Title

stata.com

bayesmh — Bayesian regression using Metropolis–Hastings algorithm

Description Quick start Menu Syntax
Options Remarks and examples Stored results Methods and formulas
References Also see

Description

`bayesmh` fits a variety of Bayesian models using an adaptive 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)))
```

Random-intercept logistic regression of y1 on x1 with group variable gr and zero-mean normal prior with variance parameter {var} for the random-intercept parameters

```
bayesmh y1 x1, likelihood(logit) reffects(gr)            ///
     prior({y1:i.gr}, normal(0, {var}))                  ///
     prior({y1: x1 _cons}, flat) prior({var}, jeffreys)
```

Menu

Statistics > Bayesian analysis > Estimation

Syntax

Univariate linear models

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

Probability distributions

Univariate distributions

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

Multiple-equation distribution specifications

```
bayesmh (deqspec) [(deqspec)] [...] [if] [in] [weight],
       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*]) [...]

The syntax of *deqspec* is

[*eqname*:] *depvar* [*if*] [*in*] [*weight*], likelihood(*distribution*)

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
binomial(*n*)	binomial regression with logit link and number of trials *n*
binlogit(*n*)	synonym for binomial()
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`

distribution	Description
`dexponential(`*beta*`)`	exponential distribution with scale parameter *beta*
`dbernoulli(`*p*`)`	Bernoulli distribution with success probability *p*
`dbinomial(`*p*`,`*n*`)`	binomial distribution with success probability *p* and number of trials *n*
`dpoisson(`*mu*`)`	Poisson distribution with mean *mu*

A distribution argument is a model parameter, *paramspec*, or a substitutable expression, *subexpr*, containing model parameters. An *n* argument may be a number; an expression, *expr*; or a variable name, *varname*. See *Specifying arguments of likelihood models and prior distributions*.

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*.

When `mvnormal()` or `mvnormal0()` of dimension *d* is applied to *paramref* with *n* parameters ($n \neq d$), *paramref* is reshaped into a matrix with *d* columns, and its rows are treated as independent samples from the specified `mvnormal()` distribution. If such reshaping is not possible, an error is issued. See example 25 for application of this feature.

options	Description
Model	
noconstant	suppress constant term; not allowed with ordered models, nonlinear models, and probability distributions
*likelihood(*lspec*)	distribution for the likelihood model
*prior(*priorspec*)	prior for model parameters; this option may be repeated
dryrun	show model summary without estimation
Model 2	
redefine(*label*:i.*varname*)	specify a random-effects linear form; this option may be repeated
xbdefine(*label*:*varlist*)	specify a linear form
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
initrandom	specify random initial 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
showreffects(*paramref*)	specify random-effects parameters to be included in 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(), redefine(), 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
reffects	requests that all parameters in a block be treated as random-effects parameters
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 of these 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(`*lspec*`)` specifies the distribution of the data. This option specifies the likelihood portion of the Bayesian model. This option is required. *lspec* is one of *modelspec* or *distribution*.

modelspec specifies one of the supported likelihood distributions for regression models. 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*.

distribution specifies one of the supported distributions for modeling the dependent variable. A distribution argument must be a model parameter specified in `{}` or a substitutable expression containing model parameters; see *Declaring model parameters* and *Specifying arguments of likelihood models and prior distributions*. A number of trials, *n*, of the binomial distribution may be a real number (#), a Stata expression, or a variable name. For an example of modeling outcome distributions directly, see *Beta-binomial model*.

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

`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`, `binomial()`, `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`.

`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

`reffects(`*varname*`)` specifies a random-effects variable, a variable identifying the group structure for the random effects, with univariate linear models. This option is useful for fitting two-level random-intercept models. A random-effects variable is treated as a factor variable with no base level. As such, you can refer to random-effects parameters or, simply, random effects associated with *varname* using a conventional factor-variable notation. For example, you can use {*depvar*`:i.`*varname*`}` to refer to all random-effects parameters of *varname*. These parameters must be included in a single prior statement, usually a normal distribution with variance specified by an additional parameter. The random-effects parameters are assumed to be conditionally independent across levels of *varname* given all other model parameters. The random-effects parameters are automatically grouped in one block and are thus not allowed in the `block()` option. See example 23.

`redefine(`*label*`:i.`*varname*`)` specifies a random-effects linear form that can be used in substitutable expressions. You can use {*label*`:`} to refer to the linear form in substitutable expressions. You can specify {*label*`:i.`*varname*} to refer to all random-effects parameters associated with *varname*. The random-effects parameters are automatically grouped in one block and are thus not allowed in the `block()` option. This option is useful for fitting multilevel models and can be repeated. See example 29.

`xbdefine(`*label*`:`*varlist*`)` specifies a linear form of the variables in *varlist* that can be used in substitutable expressions. You can use the specification {*label*`:`} to refer to the linear form in substitutable expressions. For any *varname* in *varlist*, you can use {*label*`:`*varname*} to refer to the corresponding parameter. This option is useful with nonlinear specifications when the linear form contains many variables and provides more efficient computation in such cases.

`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`, `reffects`, `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 `reffects`, `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.

`reffects` specifies that the parameters associated with the levels of a factor variable included in the likelihood specification be treated as random-effects parameters. Random-effects parameters must be included in one prior statement and are assumed to be conditionally independent across levels of a grouping variable given all other model parameters. `reffects` requires that parameters be specified as {*depvar*`:i.`*varname*`}`, where `i.`*varname* is the corresponding factor variable in the likelihood specification, and may not be combined with `block()`'s suboptions `gibbs` and `split`. This option is useful for fitting hierarchical or multilevel models. See example 25 for details.

`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. This option cannot be combined with `initrandom`.

`initrandom` requests that the model parameters be initialized randomly. Random initial values are generated from the prior distributions of the model parameters. If you want to use fixed initial values for some of the parameters, you can specify them in the `initial()` option or during parameter declarations in the `likelihood()` option. Random initial values are not available for parameters with `flat`, `density()`, `logdensity()`, and `jeffreys()` priors; you must provide fixed initial values for such parameters. This option cannot be combined with `nomleinitial`.

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 for these parameters and log marginal likelihood will not be available. 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.

showreffects(*paramref*) is used with option reffects() and specifies a list of random-effects parameters to be included in the output. By default, all random-effects parameters introduced by reffects() are excluded from the output as if you have specified the noshow() option.

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

stata.com

Remarks are presented under the following headings:

Using bayesmh
Setting up a posterior model
  Likelihood model
  Prior distributions
  Declaring model parameters
  Referring to model parameters
  Specifying arguments of likelihood models and prior distributions
  Substitutable expressions
  Checking model specification
Specifying MCMC sampling procedure
  Reproducing results
  Burn-in period and MCMC sample size
  Improving efficiency of the MH algorithm—blocking of parameters
  Gibbs and hybrid MH sampling
  Adaptation of the MH algorithm
  Specifying initial values
Summarizing and reporting results
  Posterior summaries and credible intervals
  Saving MCMC results
Convergence of MCMC

Examples are presented under the following headings:

Getting started examples
  Mean of a normal distribution with a known variance
  Mean of a normal distribution with an unknown variance
  Simple linear regression
  Multiple linear regression
  Improving efficiency of MH sampling
  Graphical diagnostics using multiple chains
Logistic regression model: a case of nonidentifiable parameters
Ordered probit regression
Beta-binomial model
Multivariate regression
Panel-data and multilevel models

Two-level random-intercept model or panel-data model
Linear growth curve model—a random-coefficient model
Mixed-effects logistic regression
Bayesian analysis of change-point problem
Bioequivalence in a crossover trial
Random-effects meta-analysis of clinical trials
Item response theory

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 primarily 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() ...
```

You can also model an outcome distribution directly by specifying one of the supported probability distributions.

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 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 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:x}, logdensity(ln(2.5)-ln(2.5^2+{y:x}^2)-ln(_pi)))
> prior({y:_cons}, logdensity(ln(10)-ln(10^2+{y:_cons}^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.

In addition to fixed initial values, you may request random initial values for all model parameters by specifying the initrandom option. Random initial values are generated from the prior distributions of the parameters, except for parameters that are assigned flat, density(), logdensity(), or jeffreys() prior distributions. For such parameters, you must specify fixed initial values, or bayesmh will issue an error. See *Graphical diagnostics using multiple chains* for an example.

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. Also, the Laplace–Metropolis approximation for the log marginal likelihood will not be available because its computation requires simulation results for all model parameters.

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**.

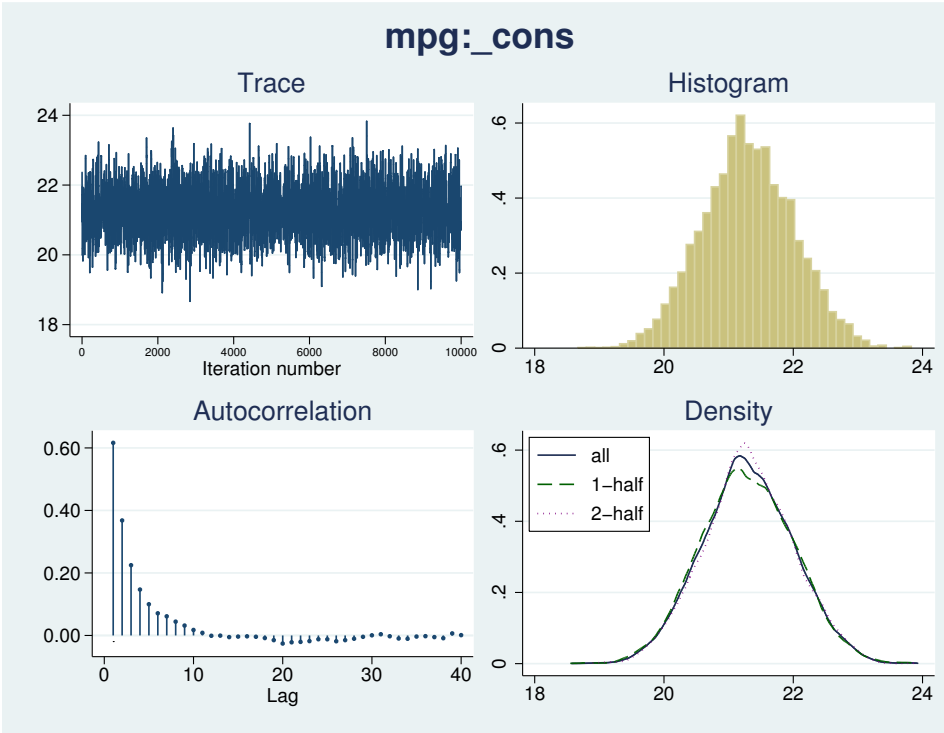
Because we used a completely noninformative prior, our results should be the same as frequentist results. In this Bayesian model, the posterior distribution of the constant parameter is known to be normal with a mean equal to the sample average. In the frequentist domain, the MLE of the constant is also 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 *Graphical diagnostics using multiple chains* for an example of using multiple chains to assess convergence. Also 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.

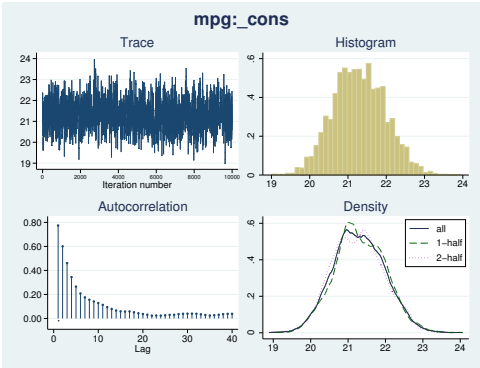
```
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, flat) prior({var}, jeffreys)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
Priors:
  {mpg:_cons} ~ 1 (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 $\mu_{\text{weight}} = -0.5$, $\mu_{\text{cons}} = 40$, $\nu_0 = 10$, 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

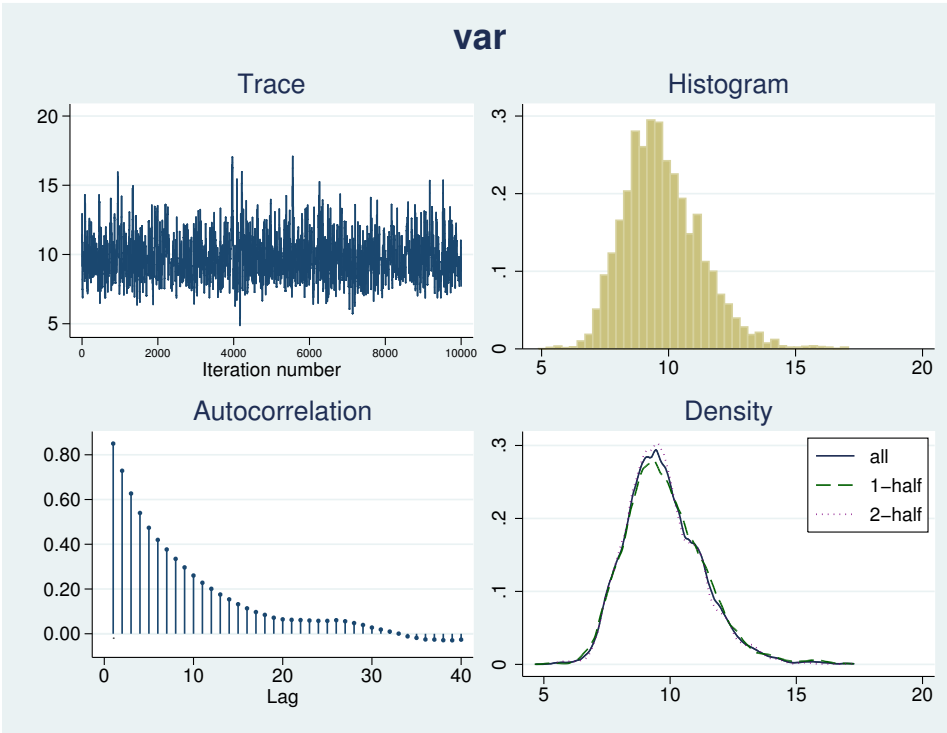
Our first simulation is performed using the default settings for the algorithm. Specifically, all three model parameters are placed in one simulation block and are updated simultaneously, as our block summary indicates.

```
. 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}
```



