bmapredict — Predictions after BMA regression

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

Description

`bmapredict` computes Bayesian model averaging (BMA) predictions using current estimation results produced by `bmaregress`. Certain predictions such as credible intervals (CrIs) require that you also run `bmacoefsample` with the `saving()` option after `bmaregress`.

`bmareps` generates a random subset of Markov chain Monte Carlo (MCMC) replicates of simulated outcomes from the entire MCMC sample and stores them as new variables in the current dataset. This command is useful for checking model fit. `bmareps` requires that you first save the MCMC sample of model parameters by using the `saving()` option with the `bmacoefsample` command.

Quick start

Compute and save posterior predictive means in a variable `pmean1` after a BMA regression with a fixed g

```
bmaregress y x1-x12
bmapredict pmean1, mean
```

Compute and save posterior predictive means in a variable `pmean2` after a BMA regression with a random g; requires that `bmacoefsample` be run after `bmaregress`

```
bmaregress y x1-x12, gprior(robust) saving(bmamodel1)
bmacoefsample, saving(bmacoef1)
bmapredict pmean2, mean
```

Compute 95% CrIs of simulated outcome, and save them in variables `cri_l` and `cri_u`; requires that `bmacoefsample` be run after `bmaregress` with a fixed or random g

```
bmaregress y x1-x12, saving(bmamodel2)
bmacoefsample, saving(bmacoef2)
bmapredict cri_l cri_u, cri
```

Same as above, but simulate an entire predictive outcome dataset, and save it in `bmaypred.dta`

```
bmapredict, saving(bmaypred)
```

Equivalent to the above, but use `bayespredict` and save results in `bayesypred.dta`

```
bayespredict {_ysim}, saving(bayesypred)
```

Write your own program `yfuncprog` to compute a more complicated function of a simulated outcome, and use `bayespredict` to simulate its posterior distribution

```
bayespredict (yfunc: @yfuncprog {_ysim}), saving(bayesyfuncpred)
```

Generate three replicates from the posterior predictive distribution, save them as `predy1`, `predy2`, and `predy3`, and specify random-number seeds for reproducibility

```
bmaregress y x1-x12, saving(bmamodel3) rseed(123)
bmacoefsample, saving(bmacoef3) rseed(982)
bmareps predy*, nreps(3) rseed(23672)
```

Generate log predictive-scores (always available and do not require bmacoefsample)

```
bmapredict lps, lps
```

Menu

Statistics > Bayesian model averaging > Predictions

Syntax

Syntax is presented under the following headings:

Compute analytical posterior predictive summaries (available only with fixed g)
Compute MCMC-sample posterior predictive summaries (available after bmacoefsample)
Compute predictions of simulated outcome (available after bmacoefsample)
Generate a subset of MCMC replicates of simulated outcome (available after bmacoefsample)
Compute log predictive-scores

Compute analytical posterior predictive summaries (available only with fixed g)

Analytical posterior mean

```
bmapredict [type] newvar [if] [in], mean [analytic]
```

Analytical posterior standard deviation

```
bmapredict [type] newvar [if] [in], std [analytic]
```

Analytical posterior predictive means and standard deviations are available after bmaregress only for models with a fixed g, in which case the analytic option is implied.

Compute MCMC-sample posterior predictive summaries (available after bmacoefsample)

Posterior mean of simulated outcome

```
bmapredict [type] newvar [if] [in], mean [mcmcsample meanopts simopts]
```

Posterior median or posterior standard deviation of simulated outcome

```
bmapredict [type] newvar [if] [in], median|std [mcmcsample simopts]
```

Credible intervals of simulated outcome

```
bmapredict [type] newvar_l newvar_u [if] [in], cri [mcmcsample criopts simopts]
```

MCMC-sample posterior predictive summaries are available only after bmacoefsample with the saving() option is used after bmaregress. For models with a random g, option mcmcsample is implied for mean and std. This option is always implied for median and cri.

Compute predictions of simulated outcome (available after bmacoefsample)

Simulate outcome and save in a file

`bmapredict` [*if*] [*in*], `saving(`*filename*[, `replace`]) [*simopts*]

Also see *Syntax* in [BAYES] **bayespredict**, particularly for predictions of functions of a simulated outcome.

