Bayesian Model Averaging with BMS

for BMS Version 0.3.5

Martin Feldkircher and Stefan Zeugner

August 3, 2022

Abstract

This manual is a brief introduction to applied Bayesian Model Averaging with the R package `BMS`. The manual is structured as a hands-on tutorial for readers with few experience with BMA. Readers from a more technical background are advised to consult the table of contents for formal representations of the concepts used in BMS.
For other tutorials and more information, please refer to `http://bms.zeugner.eu`.

Contents

1 A Brief Introduction to Bayesian Model Averaging

This section reiterates some basic concepts, and introduces some notation for readers with limited knowledge of BMA. Readers with more experience in BMA should skip this chapter and directly go to section 2. For a more thorough introduction to BMA, consult Hoeting et al. (1999). Note that a version this vignette has been published as Feldkircher and Zeugner (2015). If needed, please cite from the latter reference.

1.1 Bayesian Model Averaging

Bayesian Model Averaging (BMA) addresses model uncertainty in a canonical regression problem. Suppose a linear model structure, with y being the dependent variable, α_γ a constant, β_γ the coefficients, and ε a normal IID error term with variance σ^2:

$$y = \alpha_\gamma + X_\gamma \beta_\gamma + \varepsilon \qquad \varepsilon \sim N(0, \sigma^2 I) \tag{1}$$

A problem arises when there are many potential explanatory variables in a matrix X: Which variables $X_\gamma \in \{X\}$ should be then included in the model? And how important are they? The direct approach to do inference on a single linear model that includes all variables is inefficient or even infeasible with a limited number of observations.

BMA tackles the problem by estimating models for all possible combinations of $\{X\}$ and constructing a weighted average over all of them. If X contains K potential variables, this means estimating 2^K variable combinations and thus 2^K models. The model weights for this averaging stem from posterior model probabilities that arise from Bayes' theorem:

$$p(M_\gamma|y,X) \;=\; \frac{p(y|M_\gamma,X)p(M_\gamma)}{p(y|X)} \;= \frac{p(y|M_\gamma,X)p(M_\gamma)}{\sum_{s=1}^{2^K} p(y|M_s,X)p(M_s)} \tag{2}$$

Here, $p(y|X)$ denotes the *integrated* likelihood which is constant over all models and is thus simply a multiplicative term. Therefore, the posterior model probability (PMP) $p(M_\gamma|y,X)$ is proportional to[1] the marginal likelihood of the model $p(y|M_\gamma,X)$ (the probability of the data given the model M_γ) times a prior model probability $p(M_\gamma)$ – that is, how probable the researcher thinks model M_γ before looking at the data. Renormalization then leads to the PMPs and thus the model weighted posterior distribution for any statistic θ (e.g. the coefficients β):

$$p(\theta|y,X) = \sum_{\gamma=1}^{2^K} p(\theta|M_\gamma,y,X)p(M_\gamma|X,y)$$

The model prior $p(M_\gamma)$ has to be elicited by the researcher and should reflect prior beliefs. A popular choice is to set a uniform prior probability for each model $p(M_\gamma) \propto 1$ to represent the lack of prior knowledge. Further model prior options will be explored in section 3.

1.2 Bayesian Linear Models and Zellner's g prior

The specific expressions for marginal likelihoods $p(M_\gamma|y,X)$ and posterior distributions $p(\theta|M_\gamma,y,X)$ depend on the chosen estimation framework. The literature standard is to use a 'Bayesian regression' linear model with a specific prior structure called 'Zellner's g prior' as will be outlined in this section.[2]

For each individual model M_γ suppose a normal error structure as in (1). The need to obtain posterior distributions requires to specify the priors on the model parameters. Here, we place 'improper' priors on the constant and error variance, which means they are evenly distributed over their domain: $p(\alpha_\gamma) \propto 1$, i.e. complete prior uncertainty where the prior is located. Similarly, set $p(\sigma) \propto \sigma^{-1}$.

The crucial prior is the one on regression coefficients β_γ: Before looking into the data (y,X), the researcher formulates her prior beliefs on coefficients into a normal distribution with a specified mean and variance. It is common to assume a conservative prior mean of zero

[1] Proportionality is expressed with the sign $\propto$: i.e. $p(M_\gamma|y,X) \;\propto\; p(y|M_\gamma,X)p(M_\gamma)$

[2] Note that the presented framework is very similar to the natural normal-gamma-conjugate model - which employs proper priors for α and σ. Nonetheless, the resulting posterior statistics are virtually identical.

for the coefficients to reflect that not much is known about them. Their variance structure is defined according to Zellner's g: $\sigma^2(\frac{1}{g}X'_\gamma X_\gamma)^{-1}$:

$$\beta_\gamma|g \sim N\left(0, \sigma^2\left(\frac{1}{g}X'_\gamma X_\gamma\right)^{-1}\right)$$

This means that the researcher thinks coefficients are zero, and that their variance-covariance structure is broadly in line with that of the data X_γ. The hyperparameter g embodies how certain the researcher is that coefficients are indeed zero: A small g means few prior coefficient variance and therefore implies the researcher is quite certain (or conservative) that the coefficients are indeed zero. In contrast, a large g means that the researcher is very uncertain that coefficients are zero.

The posterior distribution of coefficients reflects prior uncertainty: Given g, it follows a t-distribution with expected value $E(\beta_\gamma|y, X, g, M_\gamma) = \frac{g}{1+g}\hat{\beta}_\gamma$, where $\hat{\beta}_\gamma$ is the standard OLS estimator for model γ. The expected value of coefficients is thus a convex combination of OLS estimator and prior mean (zero). The more conservative (smaller) g, the more important is the prior, and the more the expected value of coefficients is shrunk toward the prior mean zero. As $g \to \infty$, the coefficient estimator approaches the OLS estimator. Similarly, the posterior variance of β_γ is affected by the choice of g:[3]

$$Cov(\beta_\gamma|y, X, g, M_\gamma) = \frac{(y-\bar{y})'(y-\bar{y})}{N-3}\frac{g}{1+g}\left(1 - \frac{g}{1+g}R^2_\gamma\right)(X'_\gamma X_\gamma)^{-1}$$

I.e. the posterior covariance is similar to that of the OLS estimator, times a factor that includes g and R^2_γ, the OLS R-squared for model γ. The appendix shows how to apply the function `zlm` in order to estimate such models out of the BMA context.

For BMA, this prior framwork results into a very simple marginal likelihood $p(y|M_\gamma, X, g)$, that is related to the R-squared and includes a size penalty factor adjusting for model size k_γ:

$$p(y|M_\gamma, X, g) \propto (y-\bar{y})'(y-\bar{y})^{-\frac{N-1}{2}}(1+g)^{-\frac{k_\gamma}{2}}\left(1 - \frac{g}{1+g}\right)^{-\frac{N-1}{2}}$$

The crucial choice here concerns the form of the hyperparameter g. A popular 'default' approach is the 'unit information prior' (UIP), which sets $g = N$ commonly for all models and thus attributes about the same information to the prior as is contained in one observation. (Please refer to section 5 for a discussion of other g-priors.)[4]

2 A BMA Example: Attitude Data

This section shows how to perform basic BMA with a small data set and how to obtain posterior coefficient and model statistics.

2.1 Model Sampling

Equipped with this basic framework, let us explore one of the data sets built into R: The 'attitude' dataset describes the overall satisfaction rating of a large organization's employees, as well as several specific factors such as `complaints`, the way of handling complaints within the organization (for more information type `help(attitude)`). The data includes 6 variables, which means $2^6 = 64$ model combinations. Let us stick with the UIP g-prior (in this case $g = N = 30$). Moreover, assume uniform model priors (which means that our expected prior model parameter size is $K/2 = 3$).

First load the data set by typing

```
> data(attitude)
```

In order to perform BMA you have to load the BMS library first, via the command:

```
> library(BMS)
```

[3] here, N denotes sample size, and $\bar{y}$ the sample mean of the response variable

[4] Note that BMS is, in principle not restricted to Zellner's g-priors, as quite different coefficient priors might be defined by R-savy users.

Now perform Bayesian model sampling via the function `bms`, and write results into the variable `att`.

```
> att = bms(attitude, mprior = "uniform", g="UIP", user.int=F)
```

`mprior = "uniform"` means to assign a uniform model prior, `g="UIP"`, the unit information prior on Zellner's g. The option `user.int=F` is used to suppress user-interactive output for the moment.[5] The first argument is the data frame `attitude`, and `bms` assumes that its first column is the response variable.[6]

2.2 Coefficient Results

The coefficient results can be obtained via

```
> coef(att)
```

```
                PIP    Post Mean    Post SD Cond.Pos.Sign Idx
complaints 0.9996351  0.684449094 0.13038429    1.00000000   1
learning   0.4056392  0.096481513 0.15135419    1.00000000   3
advance    0.2129325 -0.026686161 0.09133894    0.00000107   6
privileges 0.1737658 -0.011854183 0.06143387    0.00046267   2
raises     0.1665853  0.010567022 0.08355244    0.73338938   4
critical   0.1535886  0.001034563 0.05465097    0.89769774   5
```

The above matrix shows the variable names and corresponding statistics: The second column `Post Mean` displays the coefficients averaged over all models, including the models wherein the variable was not contained (implying that the coefficient is zero in this case). The covariate `complaints` has a comparatively large coefficient and seems to be most important. The importance of the variables in explaining the data is given in the first column `PIP` which represents posterior inclusion probabilities - i.e. the sum of PMPs for all models wherein a covariate was included. We see that with 99.9%, virtually all of posterior model mass rests on models that include `complaints`. In contrast, `learning` has an intermediate PIP of 40.6%, while the other covariates do not seem to matter much. Consequently their (unconditional) coefficients[7] are quite low, since the results quite often include models where these coefficients are zero.

The coefficients' posterior standard deviations (`Post SD`) reflect further evidence: `complaints` is certainly positive, while `advance` is most likely negative. In fact, the coefficient sign can also be inferred from the fourth column `Cond.Pos.Sign`, the 'posterior probability of a positive coefficient expected value conditional on inclusion', respectively 'sign certainty'. Here, we see that in all encountered models containing this variables, the (expected values of) coefficients for `complaints` and `learning` were positive. In contrast, the corresponding number for `privileges` is near to zero, i.e. in virtually all models that include `privileges`, its coefficient sign is negative. Finally, the last column `idx` denotes the index of the variables' appearance in the original data set, as our results are obviously sorted by PIP.

Further inferring about the importance of our variables, it might be really more interesting to look at their standardized coefficients.[8] Type:

```
> coef(att, std.coefs=T, order.by.pip=F, include.constant=T)
```

```
                 PIP     Post Mean    Post SD Cond.Pos.Sign Idx
complaints  0.9996351  0.7486734114 0.14261872    1.00000000   1
privileges  0.1737658 -0.0119154065 0.06175116    0.00046267   2
learning    0.4056392  0.0930292869 0.14593855    1.00000000   3
raises      0.1665853  0.0090258498 0.07136653    0.73338938   4
critical    0.1535886  0.0008409819 0.04442502    0.89769774   5
```

[5] Note that the argument `g="UIP"` is actually redundant, as this is the default option for `bms`. The default model prior is somewhat different but does not matter very much with this data. Therefore, the command `att = bms(attitude)` gives broadly similar results.

[6] The specification of data can be supplied in different manners, e.g. in 'formulas'. Type `help(lm)` for a comparable function.

[7] Unconditional coefficients are defined as $E(\beta|y,X) = \sum_{\gamma=1}^{2^K} p(\beta_\gamma|,y,X,M_\gamma)p(M_\gamma|y,X)$ i.e. a weighted average over all models, including those where this particular coeffiecnt was restricted to zero. A conditional coeffienct in contrast, is 'conditional on inclusion', i.e. a weighted average only over those models where its regressor was included. Conditional coefficients may be obtained with the command `coef(att, condi.coef =TRUE)`.

[8] Standardized coefficients arise if both the response y and the regressors X are normalized to mean zero and variance one – thus effectively bringing the data down to same order of magnitude.

```
advance     0.2129325 -0.0225561446 0.07720309    0.00000107   6
(Intercept) 1.0000000  1.2015488514         NA            NA   0
```

The standardized coefficients reveal similar importance as discussed above, but one sees that `learning` actually does not matter much in terms of magnitude. Note that `order.by.pip=F` represents covariates in their original order. The argument `include.constant=T` also prints out a (standardized) constant.

