bayestest model — Hypothesis testing using model posterior probabilities

### Description

`bayestest model` computes posterior probabilities of Bayesian models fit using the `bayesmh` command or the `bayes` prefix. These posterior probabilities can be used to test hypotheses about model parameters. The command reports marginal likelihoods, prior probabilities, and posterior probabilities for all tested models.

### Quick start

Compute posterior probabilities of models corresponding to previously saved estimation results M1 and M2

```
bayestest model M1 M2
```

As above, but specify prior probabilities for models

```
bayestest model M1 M2, prior(0.3 0.7)
```

### Menu

Statistics > Bayesian analysis > Hypothesis testing using model posterior probabilities
Syntax

bayestest model [namelist] [ , options]

where namelist is a name, a list of names, _all, or *. A name may be ., meaning the current (active) estimates. _all and * mean the same thing.

<table>
<thead>
<tr>
<th>options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main</td>
<td>Description</td>
</tr>
<tr>
<td>prior(numlist)</td>
<td>specify prior probabilities for tested models; default is all models are equally likely</td>
</tr>
<tr>
<td>* chains(_all</td>
<td>numlist)</td>
</tr>
<tr>
<td>* sechains</td>
<td>compute results separately for each chain</td>
</tr>
<tr>
<td>Advanced</td>
<td>Description</td>
</tr>
<tr>
<td>marglmethod(method)</td>
<td>specify marginal-likelihood approximation method; default is to use Laplace–Metropolis approximation, lmetropolis; rarely used</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lmetropolis</td>
<td>Laplace–Metropolis approximation; default</td>
</tr>
<tr>
<td>hmean</td>
<td>harmonic-mean approximation</td>
</tr>
</tbody>
</table>

Options

Main

prior(numlist) specifies prior probabilities for models. By default, all models are assumed to be equally likely. You may specify probabilities for all tested models, in which case the probabilities must sum to one. Alternatively, you may specify probabilities for all but the last model, in which case the sum of the specified probabilities must be less than one, and the probability for the last model is computed as one minus this sum.

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.

sechains 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 sechains is relevant only when option nchains() is specified with bayesmh or the bayes prefix.

Advanced

marglmethod(method) specifies a method for approximating the marginal likelihood. method is either lmetropolis, the default, for Laplace–Metropolis approximation or hmean for harmonic-mean approximation. This option is rarely used.
Remarks and examples

Remarks are presented under the following headings:
- Introduction
- Testing nested hypotheses
- Comparing models with different priors

Introduction

In this entry, we describe hypothesis testing by computing model posterior probabilities, probabilities of Bayesian models given observed data. For interval hypothesis testing, see [BAYES] bayestest interval.

The bayestest model command computes posterior probabilities for specified models. The computed probabilities can be used to compare which model is more likely among considered models given observed data. You can compare models that differ only in several covariates or models with completely different regression functions, such as linear and nonlinear models. You can compare models with different outcome distributions or with different prior distributions or both. The only requirements are that the considered models have proper posterior distributions and that the same data are used to fit the models. If MCMC is used to approximate posterior distributions, convergence of MCMC should also be verified before model comparison.

The results reported by bayestest model are related to Bayes factors; see [BAYES] bayesstats ic to compute Bayes factors.

To use bayestest model, you must store estimation results after each Bayesian model of interest. You can use estimates store (see [R] estimates store) to store estimation results after bayesmh or the bayes prefix, as you can with other estimation commands, provided you also saved simulation results from bayesmh or the bayes prefix using the saving() option. See Storing estimation results after Bayesian estimation in [BAYES] Bayesian postestimation for details.

Testing nested hypotheses

Consider the following Bayesian regression model for auto.dta,

\[ \text{mpg} = \beta_0 + \beta_1 \text{weight} + \beta_2 \text{length} + \epsilon \]

where weight and length are the original weight and length variables rescaled to have similar scale as mpg.

We assume that errors are normally distributed: \( \epsilon \sim \text{normal}(0, \sigma^2) \). We also assume a noninformative Jeffreys prior for the parameters: \((\beta, \sigma^2) \sim 1/\sigma^2\). Suppose that we are interested in testing whether there is a relationship between mileage and weight and length of cars. We will consider four models: the mean-only model, the model with weight only, the model with length only, and the full model with both covariates.

