Bayesian Model Averaging (BMA) with uncertain Spatial Effects

A Tutorial

Martin Feldkircher

This version: October 2010

This file illustrates the computer code to use spatial filtering in the context of Bayesian Model Averaging (BMA). For more details and in case you use the code please cite Crespo Cuaresma and Feldkircher (2010). To get the code started you need to install the R-packages `spdep`, `BMS` (Version $\geq$ 0.2.5) and the add on package `spatBMS`. For an introduction to `BMS` see `http://bms.zeugner.eu` and the tutorials therein. The following file has been tested with `R.2.11`.

1 A Primer to Spatial Filtering and BMA

Consider a cross-sectional regression of the following form:

$$y = \alpha\iota_N + \rho\mathbf{W}y + \mathbf{X}_k\vec{\chi}_k + \sigma\varepsilon \tag{1}$$

where y is an N-dimensional column vector of the dependent variable, α is the intercept term, ι_N is an N-dimensional column vector of ones, $\mathbf{X}_k = (\mathbf{x}_1 \dots \mathbf{x}_k)$ is a matrix whose columns are stacked data for k explanatory variables and $\vec{\chi}_k = (\chi_1 \dots \chi_k)'$ is the k-dimensional parameter vector corresponding to the variables in $\mathbf{X}_k$. The spatial autocorrelation structure is specified via a spatial weight matrix $\mathbf{W}$. The coefficient ρ attached to $\mathbf{W}$ reflects the degree of spatial autocorrelation. Equation (1) is in the vein of a parametric spatial model where the spatial parameter ρ is often interpreted as a spillover parameter.
In this setting, on top of the uncertainty regarding the choice of explanatory variables an extra degree of uncertainty arises: we do *not* know the actual nature of the spatial interactions which we model through the spatial autoregressive term in equation (1), that is, if

we conduct inference conditional on $\mathbf{W}$. Spatial autocorrelation will be observable whenever the phenomenon under study is a spatial process or omitted variables cause spatial variation in the residuals (Tiefelsdorf and Griffith, 2007). Note that both arguments typically apply to economic cross-section data, where economic units interact with each other and omitted variables decrease the level of confidence in econometric analysis. Since inference from the SAR model is conditional on the weight matrix $\mathbf{W}$, which has to be exogenously specified, explicitly accounting for this source of model uncertainty is a natural generalization to uncertainty in the nature of $\mathbf{X}_k$ in the framework of BMA. In most applications there is little theoretical guidance on which structure to put on the weight matrix rendering its specification a serious challenge.

1.1 Spatial Filtering

The spatial filtering literature seeks to remove residual spatial autocorrelation patterns prior to estimation and is in principle not interested in directly estimating ρ in (1). The approach put forward by Getis and Griffith (2002) and Tiefelsdorf and Griffith (2007), is based on an eigenvector decomposition of a transformed $\mathbf{W}$ matrix, where the transformation depends on the underlying spatial model. The eigenvectors $\{e_i\}$ are included as additional explanatory variables and the regression equation (1) becomes:

$$y = \alpha\iota_N + \sum_{i=1}^{E} \gamma_i \vec{e_i} + \mathbf{X}_k\vec{\chi}_k + \sigma\varepsilon, \tag{2}$$

where each eigenvector $\vec{e_i}$ spans one of the spatial dimensions. By introducing the eigenvectors into the regression, we explicitly take care of (remaining) spatial patterns in the residuals. Furthermore spatial commonalities among the covariates in $\mathbf{X}_k$ are conditioned out. This reduces the degree of multicollinearity and further separates spatial effects from the 'intrinsic' impact the employed regressors exert on the dependent variable.

