MODELAVG: A MATLAB Toolbox for Postprocessing of Model Ensembles

Jasper A. Vrugt[a,b]

[a] *Department of Civil and Environmental Engineering, University of California Irvine, 4130 Engineering Gateway, Irvine, CA 92697-2175*
[b] *Department of Earth System Science, University of California Irvine, Irvine, CA*

MODELAVG MANUAL

Abstract

Model averaging is statistical method that is widely used to quantify the conceptual uncertainty of environmental system models and to improve the sharpness and skill of forecast ensembles of multi-model prediction systems. Here, I present a MATLAB toolbox for postprocessing of forecast ensembles. This toolbox, called MODELAVG implements many different model averaging techniques, including methods that provide point forecasts only, and methods that produce a forecast distribution of the variable(s) of interest. MCMC simulation with DREAM is used for averaging methods without a direct closed-form solution of their point forecasts. The toolbox returns to the user (among others) a vector (or matrix with posterior samples) of weights and (if appropriate) standard deviation(s) of the members' forecast distribution, a vector of averaged forecasts (and performance metrics thereof), and (if appropriate) estimates of the width and coverage of the forecast distribution, and convergence diagnostics of the DREAM algorithm. The toolbox also creates many different figures with the results of each method. Three case studies illustrate the capabilities of the MODELAVG toolbox.

Keywords: , Model averaging, Information criterion averaging, Bayes information criterion, Equal weights averaging, Granger-Ramanathan averaging, Bates-Granger averaging, Mallows model averaging, Bayesian model averaging, Markov chain Monte Carlo simulation, DREAM

Email address: `jasper@uci.edu` (Jasper A. Vrugt)
URL: `http://faculty.sites.uci.edu/jasper` (Jasper A. Vrugt), `http://scholar.google.com/citations?user=zkNXecUAAAAJ&hl=en` (Jasper A. Vrugt)

1. Introduction and Scope

Multi-model ensemble prediction systems are used by many agencies in the world to forecast the behavior of complex systems. Such systems much better capture the uncertainty of the initial states, boundary conditions, and model physics, and therefore produce more skillful predictions than forecasts derived from a single model run. Yet, as most ensemble prediction systems do not account perfectly for all sources of uncertainty, some postprocessing is necessary to provide accurate forecasts.

Model averaging is a statistical methodology that can be used to improve the skill of a multi-model ensemble. This methodology can also be used to quantify conceptual model uncertainty as predictions generated by a single model are prone to statistical bias (by reliance on an invalid model) and underestimation of uncertainty (by under-sampling the feasible model space) (*Raftery et al.*, 1999; *Hoeting et al.*, 1999; *Neuman*, 2003; *Raftery et al.*, 2005; *Vrugt et al.*, 2006). Figure 1 illustrates the concept of model averaging. Consider that at a given time we have available the output of multiple different models. These models do not necessarily have to be calibrated. Now the goal is to weight the different models in such a way that the weighted estimate (model) is a better (point) predictor of the observed system behavior (data) than any of the individual models of the ensemble. Moreover, the density of the averaged model is hopefully a good estimator of the total predictive uncertainty.

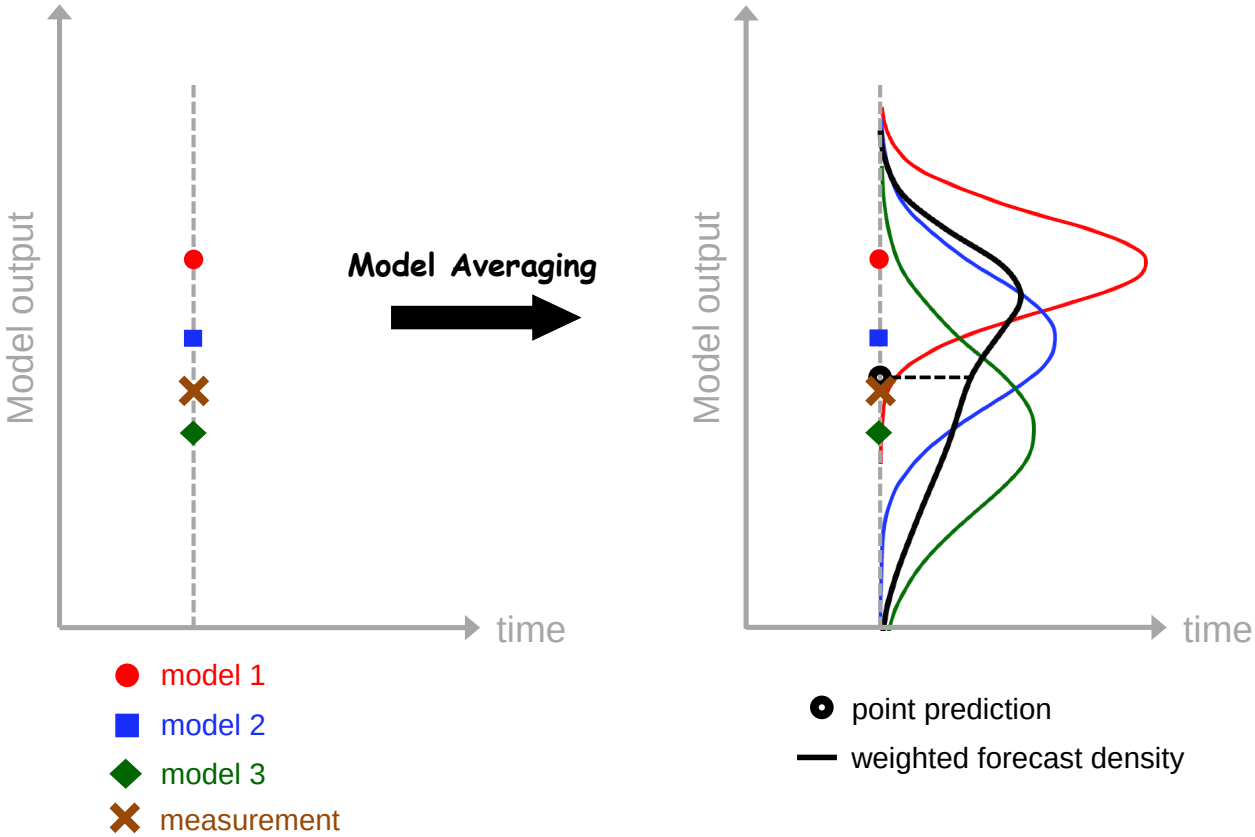


Figure 1: Schematic illustration of model averaging using a three member ensemble and single prediction of interest. The forecast of each models are displayed with the solid red circle, blue square and green diamond, respectively, and the verifying observation is indicated separately with the brown "×" symbol. The dotted black line connected with the symbol "o" denotes the weighted average of the forecasts of the three different models. This point predictor satisfies the underlying premise of model averaging as it is in better agreement with the data (smaller distance) than any of the three models of the ensemble. Some model averaging methods also construct a predictive density of this averaged forecast. This overall forecast pdf (solid black line) can be used for probabilistic forecasting and uncertainty analysis, and is simply a weighted average of each ensemble members predictive distribution (indicated with solid red, blue and green lines) centered at their respective forecasts.