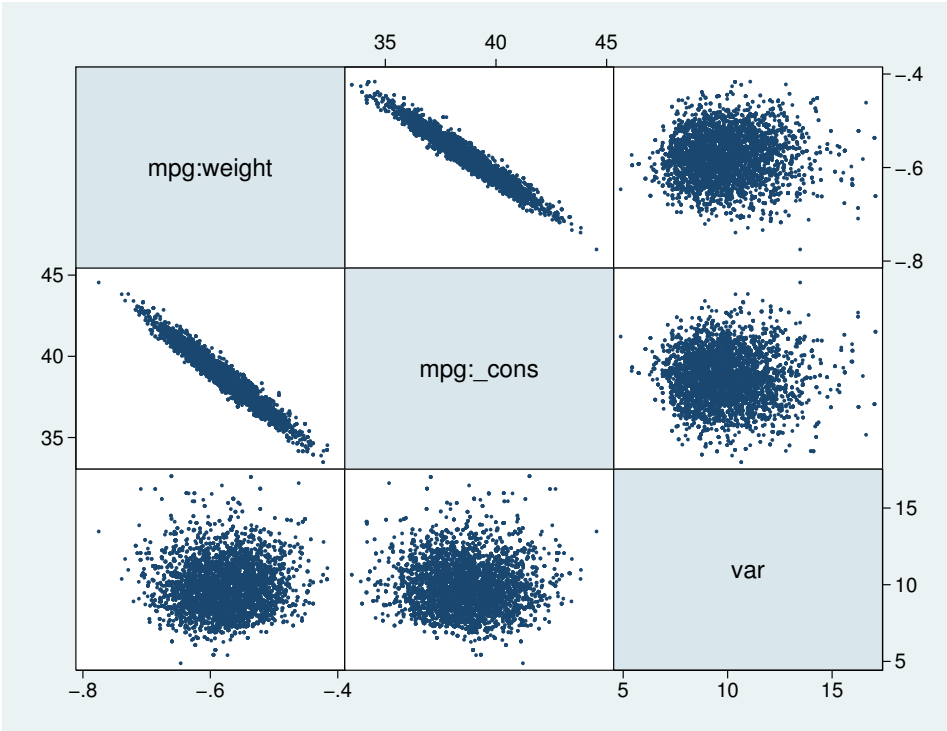
A graphical summary for the `{var}` parameter does not show any obvious problems. The trace plot reveals a good coverage of the domain of the marginal distribution, while the histogram and kernel density plots resemble the shape of an expected inverse-gamma distribution. The autocorrelation dies off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

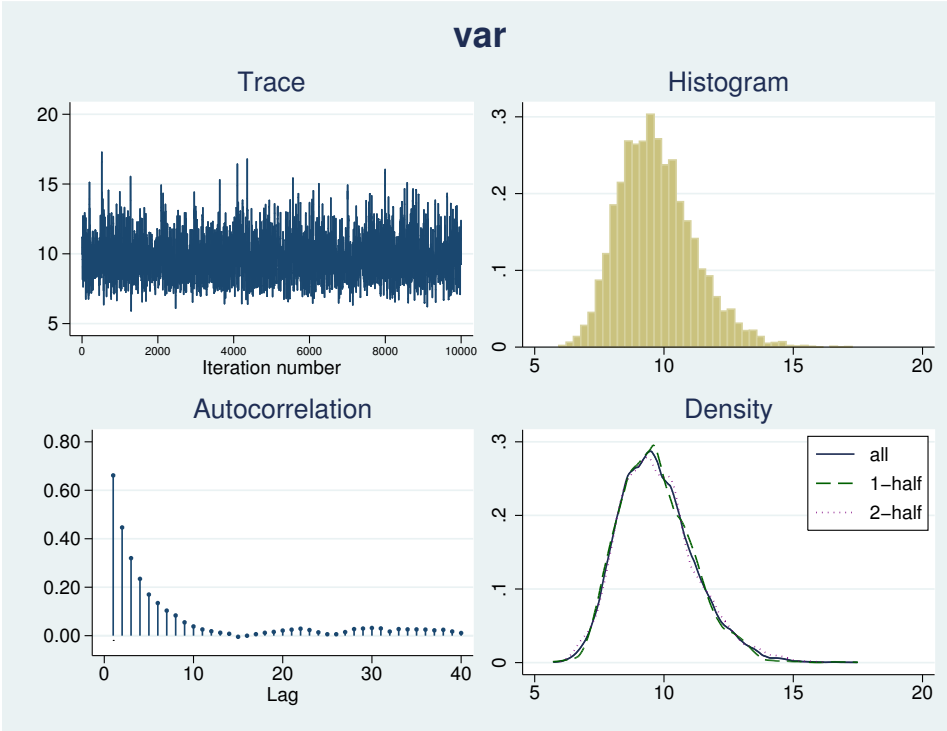
In cases like this, we can expect higher sampling efficiency if we place {var} in a separate block. We can do this by including the option `block({var})`. The other two parameters, {mpg:weight} and {mpg:_cons}, will be automatically considered as a second block.

```
. 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.

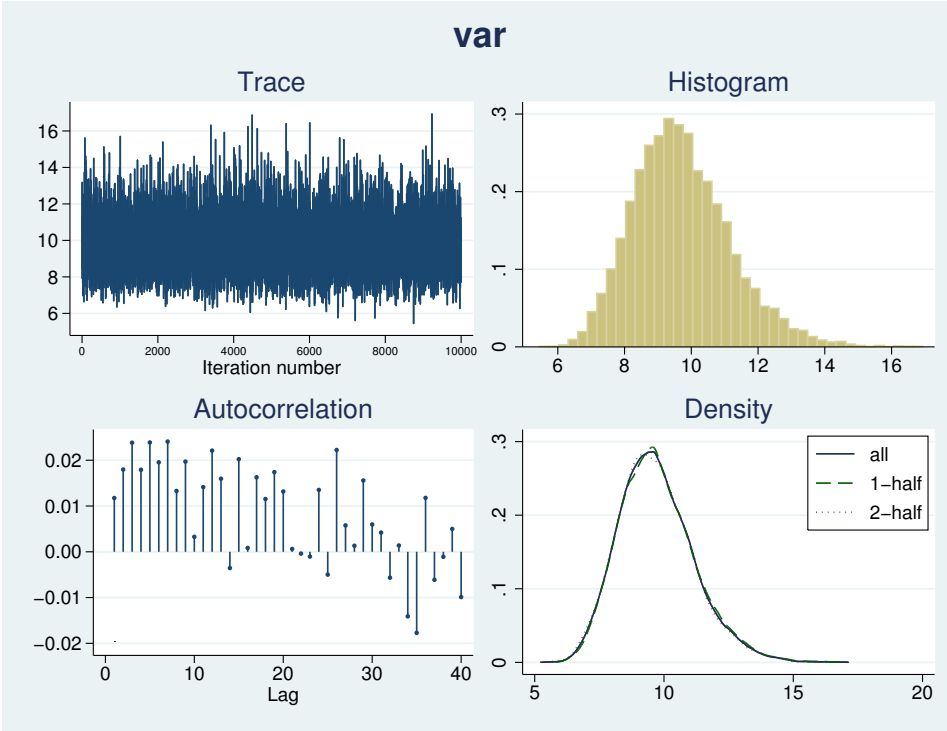
To request Gibbs sampling, we specify suboption `gibbs` within option `block()`.

```
. 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

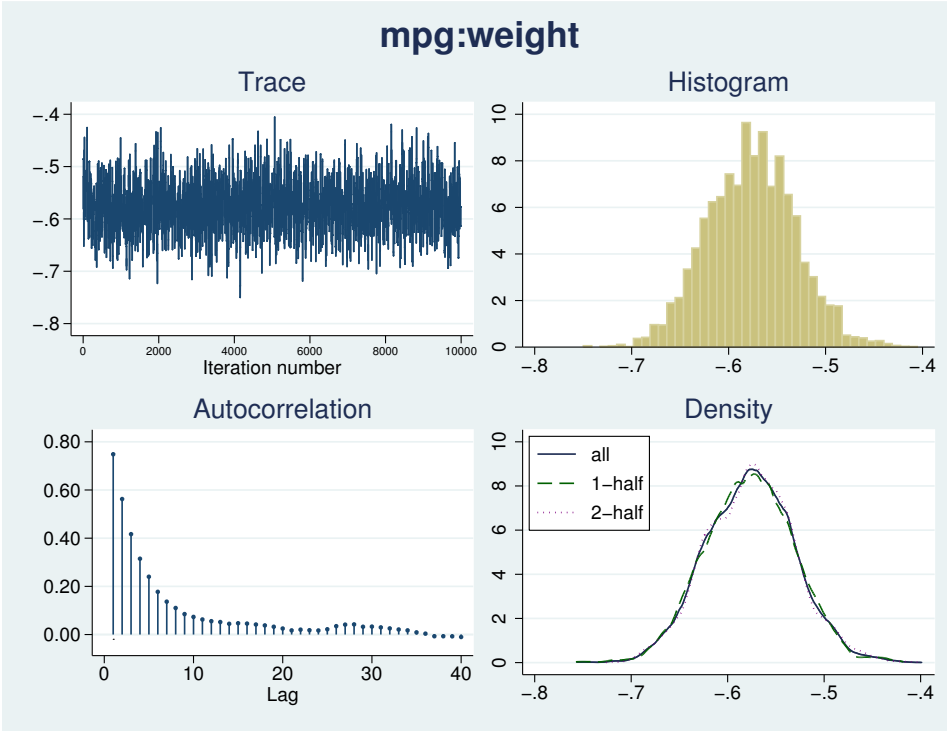
Continuing example 13, there is still room for improvement in our model in terms of sampling efficiency. The efficiency of the regression coefficients is now low relative to the variance efficiency.

```
. 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.

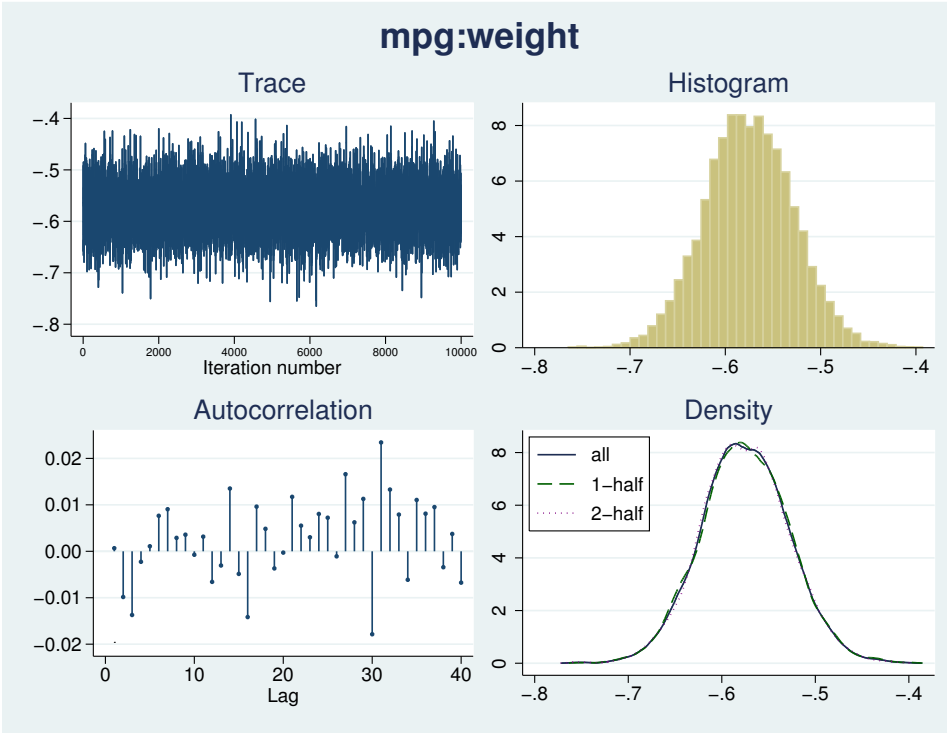
To request Gibbs sampling for the regression coefficients, we must place them in a separate block.

```
. 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)
```

◁

Graphical diagnostics using multiple chains

To assess the convergence of MCMC simulations of a Bayesian model, the literature often recommends comparing the results of multiple simulation sequences; see, for example, chapter 11.4 in Gelman et al. (2014). In this section, we show how one can perform multiple simulation runs using bayesmh and visually compare the results using trace plots.

We use a Bayesian multiple linear regression model from example 11. For brevity, we simulate only two MCMC chains, but the approach can be easily extended to more than two chains. It is essential for the two chains to have different initial values dispersed over the range of model parameter values. With bayesmh, you can provide fixed initial values by using the initial() option or when declaring parameters in the likelihood() option, or you can request random initial values by specifying the initrandom option.

We simulate two samples of size 5,000 and save the simulation results as `sim1.dta` and `sim2.dta`. Below we show the `bayesmh` specifications and the estimation results.

```
. set seed 14

. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, normal(0,100)) prior({var}, igamma(10,10))
> mcmcsize(5000) nomodelsummary initrandom saving(sim1)
Burn-in ...
Simulation ...

