD credible interval. The highest posterior density (HPD) credible interval is a type of a credible interval with the highest marginal posterior density. An HPD interval has the shortest width among all other credible intervals. For some multimodal marginal distributions, HPD may not exists. See *highest posterior density region, HPD region*.

highest posterior density region, HPD region. The highest posterior density (HPD) region for model parameters has the highest marginal posterior probability among all domain regions. Unlike an HPD credible interval, an HPD region always exist.

hybrid MH sampling, hybrid MH sampler. A hybrid MH sampler is an MCMC method in which some blocks of parameters are updated using the MH algorithms and other blocks are updated using Gibbs sampling.

hyperparameter. In Bayesian analysis, hyperparameter is a parameter of a prior distribution, in contrast to a model parameter.

hyperprior. In Bayesian analysis, hyperprior is a prior distribution of hyperparameters. See *hyperparameter*.

improper prior. A prior is said to be improper if it does not integrate to a finite number. Uniform distributions over unbounded intervals are improper. Improper priors may still yield proper posterior distributions. When using improper priors, however, one has to make sure that the resulting posterior distribution is proper for Bayesian inference to be invalid.

independent a posteriori. Parameters are considered independent a posteriori if their marginal posterior distributions are independent; that is, their joint posterior distribution is the product of their individual marginal posterior distributions.

independent a priori. Parameters are considered independent a priori if their prior distributions are independent; that is, their joint prior distribution is the product of their individual marginal prior distributions.

informative prior. An informative prior is a prior distribution that has substantial influence on the posterior distribution.

interval hypothesis testing. Interval hypothesis testing performs interval hypothesis tests for model parameters and functions of model parameters.

interval test. In Bayesian analysis, an interval test applied to a scalar model parameter calculates the marginal posterior probability for the parameter to belong to the specified interval.

Jeffreys prior. The Jeffreys prior of a vector of model parameters $\boldsymbol{\theta}$ is proportional to the square root of the determinant of its Fisher information matrix $I(\boldsymbol{\theta})$. Jeffreys priors are locally uniform and, by definition, agree with the likelihood function. Jeffreys priors are considered noninformative priors that have minimal impact on the posterior distribution.

marginal distribution. In Bayesian context, a distribution of the data after integrating out parameters from the joint distribution of the parameters and the data.

marginal likelihood. In the context of Bayesian model comparison, a marginalized over model parameters $\boldsymbol{\theta}$ likelihood of data $\mathbf{y}$ for a given model M, $P(\mathbf{y}|M) = m(\mathbf{y}) = \int P(\mathbf{y}|\boldsymbol{\theta}, M)P(\boldsymbol{\theta}|M)d\boldsymbol{\theta}$. Also see *Bayes factor*.

marginal posterior distribution. In Bayesian context, a marginal posterior distribution is a distribution resulting from integrating out all but one parameter from the joint posterior distribution.

Markov chain. Markov chain is a random process that generates sequences of random vectors (or states) and satisfies the Markov property: the next state depends only on the current state and not on any of the previous states. MCMC is the most common methodology for simulating Markov chains.

matrix model parameter. A matrix model parameter is any model parameter that is a matrix. Matrix elements, however, are viewed as scalar model parameters.

Matrix model parameters are defined and referred to within the `bayesmh` command as {*param*,`matrix`} or {*eqname*:*param*,`matrix`} with the equation name *eqname*. For example, `{Sigma, matrix}` and `{Scale:Omega, matrix}` are matrix model parameters. Individual matrix elements cannot be referred to within the `bayesmh` command, but they can be referred within postestimation commands accepting parameters. For example, to refer to the individual elements of the defined above, say, 2×2 matrices, use `{Sigma_1_1}`, `{Sigma_2_1}`, `{Sigma_1_2}`, `{Sigma_2_2}` and `{Scale:Omega_1_1}`, `{Scale:Omega_2_1}`, `{Scale:Omega_1_2}`, `{Scale:Omega_2_2}`, respectively. See [BAYES] **bayesmh**.

matrix parameter. See *matrix model parameter*.

MCMC, Markov chain Monte Carlo. MCMC is a class of simulation-based methods for generating samples from probability distributions. Any MCMC algorithm simulates a Markov chain with a target distribution as its stationary or equilibrium distribution. The precision of MCMC algorithms increases with the number of iterations. The lack of a stopping rule and convergence rule, however, makes it difficult to determine for how long to run MCMC. The time needed to converge to the target distribution within a prespecified error is referred to as mixing time. Better MCMC algorithms have faster mixing times. Some of the popular MCMC algorithms are random-walk Metropolis, Metropolis–Hastings, and Gibbs sampling.

