Title

stata.com

bayesstats summary — Bayesian summary statistics

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

Description

`bayesstats summary` calculates and reports posterior summary statistics for model parameters and functions of model parameters using current Bayesian estimation results. Posterior summary statistics include posterior means, posterior standard deviations, MCMC standard errors (MCSE), posterior medians, and equal-tailed credible intervals or highest posterior density (HPD) credible intervals.

Quick start

Posterior summaries for all model parameters after a Bayesian regression model

```
bayesstats summary
```

As above, but only for parameters {y:x1} and {y:x2}

```
bayesstats summary {y:x1} {y:x2}
```

Same as above

```
bayesstats summary {y:x1 x2}
```

Posterior summaries for elements 1,1 and 2,1 of matrix parameter {S}

```
bayesstats summary {S_1_1 S_2_1}
```

Posterior summaries for all elements of matrix parameter {S}

```
bayesstats summary {S}
```

Posterior summaries with HPD instead of equal-tailed credible intervals and with credible level of 90%

```
bayesstats summary, hpd clevel(90)
```

Posterior summaries with MCSE calculated using batch means

```
bayesstats summary, batch(100)
```

Posterior summaries for functions of scalar model parameters

```
bayesstats summary ({y:x1}-{y:_cons}) (sd:sqrt({var}))
```

Posterior summaries for the log-likelihood and log-posterior functions

```
bayesstats summary _loglikelihood _logposterior
```

Posterior summaries for selected model parameters and functions of model parameters and for log-likelihood and log-posterior functions using abbreviated syntax

```
bayesstats summary {var} ({y:x1}-{y:_cons}) _ll _lp
```

Posterior summaries of the simulated outcome

```
bayespredict {_ysim}, saving(predres)
bayesstats summary {_ysim} using predres
```

Posterior summaries of the mean across observations of the simulated outcome labeled as `mymean`

```
bayesstats summary (mymean: @mean({_ysim})) using predres
```

Menu

Statistics > Bayesian analysis > Summary statistics

Syntax

Syntax is presented under the following headings:

Summary statistics for model parameters
Summary statistics for predictions

Summary statistics for model parameters

Summary statistics for all model parameters

```
bayesstats summary [, options showreffects[(reref)]]

bayesstats summary _all [, options showreffects[(reref)]]
```

Summary statistics for selected model parameters

```
bayesstats summary paramspec [, options]
```

Summary statistics for expressions of model parameters

```
bayesstats summary exprspec [, options]
```

Summary statistics of log-likelihood or log-posterior functions

```
bayesstats summary _loglikelihood|_logposterior [, options]
```

Full syntax

```
bayesstats summary spec [spec ...] [, options]
```

paramspec can be one of the following:

{*eqname*:*param*} refers to a parameter *param* with equation name *eqname*;

{*eqname*:} refers to all model parameters with equation name *eqname*;

{*eqname*:*paramlist*} refers to parameters with names in *paramlist* and with equation name *eqname*; or

{*param*} refers to all parameters named *param* from all equations.

In the above, *param* can refer to a matrix name, in which case it will imply all elements of this matrix. See *Different ways of specifying model parameters* in [BAYES] **Bayesian postestimation** for examples.

exprspec is an optionally labeled expression of model parameters specified in parentheses:

([*exprlabel* :] *expr*)

exprlabel is a valid Stata name, and *expr* is a scalar expression that may not contain matrix model parameters. See *Specifying functions of model parameters* in [BAYES] **Bayesian postestimation** for examples.

`_loglikelihood` and `_logposterior` also have respective synonyms `_ll` and `_lp`.

spec is one of *paramspec*, *exprspec*, `_loglikelihood` (or `_ll`), or `_logposterior` (or `_lp`).