Predictions for a simulated outcome are available only after `bmacoefsample` with the `saving()` option is used after `bmaregress`.

Generate a subset of MCMC replicates of simulated outcome (available after bmacoefsample)

`bmareps` [*type*] *newrepspec* [*if*] [*in*], `nreps(`#`)` [*simopts*]

newrepspec is *newvar* with `nreps(1)` for a single replicate and *stub*`*` with `nreps(`#`)`, where # is greater than 1, for multiple replicates.

Replicates of a simulated outcome are available only after `bmacoefsample` with the `saving()` option is used after `bmaregress`.

Compute log predictive-scores

`bmapredict` [*type*] *newvar* [*if*] [*in*], `lps`

simopts	Description
Simulation	
`rseed(`#`)`	random-number seed
[`no`]`dots`	suppress or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is `dots`
`dots(`#[, `every(`#`)`]`)`	display dots as simulation is performed

meanopts	Description
Main	
`mcse(`*newvar*`)`	create *newvar* containing Monte Carlo standard errors (MCSEs)
Advanced	
`batch(`#`)`	specify length of block for batch-means calculations; default is `batch(0)`
`corrlag(`#`)`	specify maximum autocorrelation lag; default varies
`corrtol(`#`)`	specify autocorrelation tolerance; default is `corrtol(0.01)`

criopts	Description
Main	
`clevel(`#`)`	set CrI level; default is `clevel(95)`
`hpd`	calculate highest posterior density (HPD) CrIs instead of the default equal-tailed CrIs

Options

Options are presented under the following headings:

Options for analytical posterior predictive summaries
Options for MCMC-sample posterior predictive summaries
Options for predictions of simulated outcome
Options for bmareps
Option for log predictive-scores

Options for analytical posterior predictive summaries

Main

`mean` with `analytic` calculates analytical posterior predictive means of an outcome and stores them as a new variable in the current dataset. For models with fixed g, option `analytic` is implied.

`std` with `analytic` calculates analytical posterior predictive standard deviations of an outcome and stores them as a new variable in the current dataset. For models with fixed g, option `analytic` is implied.

`analytic` specifies to use analytical expressions to calculate posterior predictive means or standard deviations. This option is implied for models with fixed g, and it is not available with random g. Analytical computations are available directly after `bmaregress`.

Options for MCMC-sample posterior predictive summaries

Main

`mean` with `mcmcsample` calculates MCMC-sample posterior predictive means of a simulated outcome and stores them as a new variable in the current dataset. For models with random g, option `mcmcsample` is implied.

`median` calculates posterior medians of a simulated outcome and stores them as a new variable in the current dataset. Option `mcmcsample` is always implied with `median`.

`std` with `mcmcsample` calculates MCMC-sample posterior predictive standard deviation of a simulated outcome and stores them as a new variable in the current dataset. For models with random g, option `mcmcsample` is implied.

`cri` calculates CrIs for a simulated outcome and stores the corresponding lower and upper bounds in two new variables in the current dataset. Option `mcmcsample` is always implied with `cri`.

`mcmcsample` specifies to estimate posterior predictive means or standard deviations from the MCMC sample of model parameters instead of using analytical expressions. This option is the only choice for models with random g and thus is implied with random g.

`mcse(`*newvar*`)` is for use in a combination with option `mean`. It adds *newvar* of storage type *type* containing MCSEs for the posterior means of a simulated outcome.

`clevel(#)` specifies the credible level, as a percentage, for equal-tailed and HPD CrIs. The default is `clevel(95)` or as set by [BAYES] **set clevel**. This option requires that `cri` also be specified.

`hpd` calculates the HPD CrIs instead of the default equal-tailed CrIs. This option requires that `cri` also be specified.

Simulation

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

nodots, dots, dots(#), and dots(#, every(#)) specify to suppress or display dots during simulation. nodots, the default, suppresses the display of dots. dots displays dots every 100 iterations and iteration numbers every 1,000 iterations; it is a synonym for dots(100, every(1000)). dots(#) displays a dot every # iterations. If dots(..., every(#)) is specified, then an iteration number is displayed every #th iteration instead of a dot. dots(, every(#)) is equivalent to dots(1, every(#)).