In a frequentist setting, the four models correspond to the following hypotheses: \( H_0: \beta_1 = 0, \beta_2 = 0 \), \( H_0: \beta_1 = 0 \), and \( H_0: \beta_2 = 0 \). In a Bayesian setting, we cannot formulate point hypotheses for parameters with continuous distributions; see [BAYES] bayestest interval for examples. However, we can compute probabilities of how likely each of the four models is given the observed data.
Let’s load auto.dta and generate rescaled versions of `weight` and `length`.

```stata
. use https://www.stata-press.com/data/r16/auto
    (1978 Automobile Data)
. generate weight1 = weight/100
. generate length1 = length/10
```

Next, we fit the four models using `bayesmh`. We use the `saving()` option to save the simulation datasets so that we can store estimation results of each model for later use with `bayestest model`.

The first model we fit is the mean-only model. We store its estimation results as `meanonly`.

```stata
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
>   prior({mpg:}, flat) prior({var}, jeffreys)
>   saving(meanonly_simdata) burnin(3500)
note: adaptation option `maxiter()' changed to 35
Burn-in ...
Simulation ...
Model summary

Likelihood:
    mpg ~ normal({mpg:_cons},{var})
Priors:
    {mpg:_cons} ~ 1 (flat)
    {var} ~ jeffreys

Bayesian normal regression
Random-walk Metropolis-Hastings sampling
MCMC iterations = 13,500
Burn-in = 3,500
MCMC sample size = 10,000
Number of obs = 74
Acceptance rate = .2627
Efficiency: min = .105
            avg = .1064
            max = .1078
Log marginal-likelihood = -234.64617

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>MCSE</th>
<th>Median</th>
<th>95% Cred. Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>34.80707</td>
<td>5.963995</td>
<td>.181615</td>
<td>34.23247</td>
<td>24.9129 - 47.6883</td>
</tr>
</tbody>
</table>
```

To accommodate the Jeffreys prior for the parameters, we specify suboption `flat` within the `prior()` option for coefficients to request the flat prior with the density of 1 and suboption `jeffreys` within `prior()` for the variance parameter to request a Jeffreys prior. We also specify a longer burn-in period to improve convergence of MCMC samples for all examples. (Remember to use `bayesgraph` to check convergence of MCMC.)
We fit the second model containing only covariate `length1` and store its results as `length`:

```
. set seed 14
. bayesmh mpg length1, likelihood(normal({var}))
    > prior({mpg:}, flat) prior({var}, jeffreys)
    > saving(length_simdata) burnin(3500)
note: adaptation option `maxiter()` changed to 35
Burn-in ...
Simulation ...
Model summary

Likelihood:
   mpg ~ normal(xb_mpg,{var})

Priors:
   {mpg:length1 _cons} ~ 1 (flat)
   {var} ~ jeffreys

(1) Parameters are elements of the linear form xb_mpg.

Bayesian normal regression
Random-walk Metropolis-Hastings sampling
MCMC iterations = 13,500
Burn-in = 3,500
MCMC sample size = 10,000
Number of obs = 74
Acceptance rate = .2865
Efficiency: min = .0771
            avg = .07938
            max = .08286

Log marginal-likelihood = -198.7678

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>MCSE</th>
<th>Median</th>
<th>[95% Cred. Interval]</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpg</td>
<td>-2.069861</td>
<td>.1882345</td>
<td>.006539</td>
<td>-2.068094</td>
<td>-2.44718 -1.706264</td>
</tr>
<tr>
<td>length1</td>
<td>60.20346</td>
<td>3.562119</td>
<td>.127411</td>
<td>60.20927</td>
<td>53.34306 67.22423</td>
</tr>
<tr>
<td>_cons</td>
<td>12.88852</td>
<td>2.273808</td>
<td>.081887</td>
<td>12.62042</td>
<td>9.169482 18.16685</td>
</tr>
</tbody>
</table>

file length_simdata.dta saved
. estimates store length
```
We fit the third model containing only covariate `weight1` and store its results as `weight`:

```stata
. set seed 14
. bayesmh mpg weight1, likelihood(normal({var}))
  > prior({mpg:}, flat) prior({var}, jeffreys)
  > saving(weight_simdata) burnin(3500)
note: adaptation option `maxiter()` changed to 35
Burn-in ... Simulation ...
Model summary

Likelihood:
  mpg ~ normal(xb_mpg,{var})

Priors:
  `{mpg:weight1 _cons} ~ 1 (flat)
   {var} ~ jeffreys

(1) Parameters are elements of the linear form xb_mpg.

Bayesian normal regression
Random-walk Metropolis-Hastings sampling
  MCMC iterations = 13,500
  Burn-in = 3,500
  MCMC sample size = 10,000
  Number of obs = 74
  Acceptance rate = .1735
  Efficiency: min = .0463
table: avg = .06694
          max = .07989

Log marginal-likelihood = -198.20751

<table>
<thead>
<tr>
<th></th>
<th>Equal-tailed [95% Cred. Interval]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
</tr>
<tr>
<td>mpg</td>
<td>-.6014409</td>
</tr>
<tr>
<td>weight1</td>
<td>39.45934</td>
</tr>
<tr>
<td>_cons</td>
<td></td>
</tr>
<tr>
<td>var</td>
<td>12.13997</td>
</tr>
</tbody>
</table>
```