Summary statistics for predictions

Summary statistics for simulated outcomes, residuals, and more

`bayesstats summary` *yspec* [*yspec* ...] `using` *predfile* [`,` *options*]

Summary statistics for expressions of simulated outcomes, residuals, and more

`bayesstats summary` (*yexprspec*) [(*yexprspec*) ...] `using` *predfile* [`,` *options*]

Summary statistics for Mata functions of simulated outcomes, residuals, and more

`bayesstats summary` (*funcspec*) [(*funcspec*) ...] `using` *predfile* [`,` *options*]

Full syntax

`bayesstats summary` *predspec* [*predspec* ...] `using` *predfile* [`,` *options*]

predfile is the name of the dataset created by `bayespredict` that contains prediction results.

yspec is {*ysimspec* | *residspec* | *muspec* | *label*}.

ysimspec is `{_ysim#}` or `{_ysim#[`*numlist*`]}`, where `{_ysim#}` refers to all observations of the #th simulated outcome and `{_ysim#[`*numlist*`]}` refers to the selected observations, *numlist*, of the #th simulated outcome. `{_ysim}` is a synonym for `{_ysim1}`.

residspec is `{_resid#}` or `{_resid#[`*numlist*`]}`, where `{_resid#}` refers to all residuals of the #th simulated outcome and `{_resid#[`*numlist*`]}` refers to the selected residuals, *numlist*, of the #th simulated outcome. `{_resid}` is a synonym for `{_resid1}`.

muspec is `{_mu#}` or `{_mu#[`*numlist*`]}`, where `{_mu#}` refers to all expected values of the #th outcome and `{_mu#[`*numlist*`]}` refers to the selected expected values, *numlist*, of the #th outcome. `{_mu}` is a synonym for `{_mu1}`.

label is the name of the function simulated using `bayespredict`.

With large datasets, specifications `{_ysim#}`, `{_resid#}`, and `{_mu#}` may use a lot of time and memory and should be avoided. See *Generating and saving simulated outcomes* in [BAYES] **bayespredict**.

yexprspec is [*exprlabel* :] *yexpr*, where *exprlabel* is a valid Stata name and *yexpr* is a scalar expression that may contain individual observations of simulated outcomes, `{_ysim#[#]}`; individual expected outcome values, `{_mu#[#]}`; individual simulated residuals, `{_resid#[#]}`; and other scalar predictions, {*label*}.

funcspec is [*label*:]@*func*(*arg1*[, *arg2*]), where *label* is a valid Stata name; *func* is an official or user-defined Mata function that operates on column vectors and returns a real scalar; and *arg1* and *arg2* are one of {_ysim[#]}, {_resid[#]}, or {_mu[#]}. *arg2* is primarily for use with user-defined Mata functions; see *Defining test statistics using Mata functions* in [BAYES] **bayespredict**.

predspec is one of *yspec*, (*yexprspec*), or (*funcspec*). See *Different ways of specifying predictions and their functions* in [BAYES] **Bayesian postestimation**.

options	Description
Main	
clevel(#)	set credible interval level; default is clevel(95)
hpd	display HPD credible intervals instead of the default equal-tailed credible intervals
batch(#)	specify length of block for batch-means calculations; default is batch(0)
* chains(_all | *numlist*)	specify which chains to use for computation; default is chains(_all)
* sepchains	compute results separately for each chain
skip(#)	skip every # observations from the MCMC sample; default is skip(0)
nolegend	suppress table legend
display_options	control spacing, line width, and base and empty cells
Advanced	
corrlag(#)	specify maximum autocorrelation lag; default varies
corrtol(#)	specify autocorrelation tolerance; default is corrtol(0.01)

*Options chains() and sepchains are relevant only when option nchains() is used with bayesmh or the bayes prefix.

Options

Main

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

hpd displays the HPD credible intervals instead of the default equal-tailed credible intervals.

batch(#) specifies the length of the block for calculating batch means 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().

chains(_all | *numlist*) specifies which chains from the MCMC sample to use for computation. The default is chains(_all) or to use all simulated chains. Using multiple chains, provided the chains have converged, generally improves MCMC summary statistics. Option chains() is relevant only when option nchains() is specified with bayesmh or the bayes prefix.

sepchains specifies that the results be computed separately for each chain. The default is to compute results using all chains as determined by option chains(). Option sepchains is relevant only when option nchains() is specified with bayesmh or the bayes prefix.

showreffects and showreffects(*reref*) are for use after multilevel models, and they specify that the results for all or a list *reref* of random-effects parameters be provided in addition to other model parameters. By default, all random-effects parameters are excluded from the results to conserve computation time.