Advanced

The advanced options are available only with option mean.

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

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.

Options for predictions of simulated outcome

Main

saving(*filename*[, replace]) saves the simulated outcome in *filename*.dta. It also saves auxiliary estimation results in *filename*.ster, which is accessible by specifying estimates use *filename*. The replace option specifies to overwrite *filename*.dta and *filename*.ster if they exist.

Simulation

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

nodots, dots, dots(#), and dots(#, every(#)) specify to suppress or display dots during simulation. nodots, the default, suppresses the display of dots. dots displays dots every 100 iterations and iteration numbers every 1,000 iterations; it is a synonym for dots(100, every(1000)). dots(#) displays a dot every # iterations. If dots(..., every(#)) is specified, then an iteration number is displayed every #th iteration instead of a dot. dots(, every(#)) is equivalent to dots(1, every(#)).

Options for bmareps

Main

`nreps(#)` specifies the number of MCMC replicates of simulated outcome to be drawn at random from the entire sample of MCMC replicates. # must be an integer between 1 and the MCMC sample size, inclusively. The generated replicates are stored as new variables in the current dataset. For a single replicate, `nreps(1)`, you specify one new variable name. For multiple replicates, you specify a *stub*`*`, in which case the replicates will be stored in variables *stub*1, *stub*2, . . ., *stub*R, where R is the number of replicates specified in `nreps()`.

Simulation

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

`nodots`, `dots`, `dots(#)`, and `dots(#, every(#))` specify to suppress or display dots during simulation. `nodots`, the default, suppresses the display of dots. `dots` displays dots every 100 iterations and iteration numbers every 1,000 iterations; it is a synonym for `dots(100, every(1000))`. `dots(#)` displays a dot every # iterations. If `dots(..., every(#))` is specified, then an iteration number is displayed every #th iteration instead of a dot. `dots(, every(#))` is equivalent to `dots(1, every(#))`.

Option for log predictive-scores

Main

`lps` calculates log predictive-scores and stores them as a new variable in the current dataset.

Remarks and examples

Bayesian predictions include a simulated outcome, which is a sample from the posterior predictive distribution of the fitted Bayesian model, and their functions; see [BAYES] **bayespredict** for details. In the context of BMA, this posterior predictive distribution also accounts for model uncertainty. Often of interest are summaries of this distribution such as means, medians, standard deviations, and CrIs.

`bmapredict` simulates an outcome from its BMA posterior predictive distribution and, optionally, saves it in a dataset specified in the `saving()` option. It also computes BMA posterior predictive summaries and stores them as new variables in the current dataset when you specify one of options `mean`, `median`, `std`, or `cri`.

For BMA models with a fixed g, you can compute posterior predictive means and standard deviations analytically or, if an MCMC sample of model parameters is available, from the simulated MCMC predictive sample. The analytical computations are always the default whenever they are available; that is, the `analytic` option is implied with the `mean` and `std` options with a fixed g. For BMA models with a random g, analytical computations are not available, and thus the MCMC-sample computations are the only choice—the `mcmcsample` option is implied with a random g with `mean` and `std`. For other summaries such as posterior predictive medians and CrIs (options `median` and `cri`), only MCMC-sample estimates are available regardless of whether parameter g is fixed or random; that is, the `mcmcsample` option is always implied.

MCMC-sample computations require that you run the bmacoefsample command with the saving() option following bmaregress to generate and save a posterior sample of model parameters, which will be used to produce an MCMC sample of a simulated outcome from its posterior predictive distribution; see *Methods and formulas*.

Once a posterior MCMC sample of model parameters is available (by running bmacoefsample), you can also use bayespredict, one of the standard Bayesian postestimation commands, to obtain BMA predictions. This command is useful, for instance, if you need to compute any functions of the simulated outcome; see [BAYES] **bayespredict**. In fact, typing

```
. bmapredict, saving(bmaysim)
```

is equivalent to typing

```
. bayespredict {_ysim}, saving(bmaysim)
```

bmareps generates a random subset of MCMC replicates of a simulated outcome from the entire MCMC sample and stores them as new variables in the current dataset. This command is useful for checking model fit. It is similar to the bayesreps command as described in [BAYES] **bayespredict**.