Bayesian normal regression                      MCMC iterations  =      7,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =      5,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2597
                                                Efficiency:  min =     .06487
                                                             avg =     .07218
Log marginal likelihood = -226.83819                         max =     .07653

------------------------------------------------------------------------------
             |                                                 Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
      weight |  -.5779557   .0436132   .002422  -.5769825  -.6645717  -.4929187
       _cons |   38.66309   1.379251   .070509   38.61303   36.02204   41.29802
-------------+----------------------------------------------------------------
         var |   9.813336   1.439047   .074239   9.698644   7.313202   13.01305
------------------------------------------------------------------------------

file sim1.dta saved

. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, normal(0,100)) prior({var}, igamma(10,10))
> mcmcsize(5000) nomodelsummary initrandom saving(sim2)
Burn-in ...
Simulation ...

Bayesian normal regression                      MCMC iterations  =      7,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =      5,000
                                                Number of obs    =         74
                                                Acceptance rate  =        .24
                                                Efficiency:  min =     .09555
                                                             avg =     .09985
Log marginal likelihood = -226.83719                         max =      .1048

------------------------------------------------------------------------------
             |                                                 Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
      weight |  -.5733346   .0462179   .002019  -.5763298  -.6637416  -.4810475
       _cons |   38.56422   1.426843   .065279   38.62277   35.72788   41.38445
-------------+----------------------------------------------------------------
         var |    9.60628   1.361131   .061125   9.47494    7.28224   12.59361
------------------------------------------------------------------------------

file sim2.dta saved
```

The average simulation efficiency of both runs is above 7% and seems adequate. There is no indication of convergence problems. Nevertheless, inspecting the trace plots can provide additional reassurance. In particular, by comparing the trace plots of a model parameter based on different simulation sequences, we can detect convergence irregularities and assess the overlap of the simulated marginal distributions for this parameter. If the MCMC chains have converged, we should not observe substantial differences between the trace plots or between the sampled marginal distributions.

To draw overlaid trace plots, we need to combine the two simulation datasets in one dataset in the long form. We load the first simulation dataset and append the second simulation dataset to the first one. We also generate an indicator variable `chain`. The variable `chain` equals 0 for the records of the first chain and 1 for the records of the second chain.

```
. clear
. use sim1
. append using sim2, generate(chain)
```

In example 11, we rescaled variable `weight` of the `auto` dataset and thus modified the data in memory. We used the `clear` option when we loaded the `sim1` dataset to replace the data in memory, even though the current data have not been saved to disk.

To avoid duplicates, we save the simulation results in a compressed form, recording unique values and their frequencies. We need to expand the dataset and create a unique iteration number for each chain. In the original simulation datasets, the index and duplicates are stored in the `_index` and `_frequency` variables, respectively. We expand the combined dataset using the `_frequency` variable and sort it by using `chain` and `_index`. Then, we generate the variable `iter` to index the records in the two chains.

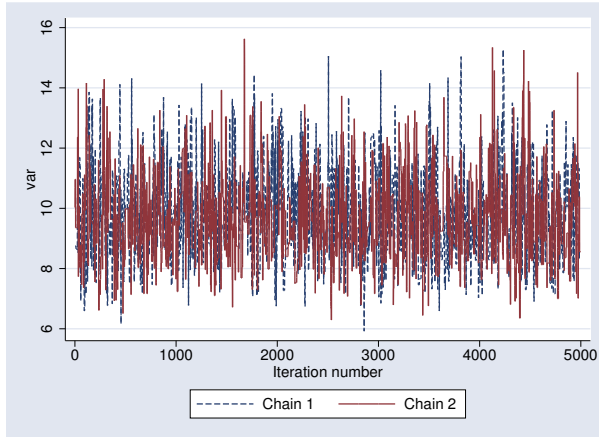
```
. expand _frequency
(7,500 observations created)
. sort chain _index
. by chain: generate iter = _n
```

Finally, we relabel the variables of interest to match the model parameter names. The scalar model parameter names are stored in `e(scparams)`, and their corresponding variable names in the simulation dataset are stored in `e(postvars)`.

```
. display e(scparams)
mpg:weight mpg:_cons var
. display e(postvars)
eq1_p1 eq1_p2 eq0_p1
. label variable eq1_p1 "mpg:weight"
. label variable eq1_p2 "mpg:_cons"
. label variable eq0_p1 "var"
. label variable iter "Iteration number"
```

We are now ready to draw the trace plots of the model parameters. For example, we can overlay the two trace plots of the {var} parameter as follows. We use the `xtset` command to declare the data to be panel data identified by the `chain` variable and specify `iter` as a time variable. We then draw the trace plots using the time-series plotting command `tsline`.

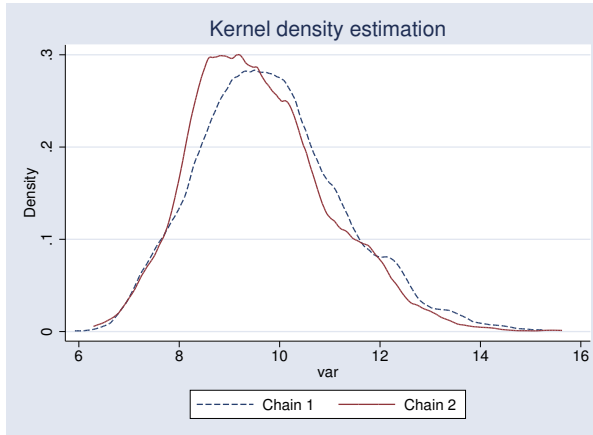
```
. xtset chain iter
       panel variable:  chain (strongly balanced)
        time variable:  iter, 1 to 5000
                delta:  1 unit
. twoway (tsline eq0_p1 if chain==0, lpattern(-))
>        (tsline eq0_p1 if chain==1, lpattern(l)),
>        legend(label(1 "Chain 1") label(2 "Chain 2"))
```



The two trace plots of {var} look similar and seem to cover approximately the same marginal posterior domain. However, the variability in the first chain does seem slightly greater, which is also evident from the reported standard deviations of {var}, 1.44 in the first run and 1.36 in the second. The estimated posterior means of {var} are somewhat different, 9.81 and 9.61, which suggests that we should either run longer MCMC simulations or improve the sampling efficiency, as we demonstrated in example 12, example 13, and example 14.

Overlaid density plots using kdensity provide another aspect of comparing multiple simulation sequences.

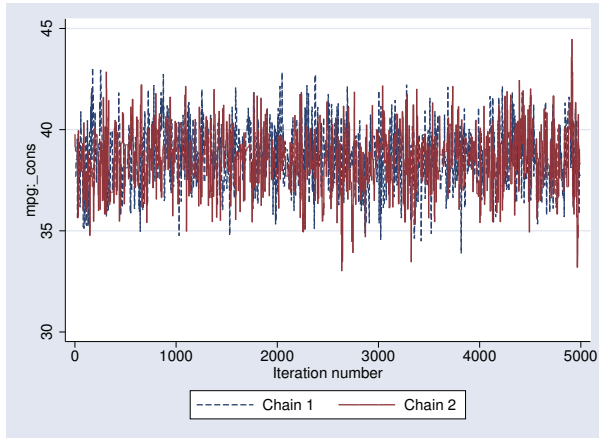
```
. twoway (kdensity eq0_p1 if chain==0, lpattern(-))
>        (kdensity eq0_p1 if chain==1, lpattern(l)),
>        legend(label(1 "Chain 1") label(2 "Chain 2"))
>        xtitle("var") ytitle("Density") title("Kernel density estimation")
```



The overlaid kdensity plots of {var} based on the two simulated chains clearly show that although the domains of the simulated marginal distributions overlap, there are also noticeable differences in the distribution of mass, which thus confirms the need for longer MCMC runs.

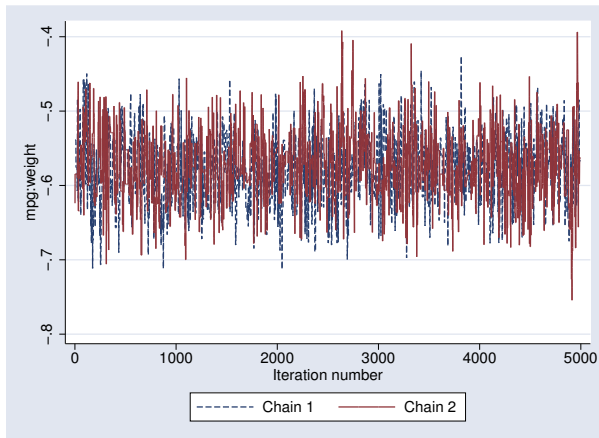
Similarly, we can draw the overlaid trace plots for parameter {mpg:_cons}.

```
. twoway (tsline eq1_p2 if chain==0, lpattern(-))
>        (tsline eq1_p2 if chain==1, lpattern(l)),
>        legend(label(1 "Chain 1") label(2 "Chain 2"))
```



The overlaid trace plots of the {mpg:_cons} parameter do not show any substantial differences or any convergence problems. However, increasing the MCMC sample sizes will further diminish the difference between the estimated posterior means of {mpg:_cons}, 38.66 and 38.56. Now, let's draw the overlaid plots for {mpg:weight}.

```
. twoway (tsline eq1_p1 if chain==0, lpattern(-))
>        (tsline eq1_p1 if chain==1, lpattern(l)),
>        legend(label(1 "Chain 1") label(2 "Chain 2"))
```



Again the overlaid trace plots of the {mpg:weight} parameter do not show any substantial differences or any convergence problems.

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
------------------------------------------------------------------------------
```

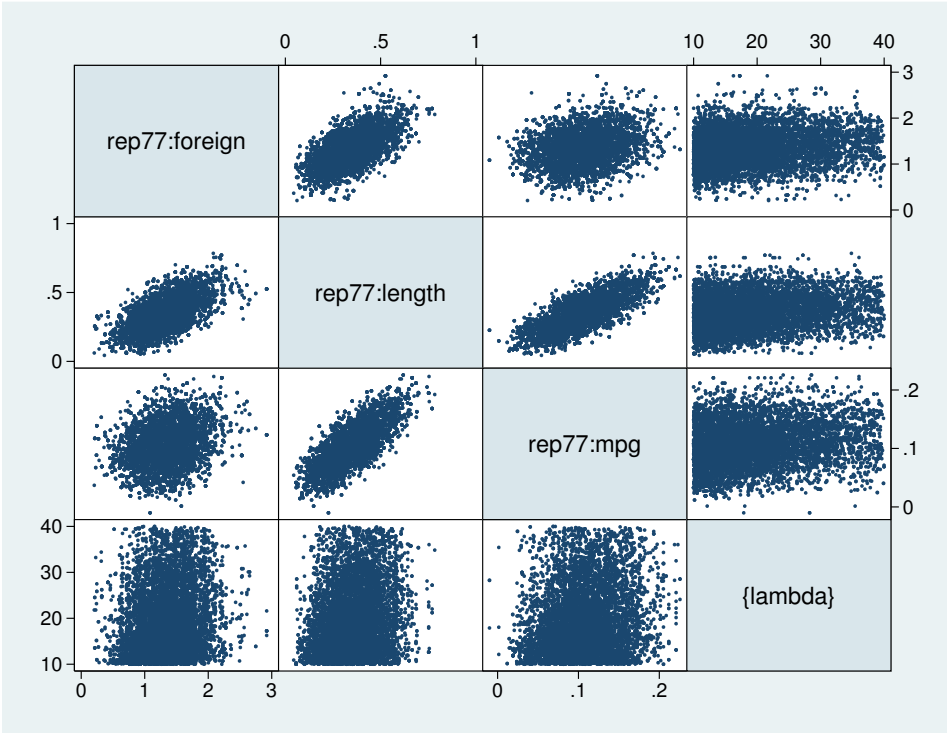
When we specify `dots` or `dots()`, `bayesmh` displays dots as simulation is performed. The burn-in and simulation iterations are displayed separately. During the adaptation period, iterations are displayed with a symbol `a` instead of a dot. This indicates the period during which the proposal distribution 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.

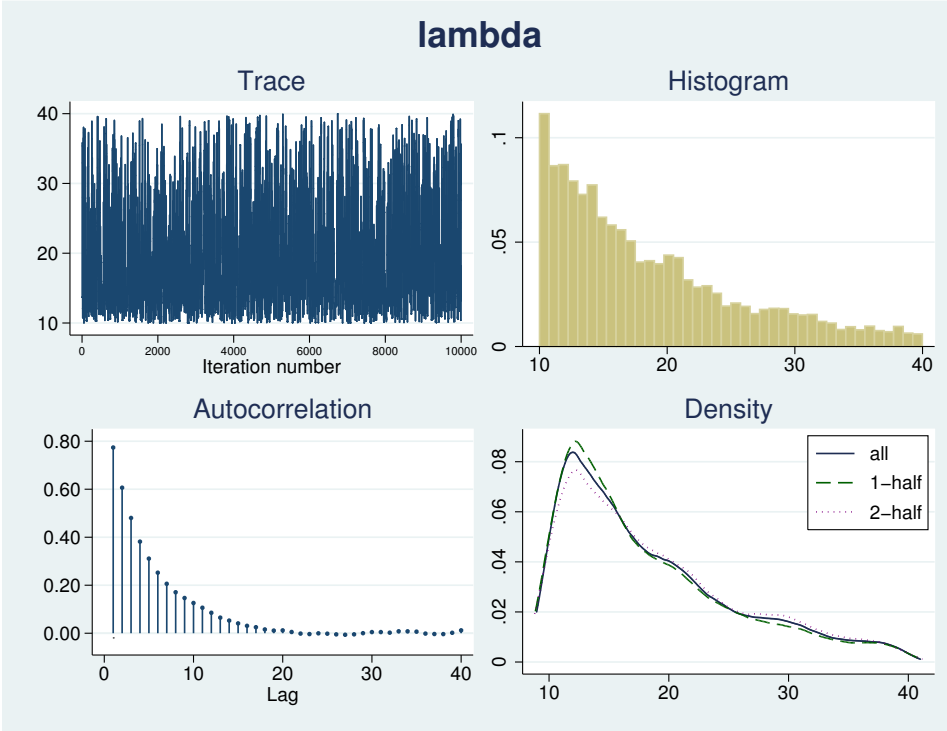
We placed the hyperparameter `{lambda}` in a separate block because we wanted to sample this nuisance parameter independently from the other model parameters. Based on the bivariate scatterplots, this parameter does appear to be independent of other model parameters a posteriori.

```
. 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 specifying one of the four distributions supported by bayesmh in the likelihood() option: dexponential(), dbernoulli(), dbinomial(), or dpoisson().

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 the option likelihood(dbinomial({theta},20)). The infection probability θ is represented by {theta}.