1.2 BMA with uncertain spatial effects

From a Bayesian perspective, the problem of obtaining estimates of the parameter associated with a covariate under uncertainty in both the nature of $\mathbf{W}$ and $\mathbf{X}_k$ can be handled in a straightforward manner. Let us assume that we are interested in a particular regression coefficient, β. Denote the set of potential models by $\mathcal{M} = \{M_1^1, M_2^1, \ldots, M_{2^K}^1, \ldots M_1^2, \ldots, M_{2^K}^2, \ldots, M_1^Z, \ldots, M_{2^K}^Z\}$, where K stands for the number of potential explanatory variables and Z the number of candidate spatial weighting matrices $\mathbf{W}_z$, $z = 1, \ldots, Z$ each with associated set of eigenvectors E_z. The cardinality of $\mathcal{M}$ is therefore $2^K \times Z$. A particular model, say M_k^z, is characterized by its parameter vector $\theta_k^z = (\alpha,\ \chi_k, \gamma_z)$ corresponding to the intercept term included in all models, the coefficients on the regressors entering the model and the coefficients on the set of eigenvectors E_z related to $\mathbf{W}_z$. In the BMA framework, the posterior distribution of β takes now the form of

$$p(\beta|y) = \sum_{j=1}^{2^K} \sum_{z=1}^{Z} p(\beta|M_j^z, y) p(M_j^z|y) \tag{3}$$

with y denoting the data and β the coefficient of interest. Inference on β is based on single inferences under models $j = 1, \ldots, 2^K \times Z$ weighted by their respective posterior model probabilities, $p(M_j^z|y)$, which in turn depend on the corresponding matrix of spatial weights. For more technical details please see Crespo Cuaresma and Feldkircher (2010).

1.3 An Illustration: The Boston Housing Data

In `R` we will do spatial filtering with the 'Boston housing data' which has been originally published in Harrison and Rubinfeld (1978). The dependent variable (`CMEDV`) contains the corrected median value of owner-occupied homes in USD 1000's for 506 observations. Among the explanatory variables we have per capita crime (`CRIM`), the pupil-teacher ratio by town (`PTRATIO`), the proportion of owner-occupied units built prior to 1940 (`AGE`), the proportion of non-retail business acres per town (`INDUS`), a variable that is proportional to the share of Afro Americans per town (`B`) and a variable reflecting the nitric oxides concentration (`NOX`) among others. For more details please see `?dataBoston`. We start with loading the data in `R` :

```
> rm(list = ls())
> library(spdep)
> library(spatBMS)
> library(BMS)
> data(dataBoston)
> data(boston.soi)
```

As in the earlier analyses of these data, we take logarithms of the variables `CMEDV`, `DIS`, `RAD` and `LSTAT` and squares of the regressors `RM` (`RM#RM`) and `NOX` (`NOX#NOX`) to model potential non-linearities. These transformations have been already carried out and the transformed variables stored in `dataBoston`.
We proceed with the construction of several weight matrices. To keep the example simple, we limit the space of candidate **W** matrices to five:

- The matrix boston.soi already comes along with the boston data set (W_0).
- A perturbation of the boston.soi matrix (W_1).
- A second perturbation of the boston.soi matrix (W_2).
- A 4 nearest neighbor matrix (K_{NN4})
- A 6 nearest neighbor matrix (K_{NN6})

The first matrix is included in the spdep package and constitutes probably a first order contiguity matrix. This class of matrices assigns positive (identical) weights to observations that share a common border. See Anselin (1988) and Crespo Cuaresma and Feldkircher (2010) for an interpretation of different coding schemes. For the sake of illustration we will randomly perturb the matrix using the function jitterW_binary which is provided in the corresponding Sweave file to this pdf.
The peturbation randomly adds / drops N=5 neighborhood relationships of the boston.soi weight matrix. The function takes as argument a matrix object, thus we have to transform boston.soi with the function nb2mat into a matrix.

```
> W0 <- boston.soi
> bostMat = nb2mat(boston.soi, style = "B")
> W1 = jitterW_binary(bostMat, N = 5)
> W1 = mat2listw(W1$Wmat)
> W1 = W1$neighbours
> W2 = jitterW_binary(bostMat, N = 5)
> W2 = mat2listw(W2$Wmat)
> W2 = W2$neighbours
```