2.3 Other Results

Other basic information about the sampling procedure can be obtained via [9].

```
> summary(att)
```

```
Mean no. regressors              Draws             Burnins
          "2.1121"                "64"                "0"
              Time  No. models visited      Modelspace 2^K
  "0.0253396 secs"                "64"                "64"
         % visited         % Topmodels            Corr PMP
             "100"               "100"                "NA"
          No. Obs.         Model Prior             g-Prior
             "30"       "uniform / 3"               "UIP"
   Shrinkage-Stats
       "Av=0.9677"
```

It reiterates some of the facts we already know, but adds some additional information such as `Mean no. regressors`, posterior expected model size (cf. section 3).

Finally, let's look into which models actually perform best: The function `topmodels.bma` prints out binary representations for all models included, but for the sake of illustration let us focus on the best three:[10]

```
> topmodels.bma(att)[,1:3]
```

```
                20       28       29
complaints       1     1.00     1.00
privileges       0     0.00     0.00
learning         0     1.00     1.00
raises           0     0.00     0.00
critical         0     0.00     0.00
advance          0     0.00     1.00
PMP (Exact) 101237 57068.95 22938.02
PMP (MCMC)  101237 57068.95 22938.02
```

We see that the output also includes the posterior model probability for each model.[11] The best model, with 29% posterior model probability,[12] is the one that only includes `complaints`. However the second best model includes `learning` in addition and has a PMP of 16%. Use the command `beta.draws.bma(att)` to obtain the actual (expected values of) posterior coefficient estimates for each of these models.

In order to get a more comprehensive overview over the models, use the command

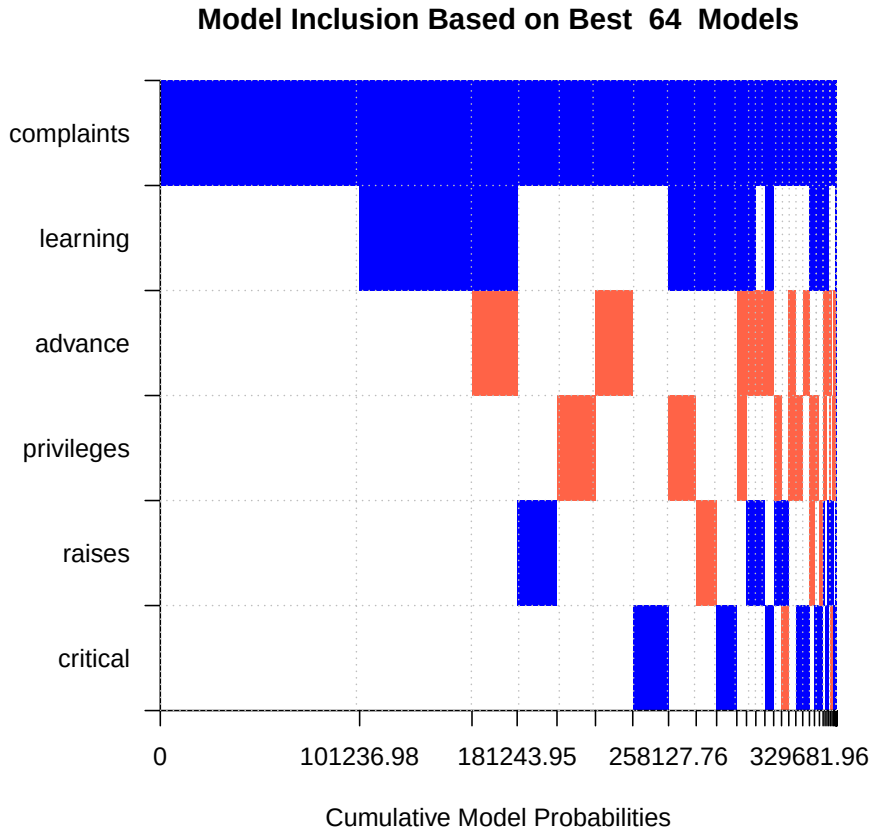
```
> image(att)
```

[9] Note that the command `print(att)` is equivalent to `coef(att); summary(att)`

[10] `topmodel.bma` results in a matrix in which each row corresponds to a covariate and each column to a model (ordered left-to-right by their PMP). The best three models are therefore in the three leftmost columns resulting from `topmodel.bma`, which are extracted via index assignment `[, 1:3]`.

[11] To access the PMP for any model, consider the function `pmpmodel` – cf. `help(pmpmodel)` .

[12] The differentiation between `PMP (Exact)` and `PMP (MCMC)` is of importance if an MCMC sampler was used – cf. section 4.3



Here, blue color corresponds to a positive coefficient, red to a negative coefficient, and white to non-inclusion (a zero coefficient). On the horizontal axis it shows the best models, scaled by their PMPs. We see again that the best model with most mass only includes `complaints`. Moreover we see that complaints is included in virtually all model mass, and unanimously with a positive coefficient. In contrast, `raises` is included very little, and its coefficient sign changes according to the model. (Use `image(att,yprop2pip=T)` for another illustrating variant of this chart.)

3 Model Size and Model Priors

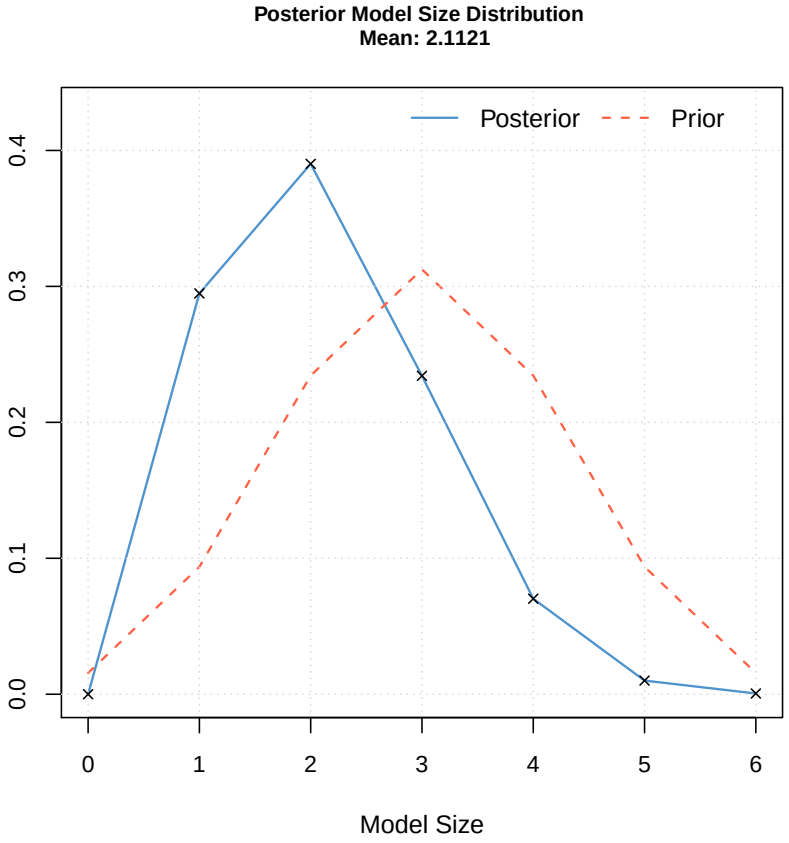
Invoking the command `summary(att)` yielded the important posterior statistic `Mean no. regressors`, the posterior expected model size (i.e. the average number of included regressors), which in our case was 2.11. Note that the posterior expected model size is equal to the sum of PIPs – verify via

```
> sum(coef(att)[,1])

[1] 2.112147
```

This value contrasts with the prior expected model size implictely used in our model sampling: With 2^K possible variable combinations, a uniform model prior means a common prior model probability of $p(M_\gamma) = 2^{-K}$. However, this implies a prior expected model size of $\sum_{k=0}^{K} \binom{K}{k} k 2^{-K} = K/2$. Moreover, since there are more possible models of size 3 than e.g. of size 1 or 5, the uniform model prior puts more mass on intermediate model sizes – e.g. expecting a model size of $k_\gamma = 3$ with $\binom{6}{3} 2^{-K} = 31\%$ probability. In order to examine how far the posterior model size distribution matches up to this prior, type:

```
> plotModelsize(att)
```



We see that while the model prior implies a symmetric distribution around $K/2 = 3$, updating it with the data yields a posterior that puts more importance on parsimonious models.

In order to illustrate the impact of the uniform model prior assumption, we might consider other popular model priors that allow more freedom in choosing prior expected model size and other factors.

3.1 Binomial Model Prior

The binomial model prior constitutes a simple and popular alternative to the uniform prior we just employed. It starts from the covariates' viewpoint, placing a common and fixed inclusion probability θ on each regressor. The prior probability of a model of size k is therefore the product of inclusion and exclusion probabilities:

$$p(M_\gamma) = \theta^{k_\gamma}(1-\theta)^{K-k_\gamma}$$

Since expected model size is $\bar{m} = K\theta$, the researcher's prior choice reduces to eliciting a prior expected model size $\bar{m}$ (which defines θ via the relation $\theta = \bar{m}/K$). Choosing a prior model size of $K/2$ yields $\theta = \frac{1}{2}$ and thus exactly the uniform model prior $p(M_\gamma) = 2^{-K}$. Therefore, putting prior model size at a value $< \frac{1}{2}$ tilts the prior distribution toward smaller model sizes and vice versa. For instance, let's impose this fixed inclusion probability prior such that prior model size equals $\bar{m} = 2$: Here, the option `user.int=T` directly prints out the results as from `coef` and `summary`.[13]

```
> att_fixed = bms(attitude, mprior="fixed", mprior.size=2, user.int=T)

                  PIP      Post Mean    Post SD Cond.Pos.Sign Idx
complaints 0.99971415  0.7034253730 0.12131094    1.00000000   1
learning   0.23916017  0.0536357004 0.11957391    1.00000000   3
advance    0.10625062 -0.0103177406 0.05991418    0.00000250   6
privileges 0.09267430 -0.0057118663 0.04446276    0.00040634   2
```

[13] The command `g="UIP"` was omitted here since `bms` sets this by default anyway.

```
raises     0.09089754  0.0061503218 0.06011618    0.81769332   4
critical   0.08273046  0.0002573042 0.03992658    0.92899714   5

Mean no. regressors               Draws             Burnins
          "1.6114"                "64"                "0"
              Time  No. models visited      Modelspace 2^K
 "0.01743269 secs"                "64"               "64"
         % visited         % Topmodels            Corr PMP
             "100"               "100"               "NA"
          No. Obs.         Model Prior             g-Prior
              "30"         "fixed / 2"               "UIP"
    Shrinkage-Stats
        "Av=0.9677"

Time difference of 0.01743269 secs
```

As seen in `Mean no. regressors`, the posterior model size is now 1.61 which is somewhat smaller than with uniform model priors. Since posterior model size equals the sum of PIPs, many of them have also become smaller than under `att` But interestingly, the PIP of `complaints` has remained at near 100%.

3.2 Custom Prior Inclusion Probabilities

In view of the pervasive impact of `complaints`, one might wonder whether its importance would also remain robust to a greatly unfair prior. For instance, one could define a prior inclusion probability of only $\theta = 0.01$ for the `complaints` while setting a 'standard' prior inclusion probability of $\theta = 0.5$ for all other variables. Such a prior might be submitted to `bms` by assigning a vector of prior inclusion probabilities via its `mprior.size` argument:[14]

```
> att_pip = bms(attitude, mprior="pip", mprior.size=c(.01,.5,.5,.5,.5), user.int=F)
```

But the results (obtained with `coef(att_pip)`) show that `complaints` still retains its PIP of near 100%. Instead, posterior model size decreases (as evidenced in a call to `plotModelsize(att_pip)`), and all other variables obtain a far smaller PIP.