```
. set obs 1
number of observations (_N) was 0, now 1
. generate y = 0
. set seed 14
. bayesmh y, likelihood(dbinomial({theta},20))
> prior({theta}, beta(2,20)) initial({theta} 0.01)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  y ~ binomial({theta},20)
Prior:
  {theta} ~ beta(2,20)
------------------------------------------------------------------------------

Bayesian binomial model                         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
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
       theta |  .0467973   .0317862   .000808   .039931   .0051255   .1277823
------------------------------------------------------------------------------
```

The estimated posterior mean for {theta} 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.

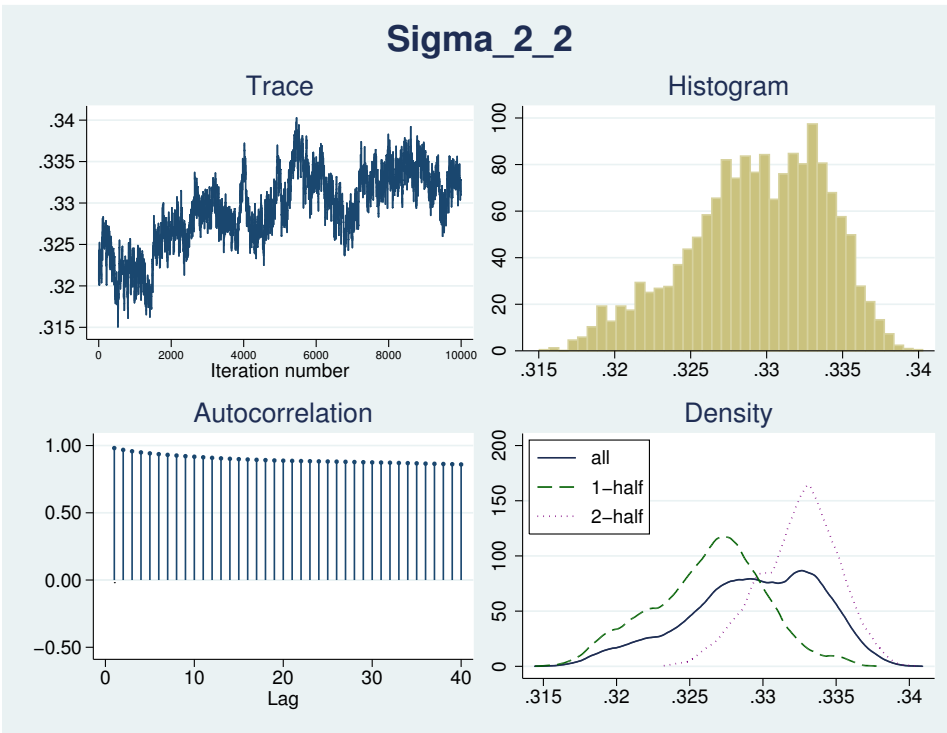
A closer inspection of the ESS table reveals very low sampling efficiencies for the elements of the covariance matrix `{Sigma}`.

```
. 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%.

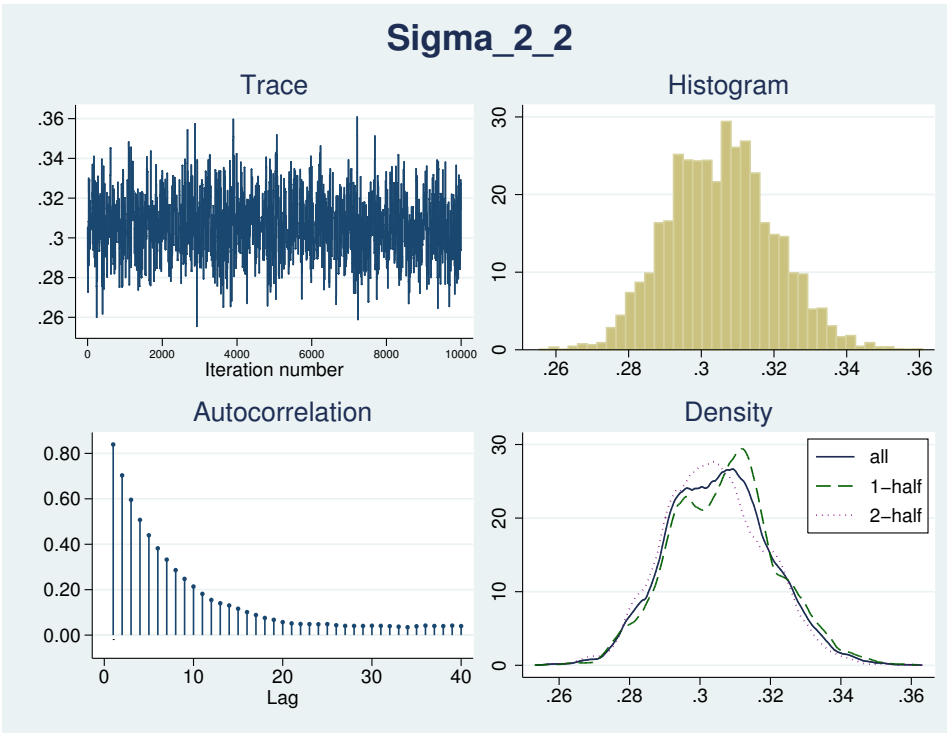
Sampling efficiencies of the elements of the covariance matrix improved substantially.

```
. 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.

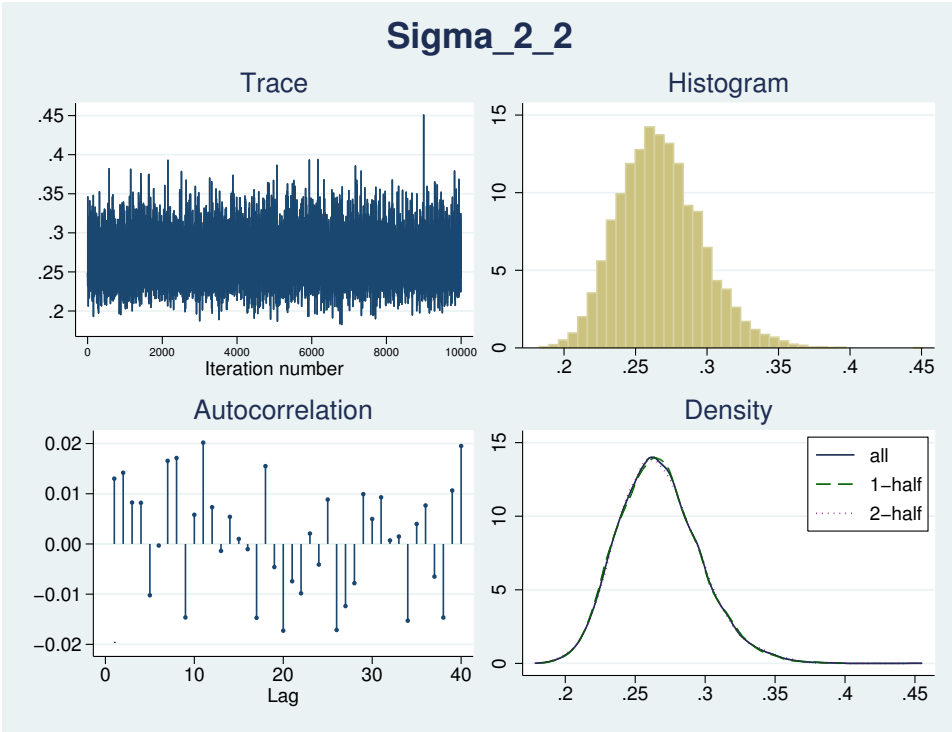
```
. set seed 14
. bayesmh (mpg) (weight) (length), likelihood(mvnormal({Sigma,m}))
> prior({mpg:_cons} {weight:_cons} {length:_cons}, normal(0,100))
> prior({Sigma,m}, iwishart(3,100,I(3)))
> block({mpg:_cons} {weight:_cons} {length:_cons})
> block({Sigma,m}, gibbs) dots
Burn-in 2500 aaaaaaaaa1000aaaaaaaaa2000aaa.. 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  =      .5942
                                                 Efficiency:  min =     .06842
                                                              avg =      .6659
Log marginal likelihood = -240.48717                          max =      .9781
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
mpg						
_cons	2.128801	.0457224	.00164	2.128105	2.041016	2.215
weight						
_cons	3.020533	.0609036	.002328	3.021561	2.908383	3.143715
length						
_cons	1.880409	.0197061	.000725	1.881133	1.843106	1.918875
Sigma_1_1	.150733	.0164464	.000166	.1495231	.1219304	.1869429
Sigma_2_1	-.1571622	.0196803	.000201	-.156005	-.1995812	-.1224243
Sigma_3_1	-.0443725	.0060229	.000061	-.0439466	-.0571876	-.0338685
Sigma_2_2	.2673525	.029205	.0003	.2654589	.2163041	.3305366
Sigma_3_2	.0708095	.0085435	.000087	.0702492	.0557448	.0893794
Sigma_3_3	.0273506	.0029932	.000031	.0271362	.0220723	.0337994

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.

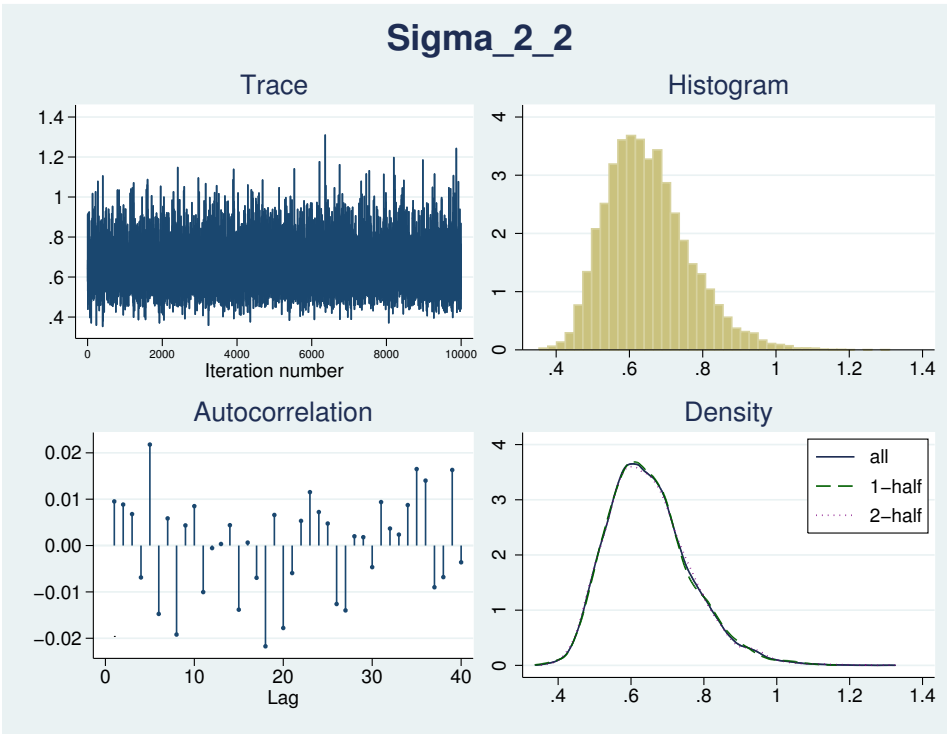
❑ Technical note: Adaptation tolerance is not met

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 five 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.

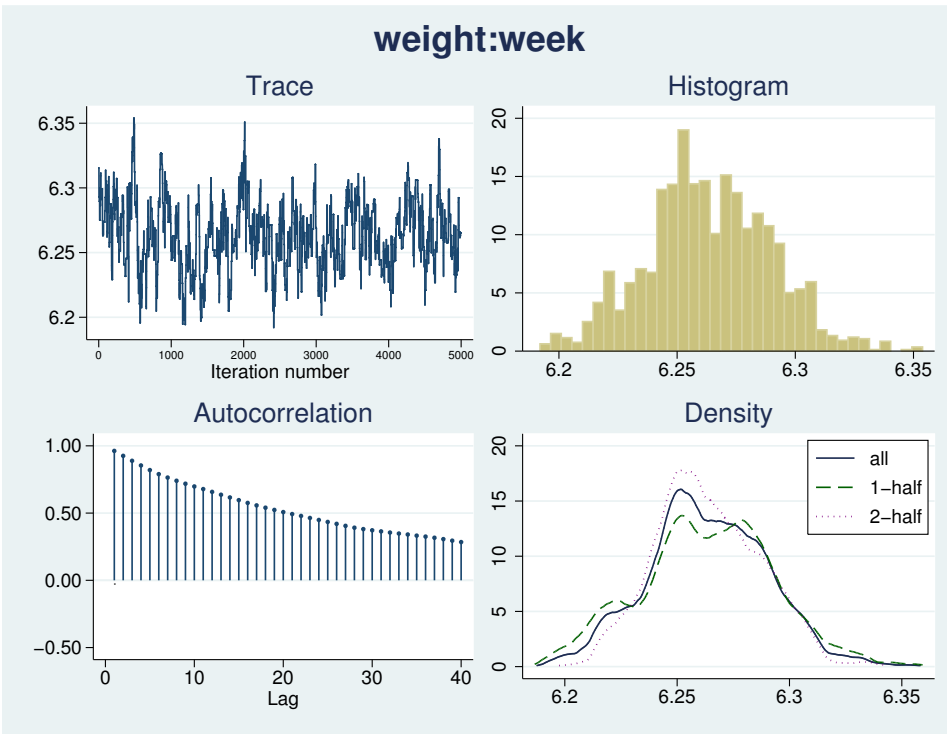
```
. bayesstats summary {weight:week _cons} {var_0} {var_id}
Posterior summary statistics                       MCMC sample size =     5,000
```

	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.