To formalize the various model averaging strategies considered herein, let me denote by $\tilde{\mathbf{Y}} = \{\tilde{y}_1, \ldots, \tilde{y}_n\}$ a $n \times 1$ vector of measurements of a certain quantity of interest. These observations can be made at different times and locations in space, yet without loss of generality I conveniently ignore these two coordinates. Further assume that there is an ensemble of K different models that predict the observed data. The point forecasts of each model available with associated point forecasts D_{jk} where $k = \{1, \ldots, K\}$ and $j = \{1, \ldots, n\}$. If we merge the different model forecasts in a $n \times K$ matrix $\mathbf{D}$ then a weighted average can be readily constructed to predict the entity, $\tilde{\mathbf{Y}}$ of interest

$$\tilde{y}_j = \mathbf{D}_j^T \boldsymbol{\beta} + \varepsilon_j = \sum_{k=1}^{K} \beta_k D_{jk} + \varepsilon_j, \tag{1}$$

where $\mathbf{D}_j$ is a $1 \times K$ vector that stores the forecasts of each of the K models at a given location and time

($= j$th row of matrix $\mathbf{D}$), $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_K\}$ denotes the weight vector, the symbol T denotes transpose, and $\{\varepsilon_j\}$ is a white noise sequence, which will be assumed to have a normal distribution with zero mean and unknown variance. In the remainder of this manual, the index j is used to mean "$foralli \in \{1, \ldots, n\}$".

A bias correction step of the individual forecasts is performed prior to the construction of the weights. For instance, a linear transformation of the form

$$\tilde{D}^{\mathrm{b}}_{jk} = a_k + b_k D_{jk}, \tag{2}$$

will often suffice. The coefficients a_k and b_k for each of the models, $\{k = 1, \ldots, K\}$ can be calculated by ordinary least squares using the simple regression model

$$\tilde{y}_j = a_k + b_k D_{jk} + \varepsilon_j, \tag{3}$$

and the observations in the calibration set. This bias correction steps leads typically to a small improvement of the predictive performance of each model of the ensemble with a_k close to zero and b_k close to unity. If the calibration set is very small, the ordinary least squares estimates become unstable, and bias correction may distort the ensemble (*Vrugt and Robinson*, 2007). Although a (linear) bias correction is recommended for each of the constituent models of the ensemble, such correction is not made explicit in subsequent notation. For convenience, I simply continue to use the notation D_{jk} rather than D^{b}_{jk} for the bias corrected predictors of $\tilde{y}_j$.

The point forecasts associated with model (1) are

$$y^{\mathrm{e}}_j = \mathbf{D}^T_j \boldsymbol{\beta} = \sum_{k=1}^{K} \beta_k D_{jk}, \tag{4}$$

where the superscript "e" is used to indicate the expected (predicted) value of the averaged model.

In this manual, I introduce a MATLAB toolbox for postprocessing of forecast ensembles. This toolbox, called MODELAVG implements a large number of model averaging techniques, including methods that provide only a point forecast, and methods that produce a forecast distribution of the variable(s) of interest. MCMC simulation with DREAM is used for averaging methods without a direct closed-form solution of their (optimal) point forecasts. The toolbox returns to the user (among others) a vector (or matrix with posterior samples) of weights and (if appropriate) standard deviation(s) of the members' forecast distribution, a vector of averaged forecasts (and performance metrics thereof), and (if appropriate) estimates of the width and coverage of the forecast distribution, and convergence diagnostics of the DREAM algorithm. The various options of the toolbox are illustrated using three different case studies involving ensemble forecasts of river discharge, sea surface temperature and sea level pressure. These studies serve as templates for other data sets.

The remainder of this manual is organized as follows. Section 2 summarizes the theory of each of the model averaging methods that is available to the user. This is followed in section 3 with a detailed description of the MODEALVG toolbox. In this section I discuss each of the input and output arguments of the toolbox, and discuss the various options available to the user. Section 4 illustrates the different functionalities of the toolbox by application to three different forecast ensembles. Section 5 highlights recent research efforts aimed

at further improving the sharpness and coverage of the ensemble. Finally, section 6 provides a summary of the content of this manual.

2. Model Averaging methods

The MATLAB toolbox MODELAVG implements seven different model averaging techniques. These methods will be described in this section. Some of these methods restrict the weights of the ensemble members to the unit simplex, $\Delta^K \in \mathbb{R}^K$ or $\{\beta_k \geq 0 \text{ and } \sum_{k=1}^K \beta_k = 1\}$. Other methods in the toolbox relax this assumption and allow for positive and negative values of the weights, $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_K\}$.

2.1. Equal weights averaging

Equal weights averaging (EWA) assumes that each member of the ensemble has a similar value of the weight,

$$\boldsymbol{\beta}_{\text{EWA}} = \left(\frac{1}{K}, \ldots, \ldots, \frac{1}{K}\right). \tag{5}$$

These weights are independent of the training data set, $\tilde{\mathbf{Y}}$ and result in a weighted forecast, $y_j^{\text{e}} = \frac{1}{K}\sum_{k=1}^K D_{jk}$ which is simply equivalent to the mean ensemble prediction.

2.2. Bates-Granger averaging

A well-known choice, proposed by *Bates and Granger* (1969), is to weight each model by one over its forecast variance, $\beta_k = 1/\hat{\sigma}_k^2$ where the error variance, $\hat{\sigma}_k^2$ of the kth model is derived from its forecast errors of the calibration period, $\hat{\sigma}_k^2 = \frac{1}{n}\sum_{j=1}^n (\tilde{y}_j - D_{jk})^2$. If the models' forecasts are unbiased and their errors uncorrelated, these weights are optimal in the sense of producing predictors with the smallest possible Root Mean Square Error (RMSE). To enforce the weights to lie on Δ^K they are normalized as follows

$$\beta_{\text{BGA},k} = \frac{1/\hat{\sigma}_k^2}{\sum\limits_{k=1}^K 1/\hat{\sigma}_k^2} \tag{6}$$

so that they add up to one. In the remainder of this manual, I use the acronym BGA for Bates-Granger averaging.

2.3. Information criterion averaging

Information criterion averaging (ICA) was proposed by *Buckland et al.* (1997) and *Burnham and Anderson* (2002) and calculates the weights as follows

$$\beta_{\text{ICA},k} = \frac{\exp\left(-\frac{1}{2}I_k\right)}{\sum\limits_{k=1}^K \exp\left(-\frac{1}{2}I_k\right)}, \tag{7}$$

where I_k is an information criterion that depends on the complexity and goodness-of-fit of each model

$$I_k = -2\log(L_k) + q(p_k), \tag{8}$$

where L_k is the maximum likelihood of model k, and $q(p_k)$ signifies a penalty term which corrects for the number of model parameters. I consider herein Akaike's information criterion (AIC), for which $q(p) = 2p$, and Bayes information criterion (BIC), for which $q(p) = p\log(n)$, where n denotes the size of the calibration data set. I refer to the model averaging method of Equation (7) for IC and BIC as AICA and BICA, respectively, and to their respective weights values as $\boldsymbol{\beta}_{\mathrm{AICA}}$ and $\boldsymbol{\beta}_{\mathrm{BICA}}$. In the literature these methods are sometimes referred to as smooth AIC and smooth BIC, respectively. I assume that the number of parameters of each model are stored in the K-vector $\mathbf{p}$, and thus $\mathbf{p} = \{p_1, \ldots, p_n\}$.

To evaluate the information criteria numerically, it is convenient to assume, as I do herein, that the errors of the individual models are normally distributed. In this case, the log-likelihood of the kth model of the ensemble, $\log(L_k)$ can be calculated from

$$-2\log(L_k) = n\log\hat{\sigma}_k^2 + n \tag{9}$$

2.4. Granger-Ramanathan averaging

The weighting schemes described above do not exploit the covariance structure that may be present in the forecast errors of the individual models. A natural way to exploit the presence of covariances is to implement ordinary least squares (OLS) using the regression model of Equation (4).

Granger and Ramanathan (1984) suggests the following OLS estimates of the weights

$$\boldsymbol{\beta}_{\mathrm{GRA}} = \left(\mathbf{D}^T\mathbf{D}\right)^{-1}\mathbf{D}^T\tilde{\mathbf{Y}}, \tag{10}$$

where $\mathbf{D}$ is the $n \times K$ matrix of ensemble forecasts and $\tilde{\mathbf{Y}}$ signifies the $n \times 1$ vector with observations of the calibration data set. The OLS estimator can be shown to be the best linear unbiased estimator of $\boldsymbol{\beta}$. I conveniently refer to this model averaging method as GRA.