3.3 Beta-Binomial Model Priors

Like the uniform prior, the fixed common θ in the binomial prior centers the mass of of its distribution near the prior model size. A look on the prior model distribution with the following command shows that the prior model size distribution is quite concentrated around its mode.

```
> plotModelsize(att_fixed)
```

This feature is sometimes criticized, in particular by Ley and Steel (2009): They note that to reflect prior uncertainty about model size, one should rather impose a prior that is less tight around prior expected model size. Therefore, Ley and Steel (2009) propose to put a *hyperprior* on the inclusion probability θ, effectively drawing it from a Beta distribution. In terms of researcher input, this prior again only requires to choose the prior expected model size. However, the resulting prior distribution is considerably less tight and should thus reduce the risk of unintended consequences from imposing a particular prior model size.[15]

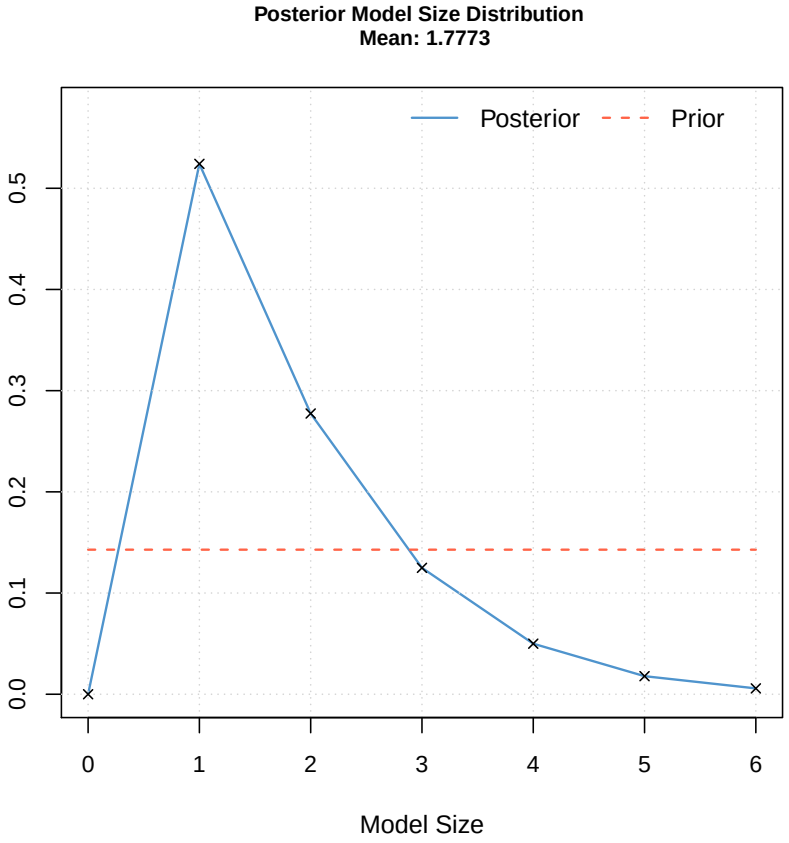
For example, take the beta-binomial prior with prior model size $K/2$[16] – and compare this to the results from `att` (which is equivalent to a fixed θ model prior of prior model size $K/2$.)

```
> att_random = bms(attitude, mprior="random", mprior.size=3, user.int=F)
> plotModelsize(att_random)
```

[14] This implies a prior model size of $\bar{m} = 0.01 + 5 \times 0.5 = 2.51$

[15] Therefore, the beta-binomial model prior with random theta is implemented as the default choice in `bms`.

[16] Note that the arguments here are actually the default values of `bms`, therefore this command is equivalent to `att_random=bms(attitude)`.

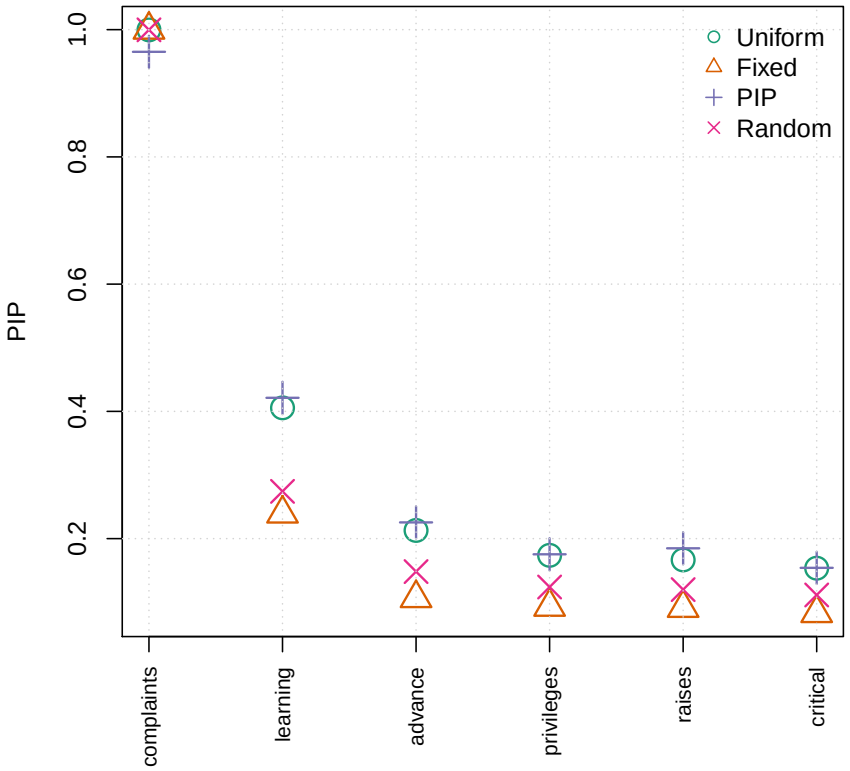


With the beta-binomial specification and prior model size $\bar{m} = K/2$, the model prior is completely flat over model sizes, while the posterior model size turns out to be 1.73. In terms of coefficient and posterior model size distribution, the results are very similar to those of `att_fixed`, even though the latter approach involved a tighter model prior. Concluding, a decrease of prior importance by the use of the beta-binomial framework supports the results found in `att_fixed`.

We can compare the PIPs from the four approaches presented so far with the following command:[17]

```
> plotComp(Uniform=att, Fixed=att_fixed, PIP=att_pip, Random=att_random)
```

[17] This is equivalent to the command `plotComp(att, att_fixed, att_pip, att_random)`



Here as well, `att_fixed` (Fixed) and `att_random` (Random) display similar results with PIPs plainly smaller than those of `att` (Uniform).

Note that the appendix contains an overview of the built-in model priors available in BMS. Moreover, BMS allows the user to define any custom model prior herself and straightforwardly use it in `bms` - for examples, check `http://bms.zeugner.eu/custompriors.php`. Another concept relating to model priors is to keep fixed regressors to be included in every sampled model: Section A.4 provides some examples.

4 MCMC Samplers and More Variables

4.1 MCMC Samplers

With a small number of variables, it is straightforward to enumerate all potential variable combinations to obtain posterior results. For a larger number of covariates, this becomes more time intensive: enumerating all models for 25 covariates takes about 3 hours on a modern PC, and doing a bit more already becomes infeasible: With 50 covariates for instance, there are more than a quadrillion ($\approx 10^{15}$) potential models to consider. In such a case, MCMC samplers gather results on the most important part of the posterior model distribution and thus approximate it as closely as possible. BMA mostly relies on the Metropolis-Hastings algorithm, which 'walks' through the model space as follows:

At step i, the sampler stands at a certain 'current' model M_i with PMP $p(M_i|y, X)$. In step $i + 1$ a candidate model M_j is proposed. The sampler switches from the current model to model M_j with probability $p_{i,j}$:

$$p_{i,j} = \min(1, p(M_j|y, X)/p(M_i|y, x))$$

In case model M_j is rejected, the sampler moves to the next step and proposes a new model M_k against M_i. In case model M_j is accepted, it becomes the current model and has to survive against further candidate models in the next step. In this manner, the number of times each model is kept will converge to the distribution of posterior model probabilities $p(M_i|y, X)$.

In addition to enumerating all models, BMS implements two MCMC samplers that differ in the way they propose candidate models:

- *Birth-death sampler* (`bd`): This is the standard model sampler used in most BMA routines. One of the K potential covariates is randomly chosen; if the chosen covariate forms already part of the current model M_i, then the candidate model M_j will have the same set of covariates as M_i but for the chosen variable ('dropping' a variable). If the chosen covariate is not contained in M_i, then the candidate model will contain all the variables from M_i plus the chosen covariate ('adding' a variable).
- *Reversible-jump sampler* (`rev.jump`): Adapted to BMA by Madigan and York (1995) this sampler either draws a candidate by the birth-death method with 50% probability. In the other case (chosen with 50% probability) a 'swap' is proposed, i.e. the candidate model M_j randomly drops one covariate with respect to M_i and randomly adds one chosen at random from the potential covariates that were not included in model M_i.
- *Enumeration* (`enum`): Up to fourteen covariates, complete enumeration of all models is the default option: This means that instead of an approximation by means of the aforementioned MCMC sampling schemes *all* possible models are evaluated. As enumeration becomes quite time-consuming or infeasible for many variables, the default option is `mcmc="bd"` in case of $K > 14$, though enumeration can still be invoked with the command `mcmc="enumerate"`.

The quality of an MCMC approximation to the actual posterior distribution depends on the number of draws the MCMC sampler runs through. In particular, the sampler has to start out from some model[18] that might not be a 'good' one. Hence the first batch of iterations will typically not draw models with high PMPs as the sampler will only after a while converge to spheres of models with the largest marginal likelihoods. Therefore, this first set of iterations (the 'burn-ins') is to be omitted from the computation of results. In `bms`, the argument `burn` specifies the number of burn-ins, and the argument `iter` the number of subsequent iterations to be retained.

4.2 An Example: Economic Growth

In one of the most prominent applications of BMA, Fernández et al. (2001b) analyze the importance of 41 explanatory variables on long-term term economic growth in 72 countries by the means of BMA. The data set is built into BMS, a short description is available via `help(datafls)`. They employ a uniform model prior and the birth-death MCMC sampler. Their g prior is set to $g = \max(N, K^2)$, a mechanism such that PMPs asymptotically either behave like the Bayesian information criterion (with $g = N$) or the risk inflation criterion ($g = K^2$) – in `bms` this prior is assigned via the argument `g="BRIC"`.

Moreover Fernández et al. (2001b) employ more than 200 million number of iterations after a substantial number of burn-ins. Since this would take quite a time, the following example reenacts their setting with only 50,000 burn-ins and 100,000 draws and will take about 30 seconds on a modern PC:

```
> data(datafls)
> fls1 = bms(datafls, burn=50000, iter=100000, g="BRIC", mprior="uniform", nmodel=2000, mcmc="bd", us
```

Before looking at the coefficients, check convergence by invoking the `summary` command:[19]

```
> summary(fls1)
```

```
Mean no. regressors              Draws             Burnins
          "10.3757"            "1e+05"            "50000"
               Time  No. models visited      Modelspace 2^K
    "7.803406 secs"             "26106"          "2.2e+12"
          % visited         % Topmodels           Corr PMP
          "1.2e-06"                "44"           "0.8985"
           No. Obs.         Model Prior            g-Prior
               "72"    "uniform / 20.5"             "BRIC"
```

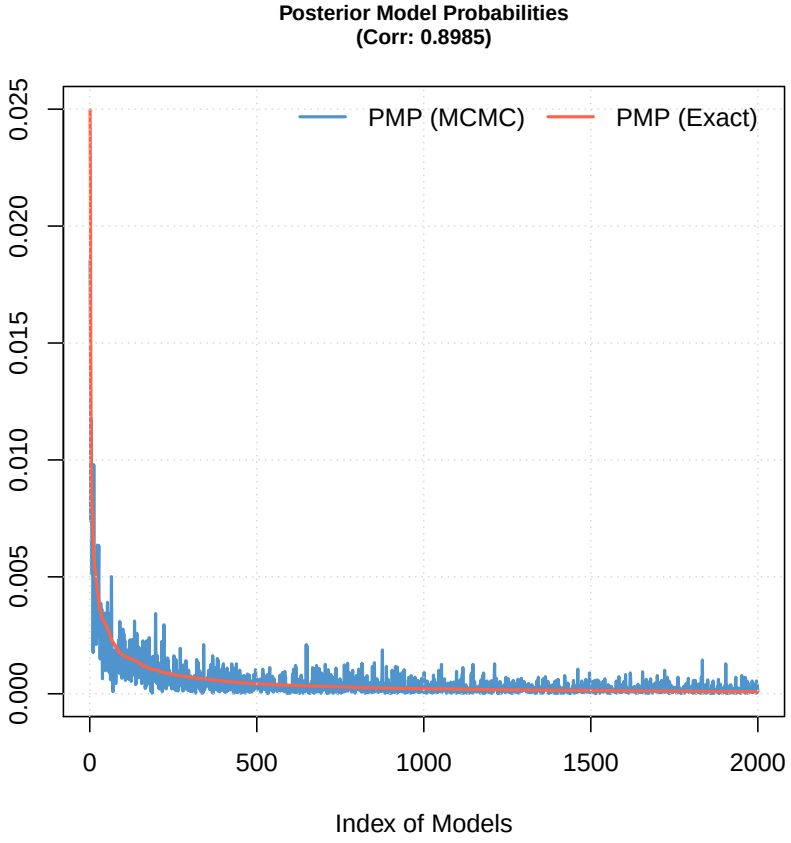
[18] `bms` has some simple algorithms implemented to choose 'good' starting models – consult the option `start.value` under `help(bms)` for more information.