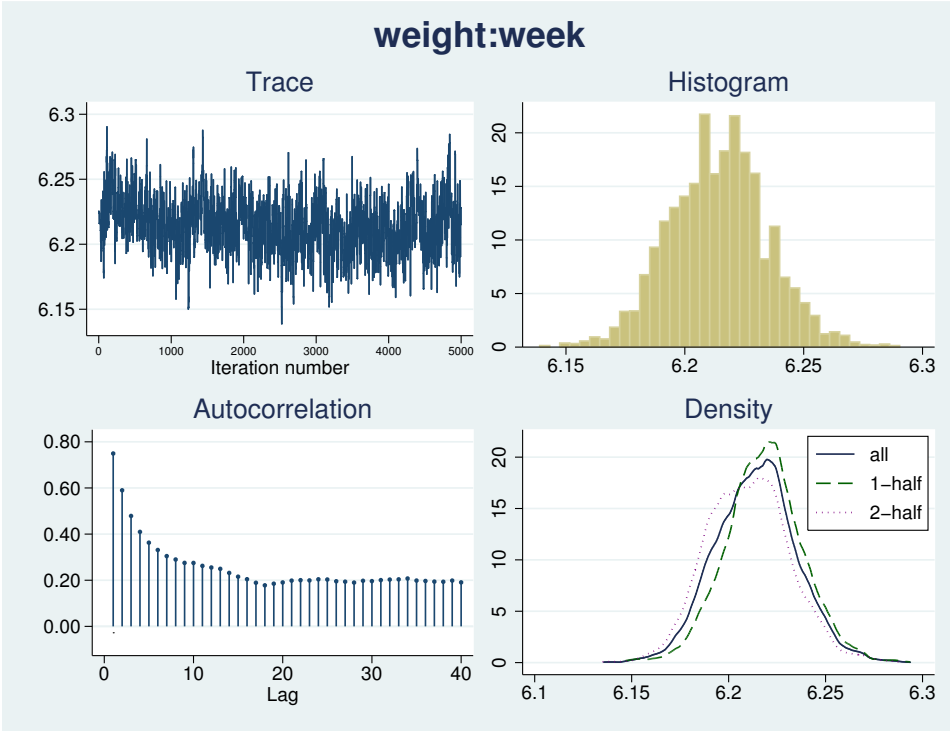
```
. bayesstats summary {weight:week _cons} {var_0} {var_id}
Posterior summary statistics                     MCMC sample size =     5,000
```

	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

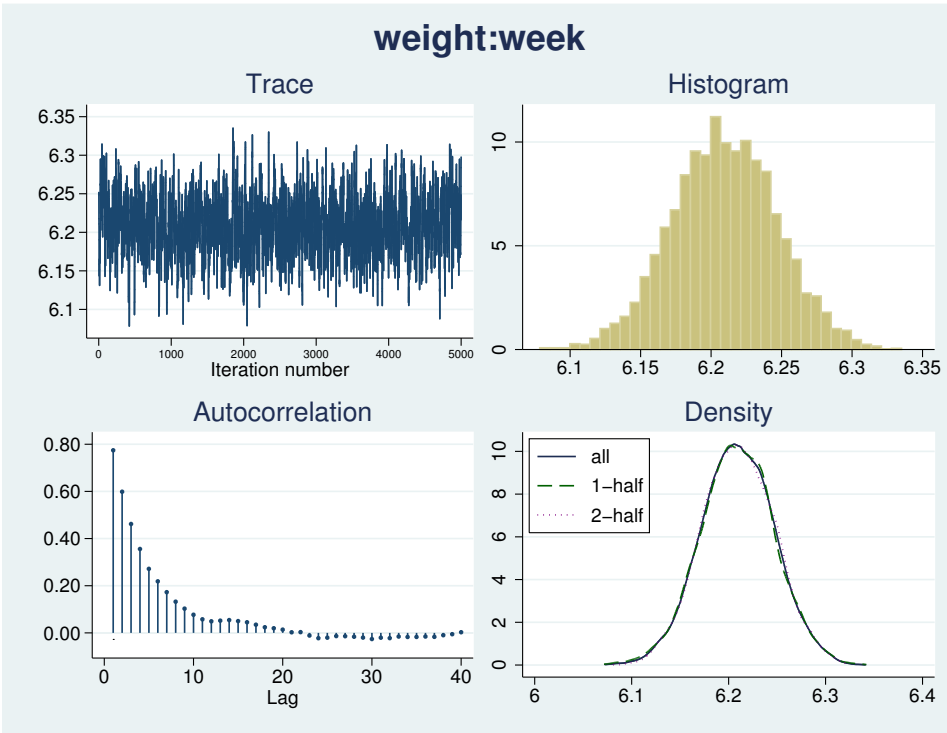
The most efficient MCMC procedure for our Bayesian model is Gibbs sampling, which can be set up as follows. To request a Gibbs sampling for a block of model parameters, we must first define them in a separate `prior()` statement and then put them in a separate `block()` with the `gibbs` suboption.

```
. 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.

◁

▷ Example 22: Fourth simulation—splitting random-effects parameters

Gibbs sampling typically provides the most efficient sampling of parameters. Full Gibbs sampling is not always available; see, for example, *Mixed-effects logistic regression* below.

In the absence of Gibbs sampling for random effects, block()'s suboption split provides the next most efficient way of sampling the random-effects parameters in bayesmh. Taking into account conditional independence of individual random effects, random-effects parameters associated with levels of the grouping variable can be sampled sequentially (as separate blocks) instead of being sampled jointly from a high-dimensional proposal distribution (as in example 20).

For example, instead of using Gibbs sampling for the random effects (as in example 21), we use block()'s suboption split for the random-effects parameters {weight:i.id}.

```
. 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({weight:_cons}, gibbs) block({weight:week}, gibbs)
> block({var_0}, gibbs) block({var_id}, gibbs)
> block({weight:i.id}, split)
> mcmcsize(5000) dots notable nomodelsummary
Burn-in 2500 aaaaaaaaa1000aaaaaaaaa2000aaaaa done
Simulation 5000 .........1000.........2000.........3000.........4000.........
> 5000 done
Bayesian normal regression                      MCMC iterations  =      7,500
Metropolis-Hastings and Gibbs sampling          Burn-in          =      2,500
                                                MCMC sample size =      5,000
                                                Number of obs    =        432
                                                Acceptance rate  =      .4823
                                                Efficiency:  min =     .04123
                                                             avg =      .1773
Log marginal likelihood = -1050.2963                         max =      .7524
```

We 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	6.206316	.0399631	.002783	6.206429	6.127974	6.28349
_cons	19.31371	.6125276	.019648	19.31878	18.08646	20.52478
var_0	4.4213	.3205769	.006464	4.407209	3.825247	5.085138
var_id	15.74962	3.448178	.056218	15.32605	10.25279	23.57063

The estimated posterior means are close to those obtained with the full Gibbs sampler in example 21, although the estimated MCMC standard errors are slightly higher. For example, the MCSE of {var_0} rises from 0.0050 to 0.0065, or about 30%.

The average sampling efficiency, 18%, is not as high as with the full Gibbs sampling in example 21 but is still high enough for reliable estimation. The caveat with using the split option for sampling the random-effects parameters is a significant decrease in speed. When speed is an issue or when the number of random-effects parameters is large, the reffects() option may be a better alternative; see example 23.

◁

▷ Example 23: Fifth simulation—using the reffects() option

The reffects() option supported by bayesmh can be used for specifying the two-level random-intercept model considered in this series of examples. It allows for faster MCMC sampling of the parameters associated with a random-effects variable compared with block()'s suboption split (see example 22).

We modify the syntax from example 22 as follows. We exclude i.id from the list of independent variables and add the reffects(id) option. We also omit the block({weight:i.id}, split) option because the blocking of the {weight:i.id} parameters is handled implicitly by the reffects() option.

```
. set seed 14

. bayesmh weight week, reffects(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({weight:_cons}, gibbs) block({weight:week}, gibbs)
> block({var_0}, gibbs) block({var_id}, gibbs)
> mcmcsize(5000) dots
Burn-in 2500 aaaaaaaaa1000aaaaaaaaa2000aaaaa done
Simulation 5000 .........1000.........2000.........3000.........4000.........
> 5000 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
Metropolis-Hastings and Gibbs sampling          Burn-in          =      2,500
                                                MCMC sample size =      5,000
                                                Number of obs    =        432
                                                Acceptance rate  =      .8475
                                                Efficiency:  min =     .03221
                                                             avg =      .3708
Log marginal likelihood = -1050.8235                         max =        .77
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
weight						
week	6.209593	.040908	.003224	6.210008	6.127663	6.288345
var_0	4.412871	.3171205	.006625	4.406072	3.834571	5.067036
weight						
_cons	19.29717	.634793	.01903	19.28847	18.0127	20.53934
var_id	15.89952	3.549986	.057213	15.46713	10.36942	24.02605

Our estimates of the variance components do not change noticeably from those in examples 21 and 22: `{var_0}` is 4.41 and `{var_id}` is 15.90.

Although the average efficiency, 0.37, of the displayed parameters is lower than the corresponding efficiency of the full Gibbs sampler in example 21, the application of the `reffects()` option results in consuming about 35% less memory during simulation and a 25% improvement in speed. The real benefit of the `reffects()` option, however, becomes apparent for models with many random-effects levels and models for which full Gibbs samplers are not available; see *Mixed-effects logistic regression* below.

When we use option `reffects()`, `bayesmh` suppresses the estimates of random-effects parameters from the output. You can use the `showreffects()` option to display them.

◁

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 $i = 1, \ldots, 9$.

▷ Example 24: Independent covariance structure for the random effects

Let us first assume that the random effects u_{0j}'s and u_{1j}'s are independent. We can 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 include random effects for pig and their interaction with `week` in our regression model. Following example 21, we add factor levels of the `id` variable to the regression by using the factor-variable specification `i.id`. We include random coefficients on `week` as `i.id#c.week`. By default, the specification 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
```

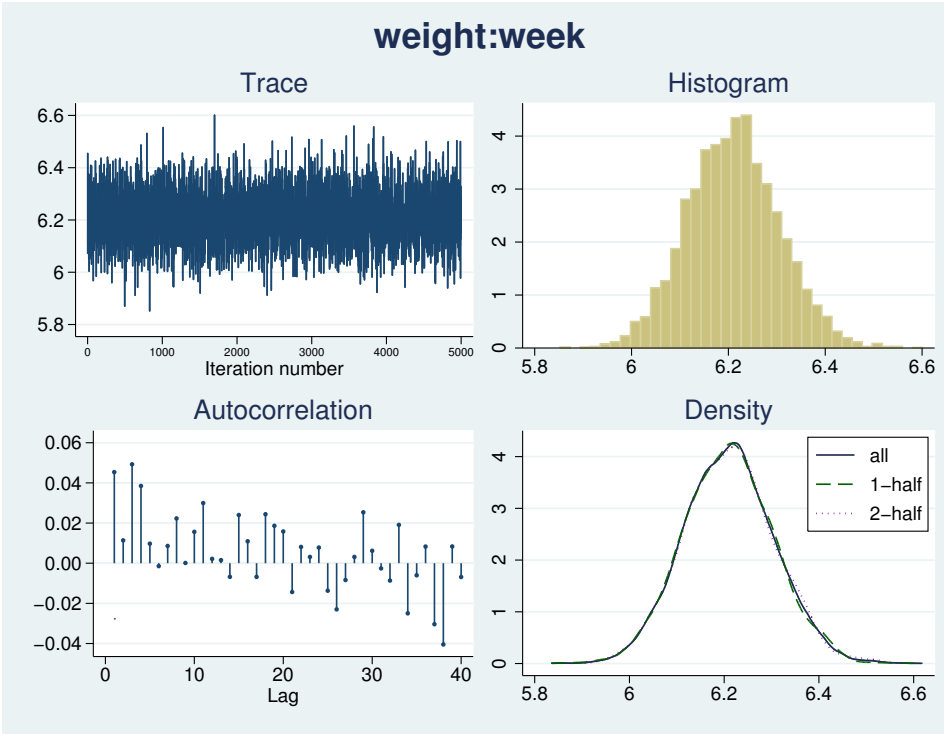
We fit our model using `bayesmh`. Following example 21, we perform blocking of parameters and use Gibbs sampling for the blocks. (We could have used the `reffects()` option as in example 23 to include random intercepts, but we want to use Gibbs sampling in this example; thus we use the factor-variable specification instead.)