It is necessary to re-transform the perturbed matrices into the nb class since the SpatialFiltering function we will use later on only allows for objects of this class as inputs. This can be done with the function mat2listw and extracting the neighborhood object by typing object$neighbours (in the example W2$neighbours).
Finally we set up two k-nearest neighbor matrices.

```
> data(boston)
> coords <- coordinates(boston.utm)
> col.knn4 <- knearneigh(coords, k = 4)
> col.knn4 = knn2nb(col.knn4)
> coords <- coordinates(boston.utm)
> col.knn6 <- knearneigh(coords, k = 6)
> col.knn6 = knn2nb(col.knn6)
```

As stated above we assume the model follows a SAR process. To free the residuals from spatial correlation we now filter the dependent variable. The SpatialFiltering function provided in the package spdep will extract the eigenvectors. A linear combination of these eigenvectors will allow us to separate spatial correlation from the dependent variable by using the identified eigenvectors as additional regressors in our econometric model.
Depending on the size of your **W** matrix, spatial filtering can take some while. The function takes the neighborhood objects we have defined above and the data to be filtered as main arguments. For more details see ?SpatialFiltering . Note also that we have set ExactEV to FALSE (quicker) which provides an approximation for our illustration example.

```
> y = as.data.frame(dataM[, 1, drop = F])
> yFilt.colGal0 = SpatialFiltering(dataM[, 1] ~ 1, ~-1, data = y,
+     nb = W0, style = "W", ExactEV = FALSE)
> yFilt.colGal1 = SpatialFiltering(dataM[, 1] ~ 1, ~-1, data = y,
```

```
+     nb = W1, style = "W", ExactEV = FALSE)
> yFilt.colGal2 = SpatialFiltering(dataM[, 1] ~ 1, ~-1, data = y,
+     nb = W1, style = "W", ExactEV = FALSE)
> yFilt.knn4 = SpatialFiltering(dataM[, 1] ~ 1, ~-1, data = y,
+     nb = col.knn4, style = "W", ExactEV = FALSE)
> yFilt.knn6 = SpatialFiltering(dataM[, 1] ~ 1, ~-1, data = y,
+     nb = col.knn6, style = "W", ExactEV = FALSE)
```

Finally we collect the eigenvectors in a list.

```
> WL.boston = list(Col0 = fitted(yFilt.colGal0), Col1 = fitted(yFilt.colGal1),
+     Col2 = fitted(yFilt.colGal1), knn4 = fitted(yFilt.knn4),
+     knn6 = fitted(yFilt.knn6))
```

Note that you can also access the extracted eigenvectors by typing e.g. `yFilt.colGal0$dataset` instead of e.g. `fitted(yFilt.colGal0)`.
Now we can start with the BMA part. All functions and input arguments of the `BMS` library are applicable. There are three important exceptions though: First, the empirical Bayes estimation as well as the hyper-g priors are so far not implemented. Thus you can specify the g-prior either by using the benchmark priors (Fernandez et al. 2001) or by any numerical number $g > 0$. See Feldkircher and Zeugner (2009) for more details on the influence of g on posterior results. Secondly, the enumeration algorithm (`mcmc=enum`) is in its current version not available for `spatFilt.bms`. Finally, to speed up calculations `BMS` provides the option `force.full.ols=FALSE`, which is not available in `spatFilt.bms`.
The additional argument `WList` of the function `spatFilt.bms` must be a list object with length corresponding to the number of weight matrices you use `length(WList)` must be greater than 1, that is you have to submit at least two sets of eigenvectors in order to use spatial filtering in the context of BMA. Each element of the list contains a matrix with the extracted eigenvectors, where the matrices do not have to have the same column dimension. In the example we have collected the eigenvectors in the object `WL.boston`. To have a quick look at the boston data set we run a short BMA chain with 1 million posterior draws (`iter=1e06`) after discarding the first 100,000 draws (`burn=1e05`). For more information regarding the other function arguments type `?spatFilt.bms`.

```
> dataM = as.matrix(apply(dataBoston, 2, as.numeric))
> model1 = spatFilt.bms(X.data = dataM, WList = WL.boston,
+     burn = 1e+05, iter = 1e+06, nmodel = 100, mcmc = "bd",
+     g = "bric", mprior = "random", mprior.size = (ncol(dataM) -
+         1)/2)
```