[19] Since MCMC sampling chains are never completely equal, the results presented here might differ from what you get on your machine.

```
Shrinkage-Stats
    "Av=0.9994"
```

Under `Corr PMP`, we find the correlation between iteration counts and analytical PMPs for the 2000 best models (the number 2000 was specified with the `nmodel=2000` argument). At 0.8985, this correlation is far from perfect but already indicates a good degree of convergence. For a closer look at convergence between analytical and MCMC PMPs, compare the actual distribution of both concepts:

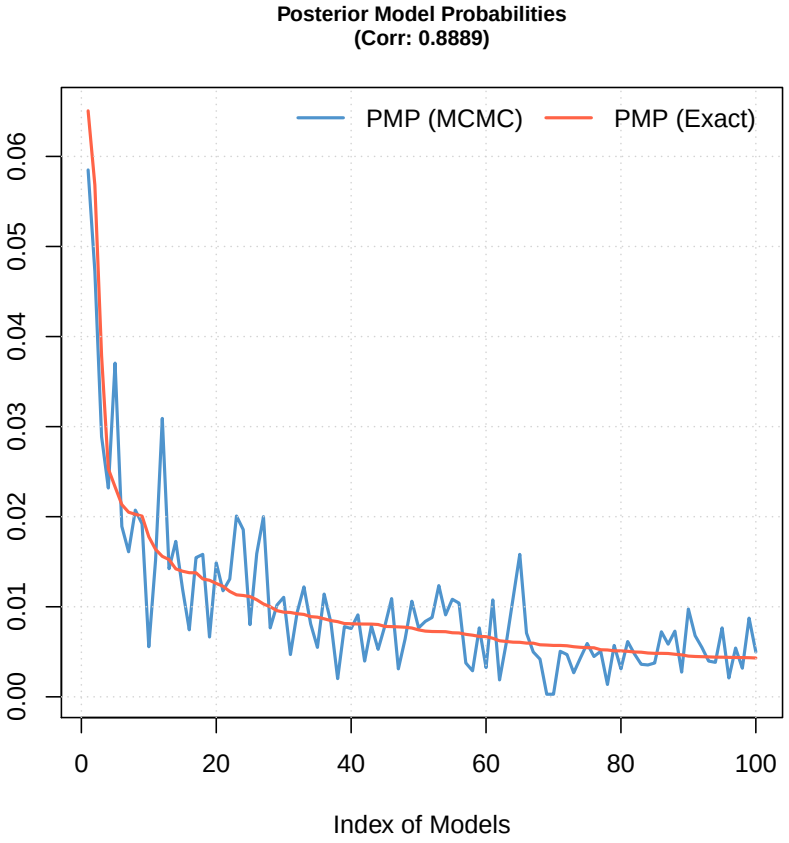
```
> plotConv(fls1)
```



The chart presents the best 2,000 models encountered ordered by their analytical PMP (the red line), and plots their MCMC iteration counts (the blue line). For an even closer look, one might just check the corresponding image for just the best 100 models with the following command:[20]

```
> plotConv(fls1[1:100])
```

[20] With `bma` objects such as `fls1`, the indexing parentheses `[]` are used to select subsets of the (ranked) best models retained in the object. For instance, while `fls1` contains 2,000 models, `fls1[1:100]` only contains the 100 best models among them. Correspondingly, `fls1[37]` would only contain the 37th-best model. Cf. `help('[.bma')`



4.3 Analytical vs. MCMC likelihoods

The example above already achieved a decent level of correlation among analytical likelihoods and iteration counts with a comparatively small number of sampling draws. In general, the more complicated the distribution of marginal likelihoods, the more difficulties the sampler will meet before converging to a good approximation of PMPs. The quality of approximation may be inferred from the number of times a model got drawn vs. their actual marginal likelihoods. Partly for this reason, `bms` retains a pre-specified number of models with the highest PMPs encountered during MCMC sampling, for which PMPs and draw counts are stored. Their respective distributions and their correlation indicate how well the sampler has converged.

However, due to RAM limits, the sampling chain can hardly retain more than a few 100,000 of these models. Instead, it computes aggregate statistics on-the-fly, taking iteration counts as model weights. For model convergence and some posterior statistics `bms` retains only the 'top' (highest PMP) `nmodel` models it encounters during iteration. Since the time for updating the iteration counts for the 'top' models grows in line with their number, the sampler becomes considerably slower the more 'top' models are to be kept. Still, if they are sufficiently numerous, those best models can already cover most of posterior model mass - in this case it is feasible to base posterior statistics on analytical likelihoods instead of MCMC frequencies, just as in the enumeration case from section 2. From `bms` results, the PMPs of 'top' models may be displayed with the command `pmp.bma`. For instance, one could display the PMPs of the best five models for `fls1` as follows:[21]

```
> pmp.bma(fls1)[1:5,]

            PMP (Exact) PMP (MCMC)
0046845800c 0.010914105    0.00810
0046844800c 0.009532020    0.00654
```

[21] `pmp.bma` returns a matrix with two columns and one row for each model. Consequently `pmp.bma(fls1)[1:5,]` extracts the first five rows and all columns of this matrix.

```
00474440008 0.006351611    0.00400
00064450008 0.004247053    0.00321
00464440008 0.003910244    0.00513
```

The numbers in the left-hand column represent analytical PMPs (`PMP (Exact)`) while the right-hand side displays MCMC-based PMPs (`PMP (MCMC)`). Both decline in roughly the same fashion, however sometimes the values for analytical PMPs differ considerably from the MCMC-based ones. This comes from the fact that MCMC-based PMPs derive from the number of iteration counts, while the 'exact' PMPs are calculated from comparing the analytical likelihoods of the best models – cf. equation (2).[22] In order to see the importance of all 'top models' with respect to the full model space, we can thus sum up their MCMC-based PMPs as follows:

```
> colSums(pmp.bma(fls1))
```

```
PMP (Exact)  PMP (MCMC)
    0.43707     0.43707
```

Both columns sum up to the same number and show that in total, the top 2,000 models account for ca. 44% of posterior model mass.[23] They should thus provide a rough approximation of posterior results that might or might not be better than the MCMC-based results. For this purpose, compare the best 5 covariates in terms of PIP by analytical and MCMC methods: `coef(fls1)` will display the results based on MCMC counts.

```
> coef(fls1)[1:5,]
```

```
              PIP     Post Mean      Post SD Cond.Pos.Sign Idx
GDP60     0.99909 -0.0161864596 0.0031207313             0  12
Confucian 0.98792  0.0566709814 0.0147170065             1  19
LifeExp   0.94225  0.0008497347 0.0003336649             1  11
EquipInv  0.92195  0.1601858531 0.0688902481             1  38
SubSahara 0.71283 -0.0110656860 0.0085394284             0   7
```

In contrast, the results based on analytical PMPs will be invoked with the `exact` argument:

```
> coef(fls1,exact=TRUE)[1:5,]
```

```
                PIP     Post Mean      Post SD Cond.Pos.Sign Idx
GDP60     1.0000000 -0.0162519441 0.0029430486             0  12
Confucian 0.9993987  0.0563416196 0.0125466528             1  19
LifeExp   0.9663382  0.0008473307 0.0002997708             1  11
EquipInv  0.9629370  0.1660656671 0.0596502222             1  38
SubSahara 0.7789498 -0.0118872756 0.0077913006             0   7
```

The ordering of covariates in terms of PIP as well as the coefficients are roughly similar. However, the PIPs under `exact = TRUE` are somewhat larger than with MCMC results. Closer inspection will also show that the analytical results downgrade the PIPs of the worst variables with respect to MCMC-PIPs. This stems from the fact that analytical results do not take into account the many 'bad' models that include 'worse' covariates and are factored into MCMC results.

Whether to prefer analytical or MCMC results is a matter of taste – however the literature prefers coefficients the analytical way: Fernández et al. (2001b), for instance, retain 5,000 models and report results based on them.

4.4 Combining Sampling Chains

The MCMC samplers described in section 4.1 need to discard the first batch of draws (the burn-ins) since they start out from some peculiar start model and may reach the altitudes of 'high' PMPs only after many iterations. Here, choosing an appropriate start model may help to speed up convergence. By default `bms` selects its start model as follows: from the full

[22] In the call to `topmodels.bma` on page 5, the PMPs under 'MCMC' and analytical ('exact') concepts were equal since 1) enumeration bases both 'top' model calculation and aggregate on-the-fly results on analytical PMPs and 2) because all possible models were retained in the object `att`.

[23] Note that this share was already provided in column `% Topmodels` resulting from the `summary` command on page 11.

model[24], all covariates with OLS t-statistics > 0.2 are kept and included in the start model. Other start models may be assigned outright or chosen according to a similar mechanism (cf. argument `start.value` in `help(bms)`).

However, in order to improve the sampler's convergence to the PMP distribution, one might actually start from several different start models. This could by particularly helpful if the models with high PMPs are clustered in distant 'regions'. For instance, one could set up the Fernández et al. (2001b) example above to get iteration chains from different starting values and combine them subsequently. Start e.g. a shorter chain from the null model (the model containing just an intercept), and use the 'reversible jump' MCMC sampler:

```
> fls2= bms(datafls, burn=20000, iter=50000, g="BRIC", mprior="uniform", mcmc="rev.jump", start.value
```

```
> summary(fls2)
```

```
Mean no. regressors              Draws             Burnins
          "10.5334"            "50000"             "20000"
               Time  No. models visited      Modelspace 2^K
     "4.59514 secs"            "10608"           "2.2e+12"
          % visited        % Topmodels            Corr PMP
          "4.8e-07"               "28"            "0.8483"
           No. Obs.        Model Prior             g-Prior
               "72"   "uniform / 20.5"              "BRIC"
    Shrinkage-Stats
        "Av=0.9994"
```

With 0.85, the correlation between analytical and MCMC PMPs is a bit smaller than the 0.9 from the `fls1` example in section 4.3. However, the results of this sampling run may be combined to yield more iterations and thus a better representation of the PMP distribution.

```
> fls_combi = c(fls1,fls2)
> summary(fls_combi)
```

```
Mean no. regressors              Draws             Burnins
          "10.4283"           "150000"             "70000"
               Time  No. models visited      Modelspace 2^K
    "12.39855 secs"            "36714"           "2.2e+12"
          % visited        % Topmodels            Corr PMP
          "1.7e-06"               "38"            "0.9337"
           No. Obs.        Model Prior             g-Prior
               "72"   "uniform / 20.5"              "BRIC"
    Shrinkage-Stats
        "Av=0.9994"
```

With 0.93, the PMP correlation from the combined results is broadly better than either of its two constituent chains `fls1` and `fls2`. Still, the PIPs and coefficients do not change much with respect to `fls1` – as evidenced e.g. by `plotComp(fls1, fls_combi, comp="Std Mean")`.