```
. 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

▷ Example 25: Unstructured covariance structure for the random effects

In this example, we assume that the random effects τ_{0j}'s and τ_{1j}'s are correlated. Again we can use the `mixed` command to fit this model by using maximum likelihood.

```
. set seed 14
. mixed weight week || id: week, cov(unstructured)
Performing EM optimization:
Performing gradient-based optimization:
Iteration 0:   log likelihood = -868.96185
Iteration 1:   log likelihood = -868.96185
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)      =    4649.17
Log likelihood = -868.96185                     Prob > chi2       =     0.0000
```

weight	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
week	6.209896	.0910745	68.18	0.000	6.031393	6.388399
_cons	19.35561	.3996387	48.43	0.000	18.57234	20.13889

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
id: Unstructured				
var(week)	.3715251	.0812958	.2419532	.570486
var(_cons)	6.823363	1.566194	4.351297	10.69986
cov(week,_cons)	-.0984378	.2545767	-.5973991	.4005234
var(Residual)	1.596829	.123198	1.372735	1.857505

```
LR test vs. linear model: chi2(3) = 764.58                Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

We modify the previous Bayesian model to account for the correlation between the random effects:

$$(\tau_{0j}, \tau_{1j}) \sim \text{i.i.d. MVN}(\beta_0, \beta_1, \Sigma)$$
$$\Sigma \sim \text{InvWishart}\{3, I(2)\}$$

$$\Sigma = \begin{bmatrix} \sigma^2_{\text{id}} & \sigma^2_{12} \\ \sigma^2_{21} & \sigma^2_{\text{week}} \end{bmatrix}$$

The elements σ^2_{id} and σ^2_{week} of Σ represent the variances of τ_{0j}'s and τ_{1j}'s, respectively, while σ_{21} is the covariance between them. We apply weakly informative inverse-Wishart prior with degree of freedom 3 and identity scale matrix.

Gibbs sampling is not available in `bayesmh` with unstructured covariance for the random effects. We thus replace `gibbs` with `reffects` in the corresponding `block()` option. This is possible because τ_{0j}'s are conditionally independent given τ_{1j}'s and vice versa. Using `block()`'s suboption `reffects` results in a more efficient sampling.

```
. set seed 14
. bayesmh weight i.id i.id#c.week, likelihood(normal({var_0})) noconstant
> prior({weight:i.id i.id#c.week},
>     mvnormal(2, {weight:_cons}, {weight:week}, {Sigma,m}))
> prior({weight:week _cons}, normal(0, 1e2))
> prior({var_0}, igamma(0.001,0.001))
> prior({Sigma,m}, iwishart(2,3,I(2)))
> block({var_0}, gibbs) block({Sigma,m}, gibbs)
> block({weight:_cons}) block({weight:week})
> block({weight:i.id}, reffects)
> block({weight:i.id#c.week}, reffects)
> noshow({weight:i.id i.id#c.week})
> mcmcsize(5000) dots
Burn-in 2500 aaaaaaaaa1000aaaaaaaaa2000aaaaa done
Simulation 5000 .........1000.........2000.........3000.........4000.........
> 5000 done
Model summary
------------------------------------------------------------------------------
Likelihood:
  weight ~ normal(xb_weight,{var_0})
Priors:
  {weight:i.id i.id#c.week} ~ mvnormal(2,{weight:_cons},{weight:week},{Sigma,m
                              }) (1)
                    {var_0} ~ igamma(0.001,0.001)
        {weight:week _cons} ~ normal(0,1e2)
Hyperprior:
  {Sigma,m} ~ iwishart(2,3,I(2))
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_weight.
Bayesian normal regression                      MCMC iterations  =      7,500
Metropolis-Hastings and Gibbs sampling          Burn-in          =      2,500
                                                MCMC sample size =      5,000
                                                Number of obs    =        432
                                                Acceptance rate  =      .5581
                                                Efficiency:  min =     .07112
                                                             avg =      .1423
Log marginal likelihood = -926.22043                         max =      .2238
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
var_0	1.607509	.1249066	.00435	1.601815	1.38134	1.860937
weight						
_cons	19.36808	.4017089	.021302	19.36764	18.52137	20.15876
week	6.201477	.0952501	.003317	6.199532	6.014793	6.389815
Sigma_1_1	6.850707	1.632765	.07773	6.60346	4.345719	10.66529
Sigma_2_1	-.0854197	.2652103	.010005	-.0803053	-.6326388	.4431884
Sigma_2_2	.400556	.0903881	.002702	.3889624	.260342	.6140122

The average sampling efficiency is about 14% with no indications for convergence problems. The posterior mean estimates of the main model parameters are close to the maximum likelihood results from `mixed`. For example, the estimates of variance components σ^2_{id}, σ_{21}, and σ^2_{week} are 6.85, -0.85, and 0.40, respectively, from `bayesmh` and 6.82, -0.98, and 0.37, respectively, from `mixed`.

◁

Mixed-effects logistic regression

Here we revisit example 1 [ME] **melogit**. The example analyzes data from the 1989 Bangladesh fertility survey (Huq and Cleland 1990). A logistic regression model applied to the response variable `c_use` uses fixed-effects variables `urban`, `age`, and `child*` and a random-effects variable, `district`, to account for the between-district variability.

A Bayesian analog of this two-level, random-intercept model using `bayesmh` is as follows. We use the `reffects()` option to specify the random-effects variable `district`. The corresponding random-effects parameters `{c_use:i.district}` are assigned a zero-mean normal prior distribution with variance `{district:var}`. A relatively weak `normal(0,100)` prior is applied to the fixed-effects parameters `{c_use:urban}`, `{c_use:age}`, `{c_use:child*}`, and `{c_use:_cons}`. The variance parameter `{district:var}` is assigned a noninformative `igamma(0.01,0.01)` prior, and a Gibbs sampler is used for it. We are not interested in the estimates of random effects in this example, so we exclude the random-effects parameters `{c_use:i.district}` from the output table.

```
. use http://www.stata-press.com/data/r14/bangladesh
(Bangladesh Fertility Survey, 1989)
. set seed 14
. bayesmh c_use urban age child*, likelihood(logit) reffects(district)
> prior({c_use:i.district}, normal(0,{district:var}))
> prior({c_use:urban age child* _cons}, normal(0, 100))
> prior({district:var}, igamma(0.01,0.01))
> block({district:var}, gibbs) noshow({c_use:i.district}) dots
Burn-in 2500 aaaaaaaaa1000aaaaaaaaa2000aaaaa done
Simulation 10000 .........1000.........2000.........3000.........4000.........
> 5000.........6000.........7000.........8000.........9000.........10000 done
Model summary
------------------------------------------------------------------------------
Likelihood:
  c_use ~ logit(xb_c_use)
Priors:
                           {c_use:i.district} ~ normal(0,{district:var})  (1)
  {c_use:urban age child1 child2 child3 _cons} ~ normal(0,100)            (1)
                                {district:var} ~ igamma(0.01,0.01)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_c_use.
Bayesian logistic regression                    MCMC iterations  =     12,500
Metropolis-Hastings and Gibbs sampling          Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =      1,934
                                                Acceptance rate  =      .4913
                                                Efficiency:  min =     .01728
                                                             avg =     .02523
Log marginal likelihood = -1240.2644                         max =     .04155
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
c_use						
urban	.7252685	.1260246	.009454	.7216279	.4789413	.9849255
age	-.0259076	.0076429	.000529	-.0259236	-.040793	-.0101205
child1	1.104812	.1540978	.008963	1.104046	.8012581	1.410451
child2	1.352477	.1890995	.014387	1.345373	.9832535	1.712931
child3	1.343504	.1793496	.012102	1.343257	.9941767	1.697041
_cons	-1.687957	.1420537	.008543	-1.683849	-1.964436	-1.405009
district						
var	.2380246	.0857548	.004207	.2269953	.1034288	.4357797

Although the average efficiency of 0.03 is not that high, there are no indications for convergence problems. (We can verify this by looking at convergence diagnostics using `bayesgraph diagnostics`.)

Our estimates of the main regression parameters are close to those obtained with the `melogit` command. The posterior mean estimate of variance parameter `{district:var}`, 0.24, is slightly larger than the corresponding estimate of 0.22 from `melogit`.

This model has a fairly large number of parameters, 67, and the logistic likelihood does not allow for efficient Gibbs sampling of regression parameters. If we do not use the `reffects()` option of `bayesmh` (or `block()`'s suboption `split` with `{c_use:i.district}`) and resort to the standard MH algorithm, we may have problems drawing a well-mixed MCMC sample.

For comparison, we show a standard `bayesmh` specification in which the `{c_use:i.district}` parameters are placed in a separate block without using the `reffects()` option. Statistically, the two model specifications are the same because they define one and the same posterior distribution. However, they use different MCMC sampling procedures.

```
. set seed 14
. bayesmh c_use urban age child* ibn.district, likelihood(logit)
> prior({c_use:i.district}, normal(0,{district:var}))
> prior({c_use:urban age child* _cons}, normal(0, 100))
> prior({district:var}, igamma(0.01,0.01))
> block({district:var}, gibbs)
> block({c_use:i.district}) noshow({c_use:i.district}) dots
Burn-in 2500 aaaaaaaaa1000aaaaaaaaa2000aaaaa done
Simulation 10000 .........1000.........2000.........3000.........4000.........
> 5000.........6000.........7000.........8000.........9000.........10000 done
Model summary
------------------------------------------------------------------------------
Likelihood:
  c_use ~ logit(xb_c_use)
Priors:
                            {c_use:i.district} ~ normal(0,{district:var})  (1)
  {c_use:urban age child1 child2 child3 _cons} ~ normal(0,100)             (1)
                                {district:var} ~ igamma(0.01,0.01)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_c_use.
Bayesian logistic regression                    MCMC iterations  =     12,500
Metropolis-Hastings and Gibbs sampling          Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =      1,934
                                                Acceptance rate  =        .53
                                                Efficiency:  min =    .007367
                                                             avg =      .0255
Log marginal likelihood = -1362.0681                         max =     .04817
------------------------------------------------------------------------------
             |                                                     Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
c_use        |
       urban |   .6929174    .119883   .013968   .6906259   .4664536   .9199054
         age |  -.0280929   .0080467   .000375  -.0280689  -.0440295  -.0126579
      child1 |   1.158416   .1697389   .011534   1.155004    .839977   1.490753
      child2 |   1.442235   .1769685   .008064   1.439614    1.09767   1.796089
      child3 |   1.447863   .1928966   .012707   1.448637   1.065645   1.836695
       _cons |  -2.348392     .14138   .010016  -2.350597  -2.621669  -2.069296
-------------+----------------------------------------------------------------
district     |
         var |   .7490145   .1557382   .014079   .7299348   .5026288   1.110885
------------------------------------------------------------------------------
Note: There is a high autocorrelation after 500 lags.
```

In this second run, we observe that the minimal sampling efficiency is less than 1% and that the MCMC convergence is questionable. For example, the reported MCMC standard error for {district:var} is about 0.014, or three times higher than the corresponding error of 0.004 in the previous run. The results from this last run are not trustworthy.

Bayesian analysis of change-point problem

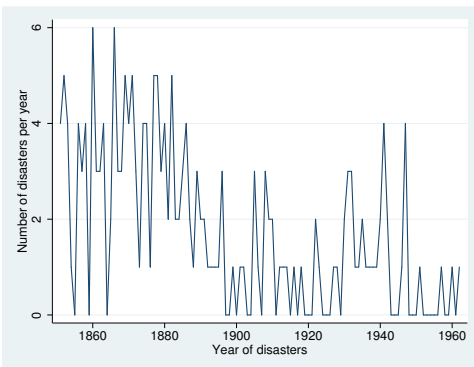
Change-point problems deal with stochastic data, usually time-series data, which undergoes some abrupt change at some time point. It is of interest to localize the point of change and estimate the properties of the stochastic process before and after the change.

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):

$$\begin{aligned}
\texttt{counts}_i &\sim \text{Poisson}(\mu_1),\ \texttt{if year}_i < \text{cp}\\
\texttt{counts}_i &\sim \text{Poisson}(\mu_2),\ \texttt{if year}_i \geq \text{cp}\\
\mu_1 &\sim 1\\
\mu_2 &\sim 1\\
\text{cp} &\sim \text{Uniform}(1851, 1962)
\end{aligned}$$

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 specify the substitutable expression