2.5. Bayesian model averaging

Hoeting et al. (1999) provide an excellent overview of the different variants of Bayesian Model Averaging (BMA) proposed in the literature. BMA differs from the other model averaging methods in this toolbox as it considers explicitly the uncertainty of each model's forecasts - and uses this uncertainty to construct a predictive distribution instead of only a weighted-average, deterministic, forecast. The BMA method offers an alternative to the selection of a single model from a number of candidate models, by weighting each candidate model according to its statistical evidence. Applications of BMA in hydrology and meteorology have been described by *Raftery et al.* (2005), *Gneiting et al.* (2005), *Vrugt and Robinson* (2007), *Vrugt et al.* (2008b) and *Bishop and Shanley* (2008).

The BMA method has several desirable properties, one of which is that it cannot only provides users with a deterministic (averaged) forecast but also with an associated forecast distribution. This forecast distribution summarizes all our knowledge about the target variable of interest, and can be used for probabilistic analysis and/or construction of 90 or 95% intervals. The BMA forecast density imposes one important constraint for the weights however, and that is that they must lie on the unit simplex, $\{\boldsymbol{\beta}|\beta_k \geq 0, k = \{1, \ldots, K\}$ and $\sum_{k=1}^{K}\beta_k = 1$. Without this restriction their values can produce rather awkward forecast distributions with densities that can even become smaller than zero. Such negative weights are tolerated if the goal of

the inference is point prediction, but cannot be sustained for density forecasts. I now describe an implementation of the BMA method that has found widespread application and use for postprocessing forecast ensembles of dynamic simulation models.

To start, lets assume that the forecasts of each model are subject to uncertainty. We can describe this uncertainty, with an (unknown) forecast distribution, $f_k(\cdot)$. This distribution expresses the prediction uncertainty of each model, $k = \{1, \ldots, K\}$ and its parameters can be inferred from a training data set. For now, I conveniently assume that the forecast distribution is centered at the forecasts of each individual model of the ensemble. We can then compute the forecast density of the BMA model, g_j, as follows

$$g_j = \sum_{k=1}^{K} \beta_k f_k(\tilde{y}_j) \tag{11}$$

The black line in Figure 1 provides an example of how the BMA density is computed from the individual models' forecast distribution, $f_k(\cdot)$. If we use for $f_k(\cdot)$ ($k = \{1, \ldots, K\}$) a normal distribution with mean equivalent to D_{jk} and variance equal to, σ_k^2

$$f_k(\tilde{y}_j | D_{jk}, \sigma_k^2) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\big(-\frac{1}{2}\sigma_k^{-2}(\tilde{y}_j - D_{jk})^2\big), \tag{12}$$

then the BMA predictive density, g_j, is simply equivalent to a Gaussian mixture distribution made up of K normal conditional distributions, each centered at their individual point forecast D_{jk} and with variance σ_k^2 (see Figure 2).

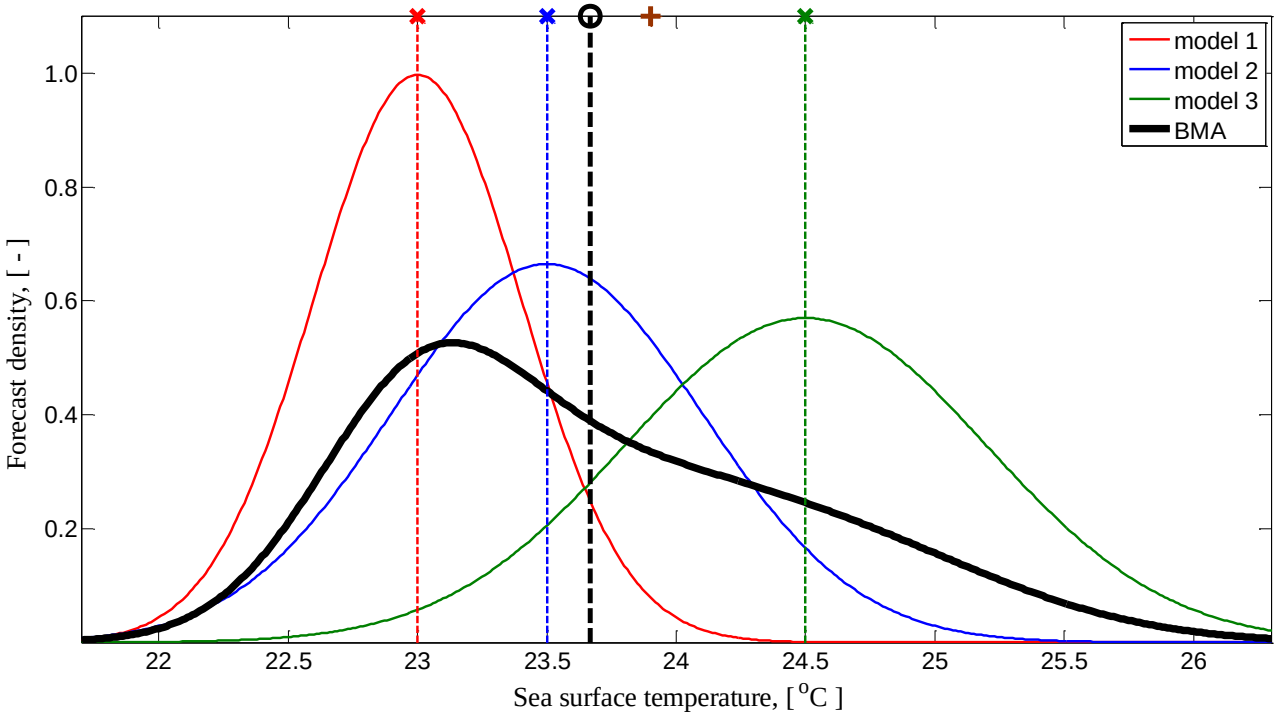


Figure 2: Schematic illustration of Bayesian model averaging using a $K = 3$ member ensemble for the sea surface temperature in degrees Celsius. The BMA predictive pdf, g_j, is indicated with the solid black line and equivalent to a weighted average of the conditional forecast distributions, $f_k(\cdot)$, of the members, $k = \{1, \ldots, K\}$ of the ensemble (displayed with solid red, blue and green lines). The forecast density of BMA can be used to compute prediction uncertainty ranges of the quantity of interest (sea surface temperature) at any desired confidence interval, $\alpha = 0.9$, 0.95 or 0.99. Also shown are the individual model forecasts ('×' symbol), the BMA deterministic point forecast ('o') symbol), and the verifying observation ('+' symbol). The deterministic point forecast of BMA can be compared to the ensemble mean and/or point predictors of other model averaging methods.

To ensure that g_j is a proper density (integrates to one), the BMA weights must lie on the unit simplex, Δ^K in $\mathbb{R}_+^K$ and thus the weights must be strictly positive and up to one

$$\sum_{k=1}^{K} \beta_k = 1 \quad ; \quad \beta_k \geq 0, \tag{13}$$

The BMA weight of each ensemble member can then be viewed as each model's relative contribution to predictive skill over the training (calibration) period. The BMA weights can thus be used to assess the usefulness of ensemble members, and this can be used as a basis for selecting ensemble members given the CPU-cost of running large ensembles (*Raftery et al.*, 2005).