The object `model1` is a standard `BMS` object. It is important, to note that *all* the reported statistics (Posterior Inclusion Probabilities, Posterior Means, Posterior Standard deviations, etc.) are *after* having integrated out uncertainty with respect to $\mathbf{W}$.
To fix ideas, we will look at the disaggregated results to - for example - assess whether a variable receives only posterior support under a particular weight matrix or to look at the posterior inclusion probabilites of the spatial weight matrices, first:

```
> model1$wTopModels[, 1:3]
```

```
                08beb0    08b6b0     08feb0
CRIM         1.0000000 1.0000000 1.00000000
ZN           0.0000000 0.0000000 0.00000000
INDUS        0.0000000 0.0000000 0.00000000
CHAS         0.0000000 0.0000000 0.00000000
AGE          1.0000000 1.0000000 1.00000000
DIS          0.0000000 0.0000000 1.00000000
RAD          1.0000000 1.0000000 1.00000000
TAX          1.0000000 1.0000000 1.00000000
PTRATIO      1.0000000 0.0000000 1.00000000
B            1.0000000 1.0000000 1.00000000
LSTAT        1.0000000 1.0000000 1.00000000
NOX          0.0000000 0.0000000 0.00000000
RM           1.0000000 1.0000000 1.00000000
NOX#NOX      0.0000000 0.0000000 0.00000000
RM#RM        1.0000000 1.0000000 1.00000000
W-Index      1.0000000 1.0000000 1.00000000
PMP (Exact) 0.4700282 0.1026632 0.04539982
PMP (MCMC)  0.4697333 0.1020738 0.04491770
```

In the example we show posterior results for the first 3 models. The line W-Index tells you which **W** has been included in the particular model. In our example the W-Index indicates that the first weight matrix in `WL.boston` (i.e. $\mathbf{W}_0$='boston.soi') has been used in the first 3 regression models.
On the contrary,

```
> topmodels.bma(model1)[, 1:3]
```

```
                45f5      45b5       47f5
CRIM         1.000000 1.0000000 1.00000000
ZN           0.000000 0.0000000 0.00000000
INDUS        0.000000 0.0000000 0.00000000
CHAS         0.000000 0.0000000 0.00000000
AGE          1.000000 1.0000000 1.00000000
DIS          0.000000 0.0000000 1.00000000
RAD          1.000000 1.0000000 1.00000000
TAX          1.000000 1.0000000 1.00000000
PTRATIO      1.000000 0.0000000 1.00000000
B            1.000000 1.0000000 1.00000000
LSTAT        1.000000 1.0000000 1.00000000
NOX          0.000000 0.0000000 0.00000000
RM           1.000000 1.0000000 1.00000000
NOX#NOX      0.000000 0.0000000 0.00000000
RM#RM        1.000000 1.0000000 1.00000000
PMP (Exact) 0.465327 0.1056594 0.04672478
PMP (MCMC)  0.463265 0.0999770 0.04399500
```

shows the aggregated results: a matrix of containing the best models along with the according posterior model probabilities (exact and frequencies) after integrating out uncertainty with respect to the weight matrices. That is, if the first two models would be the same in terms of explanatory variables but differ regarding the employed weight matrix, the posterior mass of the two models is aggregated.

```
> estimates.bma(model1)
```

```
             PIP     Post Mean      Post SD Cond.Pos.Sign Idx
B       1.000000  5.663505e-04 9.677783e-05    1.00000000  10
LSTAT   1.000000 -1.794412e-01 1.950956e-02    0.00000000  11
RM#RM   1.000000  3.755998e-02 6.808795e-03    1.00000000  15
CRIM    0.999998 -4.484651e-03 8.451294e-04    0.00000000   1
RM      0.997432 -3.609266e-01 8.760509e-02    0.00000000  13
AGE     0.991173 -1.486851e-03 4.072440e-04    0.00000000   5
RAD     0.971146  6.086177e-02 1.943944e-02    1.00000000   7
TAX     0.942929 -2.718209e-04 1.059824e-04    0.00000000   8
PTRATIO 0.834085 -1.116339e-02 6.475768e-03    0.00000000   9
DIS     0.103370 -3.949460e-03 1.905292e-02    0.00612363   6
CHAS    0.091694 -1.447677e-03 8.525083e-03    0.00000000   4
NOX#NOX 0.085756 -8.045612e-03 9.180505e-02    0.02459303  14
NOX     0.080450  2.298915e-03 1.166256e-01    0.21758856  12
ZN      0.078457  4.273656e-06 1.117140e-04    0.81679136   2
INDUS   0.077824 -4.533138e-05 5.910705e-04    0.06576377   3
```