```
({mu1}*sign(year<{cp})+{mu2}*sign(year>={cp}))
```

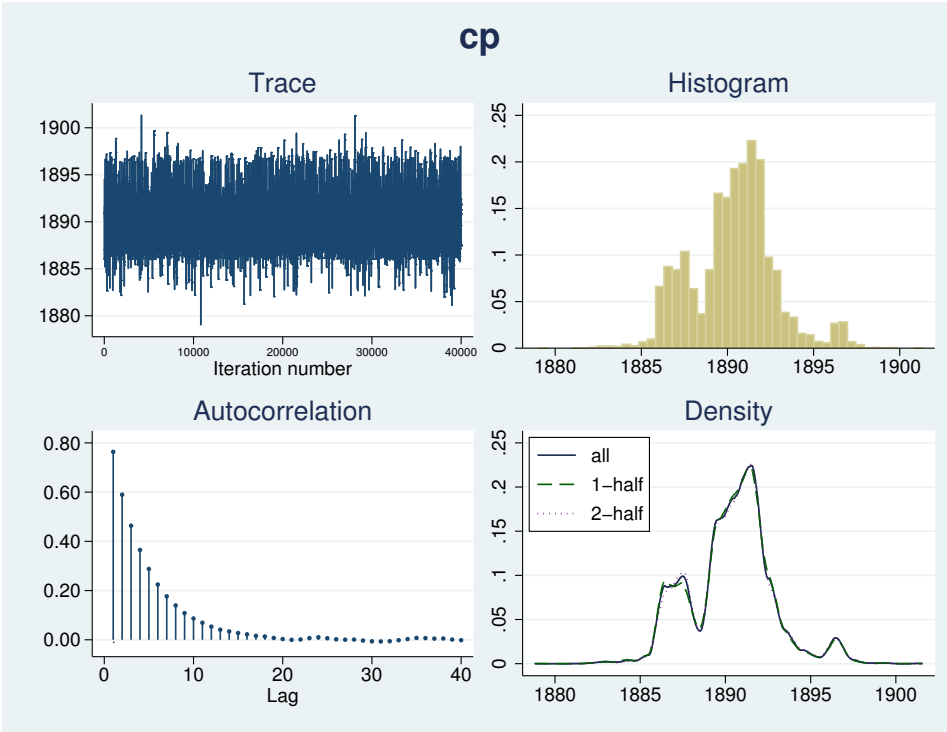
as the mean of a Poisson distribution `dpoisson()`. To ensure the feasibility of the initial state, we specify the desired initial values in option `initial()`. Because of high autocorrelation in the MCMC chain, we increase the MCMC size to achieve higher precision of our estimates. We change the default 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,
> likelihood(dpoisson({mu1}*sign(year<{cp})+{mu2}*sign(year>={cp})))
> 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.

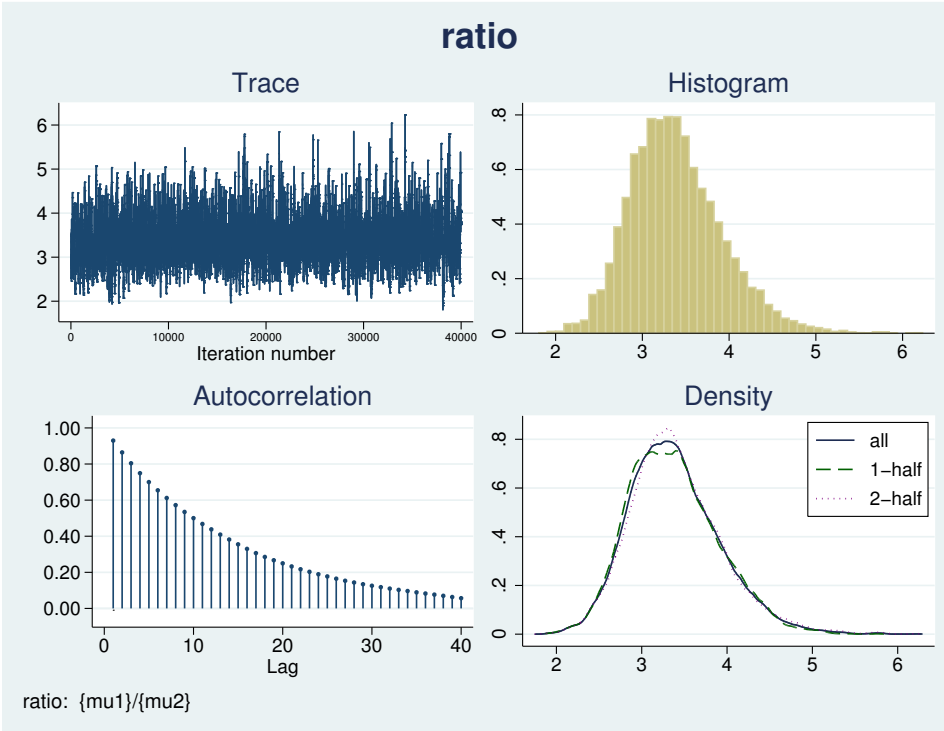
```
. bayesstats summary (ratio:{mu1}/{mu2})
Posterior summary statistics                         MCMC sample size =    40,000
       ratio : {mu1}/{mu2}
```

	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 26: 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_i's 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_i's 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`.

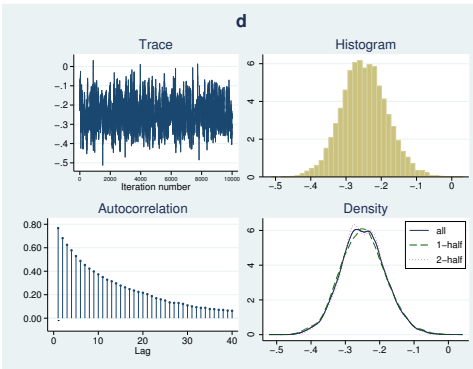
```
. bayesstats ess {d} {sig2}
Efficiency summaries    MCMC sample size =    10,000
```

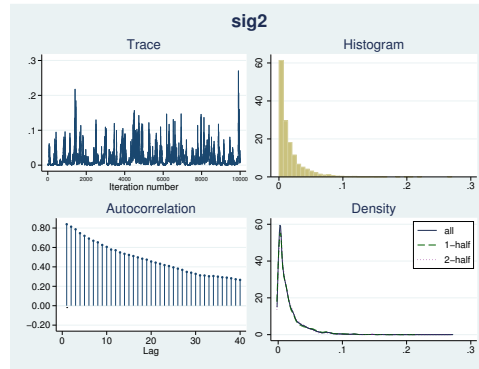
	ESS	Corr. time	Efficiency
d	184.49	54.20	0.0184
sig2	194.88	51.31	0.0195

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 27: 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_i's 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 `binomial()` 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(binomial(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).

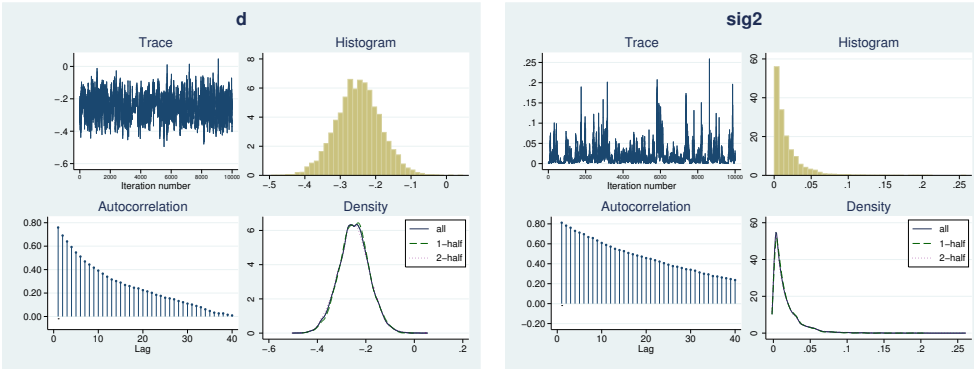
Compared with example 22, the efficiencies and other statistics for the main parameters are similar.

```
. 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}
```



Item response theory

▷ Example 28: 1PL IRT model—Rasch model

If you are not familiar with IRT, see [IRT] **irt** for an introduction to IRT concepts and terminology. Here we revisit example 1 of [IRT] **irt 1pl**. The example analyzes student responses to nine test questions and uses an abridged version of the mathematics and science data from De Boeck and Wilson (2004). The goal of the analysis is to estimate the common discrimination of the questions (items) and their individual difficulties.

An alternative formulation of the one-parameter IRT model is the Rasch (1960) model with logit link; see, for example, *Methods and formulas* of [IRT] **irt 1pl**. A typical IRT dataset consists of binary outcomes (success or failure) of J subjects, where each subject is tested on I items. Let the observation y_{ij} represent the binary outcome for item i, where $i = 1, \ldots, I$, and subject j, where $j = 1, \ldots, J$. Each item i is characterized by a level of difficulty b_i. The difficulties are not observed and must be estimated. Associated with each subject j is a latent trait level θ_j, which characterizes the ability of the subject. The model likelihood has a generalized linear regression form

$$\text{logit}\{\Pr(y_{ij} = 1|b_i, \theta_j)\} = a(\theta_j - b_i)$$

where a is a discrimination parameter. According to this likelihood model, the probability of success increases with the subject ability and decreases with item difficulty. The discrimination parameter a represents the slope of the item characteristic curves. The subject abilities are assumed to be standardized so that

$$\theta_j \sim \text{ i.i.d. } N(0, 1)$$

The discrimination parameter a can be absorbed into θ_j and b_i so that the model is reparameterized as

$$\text{logit}\{\Pr(y_{ij} = 1|\widetilde{b}_i, \widetilde{\theta}_j)\} = \widetilde{\theta}_j + \widetilde{b}_i \tag{1}$$

$$\widetilde{\theta}_j \sim \text{ i.i.d. } N(0, \sigma^2)$$

where $\sigma = a$ and $\widetilde{b}_i = -ab_i$. In addition to the above, a Bayesian formulation of the model requires, prior specifications for parameters σ^2 and $\widetilde{b}_i$. In the following example, we use

$$\sigma^2 \sim \text{InvGamma}(0.01, 0.01)$$

$$\widetilde{b}_i \sim N(0, 10)$$

To fit this model using `bayesmh`, we first need to reshape the data from example 1 of [IRT] **irt 1pl** in long format so that the answers to the nine questions are represented by the response variable `y`, while the `item` and `id` variables encode the questions and students, respectively.

```
. use http://www.stata-press.com/data/r14/masc1
(Data from De Boeck & Wilson (2004))
. generate id = _n
. quietly reshape long q, i(id) j(item)
. rename q y
```

The Rasch likelihood model can be specified with `bayesmh` using `y` as a dependent variable, `item` as an independent factor variable, and `id` as a random-effects variable. We suppress the base levels of `item` and `id` and use the `noconstant` option in the likelihood specification. The random-effects parameters `{y:i.id}` are assigned a zero-mean normal prior with variance `{var}` [σ^2 in model specification (1)]. The parameter `{var}` is assigned a noninformative inverse-gamma prior with shape 0.01 and scale 0.01, whereas the parameters `{y:i.item}` [$\widetilde{b}_i$'s in model (1)] are applied ad hoc informative `normal(0,10)` priors. Because there are many random-effects parameters `{y:i.id}`, we exclude them from the simulation results and the output table by specifying the `exclude()` option.

```
. fvset base none id item
. set seed 14
. bayesmh y i.item, noconstant reffects(id) likelihood(logit)
> prior({y:i.id}, normal(0, {var}))
> prior({y:i.item}, normal(0, 10))
> prior({var}, igamma(0.01,0.01))
> block({var}) block({y:i.item}, reffects) exclude({y:i.id}) dots
Burn-in 2500 aaaaaaaaa1000.........2000..... done
Simulation 10000 .........1000.........2000.........3000.........4000.........
> 5000.........6000.........7000.........8000.........9000.........10000 done
Model summary
------------------------------------------------------------------------------
Likelihood:
  y ~ logit(xb_y)
Priors:
    {y:i.id} ~ normal(0,{var})                                            (1)
  {y:i.item} ~ normal(0,10)                                               (1)
Hyperprior:
  {var} ~ igamma(0.01,0.01)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_y.
```

```
Bayesian logistic regression                    MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =      7,200
                                                Acceptance rate  =      .3031
                                                Efficiency:  min =     .02233
                                                             avg =     .09591
Log marginal likelihood =          .                         max =      .1139
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
y						
item						
1	.6015542	.0820766	.00258	.6053092	.438748	.7659896
2	.1025364	.0832577	.00262	.1027404	-.0551544	.2644014
3	1.547352	.0985834	.003041	1.548773	1.352163	1.737668
4	-.2704933	.081763	.002509	-.269603	-.4330444	-.1116137
5	-1.410691	.0947962	.002899	-1.41001	-1.60001	-1.232148
6	-.5911439	.0837508	.002742	-.5907655	-.756948	-.4301115
7	-1.128951	.0906917	.00292	-1.125747	-1.31619	-.9531386
8	2.060501	.1102492	.003284	2.059005	1.851561	2.277273
9	1.015636	.0875144	.002593	1.015847	.8486233	1.190524
var	.726955	.079031	.005289	.7234471	.576551	.8930412

In the simulation summary, `bayesmh` reports a modest average efficiency of about 10% with no indications for convergence problems. The log marginal likelihood is reported as missing because we used the `exclude()` option. The Laplace–Metropolis approximation of the log marginal likelihood requires that simulation results be available for all model parameters, including random-effects parameters.

To match the discrimination and question difficulty parameters of the `irt 1pl` command, we can apply the following transformation to the `bayesmh` model parameters. The common discrimination parameter equals the square-root of `{var}`, and the individual question difficulties equal the negative `{y:i.id}`'s parameters, normalized by their common discrimination. We can obtain estimates of these parameters using the `bayesstats summary` command.

```
. bayesstats summary (discr:sqrt({var}))
>                    (diff1:-{y:1bn.item}/sqrt({var}))
>                    (diff2:-{y:2.item}/sqrt({var}))
>                    (diff3:-{y:3.item}/sqrt({var}))
>                    (diff4:-{y:4.item}/sqrt({var}))
>                    (diff5:-{y:5.item}/sqrt({var}))
>                    (diff6:-{y:6.item}/sqrt({var}))
>                    (diff7:-{y:7.item}/sqrt({var}))
>                    (diff8:-{y:8.item}/sqrt({var}))
>                    (diff9:-{y:9.item}/sqrt({var})), nolegend
Posterior summary statistics                       MCMC sample size =    10,000
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
discr	.8513548	.0463711	.003113	.8505569	.7593095	.9450085
diff1	-.7082469	.1011861	.003508	-.7061776	-.9055634	-.5127043
diff2	-.1208126	.0983803	.003111	-.1213543	-.3106128	.066497
diff3	-1.821986	.1412775	.005966	-1.817721	-2.107488	-1.558985
diff4	.3184996	.0975893	.003082	.3153589	.1316555	.5159253
diff5	1.66113	.1341472	.005662	1.658582	1.414426	1.931525
diff6	.696008	.1030643	.003547	.6927263	.4945615	.9064106
diff7	1.329158	.1199949	.004557	1.325713	1.112063	1.582109
diff8	-2.426495	.1725481	.00809	-2.41446	-2.779606	-2.115298
diff9	-1.195812	.1148938	.004121	-1.19502	-1.429396	-.9759188

We observe that the reported posterior means for the common discrimination and question difficulties are close to those obtained with `irt 1pl`. For example, the estimated posterior mean for the discrimination is about 0.851, whereas the one reported by `irt 1pl` is 0.852, which is within the limits of the MCMC standard error of 0.003.

◁

In this example, we fit the Rasch model using the `reffects()` option and used transformation to estimate parameters of the corresponding 1PL IRT model. To avoid reparameterization, we could have fit the 1PL model directly using a nonlinear specification of `bayesmh`, as we demonstrate in example 29 for the 2PL IRT model. The shortcoming of the nonlinear specification, which precludes the use of the `reffects()` option, is slower execution.

▷ Example 29: 2PL IRT model

A more comprehensive IRT model is the 2PL model introduced by Birnbaum (1968), which allows the discrimination and difficulty parameters to vary between items. For a detailed description and examples of the model, see [IRT] **irt 2pl**.

A Bayesian formulation of the 2PL model allows the item-specific discrimination and difficulty parameters as well as the subject abilities to be modeled, either individually or as groups, using prior distributions.

The 2PL model likelihood has the following form,

$$\Pr(Y_{ij} = 1) = \frac{\exp\{a_i(\theta_j - b_i)\}}{1 + \exp\{a_i(\theta_j - b_i)\}}$$

where a_i's and b_i's are discrimination and difficulty parameters and θ_j's are subject abilities. This is a logistic regression model with probability of success modeled using the linear form $a_i(\theta_j - b_i)$. We assume that the probability of success increases with subject ability, which implies $a_i > 0$.

Subject ability parameters are assumed independent and distributed according to the standard normal distribution

$$\theta_j \sim N(0,1)$$

For Bayesian modeling, we additionally assume the following prior specifications:

$$\ln(a_i) \sim N(\mu_a, \sigma_a^2)$$

$$b_i \sim N(\mu_b, \sigma_b^2)$$

$$\mu_a, \mu_b \sim N(0,1)$$

$$\sigma_a^2, \sigma_b^2 \sim \text{Gamma}(1,1)$$

In the absence of prior knowledge about parameters a_i's and b_i's, we want to specify proper priors that are not subjective. Because a_i's must be positive, a common choice is to assume that $\ln(a_i)$'s are normally distributed with mean μ_a and variance σ_a^2. We assume that b_i's are normally distributed with mean μ_b and variance σ_b^2. Our prior assumption is that the questions in the study are fairly balanced in terms of discrimination and difficulty and we express this expectation by specifying $N(0,1)$ hyperpriors for μ_a and μ_b; that is, we assume that μ_a and μ_b are not that different from zero. We also put a slight prior constraint on the variability of the discrimination and difficulty parameters by assigning a gamma distribution with shape 1 and scale 1 as hyperprior distributions for σ_a^2 and σ_b^2. To demonstrate a Bayesian 2PL model, we use again the mathematics and science dataset `masc1`, reshaped in long format as in example 28.