You can also use the lps option to generate log predictive-scores. This can also be done by using bmastats lps, generate() and is provided with bmapredict for completeness; see [BMA] **bmastats lps**.

bmapredict (as well as bmareps) is a stochastic command; use the rseed() option for reproducibility. Remember that to have a complete reproducibility, you will also need to specify the rseed() option with bmacoefsample and bmaregress, which are also stochastic commands. If you do not need to reproduce each individual step, a better approach is to set a random-number seed once at the beginning of your analysis by using set seed (see [R] **set seed**).

▷ Example 1: Prediction for BMA linear regression with fixed g

Consider the Ames housing dataset (De Cock 2011), ameshouses, also used in a Kaggle competition, which describes residential houses sold in Ames, Iowa, between 2006 and 2010. It contains about 80 housing (and related) characteristics such as home size, amenities, and location. This dataset is often used for building predictive models for home sale price, saleprice. We will use BMA to model home sale price and evaluate its predictive performance. Here we will use just a few predictors to demonstrate some of the bmapredict features.

We split the dataset into two subsamples in the ratio of 2 to 1. The first subsample will be used for fitting the models and the second for testing. We use the log-transformed sale price, lsaleprice, as our outcome variable. And we generate a new variable, age, to record the age of home in years at the time of sale.

```
. use https://www.stata-press.com/data/r19/ameshouses
(Ames house data)
. splitsample, generate(sample) nsplit(2) split(2 1) rseed(18)
. generate double lsaleprice = log(saleprice)
. generate age = yrsold - yearbuilt + 1
```

We fit a BMA linear model for lsaleprice and use just a few of the available predictors for demonstration purposes. The type of sale, saletype, is included as a factor variable. The overall home quality and condition are included as ordinal predictors. We use the default specifications for model and g priors. The total number of predictors, 29, makes model enumeration impractical, so the MCMC model composition (MC3) sampling is used instead. And we specify the rseed() option for reproducibility.

```
. bmaregress lsaleprice i.saletype overallqual overallcond age
> paveddrive grlivarea totalbsmtsf garagearea garagecars
> bedroom kitchenabvgr kitchenqual bsmtqual exterqual extercond
> fullbath halfbath fireplaces wooddecksf lotarea
> if sample == 1, rseed(18)

Burn-in ...
Simulation ...
Computing model probabilities ...

Bayesian model averaging                        No. of obs         =    973
Linear regression                               No. of predictors  =     27
MC3 sampling                                               Groups  =     27
                                                           Always  =      0
                                                No. of models      =    454
                                                    For CPMP >= .9 =     85
Priors:                                         Mean model size    = 11.435
  Models: Beta-binomial(1, 1)                   Burn-in            =  2,500
   Cons.: Noninformative                        MCMC sample size   = 10,000
   Coef.: Zellner's g                           Acceptance rate    = 0.1410
       g: Benchmark, g = 973                    Shrinkage, g/(1+g) = 0.9990
  sigma2: Noninformative                        Mean sigma2        =  0.027

Sampling correlation = 0.9690
```

lsaleprice	Mean	Std. dev.	Group	PIP
overallqual	.084876	.0069513	9	1
overallcond	.0624909	.0053748	10	1
age	-.0031076	.0002871	11	1
grlivarea	.0001866	.0000233	13	1
garagecars	.091476	.0099425	16	1
fireplaces	.0570484	.0100975	25	1
wooddecksf	.0002054	.0000468	26	1
totalbsmtsf	.0000683	.0000162	14	.99776
kitchenqual	-.0276135	.0105654	19	.94116
lotarea	1.86e-06	8.67e-07	27	.89429
bedroomabvgr	.0138705	.013693	17	.56959
saletype				
New	.0184764	.0285989	6	.34325
bsmtqual	-.0047449	.0080652	20	.30144
saletype				
CWD	-.0148399	.0723153	1	.056743
paveddrive	.0020311	.0098775	12	.056586
extercond	.000576	.0029977	22	.051934
fullbath	.0005671	.0040245	23	.034588
saletype				
WD	-.0004454	.0052115	8	.032251
kitchenabvgr	-.0007089	.0061078	18	.022655
saletype				
Oth	.0021639	.0228432	7	.022015
halfbath	.0001984	.002301	24	.021049
exterqual	-.0001345	.0020265	21	.020406