file `weight_simdata.dta` saved
. estimates store weight
Finally, we fit the last model containing both covariates and store its results as full:

```
. set seed 14
. bayesmh mpg weight1 length1, likelihood(normal({var}))
  > prior({mpg:}, flat) prior({var}, jeffreys)
  > saving(full_simdata) burnin(3500)
  note: adaptation option maxiter() changed to 35
Burn-in ...
Simulation ...
Model summary

Likelihood:
  mpg ~ normal(xb_mpg,{var})
Prior:
{mpg:weight1 length1 _cons} ~ 1 (flat)
{var} ~ jeffreys

(1) Parameters are elements of the linear form xb_mpg.
Bayesian normal regression
MCMC iterations = 13,500
Random-walk Metropolis-Hastings sampling
Burn-in = 3,500
MCMC sample size = 10,000
Number of obs = 74
Acceptance rate = .2323
Efficiency: min = .05455
  avg = .06647
  max = .08085
Log marginal-likelihood = -196.86195

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>MCSE</th>
<th>Median</th>
<th>[95% Cred. Interval]</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>weight1</td>
<td>-.3977027</td>
<td>.1580411</td>
<td>.005558</td>
<td>-.401646</td>
<td>-.6965175</td>
</tr>
<tr>
<td>length1</td>
<td>-.7599159</td>
<td>.5546754</td>
<td>.021944</td>
<td>-.7502182</td>
<td>-1.907818</td>
</tr>
<tr>
<td>_cons</td>
<td>47.5913</td>
<td>6.132597</td>
<td>.262563</td>
<td>47.5666</td>
<td>35.89593</td>
</tr>
<tr>
<td>var</td>
<td>11.81753</td>
<td>1.96315</td>
<td>.07608</td>
<td>11.59273</td>
<td>8.729182</td>
</tr>
</tbody>
</table>

file full_simdata.dta saved
. estimates store full

Example 1: Computing posterior probabilities of models

We now use bayestest model to compute posterior probabilities of the four models.

. bayestest model meanonly length weight full
Bayesian model tests

|         | log(ML) | P(M)  | P(M|y)   |
|---------|---------|-------|---------|
| meanonly| -234.6462 | 0.2500 | 0.0000  |
| length  | -198.7678 | 0.2500 | 0.1055  |
| weight  | -198.2075 | 0.2500 | 0.1848  |
| full    | -196.8619 | 0.2500 | 0.7097  |

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

The mean-only model is very unlikely compared with other models. The length and weight models are somewhat likely with the respective posterior probabilities of 0.11 and 0.18, and the full model has the highest posterior probability of 0.71.
Example 2: Specifying prior probabilities of models

If we have some prior knowledge about each of the models, we can use the `prior()` option to specify prior probabilities for each model. For example, suppose that we have prior knowledge that the weight model is much more likely than the full model so that the prior probabilities are 0.1 for the mean-only model and the length model, 0.6 for the weight model, and only 0.2 for the full model.