In the same vein, the posterior inclusion probability (PIP) for the variable `RM` for example corresponds to the sum of the posterior model probabilities of all regression models including that variable and the posterior mean is calculated as the weighted (by posterior model probabilities) average of posterior means over all weight matrices. Also note that the prior over the **W** space is uniform. Other forms of (informative) priors might be implemented in a later version of the computer code.
Coming back to the disaggregated (**W**-specific) results. To calculate the posterior inclusion probabilities of the **W** matrices, you can look a the frequency with which each **W** matrix has been visited by the sampler and express this in percentages:

```
> model1$Wcount
```

```
  Col0   Col1   Col2   knn4   knn6
992117   3950   3933      0      0
```

```
> model1$Wcount/sum(model1$Wcount) * 100
```

```
   Col0    Col1    Col2    knn4    knn6
99.2117  0.3950  0.3933  0.0000  0.0000
```

In the example, the original 'boston.soi' **W** matrix along with its perturbations receives overwhelming posterior support, whereas the k-nn matrices cannot explain the spatial patterns

present in the data. If you prefer statistics based on the best models (in the example we have set `nmodel=100` meaning that results are based on a maximum of 100 models receiving highest posterior support in terms of posterior model probabilities), you can use the function `pmpW.bma`:

```
> pmpW.bma(model1)

     PMP (Exact) PMP (MCMC)
Col0  99.5443044 99.5574083
Col2   0.2278478  0.2210406
Col1   0.2278478  0.2215511
knn4   0.0000000  0.0000000
knn6   0.0000000  0.0000000
```

Usually `model1$Wcount` gives you a reasonable approximation and is directly accessible from the `model1` object.
We finally want to check whether there is remaining spatial autocorrelation present in the residuals. For that purpose we can use the 'Moran's I' test calling `lm.morantest` for the best models. Although - in order to save time - we could have a look at the 10 best models only (in the example the first 10 models already account for more than 80% of posterior mass) we carry out the residual test for all models in order to get more accurate results. The function `mTest` is wrapper for the function `lm.morantest` provided in the package `spdep`. The loop re-computes the `nmodel` best - in terms of posterior model probabilities - regressions in order to get an object of class `lm` and finally applies the `lm.morantest`.
We do this once for the eigenvector augmented regressions (the spatial filtering approach) and once in a pure OLS fashion by setting the option `variants="double"`:

```
> mTest = moranTest.bma(object = model1, variants = "double",
+     W = nb2listw(boston.soi), nmodel = 100)
```

This allows us for a direct comparison of a non-spatial regression approach and the spatial filtering BMA approach pursued in this article. If the non-spatial linear regression models do not show any patterns of spatial residual autocorrelation a standard BMA regression - as with the function `bms` - dealing solely with uncertainty with respect to the explanatory variables might be preferable.
We now extract the corresponding p-values (remember the null hypothesis corresponds to no spatial autocorrelation):

```
> pvalueMat = cbind(sapply(mTest$moran, function(x) x$p.value),
+     sapply(mTest$moranEV, function(x) x$p.value))
```

Figure 1 shows the distribution of the p-values of the 'Moran's I' test, once for pure OLS regressions (without any spatial correction, left panel) and once augmented with the eigenvectors identified by the spatial filtering algorithm (right panel).
To sum up by incorporating the eigenvectors we successfully removed spatial residual autocorrelation from the regressions. The BMA framework helped us to explicitely deal with uncertainty stemming from the construction of the weight matrices and allowed us to get