`skip(#)` specifies that every # observations from the MCMC sample not be used for computation. The default is `skip(0)` or to use all observations in the MCMC sample. Option `skip()` can be used to subsample or thin the chain. `skip(#)` is equivalent to a thinning interval of #+1. For example, if you specify `skip(1)`, corresponding to the thinning interval of 2, the command will skip every other observation in the sample and will use only observations 1, 3, 5, and so on in the computation. If you specify `skip(2)`, corresponding to the thinning interval of 3, the command will skip every 2 observations in the sample and will use only observations 1, 4, 7, and so on in the computation. `skip()` does not thin the chain in the sense of physically removing observations from the sample, as is done by, for example, `bayesmh`'s `thinning()` option. It only discards selected observations from the computation and leaves the original sample unmodified.

`nolegend` suppresses the display of the table legend, which identifies the rows of the table with the expressions they represent.

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

Advanced

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

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

Remarks and examples

stata.com

Remarks are presented under the following headings:

Introduction
Bayesian summaries for an auto data example

Introduction

`bayesstats summary` reports posterior summary statistics for model parameters and their functions using the current Bayesian estimation results. When typed without arguments, the command displays results for all model parameters. Alternatively, you can specify a subset of model parameters following the command name; see *Different ways of specifying model parameters* in [BAYES] **Bayesian postestimation**. You can also obtain results for scalar functions of model parameters; see *Specifying functions of model parameters* in [BAYES] **Bayesian postestimation**.

Sometimes, it may be useful to obtain posterior summaries of log-likelihood and log-posterior functions. This can be done by specifying `_loglikelihood` and `_logposterior` (or the respective synonyms `_ll` and `_lp`) following the command name.

You can also obtain the posterior summaries for prediction quantities when you specify the prediction dataset in the `using` specification; see *Different ways of specifying predictions and their functions* in [BAYES] **Bayesian postestimation** for how to specify prediction quantities with `bayesstats summary`.

bayesstats summary reports the following posterior summary statistics: posterior mean, posterior standard deviation, MCMC standard error, posterior median, and equal-tailed credible intervals or, if the hpd option is specified, HPD credible intervals. The default credible level is set to 95%, but you can change this by specifying the clevel() option. Equal-tailed and HPD intervals may produce very different results for asymmetric or highly skewed marginal posterior distributions. The HPD intervals are preferable in this situation.

You should not confuse the term "HPD interval" with the term "HPD region". A $\{100\times(1-\alpha)\}\%$ HPD interval is defined such that it contains $\{100\times(1-\alpha)\}\%$ of the posterior density. A $\{100\times(1-\alpha)\}\%$ HPD region also satisfies the condition that the density inside the region is never lower than that outside the region. For multimodal univariate marginal posterior distributions, the HPD regions may include unions of nonintersecting HPD intervals. For unimodal univariate marginal posterior distributions, HPD regions are indeed simply HPD intervals. The bayesstats summary command thus calculates HPD intervals assuming unimodal marginal posterior distributions (Chen and Shao 1999).

Some authors use the term "posterior intervals" instead of "credible intervals" and the term "central posterior intervals" instead of "equal-tailed credible intervals" (for example, Gelman et al. [2014]).

Bayesian summaries for an auto data example

Recall our analysis of auto.dta from example 4 in [BAYES] **bayesmh** using the mean-only normal model for mpg with a noninformative prior.

```
. use https://www.stata-press.com/data/r16/auto
(1978 Automobile Data)
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, flat) prior({var}, jeffreys)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
Priors:
  {mpg:_cons} ~ 1 (flat)
        {var} ~ jeffreys
------------------------------------------------------------------------------
Bayesian normal regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         74
                                                Acceptance rate  =      .2668
                                                Efficiency:  min =     .09718
                                                             avg =      .1021
Log marginal-likelihood =   -234.645                         max =      .1071
------------------------------------------------------------------------------
             |                                                Equal-tailed
             |      Mean   Std. Dev.     MCSE     Median  [95% Cred. Interval]
-------------+----------------------------------------------------------------
mpg          |
       _cons |   21.29222   .6828864   .021906   21.27898   19.99152   22.61904
-------------+----------------------------------------------------------------
         var |   34.76572   5.91534    .180754   34.18391    24.9129   47.61286
------------------------------------------------------------------------------
```