```
.bayestest model meanonly length weight full, prior(0.1 0.1 0.6 0.2)
```

Bayesian model tests

|          | log(ML)   | P(M)   | P(M|y)  |
|----------|-----------|--------|---------|
| meanonly | -234.6462 | 0.1000 | 0.0000  |
| length   | -198.7678 | 0.1000 | 0.0401  |
| weight   | -198.2075 | 0.6000 | 0.4210  |
| full     | -196.8619 | 0.2000 | 0.5389  |

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

Under the specified prior, posterior probabilities of the weight and full models are now more similar: 0.42 and 0.54, respectively, but the full model is still preferable.

The above is equivalent to the following prior specification:

```
.bayestest model meanonly length weight full, prior(0.1 0.1 0.6)
```

(output omitted)

Using our results, we conclude that mpg is related to both weight and length and would proceed with the full model.

After your analysis, remember to erase the saved simulation datasets you no longer need. For example, we erase all of them by typing

```
. erase meanonly_simdata.dta
. erase weight_simdata.dta
. erase length_simdata.dta
. erase full_simdata.dta
```

Comparing models with different priors

In the previous section, we used `bayestest model` to compare nested hypotheses about which covariates to include in the regression function. We can use `bayestest model` to compare models with not only different covariates but also different outcome distributions and priors for parameters.

We continue our analysis of `auto.dta`, but for simplicity, we now consider the mean-only model for mpg. Let’s compare models with two slightly different informative priors. We use an informative normal–inverse-gamma prior for both models,

\[(\beta_0|\sigma^2) \sim N(\mu_0, \sigma^2/n_0)\]
\[\sigma^2 \sim \text{InvGamma}(\nu_0/2, \nu_0\sigma_0^2/2)\]

with \(\mu_0 = 25\), \(n_0 = 10\), and \(\sigma_0^2 = 30\), but we consider two different values for the degrees of freedom: \(\nu_0 = 5\) and \(\nu_0 = 1\).
We use `bayesmh` to fit our models. Following the formulas, we specify a `normal()` prior for the constant \{mpg:\_cons\} (mean parameter) and an inverse-gamma prior `igamma()` for the variance parameter \{var\}. We specify an expression for the variance of the normal prior distribution in parentheses.

We fit the first model with $\nu_0 = 5$ and store its estimation results as `informative1`.

```
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
    > prior({mpg:}, normal(25,{var}/10))
    > prior({var}, igamma(2.5,75)) saving(inf1_simdata)
Burn-in ...
Simulation ... 
Model summary

Likelihood:  \[ mpg \sim \text{normal}\{(mpg:\_cons),{var}\} \]
Priors:  \begin{align*}
{mpg:\_cons} & \sim \text{normal}(25,{var}/10) \\
{var} & \sim \text{igamma}(2.5,75)
\end{align*}

Bayesian normal regression  \[ \text{MCMC iterations} = 12,500 \]
Random-walk Metropolis-Hastings sampling  \[ \text{Burn-in} = 2,500 \]
\[ \text{MCMC sample size} = 10,000 \]
\[ \text{Number of obs} = 74 \]
\[ \text{Acceptance rate} = .2548 \]
\[ \text{Efficiency: min} = .09065 \]
\[ \text{avg} = .1049 \]
\[ \text{max} = .1192 \]

Log marginal-likelihood = -238.55856

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>MCSE</th>
<th>Median</th>
<th>[95% Cred. Interval]</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpg _cons</td>
<td>21.71853</td>
<td>.6592655</td>
<td>.019091</td>
<td>21.69554</td>
<td>20.44644 to 23.04896</td>
</tr>
<tr>
<td>mpvar</td>
<td>35.47405</td>
<td>5.823372</td>
<td>.193417</td>
<td>34.72454</td>
<td>25.84419 to 48.2257</td>
</tr>
</tbody>
</table>
```

file `inf1_simdata.dta` saved
. estimates store informative1
We fit the second model with \( \nu_0 = 1 \) and store its estimation results as `informative2`.

```stata
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
    > prior({mpg:}, normal(25,{var}/10))
    > prior({var}, igamma(0.5,15)) saving(inf2_simdata)
Burn-in ... Simulation ...
Model summary

Likelihood:
    mpg ~ normal({mpg:cons},{var})
Priors:
    {mpg:cons} ~ normal(25,{var}/10)
    {var} ~ igamma(0.5,15)

Bayesian normal regression
Random-walk Metropolis-Hastings sampling
MCMC iterations = 12,500
Burn-in = 2,500
MCMC sample size = 10,000
Number of obs = 74
Acceptance rate = .2261
Efficiency: min = .0941
        avg = .109
        max = .1239
Log marginal-likelihood = -239.4049

    Mean    Std. Dev.     MCSE    Median  [95% Cred. Interval]
mpg
    var     35.89504    6.288571   .178665   35.17056    25.86084    50.21624

file inf2_simdata.dta saved
. estimates store informative2
```

Example 3: Comparing models with informative priors

We now use `bayestest model` to compare our models with two different informative priors.