MCMC sample. An MCMC sample is obtained from MCMC sampling. An MCMC sample approximates a target distribution and is used for summarizing this distribution.

MCMC sample size. MCMC sample size is the size of the MCMC sample. It is specified in `bayesmh`'s option `mcmcsize()`; see [BAYES] **bayesmh**.

MCMC sampling, MCMC sampler. MCMC sampling is an MCMC algorithm that generates samples from a target probability distribution.

MCMC standard error, MCSE MCSE is the standard error of the posterior mean estimate. It is defined as the standard deviation divided by the square root of ESS. MCSEs are analogs of standard errors in frequentist statistics and measure the accuracy of the simulated MCMC sample.

Metropolis–Hastings (MH) sampling, MH sampler. A Metropolis–Hastings (MH) sampler is an MCMC method for simulating probability distributions. According to this method, at each step of the Markov chain, a new proposal state is generated from the current state according to a prespecified proposal distribution. Based on the current and new state, an acceptance probability is calculated and then used to accept or reject the proposed state. Important characteristics of MH sampling is the acceptance rate and mixing time. The MH algorithm is very general and can be applied to an arbitrary target distribution. However, its efficiency is limited, in terms of mixing time, and decreases as the dimension of the target distribution increases. Gibbs sampling, when available, can provide much more efficient sampling than MH sampling.

mixing of Markov chain. Mixing refers to the rate at which a Markov chain traverses the parameter space. It is a property of the Markov chain that is different from convergence. Poor mixing indicates a slow rate at which the chain explores the stationary distribution and will require more iterations to provide inference at a given precision. Poor (slow) mixing is typically a result of high correlation between model parameters or of weakly-defined model specifications.

model hypothesis testing. Model hypothesis testing tests hypotheses about models by computing model posterior probabilities.

model parameter. A model parameter refers to any (random) parameter in a Bayesian model. Model parameters can be scalars or matrices. Examples of model parameters as defined in `bayesmh` are `{mu}`, `{scale:s}`, `{Sigma,matrix}`, and `{Scale:Omega,matrix}`. See [BAYES] **bayesmh** and, specifically, *Declaring model parameters* and *Referring to model parameters* in that entry. Also see *Different ways of specifying model parameters* in [BAYES] **bayesmh postestimation**.

model posterior probability. Model posterior probability is probability of a model M computed conditional on the observed data $\mathbf{y}$,

$$P(M|\mathbf{y}) = P(M)P(\mathbf{y}|M) = P(M)m(\mathbf{y})$$

where $P(M)$ is the prior probability of a model M and $m(\mathbf{y})$ is the marginal likelihood under model M.

noninformative prior. A noninformative prior is a prior with negligible influence on the posterior distribution. See, for example, *Jeffreys prior*.

objective prior. See *noninformative prior*.

one-at-a-time MCMC sampling. A one-at-a-time MCMC sample is an MCMC sampling procedure in which random variables are sampled individually, one at a time. For example, in Gibbs sampling, individual variates are sampled one at a time, conditionally on the most recent values of the rest of the variates.

posterior distribution, posterior. A posterior distribution is a probability distribution of model parameters conditional on observed data. The posterior distribution is determined by the likelihood of the parameters and their prior distribution. For a parameter vector $\boldsymbol{\theta}$ and data $\mathbf{y}$, the posterior distribution is given by

$$P(\boldsymbol{\theta}|\mathbf{y}) = \frac{P(\boldsymbol{\theta})P(\mathbf{y}|\boldsymbol{\theta})}{P(\mathbf{y})}$$

where $P(\boldsymbol{\theta})$ is the prior distribution, $P(\mathbf{y}|\boldsymbol{\theta})$ is the model likelihood, and $P(\mathbf{y})$ is the marginal distribution for $\mathbf{y}$. Bayesian inference is based on a posterior distribution.

posterior independence. See *independent a posteriori*.

posterior interval. See *credible interval*.