5 Alternative Formulations for Zellner's g Prior

5.1 Alternative Fixed g-Priors

Virtually all BMA applications rely on the presented framework with Zellner's g prior, and the bulk of them relies on specifying a fixed g. As mentioned in section 1.2, the value of g corresponds to the degree of prior uncertainty: A low g renders the prior coefficient distribution tight around a zero mean, while a large g implies large prior coefficient variance and thus decreases the importance of the coefficient prior.

While some popular default elicitation mechanisms for the g prior (we have seen UIP and BRIC) are quite popular, they are also subject to severe criticism. Some (e.g Fernández et al. 2001a) advocate a comparatively large g prior to minimize prior impact on the results, stay close to OLS coefficients, and represent the absolute lack of prior knowledge. Others (e.g. Ciccone and Jarociński 2010) demonstrate that such a large g may not be robust to noise innovations and risks over-fitting – in particular if the the noise component plays a substantial

[24] actually, a model with randomly drawn $\min(K, N-3)$ variables

role in the data. Again others (Eicher et al., 2009) advocate intermediate fixed values for the g priors or present alternative default specifications (Liang et al., 2008).[25]

In BMS, any fixed g-prior may be specified directly by submitting its value to the `bms` function argument `g`. For instance, compare the results for the Fernández et al. (2001b) setting when a more conservative prior such as $g = 5$ is employed (and far too few iterations are performed):

```
> fls_g5 = bms(datafls, burn=20000, iter=50000, g=5, mprior="uniform", user.int=F)
```

```
> coef(fls_g5)[1:5,]
```

```
              PIP     Post Mean      Post SD Cond.Pos.Sign Idx
GDP60     0.99102 -0.0137193578 0.0040562187    0.00000000  12
Confucian 0.93230  0.0458470035 0.0211175121    1.00000000  19
LifeExp   0.86986  0.0006646135 0.0004002169    0.99997701  11
EquipInv  0.81066  0.1013934402 0.0733896643    1.00000000  38
SubSahara 0.78428 -0.0110842168 0.0086569003    0.00066303   7
```

```
> summary(fls_g5)
```

```
Mean no. regressors              Draws             Burnins
          "20.1727"            "50000"            "20000"
               Time  No. models visited      Modelspace 2^K
    "4.999622 secs"            "44165"          "2.2e+12"
          % visited        % Topmodels           Corr PMP
            "2e-06"              "1.9"           "0.1354"
           No. Obs.        Model Prior            g-Prior
               "72"   "uniform / 20.5"          "numeric"
    Shrinkage-Stats
        "Av=0.8333"
```

The PIPs and coefficients for the best five covariates are comparable to the results from section 4.2 but considerably smaller, due to a tight shrinkage factor of $\frac{g}{1+g} = \frac{5}{6}$ (cf. section 1.2). More important, the posterior expected model size 20.2 exceeds that of `fls_combi` by a large amount. This stems from the less severe size penalty imposed by eliciting a small g. Finally, with 0.14, the correlation between analytical and MCMC PMPs means that the MCMC sampler has not at all converged yet. Feldkircher and Zeugner (2009) show that the smaller the g prior, the less concentrated is the PMP distribution, and therefore the harder it is for the MCMC sampler to provide a reasonable approximation to the actual PMP distribution. Hence the above command should actually be run with many more iterations in order to achieve meaningful results.

5.2 Model-specific g-Priors

Eliciting a fixed g-prior common to all models can be fraught with difficulties and unintended consequences. Several authors have therefore proposed to rely on model-specific priors (cf. Liang et al. 2008 for an overview), of which the following allow for closed-form solutions and are implemented in BMS:

- Empirical Bayes g – local (`EBL`): $g_\gamma = argmax_g\ p(y|M_\gamma, X, g)$. Authors such as George and Foster (2000) or Hansen and Yu (2001) advocate an 'Empirical Bayes' approach by using information contained in the data (y, X) to elicit g via maximum likelihood. This amounts to setting $g_\gamma = \max(0, F_\gamma^{OLS} - 1)$ where F_γ^{OLS} is the standard OLS F-statistic for model M_γ. Apart from obvious advantages discussed below, the `EBL` prior is not so popular since it involves 'peeking' at the data in prior formulation. Moreover, asymptotic 'consistency' of BMA is not guaranteed in this case.
- Hyper-g prior (`hyper`): Liang et al. (2008) propose putting a hyper-prior g; In order to arrive at closed-form solutions, they suggest a Beta prior on the shrinkage factor of the form $\frac{g}{1+g} \sim Beta\left(1, \frac{a}{2} - 1\right)$, where a is a parameter in the range $2 < a \leq 4$. Then, the prior expected value of the shrinkage factor is $E(\frac{g}{1+g}) = \frac{2}{a}$. Moreover, setting $a = 4$ corresponds to uniform prior distribution of $\frac{g}{1+g}$ over the interval $[0, 1]$, while $a \to 2$

[25] Note however, that g should in general be monotonously increasing in N: Fernández et al. (2001a) prove that this sufficient for 'consistency', i.e. if there is one single linear model as in equation (1), than its PMP asymptotically reaches 100% as sample size $N \to \infty$.

concentrates prior mass very close to unity (thus corresponding to $g \to \infty$). (`bms` allows to set a via the argument `g="hyper=x"`, where `x` denotes the a parameter.) The virtue of the hyper-prior is that it allows for prior assumptions about g, but relies on Bayesian updating to adjust it. This limits the risk of unintended consequences on the posterior results, while retaining the theoretical advantages of a fixed g. Therefore Feldkircher and Zeugner (2009) prefer the use of hyper-g over other available g-prior frameworks.

Both model-specific g priors adapt to the data: The better the signal-to-noise ratio, the closer the (expected) posterior shrinkage factor will be to one, and vice versa. Therefore average statistics on the shrinkage factor offer the interpretation as a 'goodness-of-fit' indicator (Feldkircher and Zeugner (2009) show that both EBL and hyper-g can be interpreted in terms of the OLS F-statistic).

Consider, for instance, the Fernández et al. (2001b) example under an Empirical Bayes prior:

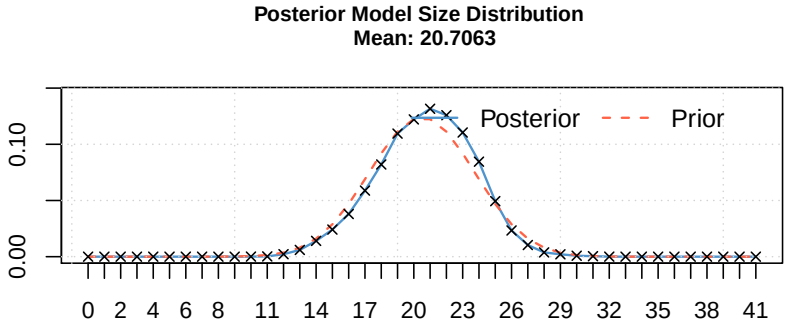
```
> fls_ebl = bms(datafls, burn=20000, iter=50000, g="EBL", mprior="uniform", nmodel=1000, user.int=F)
```

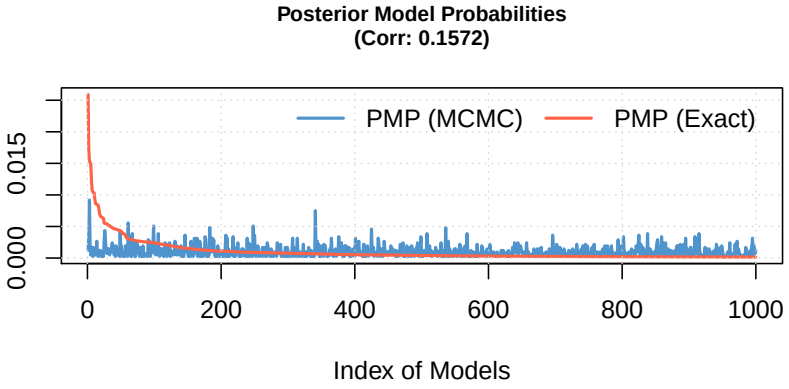
```
> summary(fls_ebl)
```

```
Mean no. regressors               Draws             Burnins
          "20.7063"             "50000"             "20000"
               Time  No. models visited      Modelspace 2^K
    "4.854738 secs"             "28381"           "2.2e+12"
          % visited         % Topmodels            Corr PMP
          "1.3e-06"               "8.3"            "0.1572"
           No. Obs.         Model Prior             g-Prior
               "72"    "uniform / 20.5"               "EBL"
    Shrinkage-Stats
        "Av=0.9606"
```

The result `Shrinkage-Stats` reports a posterior average EBL shrinkage factor of 0.961, which corresponds to a shrinkage factor $\frac{g}{1+g}$ under $g \approx 24$. Consequently, posterior model size is considerably larger than under `fls_combi`, and the sampler has had a harder time to converge, as evidenced in a quite low `Corr PMP`. Both conclusions can also be drawn from performing the `plot(fls_ebl)` command that combines the `plotModelsize` and `plotConv` functions:

```
> plot(fls_ebl)
```





The upper chart shows that posterior model size distribution remains very close to the model prior; The lower part displays the discord between iteration count frequencies and analytical PMPs.

The above results show that using a flexible and model-specific prior on Fernández et al. (2001b) data results in rather small posterior estimates of $\frac{g}{1+g}$, thus indicating that the `g="BRIC"` prior used in `fls_combi` may be set too far from zero. This interacts with the uniform model prior to concentrate posterior model mass on quite large models. However, imposing a uniform model prior means to expect a model size of $K/2 = 20.5$, which may seem overblown. Instead, try to impose smaller model size through a corresponding model prior – e.g. impose a prior model size of 7 as in Sala-i-Martin et al. (2004). This can be combined with a hyper-g prior, where the argument `g="hyper=UIP"` imposes an a parameter such that the prior expected value of g corresponds to the unit information prior ($g = N$).[26]

```
> fls_hyper = bms(datafls, burn=20000, iter=50000, g="hyper=UIP", mprior="random", mprior.size=7, nmo
```

```
> summary(fls_hyper)
```

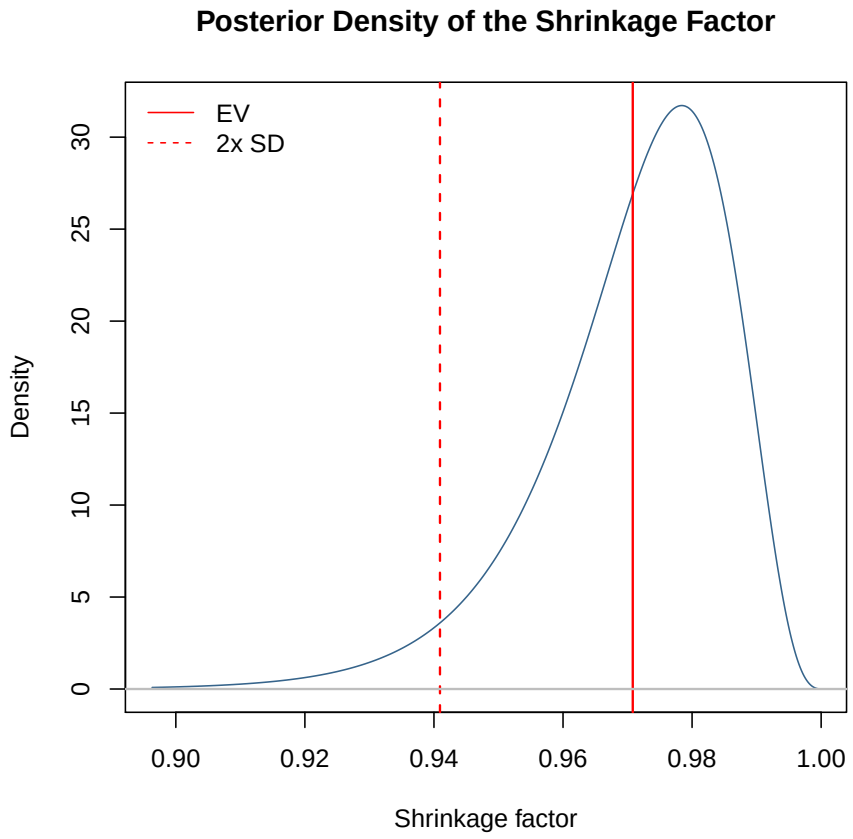
```
     Mean no. regressors                   Draws                 Burnins
                "15.1598"                 "50000"                 "20000"
                    Time       No. models visited           Modelspace 2^K
          "4.777031 secs"                 "23242"               "2.2e+12"
               % visited             % Topmodels                Corr PMP
                "1.1e-06"                    "13"                  "0.1658"
                 No. Obs.             Model Prior                 g-Prior
                     "72"            "random / 7"      "hyper (a=2.02778)"
           Shrinkage-Stats
"Av=0.9607, Stdev=0.018"
```

From `Shrinkage-Stats`, posterior expected shrinkage is 0.961, with rather tight standard deviation bounds. Similar to the EBL case before, the data thus indicates that shrinkage

[26] This is the default hyper-g prior and may therefore be as well obtained with `g="hyper "`.