saletype				
Con	.0018756	.0258891	2	.017179
ConLI	-.0011395	.0225588	4	.015548
garagearea	5.36e-08	6.54e-06	15	.014047
saletype				
ConLD	.0005622	.0100442	3	.013859
Always				
_cons	10.74662	.0676172	0	1

```
Note: Coefficient posterior means and std. dev. estimated from 454 models.
Note: Default priors are used for models and parameter g.
Note: 1 predictor with PIP less than .01 not shown.
```

The `bmaregress` commands visited a total of 454 models with an average size of about 11. There are 11 predictors with posterior inclusion probability (PIP) greater than 0.5. Whether the home is new and the height of the basement (`bsmtqual`) appear to impact the home prices, too, with respective PIPs of 0.34 and 0.3. The other predictors have PIPs below 10%.

Let's use the `bmapredict` command to compute the posterior mean predictions for `lsaleprice` on the test subsample, then compute the mean squared errors of these predictions.

```
. bmapredict pmean1 if sample == 2, mean
note: computing analytical posterior predictive means.
. generate double sqerr1 = (lsaleprice-pmean1)^2
(973 missing values generated)
. summarize sqerr1
```

Variable	Obs	Mean	Std. dev.	Min	Max
sqerr1	487	.0177806	.0354926	1.24e-07	.5608268

The estimated mean squared error for the model is 0.018.

Parameter g is fixed in our example, so `bmapredict` computed the analytical posterior predictive mean. For comparison, we can compute one based on an MCMC sample. To do this, we first need to simulate a posterior sample of model parameters by using `bmacoefsample`. To save time, we use a smaller MCMC sample of 1,000.

```
. bmaregress, saving(hprices_bmareg1)
note: file hprices_bmareg1.dta saved.
. bmacoefsample, saving(hprices_sample1) rseed(18) mcmcsize(1000) nodots
Simulation ...
file hprices_sample1.dta saved.
```

We then specify the mcmcsample option with bmapredict to compute the MCMC-sample posterior predictive mean.

```
. bmapredict pmean1_s if sample == 2, mean mcmcsample rseed(18)
note: computing posterior predictive means using simulation.
Computing predictions ...
. summarize pmean1 pmean1_s
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
       pmean1 |        487    12.03638    .3525371   11.10996   12.97869
     pmean1_s |        487    12.03606    .3524122   11.11674   12.98349
```

The two predicted means are very similar, as is expected provided the MCMC sample size is sufficiently large and the MCMC converged.

Now that we have an MCMC sample of model parameters, we can, for example, generate predictive CrIs for the test subsample and check their coverage.

```
. bmapredict cril1 criu1 if sample == 2, cri rseed(18)
note: computing credible intervals using simulation.
Computing predictions ...
. generate cover1 = lsaleprice < criu1 & lsaleprice > cril1 if sample == 2
(973 missing values generated)
. summarize cover1
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
      cover1 |        487    .9774127    .1487364          0          1
```

The coverage of the simulated 95% equal-tailed predictive CrIs is 98%, slightly larger than expected.

Example 2: Prediction for BMA linear regression with random g

The usage of bmapredict after a BMA linear regression with random g is slightly different. To illustrate the prediction workflow, let's fit the same BMA regression as in example 1 but use the random hyperg(3) prior for parameter g here.

```
. bmaregress lsaleprice i.saletype overallqual overallcond age
> paveddrive grlivarea totalbsmtsf garagearea garagecars
> bedroom kitchenabvgr kitchenqual bsmtqual exterqual extercond
> fullbath halfbath fireplaces wooddecksf lotarea
> if sample == 1, gprior(hyperg 3) rseed(18)
Burn-in ...
Simulation ...
Computing model probabilities ...
Bayesian model averaging                        No. of obs        =    973
Linear regression                               No. of predictors =     27
MC3 and adaptive MH sampling                              Groups  =     27
                                                          Always  =      0
                                                No. of models     =    678
                                                   For CPMP >= .9 =    319
Priors:                                         Mean model size   = 12.622
  Models: Beta-binomial(1, 1)                   Burn-in           =  2,500
   Cons.: Noninformative                        MCMC sample size  = 10,000
   Coef.: Zellner's g                           Acceptance rate   = 0.5261
       g: Hyper-g(3)
  sigma2: Noninformative                        Mean sigma2       =  0.027
```