```stata
. bayestest model informative1 informative2
Bayesian model tests

|         | log(ML) | P(M)  | P(M|y) |
|---------|---------|-------|-------|
| informative1 | -238.5586 | 0.5000 | 0.6998 |
| informative2  | -239.4049 | 0.5000 | 0.3002 |

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

Assuming that both models are equally likely a priori, the posterior probability of the `informative1` stored results, 0.70, is much higher than the probability of the `informative2` stored results, 0.3.
Example 4: Comparing a model with noninformative prior

A note of caution regarding comparing models with informative and noninformative priors—models with noninformative priors will often win because they are typically in most agreement with the observed data. For models with noninformative priors, most of the information about parameters is contained in a likelihood. As such, any model with an informative prior that is not in perfect agreement with the data will not fit data as well as a model with a noninformative prior.

For example, let’s fit our constant-only model using a noninformative Jeffreys prior for the parameters.

```
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
  > prior({mpg:}, flat) prior({var}, jeffreys)
  > saving(jeffreys_simdata)
Burn-in ...
Simulation ...
Model summary

Likelihood:
  mpg ~ normal({mpg:_cons},{var})

Priors:
  {mpg:_cons} ~ 1 (flat)
  {var} ~ jeffreys

Bayesian normal regression
Random-walk Metropolis-Hastings sampling
MCMC iterations = 12,500
Burn-in = 2,500
MCMC sample size = 10,000
Number of obs = 74
Acceptance rate = .2668
Efficiency: min = .09718
avg = .1021
max = .1071

Log marginal-likelihood = -234.645

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>MCSE</th>
<th>Median</th>
<th>[95% Cred. Interval]</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>34.76572</td>
<td>5.91534</td>
<td>.180754</td>
<td>34.18391</td>
<td>24.9129 - 47.61286</td>
</tr>
</tbody>
</table>
```

file jeffreys_simdata.dta saved
. estimates store jeffreys

Let’s now compare this model with our two informative models.

```
. bayestest model informative1 informative2 jeffreys
Bayesian model tests

|                  | log(ML) | P(M) | P(M|y) |
|------------------|---------|------|-------|
| informative1     | -238.5586 | 0.3333 | 0.0194 |
| informative2     | -239.4049 | 0.3333 | 0.0083 |
| jeffreys         | -234.6450 | 0.3333 | 0.9723 |

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

The posterior probability of the Jeffreys model is 0.97.
Finally, at the end of our analysis, we erase all the simulation datasets we no longer need. We erase all of them by typing

```
. erase inf1_simdata.dta
. erase inf2_simdata.dta
. erase jeffreys_simdata.dta
```

## Stored results

`bayestest model` stores the following in `r()`:

- **Macros**
  - `r(names)`: names of estimation results used
  - `r(marglmethod)`: method for approximating marginal likelihood: `lmetropolis` or `hmean`
  - `r(chains)`: chains used in the computation, if `chains()` is specified

- **Matrices**
  - `r(test)`: test results for models in `r(names)`
  - `r(test_chain#)`: matrix test for chain #, if `sepchains` is specified

## Methods and formulas

Suppose we have `r` models `M_j` for `j = 1, \ldots, r` with prior probabilities `P(M_j)` such that `\sum_{j=1}^{r} P(M_j) = 1`. Then, posterior probability for model `J` is

\[
P(M_j|y) = \frac{P(y|M_j)P(M_j)}{P(y)}
\]

where `P(y|M_j) = m_j(y)` is the marginal likelihood of `M_j` with respect to `y`, and `P(y) = \sum_{j=1}^{r} P(y|M_j)P(M_j)`. See *Methods and formulas* in [BAYES] `bayesmh` for details about computing marginal likelihood.

With multiple chains, the `bayestest model` command uses the averaged across chains log marginal-likelihood for calculations. If the `sepchains` option is specified, the results are calculated and reported separately for each chain.

## 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] `bayesstats ic` — Bayesian information criteria and Bayes factors

[BAYES] `bayesstats summary` — Bayesian summary statistics

[BAYES] `bayestest interval` — Interval hypothesis testing