bayesstats ess — Effective sample sizes and related statistics

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

Description

`bayesstats ess` calculates effective sample sizes (ESS), correlation times, and efficiencies for model parameters and functions of model parameters using current Bayesian estimation results.

Quick start

Effective sample sizes for all model parameters after a Bayesian regression model

```
bayesstats ess
```

Same as above, but only for model parameters `{y:x1}` and `{var}`

```
bayesstats ess {y:x1} {var}
```

Same as above, but skip every 5 observations from the full MCMC sample

```
bayesstats ess {y:x1} {var}, skip(5)
```

Effective sample sizes for functions of scalar model parameters

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

Same as above, and include `{y:x1}` and `{var}`

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

Menu

Statistics > Bayesian analysis > Effective sample sizes

Syntax

Syntax is presented under the following headings:

Statistics for model parameters
Statistics for predictions

Statistics for model parameters

Statistics for all model parameters

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

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

Statistics for selected model parameters

```
bayesstats ess paramspec [ , options ]
```

Statistics for expressions of model parameters

```
bayesstats ess exprspec [ , options ]
```

Full syntax

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

paramspec can be one of the following:

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

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

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

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

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

spec is one of *paramspec* or *exprspec*.

Statistics for predictions

Statistics for simulated outcomes, residuals, and more

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

Statistics for expressions of simulated outcomes, residuals, and more

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

Statistics for Mata functions of simulated outcomes, residuals, and more

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

Full syntax

`bayesstats ess` *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	
* `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 during Bayesian estimation. `collect` is allowed; see **[U] 11.1.10 Prefix commands**.

Options

Main

`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 used during Bayesian estimation.

`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 used during Bayesian estimation.

`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()`.

`corrtol(#)` specifies the autocorrelation tolerance used for calculating effective sample sizes. The default is `corrtol(0.01)`. For a given model parameter, if the absolute value of the lag-k autocorrelation is less than `corrtol()`, then all autocorrelation lags beyond the kth lag are discarded.

Remarks and examples

Remarks are presented under the following headings:

Effective sample size and MCMC sampling efficiency
Using bayesstats ess

Effective sample size and MCMC sampling efficiency

It is well known that for a random sample of T independent subjects, the standard error of the sample mean estimator is proportional to $1/\sqrt{T}$. In Bayesian inference, it is of interest to estimate the standard error of the posterior mean estimator. The posterior mean of a parameter of interest is typically estimated as a sample mean from an MCMC sample obtained from the marginal posterior distribution of the parameter of interest. Observations from an MCMC sample are not independent and are usually positively correlated, which must be taken into account when computing the standard error. Thus the standard error of the posterior mean estimator is proportional to $1/\sqrt{\text{ESS}}$, where ESS is the effective sample size for the parameter of interest. Typically, ESS is less than T, the total number of observations in the MCMC sample. We can thus interpret the posterior mean estimate as a sample mean estimate from an independent sample of size ESS. In other words, the effective sample size is an estimate of the number of independent observations that the MCMC chain represents. We say that MCMC samples with higher ESS are more efficient.

Effective sample size is directly related to the convergence properties of an MCMC sample—very low ESS relative to T suggests nonconvergence. In the extreme case of a perfectly correlated MCMC observation, ESS is 1. It is thus a standard practice to assess the quality of an MCMC sample by inspecting ESS values for all involved model parameters. Note, however, that high ESS values are not generally sufficient for declaring convergence of MCMC because pseudoconvergence, which may occur when MCMC does not explore the entire distribution, may also lead to high ESS values.

Using bayesstats ess

`bayesstats ess` reports effective sample sizes, correlation times, and efficiencies for model parameters and their functions using the current 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**. You can obtain the 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 within `bayesstats ess`.

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

```
. use https://www.stata-press.com/data/r19/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
```

	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

Example 1: Effective sample sizes for all parameters

To compute effective sample sizes and other related statistics for all model parameters, we type `bayesstats ess` without arguments after the `bayesmh` command.

```
. bayesstats ess
Efficiency summaries    MCMC sample size =    10,000
                        Efficiency:  min =    .09718
                                     avg =     .1021
                                     max =     .1071
```

	ESS	Corr. time	Efficiency
mpg			
_cons	971.82	10.29	0.0972
var	1070.99	9.34	0.1071

The closer the ESS estimates are to the MCMC sample size, the better. Also, the lower the correlation times are and the higher the efficiencies are, the better. ESS estimates can be interpreted as follows. In a sample of 10,000 MCMC observations, we have only about 972 independent observations to obtain estimates for `{mpg:_cons}` and only about 1,071 independent observations to obtain estimates for `{var}`. Correlation times are the reciprocal of efficiencies. You can interpret them as an estimated lag after which autocorrelation in an MCMC sample is small. In our example, the estimated lag is roughly 10 for both parameters. In general, efficiencies above 10% are considered good for the MH algorithm. In our example, they are about 10% for both parameters.

Alternatively, we could have listed all parameters manually:

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

Example 2: Effective sample sizes for functions of model parameters

Similarly to other Bayesian postestimation commands, `bayesstats ess` accepts expressions to compute results for functions of model parameters. For example, we can use expression `(sd:sqrt({var}))` with a label, `sd`, to compute effective sample sizes for the standard deviation of `mpg` in addition to the variance.

```
. bayesstats ess (sd:sqrt({var})) {var}
Efficiency summaries    MCMC sample size =    10,000
                        Efficiency:  min =     .1071
                                     avg =     .1082
                                     max =     .1094

          sd : sqrt({var})
```

	ESS	Corr. time	Efficiency
sd	1093.85	9.14	0.1094
var	1070.99	9.34	0.1071

ESS and efficiency are higher for the standard deviation than for the variance, which means that we need slightly more iterations to estimate `{var}` with the same precision as `sd`.

If we wanted, we could have suppressed the sd legend in the output above by specifying the nolegend option.

◁

Stored results

bayesstats ess stores the following in r():

Scalars

r(mcmcsize)	MCMC sample size used in the computation
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(ess)	matrix with effective sample sizes, correlation times, and efficiencies for parameters in r(names)
r(ess_chain#)	matrix ess for chain #, if sepchains is specified

Methods and formulas

Let θ be a scalar model parameter and $\{\theta_t\}_{t=1}^{T}$ be an MCMC sample of size T drawn from the marginal posterior distribution of θ. The effective sample size of the MCMC sample of θ is given by

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

where ρ_k is the lag-k autocorrelation of the MCMC sample, and max_lags is the maximum number less than or equal to ρ_{lag} such that for all $k = 1, \ldots,$ 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.

The lag-k autocorrelation is $\rho_k = \gamma_k/\gamma_0$, where

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

is the empirical autocovariance of lag k, and γ_0 simplifies to the sample variance. $\hat{\theta}$ is the posterior mean estimator.

Correlation time is defined as T/ESS, and efficiency is defined as the reciprocal of the correlation time, ESS/T. Because ESS is between 0 and T, inclusively, the efficiency is always between 0 and 1.

In the presence of multiple chains, the overall ESS is computed as the sum of the individual ESS statistics calculated using each chain independently. Correlation times and efficiencies are then computed using the overall ESS and the total MCMC sample size, $M \times T$, where M is the number of chains.

Also see

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

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

[BAYES] **bayesselect** — Bayesian variable selection for linear regression

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

[BAYES] **Bayesian postestimation** — Postestimation tools after Bayesian estimation

[BAYES] **bayesstats summary** — Bayesian summary statistics