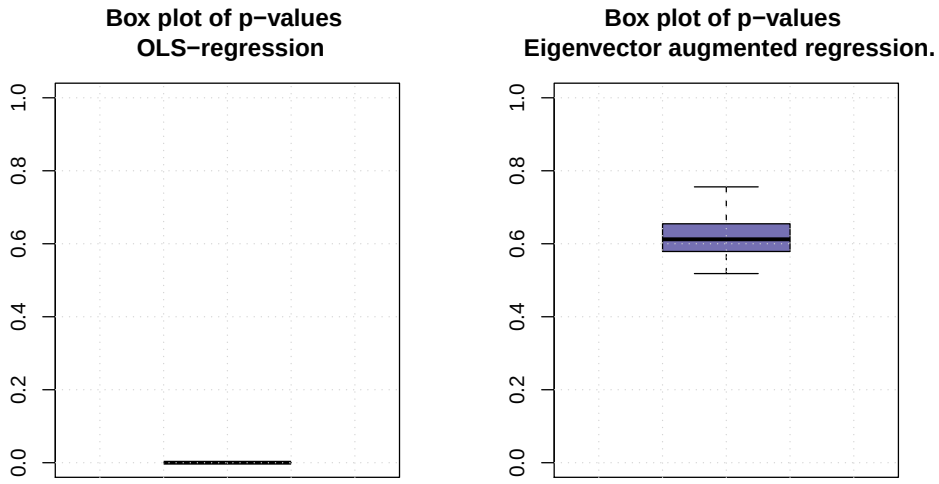


Figure 1: Distribution of p-values of a Moran's I test for spatial residual autocorrelation. Left panel shows the results for the 100 best models based on OLS regressions. Right panel includes on top of the explanatory variables the eigenvectors identified by the spatial filtering algorithm.

posterior inclusion probabilities of weighting matrices as well as the usual BMA statistics dealing with spatial correlation. Finally, once again please note that all functions of the BMA package 'BMS' are fully applicable to its spatial filtering variant. The remainder of this illustration shows posterior plots to assess convergence of the MCMC sampler, the posterior distribution of coefficients of interest, an image plot as well as a plot of the posterior model size. You can also type `spatBMS.demo` to get a short demonstration of `spatFilt.bms`'s main functions.

References

[1] Anselin, Luc, 1988. Spatial Econometrics: Methods and Models. *Kluwer Academic Publishers.*

[2] Crespo Cuaresma, Jesús and Feldkircher, Martin. 2010. Spatial Filtering, Model Uncertainty and the Speed of Income Convergence in Europe. *Working Papers 160, Oesterreichische Nationalbank.*

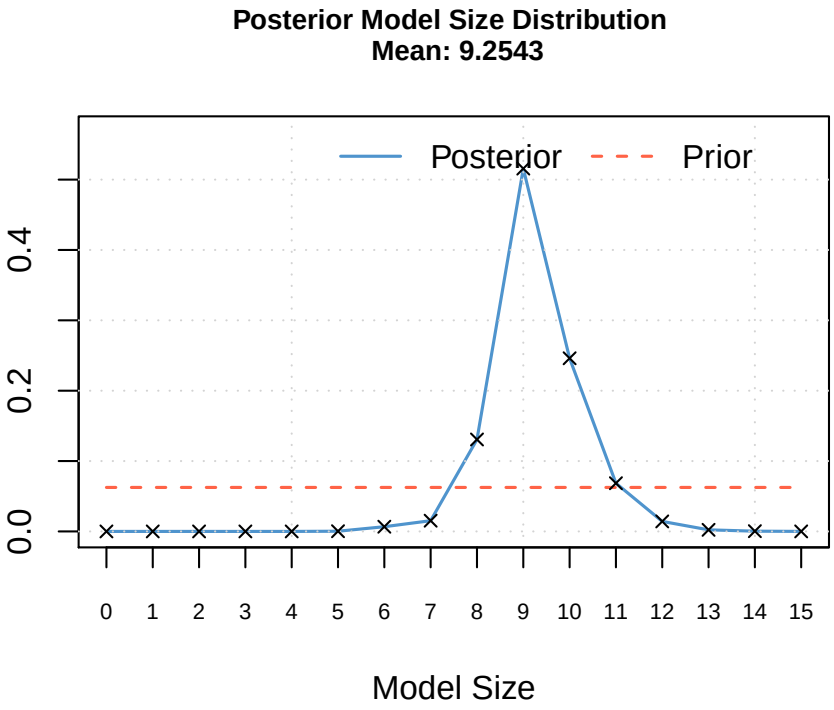
[3] Feldkircher, Martin and Zeugner, Stefan. 2009. Benchmark Priors Revisited: On Adaptive Shrinkage and the Supermodel Effect in Bayesian Model Averaging.*IMF Working Paper, WP/09/202.*