The BMA point predictor, $g_j^{\bullet}$ is simply a weighted average of the individual models of the ensemble

$$g_j^{\bullet} = \sum_{k=1}^{K} \beta_k D_{jk} \tag{14}$$

which is a deterministic forecast in its own right, whose predictive performance can be compared with the

individual models of the ensemble, or with the ensemble mean (median). If we assume a normal conditional pdf for each model of the ensemble, then the BMA forecast variance, var(·), of Equation (14) can be computed directly using (*Raftery et al.*, 2005)

$$\mathrm{var}\big(g_j^{\bullet}|D_{j1},\ldots,D_{jK}\big) = \sum_{k=1}^{K}\beta_k\big(D_{jk}-\sum_{l=1}^{K}\beta_l D_{jl}\big)^2 + \sum_{k=1}^{K}\beta_k\sigma_k^2 \tag{15}$$

This variance consists of two terms, the first representing the ensemble spread, and the second representing the within-ensemble forecast variance.

2.5.1. Inference of BMA weights and variances

Successful implementation of the BMA method requires estimates of the weights, $\boldsymbol{\beta} = \{\beta_1,\ldots,\beta_K\}$, and standard deviations, $\boldsymbol{\sigma} = \{\sigma_1,\ldots,\sigma_K\}$, of the normal conditional pdfs of the ensemble members. Their values can be estimated by maximum likelihood from the training data set. This estimator has several desirable statistical properties and involves finding the optimum of the likelihood function (*Raftery et al.*, 2005)

$$\Big(\widehat{\boldsymbol{\beta}}_{\mathrm{BMA}},\widehat{\boldsymbol{\sigma}}_{\mathrm{BMA}}\Big) = \underset{\boldsymbol{\beta}\in\Delta^K,\boldsymbol{\sigma}\in\mathbb{R}_+^K}{\arg\max}\prod_{j=1}^{n}\sum_{k=1}^{K}\beta_k f_k(\tilde{y}_j|D_{jk},\sigma_k^2). \tag{16}$$

The likelihood of the BMA parameter values, $\mathbf{x} = \{\boldsymbol{\beta},\boldsymbol{\sigma}\}$ is thus computed as follows. First, the density of the BMA mixture distribution is evaluated at each observation of the training data set, $\tilde{\mathbf{Y}}$ using Equation (11). This BMA density is simply a weighted average of the pdfs of the individual ensemble members at the respective observations. These n values are then multiplied and this product is equivalent to the likelihood on the right-hand-side of Equation 16).

Figure 2 presents an example of the density function of the BMA mixture distribution for a three model ensemble with equal weights. If the observation (symbol "✚") were part of the training data set then the density of the BMA mixture distribution (black line) at this observation would equal about 0.3.

The use of the product operator in Equation (16) can pose numerical issues due to rounding errors introduced by the floating-point arithmetic of digital computers. Indeed, if n is sufficiently large, the likelihood of Equation (16) will eventually go to zero. For algebraic simplicity and numerical stability we therefore work with the logarithm of the likelihood instead. For the normal conditional forecast distribution of Equation (12) the log-likelihood function, $\mathcal{L}(\cdot)$, is equivalent to

$$\mathcal{L}(\boldsymbol{\beta}_{\mathrm{BMA}},\boldsymbol{\sigma}_{\mathrm{BMA}}|\mathbf{D},\tilde{\mathbf{Y}}) = \sum_{j=1}^{n}\log\left\{\sum_{k=1}^{K}\beta_k\frac{1}{\sqrt{2\pi\sigma_k^2}}\exp\left[-\frac{1}{2}\sigma_k^{-2}(\tilde{y}_j-D_{jk})^2\right]\right\}. \tag{17}$$

where the summation is over all models, $k = \{1,\ldots,K\}$, and observations, $j = \{1,\ldots,n\}$, of the training data set. This thus involves the inference of $d = 2K$ parameters, namely the weight, β_k and standard deviation, σ_k of the normal distribution of each of the members of the ensemble.

Unfortunately, there are no closed-form solutions that conveniently maximize Equation (17). We therefore have to resort to an iterative solution method. In their seminal paper, *Raftery et al.* (2005) recommends the Expectation-Maximization (EM) algorithm for BMA model training. This method alternates between an expectation (E) step, which calculates the expected value of the log-likelihood of Equation (17) at the

current estimate of the BMA parameters, and a maximization (M) step, which computes new values of the parameters that maximize the expected log-likelihood value of the E step. A detailed description of the EM method appears in Appendix A of this manual. There, I also present a MATLAB code of this algorithm that can be used for BMA model training if the forecast pdf of each member is described with a normal distribution.

The EM method exhibits many desirable properties as it is relatively easy to implement, computationally efficient, and the maximization step of Equation (A.2) is designed such that that the weights are always positive and add up to one. Nonetheless, global convergence of this algorithm cannot be guaranteed as a single starting point is used in the BMA model space. Of course, multiple different initial starting points can be used, but as each trial operates independently, this is rather CPU-inefficient (*Duan et al.*, 1992). What is more, the mathematical formulations of the E and M step in the EM algorithm depend on the forecast distribution, $f_k(\cdot); k = \{1, \ldots, K\}$ that is used for the members of the ensemble. Indeed, the function EM_NORMAL in Appendix A can be used only for variables such as temperature and pressure whose conditional pdf is well described with a normal distribution (see left two plots in Figure 3). The histograms of the other three variables (C: wind speed, D: rainfall, and E: discharge) are truncated by zero and exhibit much more skew to the right. Their conditional pdf is much better approximated by a gamma distribution (*Vrugt and Robinson*, 2007; *Sloughter et al.*, 2010), yet this requires modifications to the E and M step in the EM algorithm (more later).

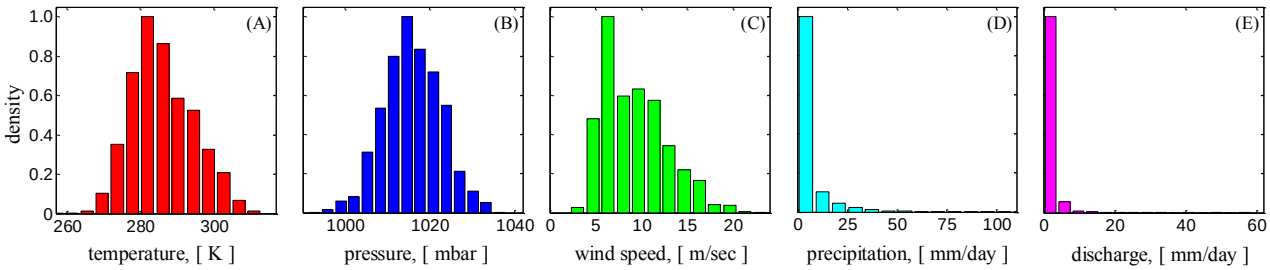


Figure 3: Histograms of daily measurement records of five different variables, including (A) outside temperature [K], (B) atmospheric pressure [mbar], (C) wind speed [m/sec], (D) precipitation [mm/day], and (E) river discharge [mm/day]. Whereas the first two variables (temperature and pressure) follow a normal distribution, the last three variables are truncated at zero and much better described with a gamma (skewed) distribution

Another limitation of the EM method is that it returns only the maximum likelihood values of the BMA model parameters without recourse to their underlying posterior uncertainty. This information is helpful to assess the usefulness of individual ensemble members (*Vrugt and Robinson*, 2007). A small ensemble has important computational advantages as fewer models need to be setup and executed. These imitations of the EM method motivated (*Vrugt et al.*, 2008b) to use Bayesian inference with the DREAM algorithm for BMA model training. This multi-chain Markov chain Monte Carlo simulation method uses differential evolution as genetic algorithm for population evolution with a Metropolis selection rule to decide whether candidate points should replace their respective parents or not. This approach scales automatically the orientation and scale of the proposal distribution en route to the target distribution, and returns not only the maximum likelihood values of the BMA model parameters but also their posterior uncertainty. The use of multiple chains also offers a robust protection against premature convergence, and opens up the use of

a wide arsenal of statistical measures to test whether DREAM has converged to the posterior distribution. What is more, the user does not have to adapt the formulation of the log-likelihood function

$$\ell(\mathbf{x}|\mathbf{D},\tilde{\mathbf{Y}}) = \sum_{j=1}^{n} \log\left\{\sum_{k=1}^{K} \beta_k f_k(\cdot)\right\}, \tag{18}$$

where $\mathbf{x}$ is a vector with the parameters of the forecast density of the BMA mixture distribution. The user only has to specify which statistical distribution to use for the $f_k(\cdot)$s of the ensemble. A detailed description of DREAM appears in Appendix B of this manual along with a basic implementation of this algorithm in MATLAB. Interested readers are also referred to the MATLAB toolbox of DREAM presented in *Vrugt* (2016).