Example 1: Summaries for all parameters

If we type `bayesstats summary` without arguments after the `bayesmh` command, we will obtain the same summary table as reported by `bayesmh`.

```
. bayesstats summary
Posterior summary statistics                       MCMC sample size =    10,000
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	21.29222	.6828864	.021906	21.27898	19.99152	22.61904
var	34.76572	5.91534	.180754	34.18391	24.9129	47.61286

The posterior mean of `{mpg:_cons}` is 21.29 and of `{var}` is 34.8. They are close to their respective frequentist analogs (the sample mean of `mpg` is 21.297, and the sample variance is 33.47), because we used a noninformative prior. Posterior standard deviations are 0.68 for `{mpg:_cons}` and 5.92 for `{var}`, and they are comparable to frequentist standard errors under this noninformative prior. The standard error estimates of the posterior means, MCSEs, are low. For example, MCSE is 0.022 for `{mpg:_cons}`. This means that the precision of our estimate is, up to one decimal point, 21.3 provided that MCMC converged. The posterior means and medians of `{mpg:_cons}` are close, which suggests that the posterior distribution for `{mpg:_cons}` may be symmetric. According to the credible intervals, we are 95% certain that the posterior mean of `{mpg:_cons}` is roughly between 20 and 23 and that the posterior mean of `{var}` is roughly between 25 and 48. We can infer from this that `{mpg:_cons}` is greater than, say, 15, and that `{var}` is greater than, say, 20, with a very high probability. (We can use [BAYES] **bayestest interval** to compute the actual probabilities.)

The above is also equivalent to typing

```
. bayesstats summary {mpg:_cons} {var}
  (output omitted)
```

Example 2: Credible intervals

By default, `bayesstats summary` reports 95% equal-tailed credible intervals. We can change the default credible level by specifying the `clevel()` option.

```
. bayesstats summary, clevel(90)
Posterior summary statistics                       MCMC sample size =    10,000
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [90% Cred. Interval]	
mpg						
_cons	21.29222	.6828864	.021906	21.27898	20.18807	22.44172
var	34.76572	5.91534	.180754	34.18391	26.28517	44.81732

As expected, 90% credible intervals are more narrow.

To calculate and report HPD intervals, we specify the hpd option.

```
. bayesstats summary, hpd
Posterior summary statistics                     MCMC sample size =    10,000
```

	Mean	Std. Dev.	MCSE	Median	HPD [95% Cred. Interval]	
mpg _cons	21.29222	.6828864	.021906	21.27898	19.94985	22.54917
var	34.76572	5.91534	.180754	34.18391	24.34876	46.12339

The posterior distribution of {mpg:_cons} is symmetric about the posterior mean; thus there is little difference between the 95% equal-tailed credible interval from example 1 and this 95% HPD credible interval for {mpg:_cons}. The 95% HPD interval for {var} has a smaller width than the corresponding equal-tailed interval in example 1.

◁

▷ Example 3: Batch-means estimator

bayesstats summary provides two estimators for MCSE: effective-sample-size and batch-means. Estimation using effective sample sizes is the default. You can use the batch(#) option to request the batch-means estimator, where # is the batch size. The optimal batch size depends on the autocorrelation in the MCMC sample. For example, if we observe that the autocorrelation for the parameters of interest is negligible after lag 100, we can specify batch(100) to estimate MCSE.