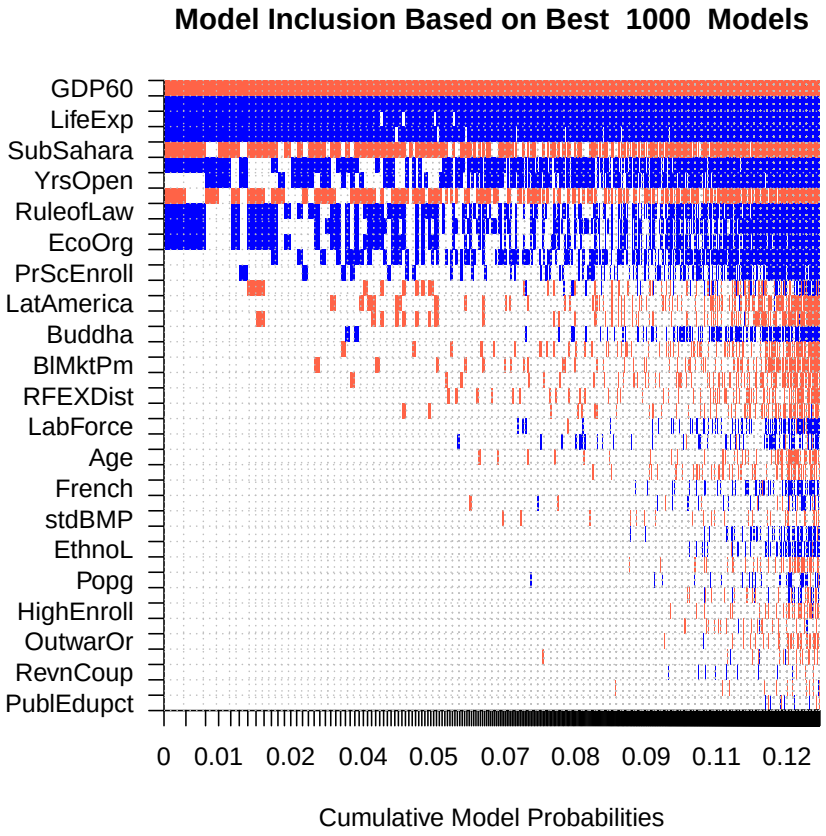
should be rather small (corresponding to a fixed g of $g \approx 24$) and not vary too much from its expected value. Since the hyper-g prior induces a proper posterior distribution for the shrinkage factor, it might be helpful to plot its density with the command below. The chart confirms that posterior shrinkage is tightly concentrated above 0.94.

```
> gdensity(fls_hyper)
```



While the hyper-g prior had an effect similar to the EBL case `fls_ebl`, the model prior now employed leaves the data more leeway to adjust posterior model size. The results depart from the expected prior model size and point to an intermediate size of ca. 15. The focus on smaller models is evidenced by charting the best 1,000 models with the `image` command:

```
> image(fls_hyper)
```

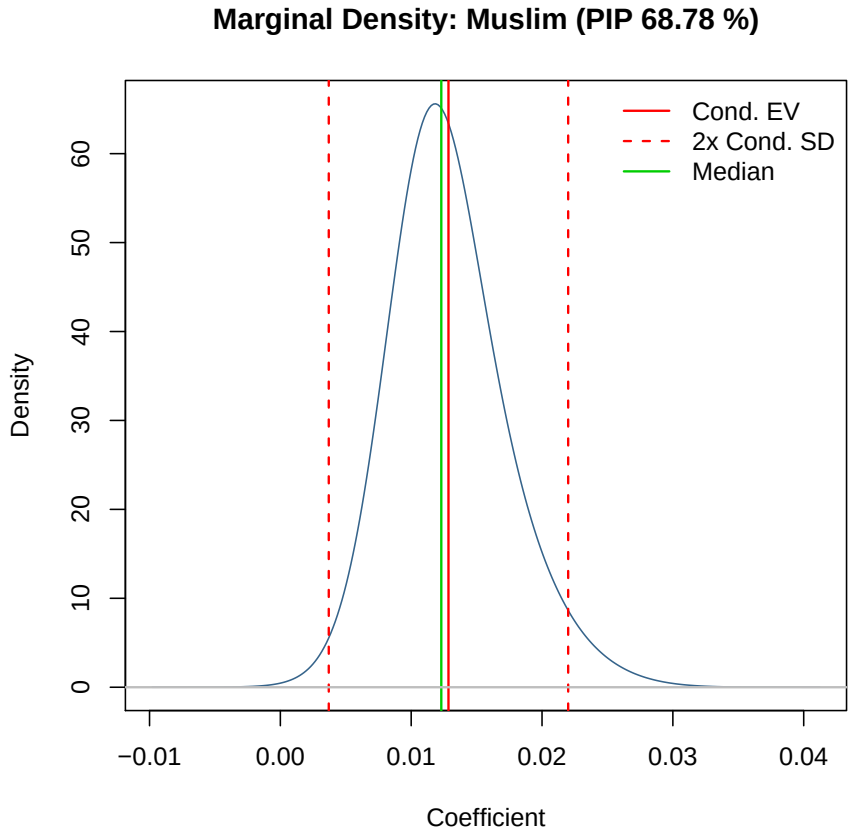


In a broad sense, the coefficient results correspond to those of `fls_combi`, at least in expected values. However, the results from `fls_hyper` were obtained under more sophisticated priors that were specifically designed to avoid unintended influence from prior parameters: By construction, the large shrinkage factor under `fls_combi` induced a quite small posterior model size of 10.4 and concentrated posterior mass tightly on the best models encountered (they make up 38% of the entire model mass). In contrast, the hyper-g prior employed for `fls_hyper` indicated a rather low posterior shrinkage factor and consequently resulted in higher posterior model size (15.2) and less model mass concentration (13%).

5.3 Posterior Coefficient Densities

In order to compare more than just coefficient expected values, it is advisable to consult the entire posterior distribution of coefficients. For instance, consult the posterior density of the coefficient for `Muslim`, a variable with a PIP of 66.3%: The `density` method produces marginal densities of posterior coefficient distributions and plots them, where the argument `reg` specifies the variable to be analyzed.

```
> density(fls_combi,reg="Muslim")
```



We see that the coefficient is neatly above zero, but somewhat skewed. The integral of this density will add up to 0.688, conforming to the analytical PIP of `Muslim`. The vertical bars correspond to the analytical coefficient conditional on inclusion from `fls_combi` as in

```
> coef(fls_combi,exact=T,condi.coef=T)["Muslim",]
```

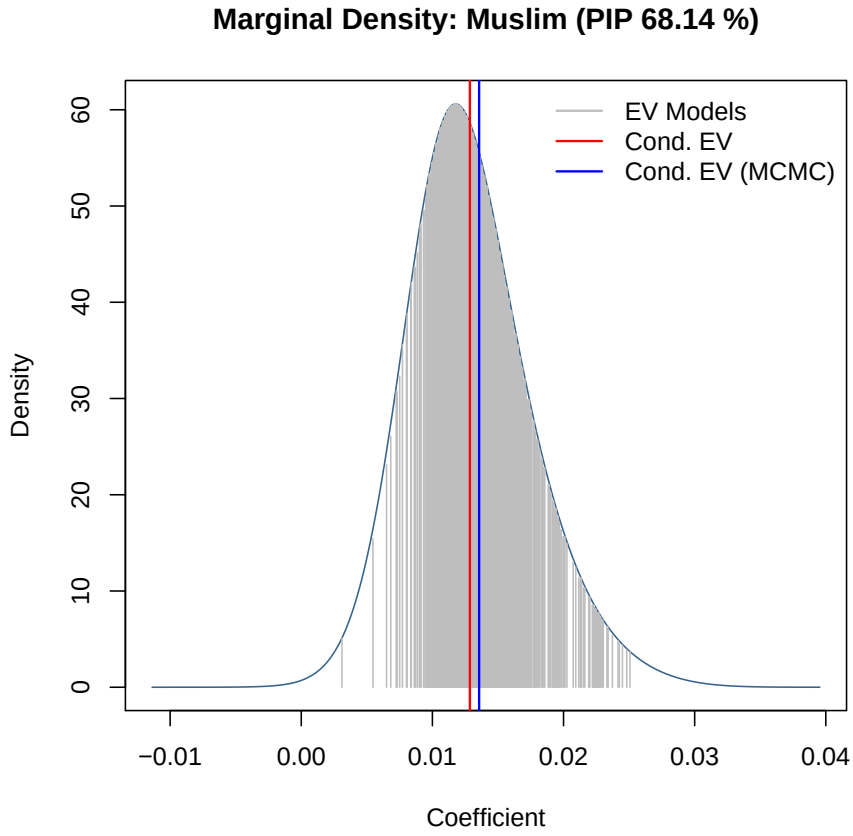
```
          PIP     Post Mean       Post SD Cond.Pos.Sign          Idx
0.687790245   0.012849494   0.004575044   1.000000000  23.000000000
```

Note that the posterior marginal density is actually a model-weighted mixture of posterior densities for each model and can this be calculated only for the top models contained in `fls_combi` (here 2041).

Now let us compare this density with the results under the hyper-g prior:[27]

```
> dmuslim=density(fls_hyper,reg="Muslim",addons="Eebl")
```

[27] Since for the hyper-g prior, the marginal posterior coefficient distribution derive from quite complicated expressions, executing this command could take a few seconds.



Here, the `addons` argument assigns the vertical bars to be drawn: the expected conditional coefficient from MCMC (`E`) results should be indicated in contrast to the expected coefficient based on analytical PMPs (`e`). In addition the expected coefficients under the individual models are plotted (`b`) and a legend is included (`l`). The density seems more symmetric than before and the analytical results seem to be only just smaller than what could be expected from MCMC results.

Nonetheless, even though `fls_hyper` and `fls_combi` applied very different g and model priors, the results for the `Muslim` covariate are broadly similar: It is unanimously positive, with a conditional expected value somewhat above 0.01. In fact 95% of the posterior coefficient mass seems to be concentrated between 0.004 and 0.024:

```
> quantile(dmuslim, c(0.025, 0.975))

      2.5%      97.5%
0.00433047 0.02352947
```

6 Predictive Densities

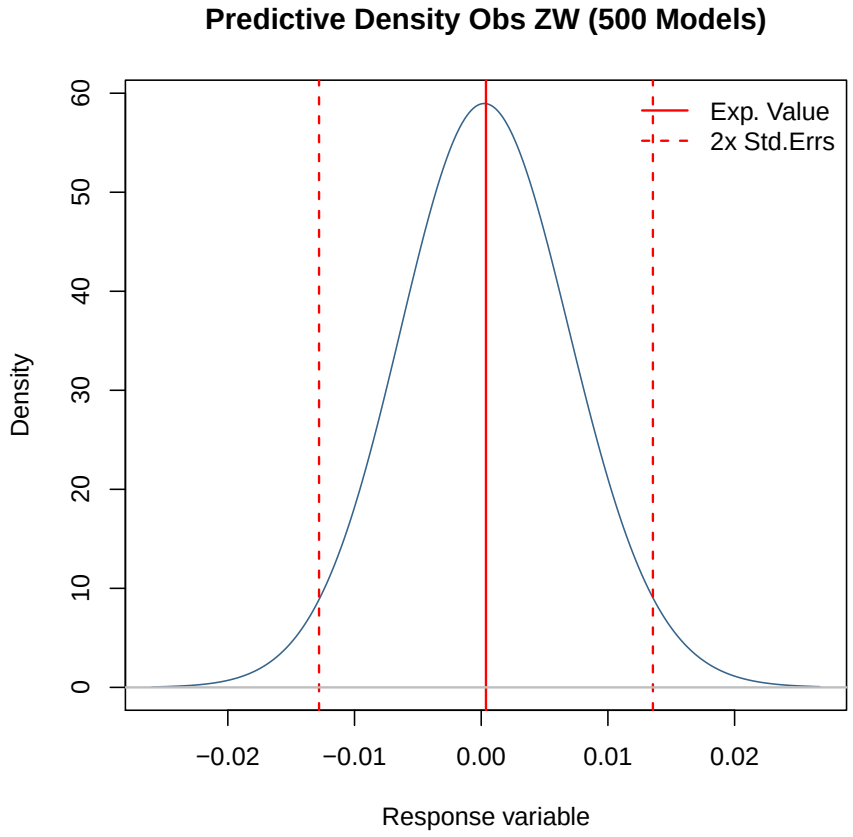
Of course, BMA lends itself not only to inference, but also to prediction. The employed 'Bayesian Regression' models naturally give rise to predictive densities, whose mixture yields the BMA predictive density – a procedure very similar to the coefficient densities explored in the previous section.