2.5.2. The BMA conditional distribution

The MATLAB toolbox presented herein allows the user to implement two different distributions for the conditional pdf, $f_k(\cdot)$, of the ensemble members. This includes the normal and gamma distribution, and allows a proper characterization of variables with/without a skew (see Figure 3). The gamma distribution is given by

$$f_k(\tilde{y}_j|a,b) \sim \frac{1}{b^a\Gamma(a)}\tilde{y}_j^{(a-1)}\exp(-\tilde{y}_j/b), \tag{19}$$

where $a > 0$ and $b > 0$ are a shape and scale parameter, respectively and $f_k(\tilde{y}_j) = 0$ if $\tilde{y}_j \leq 0$. The mean and variance of the gamma distribution are determined by the values of a and b, as follows, $\mu = ab$ and $\sigma^2 = ab^2$. Hence, the values of a and b cannot be chosen freely as their product should equate to D_{jk} and the mean of the gamma distribution centers around the actual forecast of the kth member of the ensemble. I therefore calculate the values of a and b as follows

$$a_{jk} = \frac{|D_{jk}|^2}{\sigma_k^2} \quad ; \quad b_{jk} = \frac{\sigma_k^2}{|D_{jk}|}, \tag{20}$$

and estimate the standard deviation of the gamma distribution, σ_k using MCMC simulation with DREAM. This involves the inference of $d = 2K$ parameters, namely the weight, β_k and standard deviation, σ_k of the gamma forecast distribution of each member, $k = \{1,\ldots,K\}$ of the ensemble.

Thus far I have made three important assumptions, (a) each member of the ensemble has the same "type" of forecast distribution (normal or gamma), (b) the variance of this distribution differs among the members of the ensemble, and (c) the variance is constant. The first assumption will not be contested, as the shape of the forecast distribution is determined by the frequency distribution (histogram) of the target variable, yet the second and third assumption might be too restrictive. The MATLAB toolbox of MODELAVG therefore implements four different parameterizations of the standard deviation of the forecast distribution.

(1) common constant variance: all members of the ensemble have the same standard deviation, that is $\sigma_1 = \sigma_2 = \ldots = \sigma_K$. This simplifies somewhat the inference and involves $d = K+1$ fitting parameters.
(2) individual constant variance: all members of the ensemble have their own standard deviation, and thus $\boldsymbol{\sigma} = \{\sigma_1,\ldots,\sigma_K\}$. This assumption was made in our notation thus far and results in a BMA mixture distribution with $d = 2K$ unknowns.
(3) common non-constant variance: the standard deviation of the forecast distribution is dependent on

the magnitude of the forecast. This approach is implemented using $\sigma_{jk} = cD_{jk}$, where the multiplier c applies to all models and forecasts of the ensemble. This approach involves $d = K + 1$ fitting parameters.

(4) individual non-constant variance: the standard deviation of the forecast distribution is member and forecast dependent. This approach is implemented using $\sigma_{jk} = c_k D_{jk}$, where the K multipliers are subject to inference. This involves $d = 2K$ unknowns.

The first two approaches assume a homoscedastic (constant) variance of the conditional distribution of each ensemble member. This approach might be appropriate for variables such as the temperature and pressure of the atmosphere that are known to have a constant measurement error (*Vrugt et al.*, 2006). The last two approaches assume a heteroscedastic (non-constant) variance of the conditional forecast distribution. The variance of this distribution increases with value of the forecast. This assumption is appropriate for variables such as rainfall (*Sloughter et al.*, 2007), discharge (*Vrugt and Robinson*, 2007) and wind speed (*Sloughter et al.*, 2010) whose measurement errors are known to increase with magnitude of the observation. Note, it is rather easy to formulate other models for the variance of the forecast distribution, for instance, one can augment the heteroscedastic variance with a constant value, for instance, $\sigma_{jk} = c_k D_{jk} + c_2$ so that the variance does not have to become zero if $D_{jk} = 0$. This adds one additional parameter, c_2 to the BMA forecast density of Equation (11) and now involves $d = 2K + 1$ unknowns.

The MATLAB function BMA_CALC calculates the value of the log-likelihood, $\mathcal{L}(\mathbf{x}|\mathbf{D}, \tilde{\mathbf{Y}})$ of the BMA model in Equation (16) for a given vector, $\mathbf{x}$ (input argument) of weights and standard deviations (or proxies thereof) of the ensemble members, ensemble forecast $\mathbf{D}$ and training data observations $\tilde{\mathbf{Y}}$. The fifth input argument, `options` contains the properties of the forecast distribution, and is defined by the user in the MODELAVG toolbox (see section 3).

MATLAB code of `BMA_calc`: This function computes the value of the log-likelihood (return argument) for a vector `x` with BMA weights and standard deviations (or proxies thereof) of the members' conditional distribution. The second and third input argument store the ensemble forecasts, `D` and verifying observations, `Y`, respectively, and the last input argument `options` is a structure with fields that determines the properties of the members' forecast distribution. Notation is consistent with main text. Built-in functions are highlighted with a low dash. The fields PDF and VAR of `options` store the name and variance option of the conditional distribution. The `switch` function switches among the several cases listed in the code. The function `normpdf(Y,D(:,k),sigma(:,k))` returns the probability densities of the normal distribution with mean equal to the `n` forecasts of the `k`th ensemble member, "D(:,k)", and standard deviation "sigma(:,k)", evaluated at the observed values, `Y`. The function `gampdf(Y,A(:,k),B(:,k))` computes the density at the observed values, `Y`, of the gamma distribution with shape, "A(:,k)", and scale, "B(:,k)", vectors of the `k`th ensemble member, respectively. `log(L)` computes the natural logarithm of the n likelihood values of the BMA mixture distribution, and `sum()` returns the sum of the `n` log-likelihood values.

```
function [ loglik ] = BMA_calc ( x , D , Y , options )
% This function calculates the log likelihood of the BMA mixture distribution
% Function of MODELAVG toolbox, V1.0

if nargin<4,
    error('MODELAVG:BMA_calc:TooFewInputs','Requires at least four input arguments.');
end

[PDF,VAR] = v2struct(options,{'Fieldnames','PDF','VAR'}); % Unpack fields options
[n,K] = size(D);            % Number of forecasts and number of ensemble members
beta = x(1:K)';             % Unpack weights of member's conditional pdf

switch VAR % VARIANCE OPTION -> (n x K)-matrix "sigma" with forecast standard deviations
    case {'1'} % 1: common constant variance
        sigma = x(K+1) * ones(n,K);
    case {'2'} % 2: individual constant variance
        sigma = bsxfun(@times,x(K+1:2*K),ones(n,K));
    case {'3'} % 3: common non-constant variance
        c = x(K+1); sigma = c * D;
    case {'4'} % 4: individual non-constant variance
        c = x(K+1:2*K); sigma = bsxfun(@times,c,D);
    otherwise
        error('MODELAVG:BMA_calc','Unknown variance option');
end
sigma = max(sigma,eps);     % each element (n x K)-matrix sigma at least equal to 2.22e-16

switch PDF % CONDITIONAL DISTRIBUTION -> (n x K)-matrices "A" and "B"
    case {'normal'} % Gaussian with mean "D" and standard deviation "sigma"
        A = D; B = sigma;
    case {'gamma'}  % Gamma with shape "A" and scale "B"
        mu = abs(D); var = sigma.^2; A = mu.^2./var; B = var./mu;
end

L = pdf(PDF,repmat(Y,1,K),A,B);     % (n x K)-matrix of likelihoods forecasts at Y

lik = L*beta + realmin;             % (n x 1)-vector of likelihoods BMA model at Y
loglik = sum(log(lik));             % Return log-likelihood of BMA model
```