[4] Fernández, Carmen and Ley, Eduardo and Steel, Mark F.J.. 2001. Benchmark Priors for Bayesian Model Averaging *Journal of Econometrics, 100, 381-427.*

[5] Getis, Arthur and Griffith, Daniel A. 2002. Comparative Spatial Filtering in Regression Analysis. *Geographical Analysis*, 34:2, 130-140.

[6] Harrison, D. and Rubinfeld, D.L. 1978. Hedonic prices and the demand for clean air. *Journal of Environmental Economics & Management, Vol.5, 81-102.*

[7] Tiefelsdorf, Michael and Griffith, Daniel 2007. Semiparametric filtering of spatial autocorrelation: the eigenvector approach. *Environment and Planning A, 37, 1193-1221.*



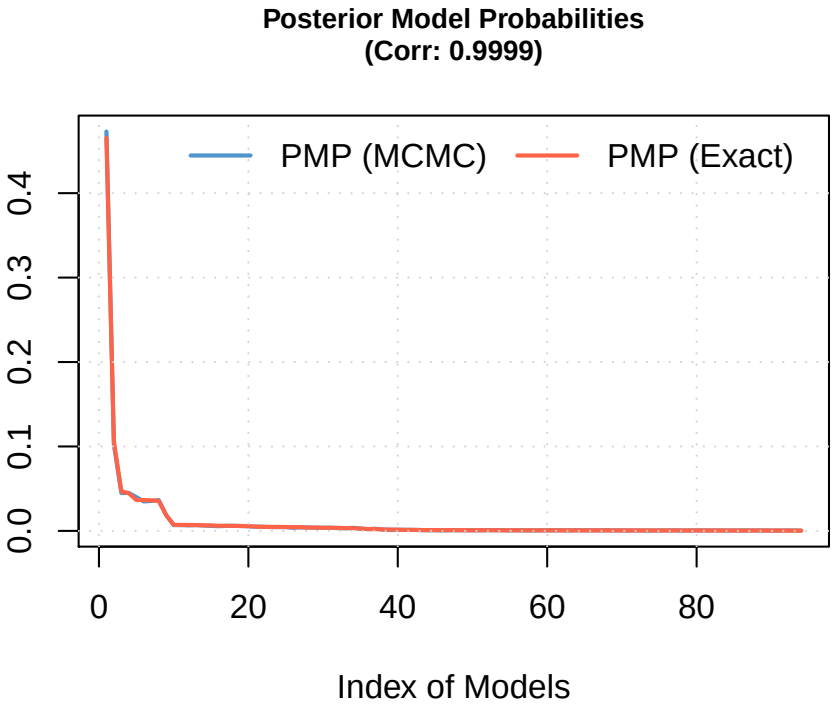


Figure 2: The top panel shows the posterior distribution of the model size indicating that a model explaining house prices contains on bottom panel a convergence plot. The correlation of > 0.99 indicates excellent convergence of the $MCMC$ algorithm.