Sampling correlation = 0.8885

lsaleprice	Mean	Std. dev.	Group	PIP
overallqual	.084121	.0070143	9	1
overallcond	.0626002	.0054331	10	1
age	-.0030242	.000306	11	1
grlivarea	.0001802	.0000232	13	1
garagecars	.0906061	.0105574	16	1
fireplaces	.0576947	.0101318	25	1
wooddecksf	.0002048	.000047	26	1
totalbsmtsf	.0000691	.0000165	14	.9968
kitchenqual	-.0278064	.0102489	19	.9515
lotarea	1.89e-06	8.37e-07	27	.9103
bedroomabvgr	.0182072	.013439	17	.7243
bsmtqual	-.0076919	.0091544	20	.4843
saletype				
New	.022116	.0305646	6	.4058
paveddrive	.0069869	.0175817	12	.1869
saletype				
CWD	-.0424225	.1177667	1	.161
extercond	.0011747	.0041903	22	.1077
garagearea	7.44e-07	.0000155	15	.0755
kitchenabvgr	-.0024381	.0112473	18	.0752
saletype				
ConLD	.0036524	.0242701	3	.0719
fullbath	.0010832	.0055586	23	.0715
saletype				
ConLw	.000148	.0221823	5	.0701
Con	.0078374	.0522285	2	.0688
WD	-.0002908	.0080338	8	.0687
halfbath	.0005647	.0038594	24	.0544
saletype				
ConLI	-.0038273	.0413487	4	.0521
Oth	.0039455	.0312492	7	.0437
exterqual	-.0003791	.0032394	21	.0416
Always				
_cons	10.75005	.0707137	0	1

Note: Coefficient posterior means and std. dev. estimated from 678 models.
Note: Default prior is used for models.

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred.	interval]
g	425.3529	206.9134	6.15565	375.5357	179.5198	963.0143
Shrinkage	.9971914	.0011648	.000033	.9973442	.9944604	.9989627

The number of visited models is larger, 678 versus 454, as is the posterior mean model size, 13 versus 11.

Unlike for BMA models with fixed g, the mean and standard deviations cannot be computed analytically with random g. All posterior predictive summaries now need to be estimated from an MCMC predictive sample of `lsaleprice`. As in example 1 for MCMC-sample mean and CrIs, we first need to run `bmacoefsample` to obtain a posterior sample of model parameters.

```
. bmaregress, saving(hprices_bmareg2)
note: file hprices_bmareg2.dta saved.
. bmacoefsample, saving(hprices_sample2) rseed(18) mcmcsize(1000) nodots
Simulation ...
file hprices_sample2.dta saved.
```

We compute posterior mean predictions and the mean squared error for these predictions. Because our BMA model uses random g, the `mcmcsample` option is automatically implied.

```
. bmapredict pmean2 if sample == 2, mean rseed(18)
note: computing posterior predictive means using simulation; option mcmcsample
      implied.
Computing predictions ...
. generate double sqerr2 = (lsaleprice-pmean2)^2
(973 missing values generated)
. summarize sqerr2
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
      sqerr2 |        487    .0180237    .0358707   7.16e-08   .5625565
```

The estimated mean squared error for this model is 0.018.

We can also compute 95% equal-tailed predictive CrIs and their coverage in this example.

```
. bmapredict cril2 criu2 if sample == 2, cri rseed(18)
note: computing credible intervals using simulation.
Computing predictions ...
. generate cover2 = lsaleprice < criu2 & lsaleprice > cril2 if sample == 2
(973 missing values generated)
. summarize cover2
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
      cover2 |        487    .9794661    .1419635          0          1
```

The estimated coverage is 98%.

The predictive performances of the two models, with fixed and random g, are similar in terms of the mean squared error and CrI coverage.