Thus, the code first computes the value of the standard deviation for each forecast and member of the ensemble. This results in the $n \times K$ matrix `sigma` which has the same numbers of rows and columns as matrix `D` with ensemble forecasts. Then, the likelihood of the BMA mixture distribution is evaluated at each observation of the training data set by taking the sum of the weighted densities of the K different conditional distributions evaluated at $\tilde{\mathbf{Y}}$. Then, the log-likelihood is computed by taking the sum of the natural log values of the n different densities. For users, it is rather straightforward to implement their own variance definition.

2.6. Mallows model averaging

Mallows model averaging (MMA) is a Frequentist solution to the problem of model averaging. The MMA method uses the following penalized sum of squared residuals objective function

$$C_n(\boldsymbol{\beta}|\mathbf{D}, \tilde{\mathbf{Y}}, \hat{\sigma}^2, \mathbf{p}) = \sum_{j=1}^{n} \left(\tilde{y}_j - \boldsymbol{\beta}^T \mathbf{D}_j\right)^2 + 2\hat{\sigma}^2 \sum_{k=1}^{K} \beta_k p_k \tag{21}$$

where, as before, p_k denotes the number of "free" parameters of the kth model of the ensemble, the symbol T signifies transpose, and $\hat{\sigma}^2$ is an estimate of the variance σ^2 of ε_j in Equation (1). This value is often set equivalent to the variance of the forecast error of the most complex model (= parameter rich) of the ensemble.

We can now find the optimal values of the MMA weights by minimizing Mallows' criterion in Equation (21) or

$$\widehat{\boldsymbol{\beta}}_{\text{MMA}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^K}{\arg\min}\, C_n(\boldsymbol{\beta}|\mathbf{D}, \tilde{\mathbf{Y}}, \hat{\sigma}^2, \mathbf{p}), \tag{22}$$

where the weights are allowed to vary freely in $\mathbb{R}^K$ and are thus not restricted to the unit simplex Δ^K. The value of $\boldsymbol{\beta}_{\text{MMA}}$ can also be estimated by maximizing the following log-likelihood function

$$\mathcal{L}(\boldsymbol{\beta}|\mathbf{D}, \tilde{\mathbf{Y}}, \hat{\sigma}^2, \mathbf{p}) \simeq -\frac{1}{2} C_n(\boldsymbol{\beta}|\mathbf{D}, \tilde{\mathbf{Y}}, \hat{\sigma}^2, \mathbf{p}), \tag{23}$$

using MCMC simulation with DREAM (*Diks and Vrugt*, 2010; *Vrugt et al.*, 2008b, 2009). Indeed, the maximum likelihood of the MMA weights is simply equivalent to the sample of DREAM with largest value of Equation (23). This sample is easy to find in MATLAB using the built-in `max` operator. The theory and MATLAB implementation of the DREAM algorithm is presented in Appendix B. A separate toolbox of this algorithm is available in MATLAB and described in detail by *Vrugt* (2016).

If so desired, we can also restrict the MMA weights to lie on the unit simplex, Δ^K, in $\mathbb{R}_+^K$ and thus to be positive and add up to one, $\beta_k \geq 0; \sum_{k=1}^{K} \beta_k = 1$. This alternative model averaging method is hereafter conveniently referred to as MMA$^{\Delta}$.

The MATLAB function `MMA_calc` listed below calculates the log-likelihood of Equation 23 for a given input vector of weights, `beta`.

MATLAB code of `MMA_calc`: This function calculates the log-likelihood of the MMA deterministic point forecasts using as input arguments, the weights `x`, ensemble forecasts, `D`, verifying observations, `Y`, number of "free" parameters of each model of the ensemble, `p`, and variance of the forecast error, `var_err` of the most complex model of the ensemble. Notation is consistent with Equation (23) in main text. Matrix algebra is used to minimize the CPU-time. Built-in functions are highlighted with a low dash. The function `sum()` calculates the sum of the squared differences between the MMA point forecast and the verifying observations, and `size(D)` returns the number of rows and columns of the matrix `D`.

```
function [ loglik ] = MMA_calc ( beta , D , Y , var_err , p );
% This function calculates the log likelihood of MMA
% Function of MODELAVG toolbox, V1.0

% B.C. Hansen, "Least Squares Model Averaging", Econometrica, vol. 75,
%   no. 4, pp. 1175-1189, 2007

if nargin<5,
    error('MODELAVG:MMA_calc:TooFewInputs','Requires at least five input arguments.');
end

G = D*beta';                                    % MMA deterministic point forecast
Cn = sum((Y - G).^2) + 2*var_err*beta*p';        % Mallows criterion: ref Equation (11)
loglik = -Cn/2;                                 % Log-likelihood of x = { MMA weights }
```

This concludes the theory of the different model averaging methods. I now describe the implementation of the theory and codes describes above in the MODELAVG toolbox in MATLAB.

3. The MODELAVG toolbox

The MODELAVG toolbox implements each of the model averaging methods described in section 2 in MATLAB and returns to the user the values of the weights, $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_K\}$ and members' standard deviation(s) (or proxies thereof) of the conditional distribution, $f_k(\cdot)$ (if BMA is used). You can download the MODELAVG toolbox from my website at the following link `http://faculty.sites.uci.edu/MODELAVG`. Appendix C explains how to setup the MODELAVG toolbox in MATLAB.

3.1. MODELAVG: MATLAB implementation

The MODELAVG toolbox can be executed from the MATLAB prompt by typing the following statement in the command window

$$\texttt{[x,output] = MODELAVG(method,D,Y,options)} \tag{24}$$

where `method` (string), `D` ($n \times K$ matrix), $\mathbf{Y}$ ($n \times 1$ vector) and `options` (structure array) are input arguments defined by the user, and `x` (vector) and output (structure array) are output arguments that are computed by the function MODELAVG and returned to the user. To minimize the number of input and output arguments of MODELAVG, the structure `options` and `output` group related variables using data

containers called fields, more of which later. The structure `options` is an optional input argument required only for information criterion averaging, Bayesian model averaging and Mallows model averaging.

A summary of the different functions of the MODELAVG toolbox appears in Appendix D. I will now discuss each of the input and output variables.

3.2. First input argument: `method`

The variable `method` defines with a string enclosed between quotes the name of the model averaging method that will be used by the function MODELAVG. The user can select among the eight different methods discussed of section 2 using the acronym (in quotes) listed in the first column of Table 1.

Table 1: Acronym used by input argument method for each of the model averaging methods of section 2. The last column summarizes for each method whether the weights are restricted to the unit simplex, Δ^K, or not.