Let us, for instance, use the information from the first 70 countries contained in `datafls` to forecast economic growth for the latter two, namely Zambia (identifier `ZM`) and Zimbabwe (identifier `ZW`). We then can use the function `pred.density` on the BMA object `fcstbma` to form predictions based on the explanatory variables for Zambia and Zimbabwe (which are in `datafls[71:72,]`).

```
> fcstbma= bms(datafls[1:70,], mprior="uniform", burn=20000, iter=50000, user.int=FALSE)
> pdens = pred.density(fcstbma, newdata=datafls[71:72,])
```

The resulting object `pdens` holds the distribution of the forecast for the two countries, conditional on what we know from other countries, and the explanatory data from Zambia and Zimbabwe. The expected value of this growth forecast is very similar to the classical point forecast and can be accessed with `pdens$fit`.[28] Likewise the standard deviations of the predictive distribution correspond to classical standard errors and are returned by `pdens$std.err`. But the predictive density for the growth in e.g. Zimbabwe might be as well visualized with the following command:[29]

```
> plot(pdens, 2)
```



Here, we see that conditional on Zimbabwe's explanatory data, we expect growth to be concentrated around 0. And the actual value in `datafls[72,1]` with 0.0046 is not too far off from that prediction. A closer look at both our densities with the function `quantile` shows that for Zimbabwe, any growth rate between -0.01 and 0.01 is quite likely.

```
> quantile(pdens, c(0.05, 0.95))
```

```
              5%         95%
ZM  0.003507328 0.02793047
ZW -0.010802109 0.01171919
```

For Zambia (`ZM`), though, the explanatory variables suggest that positive economic growth should be expected. But over our evaluation period, Zambian growth has been even worse than in Zimbabwe (with -0.01 as from `datafls["ZM",1]`).[30] Under the predictive density for Zambia, this actual outcome seems quite unlikely.

To compare the BMA prediction performs with actual outcomes, we could look e.g. at the forecast error `pdens$fit - datafls[71:72,1]`. However it would be better to take standard errors into account, and even better follow the 'Bayesian way' and evaluate the predictive density of the outcomes as follows:

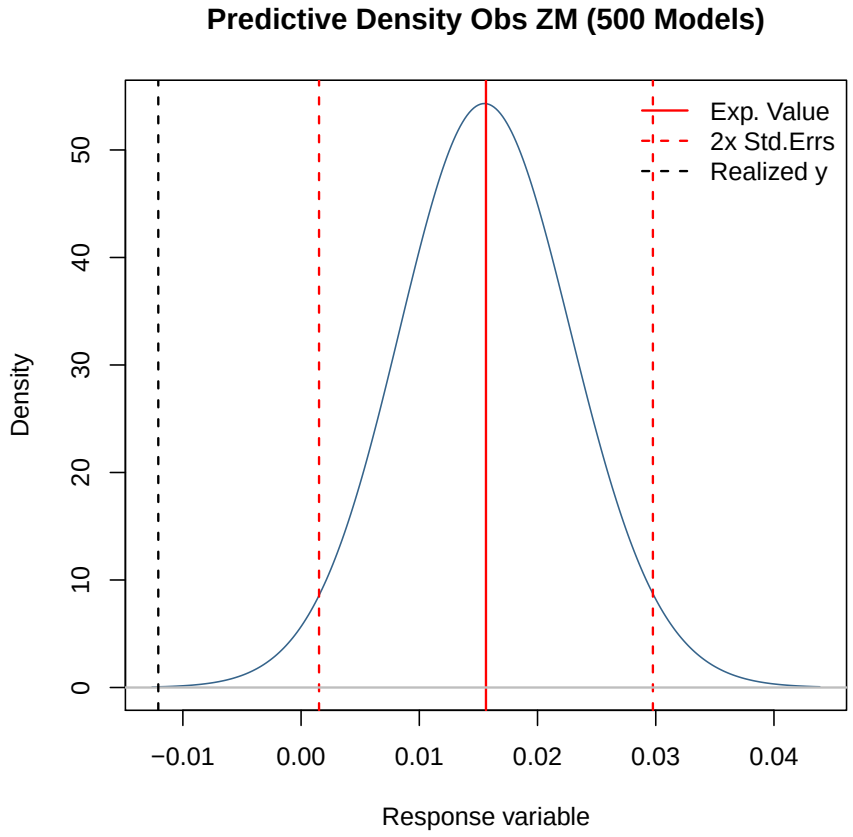
[28] Note that this is equivalent to `predict(fcstbma, datafls[71:72, ])` .
[29] Here, 2 means to plot for the second forecasted observation, in this case `ZW`, the 72-th row of `datafls`.
[30] Note that since `ZM` is the rowname of the 71-st row of `datafls`, this is equivalent to calling `datafls[71, ]`.

```
> pdens$dyf(datafls[71:72,1])
```

```
[1]  0.06422567 47.94948244
```

The density for Zimbabwe is quite high (similar to the mode of predictive density as seen in the chart above), whereas the one for Zambia is quite low. In order to visualize how bad the forecast for Zambia was, compare a plot of predictive density to the actual outcome, which is situated far to the left.

```
> plot(pdens, "ZM", realized.y=datafls["ZM",1])
```



The results for Zambia suggest either that it is an outlier or that our forecast model might not perform that well. One could try out other prior settings or data, and compare the differing models in their joint predictions for Zambia and Zimbabwe (or even more countries). A standard approach to evaluate the goodness of forecasts would be to e.g. look at root mean squared errors. However Bayesians (as e.g Fernández et al. 2001a) often prefer to look at densities of the outcome variables and combine them in a 'log-predictive score' (LPS). It is defined as follows, where $p(y_i^f|X, y, X_i^f)$ denotes predictive density for y_i^f (Zambian growth) based on the model information (y, X) (the first 70 countries) and the explanatory variables for the forecast observation (Zambian investment, schooling, etc.).

$$-\sum_i \log(p(y_i^f|X, y, X_i^f))$$

The log-predictive score can be accessed with `lps.bma`.

```
> lps.bma(pdens, datafls[71:72,1])
```

```
[1] -0.5623978
```

Note however, that the LPS is only meaningful when comparing different forecast settings.

References

Ciccone, A. and Jarociński, M. (2010). Determinants of Economic Growth: Will Data Tell? *American Economic Journal: Macroeconomics*, forthcoming.

Eicher, T., Papageorgiou, C., and Raftery, A. (2009). Determining growth determinants: default priors and predictive performance in Bayesian model averaging. *Journal of Applied Econometrics, forthcoming*.

Feldkircher, M. and Zeugner, S. (2009). Benchmark Priors Revisited: On Adaptive Shrinkage and the Supermodel Effect in Bayesian Model Averaging. *IMF Working Paper*, WP/09/202.

Feldkircher, M. and Zeugner, S. (2015). Bayesian Model Averaging Employing Fixed and Flexible Priors: The BMS Package for R. *Journal of Statistical Software*, 68.

Fernández, C., Ley, E., and Steel, M. F. (2001a). Benchmark Priors for Bayesian Model Averaging. *Journal of Econometrics*, 100:381–427.

Fernández, C., Ley, E., and Steel, M. F. (2001b). Model Uncertainty in Cross-Country Growth Regressions. *Journal of Applied Econometrics*, 16:563–576.

George, E. and Foster, D. (2000). Calibration and empirical Bayes variable selection. *Biometrika*, 87(4):731–747.

Hansen, M. and Yu, B. (2001). Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96(454):746–774.

Hoeting, J. A., Madigan, D., Raftery, A. E., and Volinsky, C. T. (1999). Bayesian Model Averaging: A Tutorial. *Statistical Science*, 14, No. 4:382–417.

Ley, E. and Steel, M. F. (2009). On the Effect of Prior Assumptions in Bayesian Model Averaging with Applications to Growth Regressions. *Journal of Applied Econometrics*, 24:4:651–674.

Liang, F., Paulo, R., Molina, G., Clyde, M. A., and Berger, J. O. (2008). Mixtures of g Priors for Bayesian Variable Selection. *Journal of the American Statistical Association*, 103:410–423.

Madigan, D. and York, J. (1995). Bayesian graphical models for discrete data. *International Statistical Review*, 63.:215–232.

Sala-i-Martin, X., Doppelhofer, G., and Miller, R. I. (2004). Determinants of Long-Term Growth: A Bayesian Averaging of Classical Estimates (BACE) Approach. *American Economic Review*, 94:813–835.

A Appendix

A.1 Available Model Priors – Synopsis

The following provides an overview over the model priors available in `bms`. Default is `mprior="random"`. For details and examples on built-in priors, consult `help(bms)`. For defining different, custom g-priors, consult `help(gprior)` or `http://bms.zeugner.eu/custompriors.php`.

Uniform Model Prior

- *Argument*: `mprior="uniform"`
- *Parameter*: none
- *Concept*: $p(M_\gamma) \propto 1$
- *Reference*: none

Binomial Model Prior

- *Argument*: `mprior="fixed"`
- *Parameter* (`mprior.size`): prior model size $\bar{m}$ (scalar); Default is $\bar{m} = K/2$
- *Concept*: $p(M_\gamma) \propto \left(\frac{\bar{m}}{K}\right)^{k_\gamma} \left(1 - \frac{\bar{m}}{K}\right)^{K-k_\gamma}$
- *Reference*: Sala-i-Martin et al. (2004)

Beta-Binomial Model Prior

- *Argument*: `mprior="random"`
- *Parameter* (`mprior.size`): prior model size $\bar{m}$ (scalar)
- *Concept*: $p(M_\gamma) \propto \Gamma(1 + k_\gamma)\Gamma(\frac{K-m}{m} + K - k_\gamma)$; Default is $\bar{m} = K/2$
- *Reference*: Ley and Steel (2009)

Custom Prior Inclusion Probabilities

- *Argument*: `mprior="pip"`
- *Parameter* (`mprior.size`): A vector of size K, detailing K prior inclusion probabilities π_i: $0 < \pi < 1 \; \forall i$
- *Concept*: $p(M_\gamma) \propto \prod_{i \in \gamma} \pi_i \; \prod_{j \notin \gamma} (1 - \pi_j)$
- *Reference*: none

Custom Model Size Prior

- *Argument*: `mprior="customk"`
- *Parameter* (`mprior.size`): A vector of size $K + 1$, detailing prior θ_j for 0 to K size models: any real >0 admissible
- *Concept*: $p(M_\gamma) \propto \theta_{k_\gamma}$
- *Reference*: none

A.2 Available g-Priors – Synopsis

The following provides an overview over the g-priors available in `bms`. Default is `g="UIP"`. For implementation details and examples, consult `help(bms)`. For defining different, custom g-priors, consult `help(gprior)` or `http://bms.zeugner.eu/custompriors.php`.

Fixed g

- *Argument*: `g=x` where `x` is a positive real scalar;
- *Concept*: Fixed g common to all models
- *Reference*: Fernández et al. (2001a)
- *Sub-options*: Unit information prior `g="UIP"` sets $g = N$; `g="BRIC"` sets $g = \max(N, K^2)$, a combination of BIC and RIC. (Note these two options guarantee asymptotic consistency.) Other options include `g="RIC"` for $g = K^2$ and `g="HQ"'` for the Hannan-Quinn setting $g = \log(N)^3$.