◁

▷ Example 3: Predictions by using bayespredict

After generating a sample of model parameters by using `bmacoefsample`, you can also use the more advanced functionality of the `bayespredict` command to compute BMA predictions. For instance, let's generate a posterior sample of mean squared prediction errors, which are functions of a simulated outcome. There is no automatic way to compute this statistic by using any of the available options of `bmapredict` or `bayespredict`, so we need to write a program that computes it.

```
. program psqerr2
  1.         args mse ysim
  2.         local touse $BAYESPR_touse
  3.         local y $BAYESPR_extravars
  4.         tempvar v
  5.         generate double `v' = (`y'-`ysim')^2 if `touse'
  6.         summarize `v', meanonly
  7.         scalar `mse' = r(mean)
  8. end
```

The psqerr2 program has two arguments: mse to store the mean squared error value and ysim that will contain the simulated outcome generated by bayespredict for each MCMC iteration. To compute the squared errors, we will need the actual outcome values, labeled as y in the program, which will be passed to the program as an extra variable. The program then uses the observed outcome values and the simulated values to compute the squared errors, using the proper estimation sample `touse'. Finally, it saves the mean of the squared errors in the scalar `mse'.

Next, we call bayespredict by using its syntax for user-defined programs; see [BAYES] **bayespredict**. We provide the name of our program and the vector of simulations of the outcome, {_ysim}. We also provide the name of the observed outcome variable, lsaleprice, in the extravars() option. And we save our simulated squared errors in the bmapred2 dataset for later use.

```
. bayespredict (sqerr2:@psqerr2 {_ysim}, extravars(lsaleprice)) if sample == 2,
> saving(bmapred2) rseed(18)
Computing predictions ...
file bmapred2.dta saved.
file bmapred2.ster saved.
```

We can use the bayesstats summary command (see [BAYES] **bayesstats summary**) to compute, for instance, the posterior mean of the mean squared error estimates.

```
. bayesstats summary {sqerr2} using bmapred2
Posterior summary statistics                     MCMC sample size =     1,000
```

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred. interval]	
sqerr2	.0461	.0030704	.000097	.0460156	.0402083	.0525146

The posterior mean is about 0.046. It is, as expected, larger than the mean squared error calculated in example 2, where we used the predicted mean outcome to compute the errors. In practice, there is no reason to compute the mean squared error via simulation, as we showed in this example. We did this purely to demonstrate a more advanced usage of bayespredict.

◁

Methods and formulas

Methods and formulas are presented under the following headings:

BMA predictions for the linear model
Analytic predictive mean and standard deviation for fixed g
Simulating outcome from its posterior predictive distribution

BMA predictions for the linear model

We consider predictions for a BMA linear model, which is fit by the `bmaregress` command. Consider a $(p+1)\times 1$ vector of regression coefficients and the intercept in the original metric, $\boldsymbol{\theta} = (\boldsymbol{\beta}', \alpha)'$ and error variance σ^2.

Let $\mathbf{X}$ be a $n \times p$ matrix of predictor values used to fit the BMA model, $\mathbf{X}^*$ be a $q \times p$ matrix of new predictor values, $\mathbf{y}^*$ be a vector of posterior predictions of size q, and $\mathbf{X}^*_{\mathbf{1}} = [\mathbf{X}^* \ \mathbf{1}_q]$, where $\mathbf{1}_q$ is a $q \times 1$ vector of ones. In some cases, certain statistics of $\mathbf{y}^*$, such as mean and standard deviation, can be computed directly; others require simulations from the predictive distribution. We consider these two cases separately.