Content `method`	Description	Δ^K
'EWA'	Equal weight averaging	Yes
'BGA'	Bates-Granger averaging	Yes
'AICA'	Akaike information criterion averaging	Yes
'BICA'	Bayes information criterion averaging	Yes
'GRA'	Granger-Ramanathan averaging	No
'BMA'	Bayesian model averaging	Yes
'MMA'	Mallows' model averaging	No
'MMA-S'	Mallows' model averaging	Yes

The function MODELAVG is case insensitive; thus, the user can input to method lower case (small) and upper case (capital) letters of the acronyms listed in Table 1. Almost all methods restrict the weights to the unit simplex, except Granger-Ramanathan averaging (`method = 'GRA'`) and variant MMA$^{\Delta}$ of Mallows Model Averaging (`method = 'MMA-S'`).

The first five model averaging methods listed in Table 1 ('EWA', 'BGA','AICA', 'BICA' and 'GRA') will execute rapidly as they have a direct closed-form solution for their weights. The last three methods ('BMA', 'MMA', and 'MMA-S') require an iterative solution with DREAM to locate the maximum likelihood values of their weights. If BMA is used, then DREAM returns as well estimates of the standard deviation of the members' conditional distribution, $f_k(\cdot); k = \{1, \ldots, K\}$.

3.3. Second input argument: `D`

The second input argument `D` of the function MODELAVG is a $n \times K$ matrix with n forecasts of each member of the ensemble. This input argument is thus equivalent to $\mathbf{D}$ and uses a separate column for each of the K models of the ensemble. Bias correction is recommended for each ensemble member particularly for model averaging methods that restrict the weights to the unit simplex. The MODELAVG toolbox has a built-in utility for linear bias correction (see Equation (2)), yet more advanced bias-correction methods can be devised by the user - more of which later.

3.4. Third input argument: `Y`

The third input argument `Y` of the MODELAVG function stores the training data set, $\tilde{\mathbf{Y}}$ with verifying observations. This $n \times 1$ vector is used to determine the weights of each model averaging method. If BMA is used, then this also includes estimates of the variance(s) of the members' forecast distribution. The number of rows of `Y` should match exactly the number of rows of the forecast ensemble stored in `D` (second input argument of function MODELAVG).

3.5. Fourth input argument: `options`

The fourth and last input argument of the function MODELAVG is the structure `options`. This input argument is optional and used only by information criterion averaging (AICA or BICA), BMA, MMA, and MMA$^{\Delta}$ although the field print applies to all methods. Table 2 summarizes the different fields of structure options and their content.

Table 2: Overview of the different fields of input argument `options`

Field `options`	Description	Options	Type	`method`
PDF	Forecast distribution	'normal'/'gamma'	string	BMA
VAR	Variance option	'1'/'2'/'3'/'4'	string	BMA
alpha	Prediction interval	e.g. 0.90/0.95/0.99	real	BMA
p	Model complexity	> 0	K-vector	AICA/BICA/MMA/MMA$^{\Delta}$
print	Screen output	'yes' or 'no'	string	All

The first three fields PDF, VAR and alpha of structure `options` are necessary input for the BMA method. The field PDF lists with a string enclosed between quotes the name of the conditional distribution, $f_k(\cdot)$ that is used for the $k = \{1, \ldots, K\}$ ensemble members. Built-in options include 'normal' and 'gamma' for a Gaussian and gamma forecast distribution, respectively.

The field VAR of structure `options` allows the user to select with a string (between quotes) the variance of the conditional distribution. The user can select among four different options,

1. (A) `VAR = '1'` : common constant variance; $\sigma_1 = \sigma_2 = \ldots = \sigma_K$.
2. (B) `VAR = '2'` : individual constant variance; $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_K\}$.
3. (C) `VAR = '3'` : common non-constant variance; $\sigma_{jk} = cD_{jk}$.
4. (D) `VAR = '4'` : individual non-constant variance; $\sigma_{jk} = c_k D_{jk}$.

These options allow for a homoscedastic and heteroscedastic variance of the conditional distribution of the ensemble members, and have been discussed in section 2.5.2 of this manual.

The field alpha of structure `options` defines the prediction interval of the BMA model. This interval is computed for each observation of the training data set, `Y`, and returned to the user in the output argument output of MODELAVG (see next section). Typical values of alpha are 0.90, 0.95 or 0.99 for a 90, 95 or 99% prediction interval of the BMA mixture distribution, respectively. If the user does not specify the field alpha or leaves empty its content then the toolbox will assume alpha = 0.95.

The field p of structure options stores the number of parameters for each model of the ensemble. Thus, p should contain K values, and have dimensions $1 \times K$ (horizontal vector) or $K \times 1$ (vertical vector),

respectively. This field is required input for information criterion averaging (AICA and BICA), MMA, and MMA$^{\Delta}$.

Finally, the field print of structure `options` controls the output writing of the MODELAVG toolbox. If the content of print equates to 'yes' then the toolbox will visualize, in many different figures, the output of each model averaging method. This output writing is suppressed if field print is set to 'no'.

The user can employ upper case and lower case letters for each of the fields of structure `options`. The same holds for the content of each field of `options`. Thus, `options.pdf = 'NORMAL'` is equally valid as `options.PDF = 'normal'`.

3.6. Output arguments

The function MODELAVG returns to the user two output arguments, `x` (vector or matrix) and `output` (structure array). The content of these two output arguments differs per model averaging method and the respective settings that are being used.

3.6.1. Return argument x

The content of `x` depends on the model averaging method that is being used (see Table 3).

Table 3: Content and number of rows and columns of output argument `x` of the MODELAVG toolbox

`method`	`VAR`	Content of `x`	Size of `x`
'EWA'		$\{\beta_1, \ldots, \beta_K\}$	$1 \times K$
'BGA'		$\{\beta_1, \ldots, \beta_K\}$	$1 \times K$
'AICA'		$\{\beta_1, \ldots, \beta_K\}$	$1 \times K$
'BICA'		$\{\beta_1, \ldots, \beta_K\}$	$1 \times K$
'GRA'		$\{\beta_1, \ldots, \beta_K\}$	$1 \times K$
'MMA'		$M \times \{\beta_1, \ldots, \beta_K, \mathcal{L}(\boldsymbol{\beta}\vert\mathbf{D}, \tilde{\mathbf{Y}}, \hat{\sigma}^2, \mathbf{p})\}$	$M \times (K+1)$
'MMA-S'		$M \times \{\beta_1, \ldots, \beta_K, \mathcal{L}(\boldsymbol{\beta}\vert\mathbf{D}, \tilde{\mathbf{Y}}, \hat{\sigma}^2, \mathbf{p})\}$	$M \times (K+1)$
'BMA'	'1'	$M \times \{\beta_1, \ldots, \beta_K, \sigma, \mathcal{L}(\boldsymbol{\beta}\vert\mathbf{D}, \tilde{\mathbf{Y}})\}$	$M \times (K+2)$
'BMA'	'2'	$M \times \{\beta_1, \ldots, \beta_K, \sigma_1, \ldots, \sigma_K, \mathcal{L}(\boldsymbol{\beta}\vert\mathbf{D}, \tilde{\mathbf{Y}})\}$	$M \times (2K+1)$
'BMA'	'3'	$M \times \{\beta_1, \ldots, \beta_K, c, \mathcal{L}(\boldsymbol{\beta}\vert\mathbf{D}, \tilde{\mathbf{Y}})\}$	$M \times (K+2)$
'BMA'	'2'	$M \times \{\beta_1, \ldots, \beta_K, c_1, \ldots, c_K, \mathcal{L}(\boldsymbol{\beta}\vert\mathbf{D}, \tilde{\mathbf{Y}})\}$	$M \times (2K+1)$

For EWA, BGA, AICA, BICA and GRA, the output argument `x` is equivalent to a horizontal vector with K values of the weights, $\{\beta_1, \ldots, \beta_K\}$. The entries of `x` correspond to the different columns of input argument `D`. Thus, entry k of `x` stores the weight of the kth column (ensemble member) of matrix `D`.