In our example, autocorrelation dies out after about lag 10 (see, for example, *Autocorrelation plots* in [BAYES] **bayesgraph** and example 1 in [BAYES] **bayesstats ess**), so we use 10 as our batch size:

```
. bayesstats summary, batch(10)
Posterior summary statistics                     MCMC sample size =    10,000
                                                 Batch size       =        10
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg _cons	21.29222	.6828864	.015315	21.27898	19.99152	22.61904
var	34.76572	5.91534	.135295	34.18391	24.9129	47.61286

```
Note: Mean and MCSE are estimated using batch means.
```

The batch-means MCSE estimates are somewhat smaller than those obtained by default using effective sample sizes.

Use caution when choosing the batch size for the batch-means method. For example, if you use the batch size of 1, you will obtain MCSE estimates under the assumption that the draws in the MCMC sample are independent, which is not true.

◁

Example 4: Subsampling or thinning the chain

You can reduce correlation between MCMC draws by thinning or subsampling the MCMC chain. You can use the `skip(#)` option to skip every # observations from the MCMC sample, which is equivalent to a thinning interval of $\# + 1$. For example, if you specify `skip(1)`, corresponding to the thinning interval of 2, `bayesstats summary` will skip every other observation in the sample and will use only observations 1, 3, 5, and so on in the computation. If you specify `skip(2)`, corresponding to the thinning interval of 3, `bayesstats summary` will skip every two observations in the sample and will use only observations 1, 4, 7, and so on in the computation. By default, no observations are skipped—`skip(0)`. Note that `skip()` does not thin the chain in the sense of physically removing observations from the sample, as is done by `bayesmh`'s `thinning()` option. It discards only selected observations from the computation and leaves the original sample unmodified.

```
. bayesstats summary, skip(9)
note: skipping every 9 sample observations; using observations 1,11,21,...
Posterior summary statistics                       MCMC sample size =     1,000
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
mpg						
_cons	21.29554	.6813796	.029517	21.27907	19.98813	22.58582
var	34.7396	5.897313	.206269	33.91782	24.9554	48.11452

We selected to skip every 9 observations, which led to a significant reduction of the MCMC sample size and thus increased our standard deviations. In some cases, with larger MCMC sample sizes, subsampling may decrease standard deviations because of the decreased autocorrelation in the reduced MCMC sample.

◁

Example 5: Summaries for expressions of model parameters

`bayesstats summary` accepts expressions to provide summaries of functions of model parameters. For example, we can use expression `(sd:sqrt({var}))` with a label, `sd`, to summarize the standard deviation of `mpg` in addition to the variance.

```
. bayesstats summary (sd:sqrt({var})) {var}
Posterior summary statistics                       MCMC sample size =     10,000
          sd : sqrt({var})
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
sd	5.87542	.4951654	.014972	5.846701	4.991282	6.900207
var	34.76572	5.91534	.180754	34.18391	24.9129	47.61286

Expressions can also be used for calculating posterior probabilities, although this can be more easily done using `bayestest interval` (see [BAYES] **bayestest interval**). For illustration, let's verify the probability that `{var}` is within the endpoints of the reported credible interval, indeed 0.95.

```
. bayesstats summary (prob:{var}>24.913 & {var}<47.613)
Posterior summary statistics                       MCMC sample size =    10,000
       prob : {var}>24.913 & {var}<47.613
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
prob	.9502	.2175424	.005301	1	0	1

Example 6: Summaries for log likelihood and log posterior

We can use reserved names `_loglikelihood` (or the synonym `_ll`) and `_logposterior` (or the synonym `_lp`) to obtain summaries of the log likelihood and log posterior for the simulated MCMC sample.

```
. bayesstats summary _ll _lp
Posterior summary statistics                       MCMC sample size =    10,000
        _ll : _loglikelihood
        _lp : _logposterior
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
_ll	-235.4162	.990654	.032232	-235.1379	-238.1236	-234.4345
_lp	-238.9507	1.037785	.034535	-238.6508	-241.7889	-237.9187

Example 7: Summaries for predicted outcomes

We continue our series of examples by computing summaries for Bayesian predictions. Let's generate Bayesian predictions of `mpg` and summarize them.

We use `bayespredict` to simulate outcome values for `mpg` for the first 10 observations from the fitted `bayesmh` model. To use `bayespredict`, we must first save the simulation results from `bayesmh` in a Stata dataset, `autosim.dta`. We then use `bayespredict` to save the prediction results in the dataset `mpgreps.dta`.