Analytic predictive mean and standard deviation for fixed g

With a fixed g, the posterior predictive mean and variance of $\mathbf{y}^*$ can be computed analytically from the posterior means of $\boldsymbol{\theta} = (\boldsymbol{\beta}', \alpha)'$ and σ^2 and posterior variance of $\boldsymbol{\theta}$ as follows,

$$E(\mathbf{y}^*|\mathbf{y}, \mathbf{X}, \mathbf{X}^*_{\mathbf{1}}) = \mathbf{X}^*_{\mathbf{1}} E(\boldsymbol{\theta}|\mathbf{y})$$
$$\mathrm{Var}(\mathbf{y}^*|\mathbf{y}, \mathbf{X}, \mathbf{X}^*_{\mathbf{1}}) = E(\sigma^2|\mathbf{y})\mathbf{I}_q + \mathbf{X}^*_{\mathbf{1}} \mathrm{Var}(\boldsymbol{\theta}|\mathbf{y}) \mathbf{X}^{*\prime}_{\mathbf{1}}$$

where $E(\boldsymbol{\theta}|\mathbf{y})$, $E(\sigma^2|\mathbf{y})$, and $\mathrm{Var}(\boldsymbol{\theta}|\mathbf{y})$ are given, respectively, by (11), (13), and (12) in *Methods and formulas* of [BMA] **bmaregress**.

The posterior mean $E(\mathbf{y}^*|\mathbf{y}, \mathbf{X}, \mathbf{X}^*_{\mathbf{1}})$ is computed with options `mean` and `analytic`. The posterior standard deviation of $\mathbf{y}^*$ is the square root of the diagonal of $\mathrm{Var}(\mathbf{y}^*|\mathbf{y}, \mathbf{X}, \mathbf{X}^*_{\mathbf{1}})$. It is computed with options `std` and `analytic`.

Simulating outcome from its posterior predictive distribution

The posterior predictive distribution of $\mathbf{y}^*$ can be estimated via MCMC simulations. Suppose we have a sample $\{\boldsymbol{\beta}_t, \alpha_t, \sigma^2_t\}_{t=1}^T$ of regression coefficients, intercepts, and error variances from a BMA linear model. Such posterior samples are generated by the `bmacoefsample` command (see [BMA] **bmacoefsample**). Here T is the MCMC sample size as stored in `e(mcmcsize2)` by `bmacoefsample`.

For each t, an error vector $\boldsymbol{\epsilon}^*_t$ and outcome $\mathbf{y}^*_t$ are simulated according to

$$\begin{aligned} \boldsymbol{\epsilon}^*_t|\boldsymbol{\beta}_t, \sigma^2_t, \mathbf{X} &\sim N_q(\mathbf{0}, \sigma^2_t \mathbf{I}_q) \\ \mathbf{y}^*_t|\mathbf{X}, \mathbf{X}^*_{\mathbf{1}} &= \alpha_t + \mathbf{X}^*\boldsymbol{\beta}_t + \boldsymbol{\epsilon}^*_t \end{aligned}$$

The result is a predictive sample $\{\mathbf{y}^*_t\}_{t=1}^T$, which is conditionally independent given $\mathbf{X}^*$ and $\mathbf{X}$.

If you use `bmapredict, saving(`*filename*`)` on the same dataset used to fit the model, the simulated outcome $\{\tilde{\mathbf{y}}\}_{t=1}^T$ of size $n \times T$ will be saved in *filename*`.dta` with the observed data matrix $\mathbf{X}$ being used in place of $\mathbf{X}^*$ in the above. This is sometimes called a replication sample. Posterior summaries corresponding to options `mean mcmcsample`, `median`, `std mcmcsample`, and `cri` are computed based on the simulated outcome $\{\tilde{\mathbf{y}}\}_{t=1}^T$ as described in *Methods and formulas* of [BAYES] **bayespredict**.

`bmareps` generates a subset of replicates $\{\tilde{\mathbf{y}}\}_{t=1}^{T_{\text{reps}}}$, where T_{reps} is determined by the `nreps(#)` option.

For the computation of log predictive-scores (option `lps`), see *Methods and formulas* of [BMA] **bmastats lps**.

Reference

De Cock, D. 2011. Ames, Iowa: Alternative to the Boston housing data as an end of semester regression project. *Journal of Statistics Education* 19(3). https://doi.org/10.1080/10691898.2011.11889627.

Also see

[BMA] **BMA postestimation** — Postestimation tools for Bayesian model averaging

[BMA] **bmaregress** — Bayesian model averaging for linear regression

[BMA] **bmacoefsample** — Posterior samples of regression coefficients

[BMA] **Glossary**

[BAYES] **bayespredict** — Bayesian predictions

[U] 20 Estimation and postestimation commands