```
bayesmh y = ({discr:}*({subj:}-{diff:})), likelihood(logit)     ///
        redefine(discr:i.item)                                  ///
        redefine(diff:i.item)                                   ///
        redefine(subj:i.id)                                     ///
        prior({subj:i.id},    normal(0, 1))                     ///
        prior({discr:i.item}, lognormal({mua}, {vara}))         ///
        prior({diff:i.item},  normal({mub}, {varb}))            ///
        prior({vara varb},    gamma(1, 1))                      ///
        prior({mua mub},      normal(0, 1))                     ///
        ...
```

To specify the 2PL model likelihood in `bayesmh`, we need to use a nonlinear specification to accommodate the varying coefficients a_i's. For `masc1.dta`, we have 9 items, where $i = 1,\ldots,9$, and 800 subjects, where $j = 1,\ldots,800$. A straightforward nonlinear specification is `({discr:i.item}*({subj:i.id}-{diff:i.item}))`. Given the large number of subjects, it may be computationally prohibitive to use this substitutable expression. A more computationally efficient way is to use the `redefine()` option to specify the random effects associated with item discrimination, item difficulty, and student ability. For example, `redefine(subj:i.id)` introduces subject random-effects parameters, one for each subject, and represents the parameters θ_j's. Similarly, we use `redefine(discr:i.item)` and `redefine(diff:i.item)` to introduce the item-specific discrimination and difficulty parameters a_i's and b_i's, respectively. The probability of success is then modeled using the expression `({discr:}({subj:}-{diff:}))`.

To achieve better sampling efficiency, we place the hyperparameters `{mua}`, `{mub}`, `{vara}`, and `{varb}` into separate blocks using the `block()`'s suboption `split`. We also initialize the discrimination and difficulty random effects with 1, because the default zeros result in an invalid initial state. We have many random-effects parameters `{subj:i.id}`, so we exclude them from the simulation results and the output table by specifying the `exclude()` option.

```
. fvset base none id item
. set seed 14
. bayesmh y = ({discr:}*({subj:}-{diff:})), likelihood(logit)
> redefine(discr:i.item) redefine(diff:i.item) redefine(subj:i.id)
> prior({subj:i.id}, normal(0, 1))
> prior({discr:i.item}, lognormal({mua}, {vara}))
> prior({diff:i.item}, normal({mub}, {varb}))
> prior({vara varb}, gamma(1, 1)) prior({mua mub}, normal(0, 1))
> block({vara varb mua mub}, split) init({discr:i.item} 1 {diff:i.item} 1)
> exclude({subj:i.id}) dots
Burn-in 2500 aaaaaaaaa1000aaaaaaaaa2000aaaaa done
Simulation 10000 .........1000.........2000.........3000.........4000.........
> 5000.........6000.........7000.........8000.........9000.........10000 done
Model summary
------------------------------------------------------------------------------
Likelihood:
  y ~ logit(xb_discr*(xb_subj-xb_diff))
Priors:
  {discr:i.item} ~ lognormal({mua},{vara})                                (1)
   {diff:i.item} ~ normal({mub},{varb})                                   (2)
     {subj:i.id} ~ normal(0,1)                                            (3)
Hyperpriors:
  {vara varb} ~ gamma(1,1)
    {mua mub} ~ normal(0,1)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_discr.
(2) Parameters are elements of the linear form xb_diff.
(3) Parameters are elements of the linear form xb_subj.
```

```
Bayesian logistic regression                    MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =      7,200
                                                Acceptance rate  =      .3657
                                                Efficiency:  min =     .01027
                                                             avg =     .04499
Log marginal likelihood =          .                         max =      .1762
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
discr						
item						
1	1.498385	.2400297	.017729	1.475542	1.077856	2.016152
2	.666128	.1119997	.005344	.6649249	.4541802	.886375
3	.9383295	.1477952	.01186	.9204028	.6765889	1.266951
4	.7994094	.1238608	.005473	.7962891	.5688815	1.059357
5	.9051624	.1482122	.011637	.9043871	.6396924	1.205136
6	.958388	.142267	.009086	.9504636	.7001779	1.252891
7	.4792205	.0899554	.008877	.4741658	.3261732	.6680243
8	1.297704	.221223	.017263	1.295304	.8673528	1.758576
9	.6670617	.1104876	.009091	.6625288	.4666832	.8896693
diff						
item						
1	-.4989148	.0830179	.005137	-.4993581	-.6742972	-.3519269
2	-.1502021	.1239443	.003577	-.1458722	-.4059771	.0751602
3	-1.710592	.2234976	.016935	-1.692729	-2.214283	-1.328619
4	.3466566	.114595	.004388	.3434778	.1346476	.5714364
5	1.605012	.2387544	.018704	1.57642	1.229628	2.138817
6	.6442216	.1147523	.006181	.6396185	.4391379	.8736935
7	2.190372	.4265088	.041367	2.130899	1.530794	3.148253
8	-1.830477	.2345021	.019047	-1.795231	-2.420293	-1.46107
9	-1.472878	.2480988	.01968	-1.446161	-2.045008	-1.072642
mua	-.157914	.1768629	.004552	-.1556469	-.5143239	.188628
vara	.2516064	.1808929	.007705	.1996404	.0597163	.7334286
mub	-.078763	.4360087	.010388	-.0791598	-.945727	.7879737
varb	1.953467	.8047981	.029965	1.806903	.8319104	3.947983

`bayesmh` reports an acceptable average efficiency of about 4%. A close inspection of the estimation table shows that the posterior mean estimates for item discrimination and difficulty are not much different from the MLE estimates obtained with the `irt 2pl` command; see example 1 in [IRT] **irt 2pl**.

◁

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#_reffects)	1 if the parameters in #th block are random effects; 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
Adaptive MH algorithm for random effects
Gibbs sampling for some likelihood-prior and prior-hyperprior configurations
Likelihood-prior configurations
Prior-hyperprior configurations
Marginal likelihood

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 $T_{\text{total}} = T_0 + (T - 1) \times \texttt{thinning()} + 1$. 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

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

   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 $u < \alpha(\boldsymbol{\theta}_*|\boldsymbol{\theta}_{t-1})$ or $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1}$, otherwise.
4. Perform adaptive iteration k. This step is performed only if $k \le$ AMAX and t mod ALEN $= 0$. Update ρ_k according to (2), update Σ_k according to (3), 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 $\boldsymbol{\theta}^1$, $\boldsymbol{\theta}^2$, …, $\boldsymbol{\theta}^B$, 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 1, probabilities are initialized with 0.5, and the remaining parameters are initialized with 0. 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.

You can specify the `initrandom` option to request random initial values for all model parameters. In this case, `bayesmh` generates random initial values from the corresponding prior distributions of the parameters, except for those that are assigned improper priors such as `flat` and `jeffreys()` or user-defined priors using the `density()` and `logdensity()` prior options. You must specify fixed initial values for all model parameters for which random initial values cannot be generated.

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, $\alpha(\boldsymbol{\theta}_*|\boldsymbol{\theta}_{t-1})$, 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

$$\rho_k = \rho_{k-1} e^{\beta_k \{\Phi^{-1}(\text{AR}_k/2) - \Phi^{-1}(\text{TAR}/2)\}} \tag{2}$$

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

$$\Sigma_k = (1 - \beta_k)\Sigma_{k-1} + \beta_k \widehat{\Sigma}_k, \quad \beta_k = \frac{\beta_0}{k^\gamma} \tag{3}$$

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 (3) 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.

Adaptive MH algorithm for random effects

Suppose that `u` is a random-effects variable that takes discrete values $1, \ldots, m$. For an independent sample $Y = \{y_{ij}\}$, where $j = 1, \ldots, m$ and where $i = 1, \ldots, n_j$, we assume that `u` takes value j for all y_{ij}, where $i = 1, \ldots, n_j$. Consider a two-level Bayesian model that includes random-effect parameters η_j, where $j = 1, \ldots, m$, one for each level of `u`, and additional parameter vector $\boldsymbol{\theta}$. We assume that, with respect to the posterior distribution of the model, the random-effects parameters η_j are conditionally independent given $\boldsymbol{\theta}$ and the data sample Y. The latter can be ensured the prior distribution of η_j's satisfies the conditional independence condition

$$\pi(\eta_1, \ldots, \eta_m|\boldsymbol{\theta}) = \prod_{j=1}^{m} \pi(\eta_j|\boldsymbol{\theta})$$

In this case, the posterior distribution admits the following factorization,

$$\Pr(\eta_1,\ldots,\eta_m,\boldsymbol{\theta}|Y) = \pi(\boldsymbol{\theta})\left\{\prod_{j=1}^{m}\pi(\eta_j|\boldsymbol{\theta})\prod_{i=1}^{n_j}\Pr(y_{ij}|\eta_j,\boldsymbol{\theta})\right\}$$

where $\pi(\boldsymbol{\theta})$ is the prior distribution of $\boldsymbol{\theta}$. This form of the posterior allows the parameters η_j's to be placed in one block and steps 1, 2, and 3 of the adaptive MH algorithm to be performed for all of them simultaneously in a vector form, as if they were a single scalar parameter.

To request the random-effects MH algorithm in `bayesmh`, use `block`'s suboption `reffects`. The same algorithm is used if one specifies the `reffects()` option. A random-effects block of parameters has a default acceptance rate of 0.44, performs adaptation of the scale ρ_k according to (2), but uses a fixed identity matrix for the proposal covariance Σ_k.

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_i's 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,

$$\begin{aligned} y_i|\boldsymbol{\theta}^b, \sigma^2 &\sim N(\mathbf{x}_i'\boldsymbol{\beta}, \sigma^2),\ i = 1, 2, \ldots, n \\ \theta_k^b &\sim \text{i.i.d. } N(\beta_0, \tau_0^2),\ k = 1, 2, \ldots, p_1 \\ \boldsymbol{\theta}^b|\sigma^2, \mathbf{y} &\sim F_b = \text{MVN}(\boldsymbol{\mu}_n, \Lambda_n) \end{aligned}$$

where β_0 and τ_0^2 are hyperparameters (prior regression coefficient and prior variance) of normal prior distributions for θ_k^b 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_i's with a mean μ; mean and variance are independent a priori,

$$\begin{aligned} y_i|\mu, \theta^b &\sim N(\mu, \theta^b),\ i = 1, 2, \ldots, n \\ \theta^b &\sim \text{InvGamma}(\alpha, \beta) \\ \theta^b|\mu, \mathbf{y} &\sim F_b = \text{InvGamma}\left(\alpha + n/2, \beta + \sum_{i=1}^{n}(y_i - \mu)^2/2\right) \end{aligned}$$

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.

Estimation of the log marginal likelihood becomes unstable for high-dimensional models such as multilevel models and may result in a missing value.

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.

Birnbaum, A. 1968. Some latent trait models and their use in inferring an examinee's ability. In *Statistical Theories of Mental Test Scores*, ed. F. M. Lord and M. R. Novick, 395–479. Reading, MA: Addison–Wesley.

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.

De Boeck, P., and M. Wilson, ed. 2004. *Explanatory Item Response Models: A Generalized Linear and Nonlinear Approach*. New York: Springer.

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.

Huq, N. M., and J. Cleland. 1990. *Bangladesh Fertility Survey 1989 (Main Report)*. National Institute of Population Research and Training.

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.

Rasch, G. 1960. *Probabilistic Models for Some Intelligence and Attainment Tests*. Copenhagen: Danish Institute of Educational Research.

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**