```
. bayesmh, saving(autosim)
note: file autosim.dta saved
. bayespredict {_ysim[1/10]}, saving(mpgreps) rseed(16)
Computing predictions ...
file mpgreps.dta saved
file mpgreps.ster saved
```

We can now summarize the prediction results by using `bayesstats summary`. We specify the prediction quantity we wish to summarize, the simulated outcome `{_ysim}` in our example, and the prediction dataset, `mpgreps.dta`, which contains the prediction quantity, in the `using` specification.

```
. bayesstats summary {_ysim} using mpgreps
Posterior summary statistics                       MCMC sample size =    10,000
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
_ysim1_1	21.24878	6.018783	.062648	21.23939	9.444973	33.07051
_ysim1_2	21.2539	5.944206	.060415	21.21638	9.421932	32.90605
_ysim1_3	21.3256	5.910363	.061595	21.31499	9.801655	33.02746
_ysim1_4	21.40651	5.963456	.059479	21.45933	9.794156	33.39388
_ysim1_5	21.19781	5.926335	.061197	21.26437	9.759916	32.80291
_ysim1_6	21.34776	5.94413	.059441	21.32314	9.771529	33.30251
_ysim1_7	21.34043	5.898474	.058985	21.34119	9.821613	33.07709
_ysim1_8	21.25329	5.957051	.05886	21.26176	9.476474	32.96236
_ysim1_9	21.25284	5.866096	.05962	21.3052	9.714165	32.82636
_ysim1_10	21.3464	5.931401	.060853	21.30528	9.670334	33.10769

`bayesstats summary` reports posterior summaries for all simulated outcomes in the prediction dataset, `mpgreps.dta`. Estimated posterior means and standard deviations are similar to the corresponding observed values for `mpg`, 21.30 and 5.79, respectively.