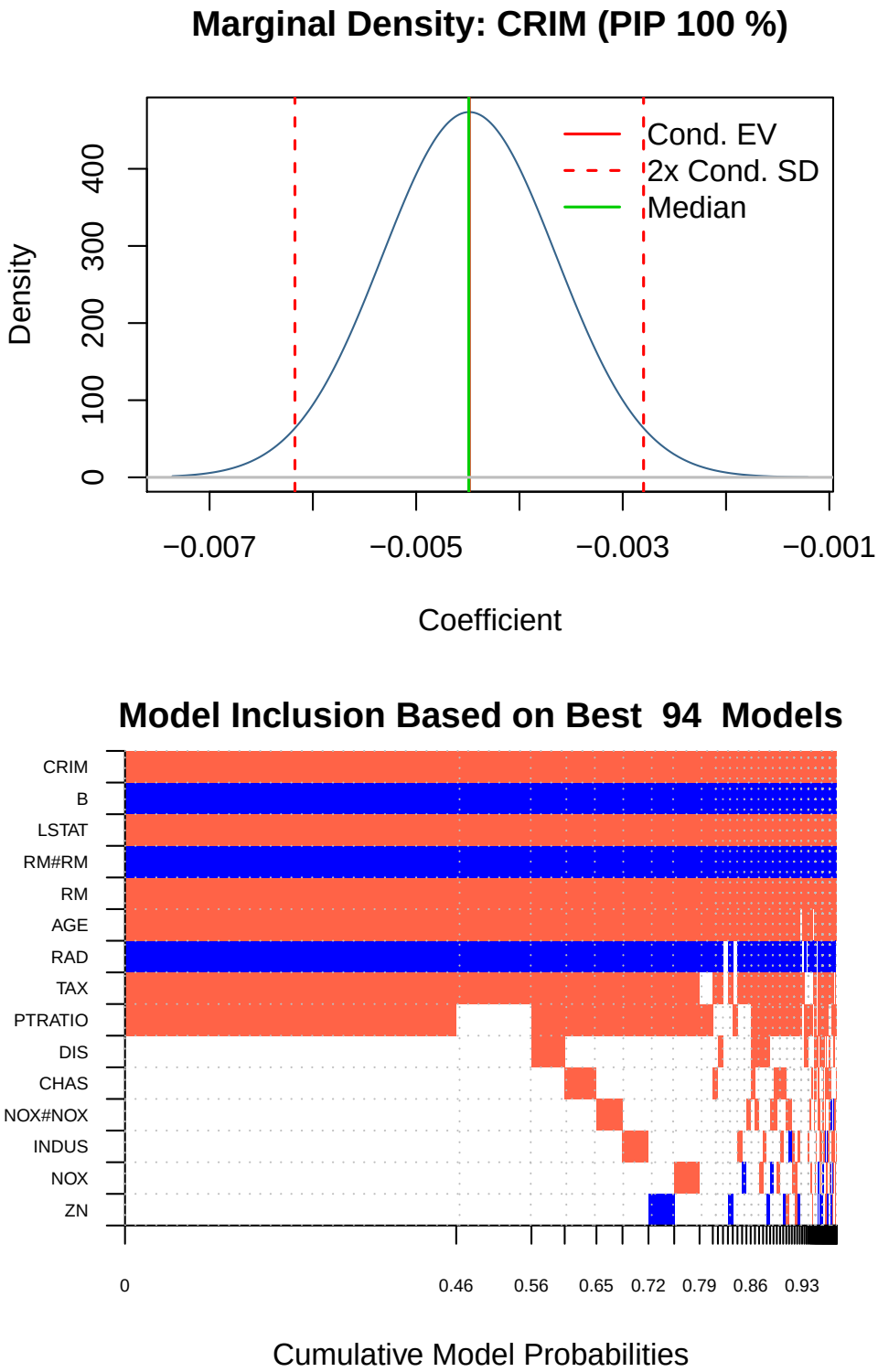


Figure 3: Top panel shows the posterior distribution of the coefficient 'CRIM', bottom panel shows an 'image' plot. The first plot indicates the negative relationship between the crime rate and house prices. The second plot illustrates the posterior inclusions probabilities and the signs (red = positive, blue = negative) of the employed regressors. The x-axis also indicates how dense the posterior model probabilities are distributed.