posterior odds. Posterior odds for $\boldsymbol{\theta}_1$ compared with $\boldsymbol{\theta}_2$ is the ratio of posterior density evaluated at $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ under a given model,

$$\frac{p(\boldsymbol{\theta}_1|\mathbf{y})}{p(\boldsymbol{\theta}_2|\mathbf{y})} = \frac{p(\boldsymbol{\theta}_1)}{p(\boldsymbol{\theta}_2)}\frac{p(\mathbf{y}|\boldsymbol{\theta}_1)}{p(\mathbf{y}|\boldsymbol{\theta}_2)}$$

In other words, posterior odds are prior odds times the likelihood ratio.

posterior predictive distribution. A posterior predictive distribution is a distribution of unobserved (future) data conditional on the currently observed data. Posterior predictive distribution is derived by marginalizing the likelihood function with respect to the posterior distribution of model parameters.

prior distribution, prior. In Bayesian statistics, prior distributions are probability distributions of model parameters formed based on some a priori knowledge about parameters. Prior distributions are independent of the observed data.

prior independence. See *independent a priori*.

prior odds. Prior odds for $\boldsymbol{\theta}_1$ compared with $\boldsymbol{\theta}_2$ is the ratio of prior density evaluated at $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ under a given model, $p(\boldsymbol{\theta}_1)/p(\boldsymbol{\theta}_2)$. Also see *posterior odds*.

proposal distribution. In the context of the MH algorithm, a proposal distribution is used for defining the transition steps of the Markov chain. In the standard random-walk Metropolis algorithm, the proposal distribution is a multivariate normal distribution with zero mean and adaptable covariance matrix.

pseudoconvergence. A Markov chain may appear to converge when in fact it did not. We refer to this phenomenon as pseudoconvergence. Pseudoconvergence is typically caused by multimodality of the stationary distribution, in which case the chain may fail to traverse the weakly connected regions of the distribution space. A common way to detect pseudoconvergence is to run multiple chains using different starting values and to verify that all of the chain converge to the same target distribution.

reference prior. See *noninformative prior*.

scalar model parameter. A scalar model parameter is any model parameter that is a scalar. For example, `{mean}` and `{hape:alpha}` are scalar parameters, as declared by the `bayesmh` command. Elements of matrix model parameters are viewed as scalar model parameters. For example, for

a 2×2 matrix parameter {Sigma,matrix}, individual elements {Sigma_1_1}, {Sigma_2_1}, {Sigma_1_2}, and {Sigma_2_2} are scalar parameters. If a matrix parameter contains a label, the label should be included in the specification of individual elements as well. See [BAYES] **bayesmh**.

scalar parameter. See *scalar model parameter*.

semiconjugate prior. A prior distribution is semiconjugate for a family of likelihood distributions if the prior and (full) conditional posterior distributions belong to the same family of distributions. For semiconjugacy to hold, parameters must typically be independent a priori; that is, their joint prior distribution must be the product of the individual marginal prior distributions. For example, the normal prior distribution for a mean parameter of a normal data distribution with an unknown variance (which is assumed to be independent of the mean a priori) is a semiconjugate prior. Semiconjugacy may provide an efficient way of sampling from posterior distributions and is used in Gibbs sampling.

stationary distribution. Stationary distribution of a stochastic process is a joint distribution that does not change over time. In the context of MCMC, stationary distribution is the target probability distribution to which the Markov chain converges. When MCMC is used for simulating a Bayesian model, the stationary distribution is the target joint posterior distribution of model parameters.

subjective prior. See *informative prior*.

subsampling the chain. See *thinning*.

thinning. Thinning is a way of reducing autocorrelation in the MCMC sample by subsampling the MCMC chain every prespecified number of iterations determined by the thinning interval. For example, the thinning interval of 1 corresponds to using the entire MCMC sample; the thinning interval of 2 corresponds to using every other sample value; and the thinning interval of 3 corresponds to using values from iterations 1, 4, 7, 10, and so on. Thinning should be applied with caution when used to reduce autocorrelation because it may not always be the most appropriate way of improving the precision of estimates.

vague prior. See *noninformative prior*.

valid initial state. See *feasible initial value*.

vanishing adaptation. See *diminishing adaptation*.

Zellner's g-prior. Zellner's g-prior is a form of a weakly informative prior for the regression coefficients in a linear model. It accounts for the correlation between the predictor variables and controls the impact of the prior of the regression coefficients on the posterior with parameter g. For example, $g = 1$ means that prior weight is 50% and $g \to \infty$ means diffuse prior.

Subject and author index

See the combined subject index and the combined author index in the *Glossary and Index*.