We can specifically examine the first observation of the replicated sample, `{_ysim_1}`, and compare it with the observed value, `mpg[1]`, of 22.

```
. bayesstats summary ({_ysim_1}>=`=mpg[1]') using mpgreps
Posterior summary statistics                       MCMC sample size =    10,000
       expr1 : _ysim1_1>=22
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred.	Interval]
expr1	.4479	.497303	.004973	0	0	1

We find that 45% of the replicates of `mpg[1]` are greater than 22. The reported probability of 0.45 is known as the posterior predictive p-value and is used for goodness-of-fit checking; see [BAYES] **bayesstats ppvalues**.

◁

Stored results

`bayesstats summary` stores the following in `r()`:

Scalars

`r(mcmcsize)`	MCMC sample size used in the computation
`r(clevel)`	credible interval level
`r(hpd)`	`1` if `hpd` is specified, `0` otherwise
`r(batch)`	batch length for batch-means calculations
`r(skip)`	number of MCMC observations to skip in the computation; every `r(skip)` observations are skipped
`r(corrlag)`	maximum autocorrelation lag
`r(corrtol)`	autocorrelation tolerance
`r(nchains)`	number of chains used in the computation

Macros

`r(names)`	names of model parameters and expressions
`r(expr_#)`	#th expression
`r(exprnames)`	expression labels
`r(chains)`	chains used in the computation, if `chains()` is specified

Matrices

`r(summary)`	matrix with posterior summaries statistics for parameters in `r(names)`
`r(summary_chain#)`	matrix `summary` for chain #, if `sepchains` is specified

Methods and formulas

Methods and formulas are presented under the following headings:

Point estimates
Credible intervals

Most of the summary statistics employed in Bayesian analysis are based on the marginal posterior distributions of individual model parameters or functions of model parameters.

Let θ be a scalar model parameter and $\{\theta_t\}_{t=1}^T$ be an MCMC chain of size T drawn from the marginal posterior distribution of θ. For a function $g(\theta)$, substitute $\{\theta_t\}_{t=1}^T$ with $\{g(\theta_t)\}_{t=1}^T$ in the formulas below. If θ is a covariance matrix model parameter, the formulas below are applied to each element of the lower-diagonal portion of θ.

Point estimates

Marginal posterior moments are approximated using the Monte Carlo integration applied to the simulated samples $\{\theta_t\}_{t=1}^T$.

Sample posterior mean and sample posterior standard deviation are defined as follows,

$$\widehat{\theta} = \frac{1}{T}\sum_{t=1}^{T}\theta_t,\ \ \widehat{s}^2 = \frac{1}{T-1}\sum_{t=1}^{T}(\theta_t - \widehat{\theta})^2$$

where $\widehat{\theta}$ and $\widehat{s}^2$ are sample estimators of the population posterior mean $E(\theta_t)$ and posterior variance $\mathrm{Var}(\theta_t)$.

With multiple chains, the posterior mean and standard deviation are estimated using the combined sample of all chains or of those that are requested in the `chains()` option as follows. Let $\{\theta_{jt}\}_{t=1}^T$ be the jth Markov chain, $j = 1, \ldots, M$, with sample mean $\widehat{\theta}_j$ and variance $\widehat{s}_j^2$. The overall sample posterior mean is

$$\widehat{\theta} = \frac{1}{MT}\sum_{j=1}^{M}\sum_{t=1}^{T}\theta_{jt}$$

and equals the average of the sample means of individual chains. Let B and W be the respective between-chains and within-chain variances

$$B = \frac{T}{M-1}\sum_{j=1}^{M}(\widehat{\theta}_j - \widehat{\theta})^2,\ \ W = \frac{1}{M}\sum_{j=1}^{M}\widehat{s}_j^2$$

The estimator of the posterior variance is

$$\widehat{s}^2 = \frac{T-1}{T}W + \frac{1}{T}B \tag{1}$$

When the chains are strongly stationary, $\widehat{s}^2$ is an unbiased estimator of the marginal posterior variance of θ (Gelman et al. 2014, sec. 11.4).

The precision of the sample posterior mean is evaluated by its standard error, also known as the Monte Carlo standard error (MCSE). Note that MCSE cannot be estimated using the classical formula for the standard error, $\widehat{s}/\sqrt{T}$, because of the dependence between θ_t's.

Let

$$\sigma^2 = \text{Var}(\theta_t) + 2\sum_{k=1}^{\infty} \text{Cov}(\theta_t, \theta_{t+k})$$

Then, $\sqrt{T}\times$MCSE approaches σ asymptotically in T.

`bayesstats summary` provides two different approaches for estimating MCSE. Both approaches try to adjust for the existing autocorrelation in the MCMC sample. The first one uses the so-called effective sample size (ESS), and the second one uses batch means (Roberts 1996; Jones et al. 2006).

The ESS-based estimator for MCSE, the default in `bayesstats summary`, is given by

$$\text{MCSE}(\widehat{\theta}) = \widehat{s}/\sqrt{\text{ESS}}$$

ESS is defined as

$$\text{ESS} = T/(1 + 2\sum_{k=1}^{\text{max\_lags}} \rho_k)$$

where ρ_k is the lag-k autocorrelation, and max_lags is the maximum number less than or equal to ρ_{lag} such that for all $k = 1, \ldots, \text{max\_lags}$, $|\rho_k| > \rho_{\text{tol}}$, where ρ_{lag} and ρ_{tol} are specified in options `corrlag()` and `corrtol()` with the respective default values of 500 and 0.01. ρ_k is estimated as γ_k/γ_0, where