Empirical Bayes (Local) g

- *Argument*: `g="EBL"`
- *Concept*: Model-specific g_γ estimated via maximum likelihood: amounts to $g_\gamma = \max(0, F_\gamma - 1)$, where $F_\gamma \equiv \frac{R^2_\gamma (N-1-k_\gamma)}{(1-R^2_\gamma)k_\gamma}$ and R^2_γ is the OLS R-squared of model M_γ.
- *Reference*: George and Foster (2000); Liang et al. (2008)
- *Sub-options*: none

Hyper-g prior

- *Argument*: `g="hyper"`
- *Concept*: A Beta prior on the shrinkage factor with $p(\frac{g}{1+g}) = B(1, \frac{a}{2} - 1)$. Parameter a ($2 < a \leq 4$) represents prior beliefs: $a = 4$ implies prior shrinkage to be uniformly distributed over $[0, 1]$, $a \to 2$ concentrates mass close to unity. Note that prior expected value of the shrinkage facor is $E(\frac{g}{1+g}) = \frac{2}{a}$.
- *Reference*: Liang et al. (2008); Feldkircher and Zeugner (2009)
- *Sub-options*: `g="hyper=x"` with `x` defining the parameter a (e.g. `g="hyper=3"` sets $a = 3$). `g="hyper"` resp. `g="hyper=UIP"` sets the prior expected shrinkage factor equivalent to the UIP prior $E(\frac{g}{1+g}) = \frac{N}{1+N}$; `g="hyper=BRIC"` sets the prior expected shrinkage factor equivalent to the BRIC prior. Note that the latter two options guarantee asymptotic consistency.

A.3 'Bayesian Regression' with Zellner's g – Bayesian Model Selection

The linear model presented in section 1.2 using Zellner's g prior is implemented under the function `zlm`. For instance, we might consider the attitude data from section 2 and estimate just the full model containing all 6 variables. For this purpose, first load the built-in data set with the command

```
> data(attitude)
```

The full model is obtained by applying the function `zlm` on the data set and storing the estimation into `att_full`. Zellner's g prior is estimated by the argument `g` just in the same way as in section 5.[31]

```
> att_full = zlm(attitude,g="UIP")
```

The results can then be displayed by using e.g. the `summary` method.

```
> summary(att_full)
```

```
Coefficients
               Exp.Val.   St.Dev.
(Intercept) 12.52405242        NA
complaints   0.59340736 0.1524868
privileges  -0.07069369 0.1285614
learning     0.30999882 0.1596262
raises       0.07909561 0.2097886
critical     0.03714334 0.1392373
advance     -0.21005485 0.1688040

 Log Marginal Likelihood:
-113.7063
 g-Prior: UIP
Shrinkage Factor: 0.968
```

The results are very similar to those resulting from OLS (which can be obtained via `summary(lm(attitude))`). The less conservative, i.e. the larger g becomes, the closer the results get to OLS. But remember that the full model was not the best model from the BMA application in section 2. In order to extract the best encountered model, use the function `as.zlm` to extract this single model for further analysis (with the argument `model` specifying the rank-order of the model to be extracted). The following command reads the best model from the BMA results in section into the variable `att_best`.

[31] Likewise, most methods applicable to `bms`, such as `density`, `predict` or `coef`, work analogously for `zlm`.

```
> att_best = as.zlm(att,model=1)
> summary(att_best)

Coefficients
              Exp.Val.   St.Dev.
(Intercept) 15.9975134        NA
complaints   0.7302676 0.1010205

 Log Marginal Likelihood:
-107.4047
 g-Prior: UIP
Shrinkage Factor: 0.968
```

As suspected, the best model according to BMA is the on including only `complaints` and the intercept, as it has the highest log-marginal likelihood (`logLik(att_best)`). In such a way, the command `as.zlm` can be combined with `bms` for 'Bayesian Model Selection', i.e. using the model prior and posterior framework to focus on teh model with highest posterior mass. Via the utility `model.frame`, this best model can be straightforwardly converted into a standard OLS model:

```
> att_bestlm = lm(model.frame(as.zlm(att)))
> summary(att_bestlm)

Call:
lm(formula = model.frame(as.zlm(att)))

Residuals:
     Min       1Q   Median       3Q      Max
-12.8799  -5.9905   0.1783   6.2978   9.6294

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 14.37632    6.61999   2.172   0.0385 *
complaints   0.75461    0.09753   7.737 1.99e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.993 on 28 degrees of freedom
Multiple R-squared:  0.6813,        Adjusted R-squared:  0.6699
F-statistic: 59.86 on 1 and 28 DF,  p-value: 1.988e-08
```

A.4 BMA when Keeping a Fixed Set of Regressors

While BMA should usually compare as many models as possible, some considerations might dictate the restriction to a subspace of the 2^K models. For complicated settings one might employ a customly designed model prior (cf. section A.1). The by far most common setting, though, is to keep some regressors fixed in the model setting, and apply Bayesian Model uncertainty only to a subset of regressors.

Suppose, for instance, that prior research tells us that any meaningful model for `attitude` (as in section 2) must include the variables `complaints` and `learning`. The only question is whether the additional four variables matter (which reduces the potential model space to $2^4 = 16$). We thus sample over these models while keeping `complaints` and `learning` as fixed regressors:

```
> att_learn = bms(attitude,mprior="uniform", fixed.reg=c("complaints", "learning") )

                PIP    Post Mean    Post SD Cond.Pos.Sign Idx
complaints 1.0000000  0.622480469 0.12718297     1.0000000   1
learning   1.0000000  0.237607970 0.15086061     1.0000000   3
advance    0.2878040 -0.053972968 0.11744640     0.0000000   6
privileges 0.1913388 -0.017789715 0.06764219     0.0000000   2
raises     0.1583504  0.001767835 0.07951209     0.3080239   4
critical   0.1550556  0.002642777 0.05409412     1.0000000   5
```

```
Mean no. regressors              Draws             Burnins
          "2.7925"               "16"                "0"
              Time  No. models visited      Modelspace 2^K
 "0.003888845 secs"               "16"               "64"
         % visited        % Topmodels            Corr PMP
              "25"              "100"               "NA"
          No. Obs.        Model Prior             g-Prior
              "30"      "uniform / 4"               "UIP"
    Shrinkage-Stats
        "Av=0.9677"

Time difference of 0.003888845 secs
```

The results show that the PIP and the coefficients for the remaining variables increase a bit compared to `att`. The higher PIPs are related to the fact that the posterior model size (as in `sum(coef(att_learn)[,1])`) is quite larger as under `att`. This follows naturally from our model prior: putting a uniform prior on all models between parameter size 2 (the base model) and 6 (the full model) implies a prior expected model size of 4 for `att_learn` instead of the 3 for `att`.[32] So to achieve comparable results, one needs to take the number of fixed regressors into account when setting the model prior parameter `mprior.size`. Consider another example:

Suppose we would like to sample the importance and coefficients for the cultural dummies in the dataset `datafls`, conditional on information from the remaining 'hard' variables. This implies keeping 27 fixed regressors, while sampling over the 14 cultural dummies. Since model uncertainty thus applies only to $2^{14} = 16,384$ models, we resort to full enumeration of the model space.

```
> fls_culture = bms(datafls,fixed.reg=c(1,8:16,24,26:41), mprior="random", mprior.size=28, mcmc="enu
```

Here, the vector `c(1,8:16,24,26:41)` denotes the indices of the regressors in `datafls` to be kept fixed.[33] Moreover, we use the beta-binomial ('random') model prior. The prior model size of 30 embodies our prior expectation that on average 1 out of the 14 cultural dummies should be included in the true model. As we only care about those 14 variables, let us just display the results for the 14 variables with the least PIP:

```
> coef(fls_culture)[28:41, ]
```

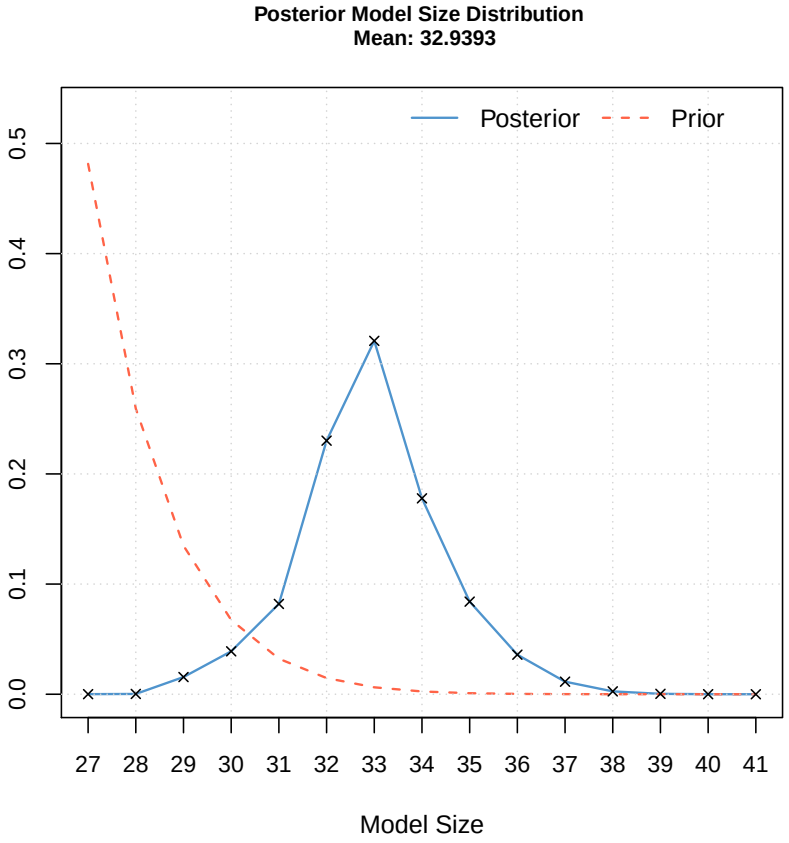
	PIP	Post Mean	Post SD	Cond.Pos.Sign	Idx
Confucian	0.99950018	6.796387e-02	0.0130198193	1.00000000	19
Hindu	0.94793751	-7.519094e-02	0.0270173174	0.00000000	21
SubSahara	0.84127891	-1.584077e-02	0.0091307736	0.00000000	7
EthnoL	0.70497895	9.238430e-03	0.0070484067	0.99999675	20
Protestants	0.57157577	-6.160916e-03	0.0061672877	0.00001431	25
Muslim	0.53068726	7.908574e-03	0.0086009027	0.99999949	23
LatAmerica	0.52063035	-6.538488e-03	0.0074843453	0.00469905	6
Spanish	0.21738032	2.105990e-03	0.0047683535	0.98325917	2
French	0.17512267	1.065459e-03	0.0027682773	0.99999954	3
Buddha	0.11583307	9.647944e-04	0.0033462133	0.99999992	17
Brit	0.10056773	4.095469e-04	0.0017468022	0.94203569	4
Catholic	0.09790780	-1.072004e-05	0.0019274111	0.45246829	18
WarDummy	0.07478332	-1.578379e-04	0.0007599415	0.00123399	5
Jewish	0.04114852	-5.614758e-05	0.0018626191	0.24834675	22

As before, we find that `Confucian` (with positive sign) as well as `Hindu` and `SubSahara` (negative signs) have the most important impact conditional on 'hard' information. Moreover, the data seems to attribute more importance to cultural dummies as we expectd with our model prior: Comparing prior and posterior model size with the following command shows how much importance is attributed to the dummies.

```
> plotModelsize(fls_culture, ksubset=27:41)
```

[32] The command `att_learn2 = bms(attitude, mprior='fixed', mprior.size=3, fixed.reg=c('complaints', 'learning') )` produces coefficients that are much more similar to `att`.

[33] Here, indices start from the first regressor, i.e. they do not take the dependent variable into account. The fixed data used above therefore corresponds to `datafls[ ,c(1,8:16,24,26:41) + 1]`.



Expected posterior model size is close to 33, which means that 6 of the cultural dummies should actually be included in a 'true' model.