$$\gamma_k = \frac{1}{T}\sum_{t=1}^{T-k}(\theta_t - \widehat{\theta})(\theta_{t+k} - \widehat{\theta})$$

is the lag-k empirical autocovariance.

With multiple chains, the overall ESS is given by the sum of the effective sample sizes of individual chains. The MCSE is then calculated using the formula

$$\text{MCSE}(\widehat{\theta}) = \widehat{s}/\sqrt{\sum_{j=1}^{M} \text{ESS}_j}$$

where $\widehat{s}$ is computed using (1) and ESS_j is the effective sample size of the jth chain.

The batch-means estimator of MCSE is obtained as follows. For a given batch of length b, the initial MCMC chain is split into m batches of size b,

$$\{\theta_{j'+1}, \ldots, \theta_{j'+b}\}\ \{\theta_{j'+b+1}, \ldots, \theta_{j'+2b}\}\ \ldots\ \{\theta_{T-b+1}, \ldots, \theta_T\}$$

where $j' = T - m \times b$ and m batch means $\widehat{\mu}_1, \ldots, \widehat{\mu}_m$ are calculated as sample means of each batch. m is chosen as the maximum number such that $m \times b \leq T$. If m is not a divisor of T, the first $T - m \times b$ observations of the sample are not used in the batch-means computation. The batch-means estimator of the posterior variance, $\widehat{s}^2_{\text{batch}}$, is based on the assumption that $\widehat{\mu}_j$s are much less correlated than the original sample draws.

The batch-means estimator of the posterior mean is

$$\widehat{\theta}_{\text{batch}} = \frac{1}{m}\sum_{j=1}^{m} \widehat{\mu}_j$$

We have $\widehat{\theta}_{\text{batch}} = \widehat{\theta}$, whenever $m \times b = T$. Under the assumption that the batch means are uncorrelated, $\widehat{s}^2_{\text{batch}} = \{1/(m-1)\} \sum_{j=1}^{m} (\widehat{\mu}_j - \widehat{\theta}_{\text{batch}})^2$ can be used as an estimator of σ^2/b. This fact justifies the batch-means estimator of MCSE given by

$$\text{MCSE}_{\text{batch}}(\widehat{\theta}) = \frac{\widehat{s}_{\text{batch}}}{\sqrt{m}}$$

The accuracy of the batch-means estimator depends on the choice of the batch length b. The higher the autocorrelation in the original MCMC sample, the larger the batch length b should be, provided that the number of batches m does not become too small; $\sqrt{T}$ is typically used as the maximum value for b. The batch length is commonly determined by inspecting the autocorrelation plot for θ. Under certain assumptions, Flegal and Jones (2010) establish that an asymptotically optimal batch size is of order $T^{1/3}$.

With multiple chains, the batch-means estimator is calculated using the combined sample of all chains or of those that are requested in the `chains()` option.

Credible intervals

Let $\theta_{(1)}, \ldots, \theta_{(T)}$ be an MCMC sample ordered from smallest to largest. Let $(1-\alpha)$ be a credible level. Then, a $\{100 \times (1-\alpha)\}\%$ equal-tailed credible interval is

$$(\theta_{([T\alpha/2])}, \theta_{([T(1-\alpha/2)])})$$

where $[\,]$ in the above imply an integer number.

A $\{100 \times (1-\alpha)\}\%$ HPD interval is defined as the shortest interval among the $\{100 \times (1-\alpha)\}\%$ credible intervals $(\theta_{(j)}, \theta_{(j+[T(1-\alpha)])})$, $j = 1, \ldots, T - [T(1-\alpha)]$.

With multiple chains, credible intervals are computed using the combined sample of all chains or of those requested with the `chains()` option; see Brooks and Gelman (1998, sec. 1.1).

References

Brooks, S. P., and A. Gelman. 1998. General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics* 7: 434–455.

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

Flegal, J. M., and G. L. Jones. 2010. Batch means and spectral variance estimators in Markov chain Monte Carlo. *Annals of Statistics* 38: 1034–1070.

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

Jones, G. L., M. Haran, B. S. Caffo, and R. Neath. 2006. Fixed-width output analysis for Markov chain Monte Carlo. *Journal of the American Statistical Association* 101: 1537–1547.

Roberts, G. O. 1996. Markov chain concepts related to sampling algorithms. In *Markov Chain Monte Carlo in Practice*, ed. W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, 45–57. Boca Raton, FL: Chapman and Hall.

Also see

[BAYES] **bayes** — Bayesian regression models using the bayes prefix

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

[BAYES] **Bayesian estimation** — Bayesian estimation commands

[BAYES] **Bayesian postestimation** — Postestimation tools for bayesmh and the bayes prefix

[BAYES] **bayesgraph** — Graphical summaries and convergence diagnostics

[BAYES] **bayespredict** — Bayesian predictions

[BAYES] **bayesstats ess** — Effective sample sizes and related statistics

[BAYES] **bayesstats ppvalues** — Bayesian predictive p-values and other predictive summaries

[BAYES] **bayestest interval** — Interval hypothesis testing