If MMA or MMA$^{\Delta}$ are used then output argument `x` consist of a $M \times (K+1)$ matrix with M posterior samples of the MMA or MMA$^{\Delta}$ weights and their corresponding values of the log-likelihood of Equation (23). These M solutions summarize the posterior of the MMA or MMA$^{\Delta}$ weights and can be used (among others) to analyze the uncertainty of the MMA weights and point forecasts.

The maximum likelihood values of the MMA and MMA$^{\Delta}$ weights, $\hat{\beta}_{\text{MMA}}$ or $\hat{\beta}_{\text{MMA}^{\Delta}}$ are stored separately in the field ML of output argument output (more of which later). Their values can also be derived from `x` by locating the row number of this matrix with largest value of the log-likelihood of Equation (23). This

row of x can be located with the MATLAB command

```
[ na , row_max ] = max(x(:,K+1))
```
(25)

and thus the maximum likelihood values of the MMA or MMA$^{\Delta}$ weights are equivalent to

```
beta_MMA = x(row_max,1:K)
```
(26)

Finally, if BMA is used for postprocessing of the forecast ensemble, then output argument x is equivalent to a $M \times (d+1)$ matrix with M posterior samples of the d parameters of the BMA mixture distribution, and corresponding value of the log-likelihood of Equation (16). The first K columns of x list the values of the BMA weights of each member of the ensemble. The content of columns $K+1$ to d of x depends on the assumed properties of the members' forecast distribution defined by the user in field VAR of structure options. For instance, if options.VAR = '1' then all K members of the ensemble are assumed to have the same standard deviation of their forecast distribution, hence $d = K+1$, and column $K+1$ of x lists the posterior values of σ. If options.VAR = '2' then each members' forecast distribution has a different standard deviation, $d = 2K$, and columns $K+1$ to d store the posterior values of $\{\sigma_1, \ldots, \sigma_K\}$. Table 3 summarizes the content of each column of x for the different settings of field VAR of structure options.

The maximum likelihood values of the parameters of the BMA mixture distribution are stored in field ML of structure options. The user can also derive these values from output argument x by locating the (posterior) sample of this matrix with largest value of the log-likelihood. The recipe of how to do this in MATLAB was given in Equations (25) and (26).

3.6.2. Return argument output

The second output argument of the function MODELAVG is a structure array called output and stores important information about the performance of each model averaging method, and (if appropriate) convergence properties of the DREAM algorithm. Most of the fields of structure output are only defined if MMA, MMA$^{\Delta}$ or BMA are used (see Table 4).

Table 4: Content of the return argument `output` of the MODELAVG function

Field `output`	Description	Content
Ye	Deterministic point forecast, Equation (4)	$n \times 1$ vector
RMSE	Root mean square error point forecast	scalar
R	Cross-correlation of point forecasts and training data	scalar
RMSE_mod	Root mean square errors ensemble members	$1 \times K$ vector
RunTime	Elapsed time	scalar
	MMA and MMA$^{\Delta}$ methods	
ML	Maximum likelihood values of weights	$1 \times K$ vector
loglik	Maximum value of log-likelihood Equation (23)	scalar
std	Posterior standard deviation weights	$1 \times K$ vector
corr	Posterior correlation coefficients of weights	$K \times K$ matrix
	BMA method	
ML	Maximum likelihood BMA model parameters	$1 \times d$ vector
loglik	Maximum value of log-likelihood Equation (16)	scalar
corr	Posterior correlation coefficients BMA parameters	$d \times d$ matrix
std	Posterior standard deviation BMA parameters	$1 \times d$ vector
pred	Prediction intervals mixture distribution	$n \times 2$ matrix
coverage	Coverage of prediction intervals	scalar
spread	Average width of prediction intervals	scalar
	BMA, MMA and MMA$^{\Delta}$ (DREAM diagnostics)	
MR_stat	Multivariate scale-reduction factor	$N \times 2$ matrix
R_stat	Univariate scale-reduction factor	$N \times (d+1)$ matrix
AR	Acceptance rate (%) of proposals	$N \times 2$ matrix

The field Ye of `options` stores the weighted forecast of each model averaging method. This $n \times 1$ vector is derived from Equation (4) using the (maximum likelihood) weights of each averaging method and ensemble forecasts of input argument `D`. The fields RMSE and R of `output` (both scalars) summarize the performance of the averaged forecast using the root mean square error and Pearson's correlation coefficient, respectively. The field RMSE_mod is a $1 \times K$ vector with RMSEs of the individual members of the ensemble, and the field RunTime (scalar) of `output` stores the CPU-time of each method.

If MMA or MMA$^{\Delta}$ are used then the structure `output` contains several other fields. The fields ML (vector), and loglik (scalar) store the maximum likelihood values of the MMA or MMA$^{\Delta}$ weights and corresponding value of the log-likelihood function of Equation (23), respectively. The fields std (vector) and corr (matrix) list the posterior standard deviations and correlation coefficients of the weights.

If the BMA method is used then structure `options` returns to the user several more fields that summarize the performance of the BMA mixture distribution. The field pred (matrix) lists the prediction intervals of the BMA forecast distribution, and coverage and spread (both scalars) store the coverage and spread of these intervals, respectively. The statistical significance of the prediction intervals is defined by the user in field alpha of input argument `options`.

Diagnostic information about the performance of the DREAM algorithm can be found in fields MR_stat (matrix), R_stat (matrix) and AR (matrix) of the return argument `output`. These fields are only defined

if MMA, MMA$^{\Delta}$, or BMA are used as these three methods use MCMC simulation with DREAM to find the maximize likelihood values of the weights (MMA and MMA$^{\Delta}$) or weights and standard deviation(s) of the members forecast distribution (BMA) The fields R_stat and MR_stat list the values of the $\hat{R}$ and $\hat{R}^d$ convergence diagnostics of *Gelman and Rubin* (1992) and *Brooks and Gelman* (1998) at different iterations, respectively. Field AR of `output` stores the acceptance rate of DREAM. The MATLAB command

```
plot(output.R_stat(:,1),output.R_stat(:,2:end))
```
(27)

plots the evolution of the $\hat{R}$ convergence diagnostic of the weights of each member of the ensemble. If BMA is used then this plot includes as well the standard deviation of the members' forecast distribution. The results in this graph can be used to judge when convergence of DREAM has been achieved and thus which samples to return in output argument `x`.

The MODELAVG toolbox not only returns to the user the variables `x` and `output` but also generates graphical output. These figures display many of the input and output variables of the toolbox, and include (i) a time series plot of the forecast ensemble with averaged forecast and verifying observations, (ii) a plot of the predictions intervals of the BMA model and corresponding observations, (iii) histograms and bivariate scatter plots of the posterior samples of DREAM, (iv) trace plots of the Markov chains sampled by DREAM, (v) trace plots of the convergence diagnostics of DREAM, and (vi) a quantile-quantile plot of the residuals of the weighted-average forecast. The main results of the toolbox are also written to the file "MODELAVG_output.txt" in the MATLAB editor. The case study section provides a screen shot of the content of this file.

3.7. Evaluation data set: structure `val`

The output variables returned by MODELAVG apply to the training data set only. The built-in script MODELAVG_EVAL allows the user to calculate performance statistics for the evaluation data set. This function can be executed as follows

```
[val] = MODELAVG_eval(method,D_eval,Y_eval,options,a,b,output)
```
(28)

where `method` and `options` are defined by user prior to running the main function, MODELAVG of the MODELAVG toolbox, `a` and `b` are $1 \times K$ vectors with intercepts and slopes of the K ensemble members derived from linear bias correction of the training data set using Equation (2), `output` is the return argument of the MODELAVG function, and `D_val` ($m \times K$ matrix) and `Y_val` ($m \times 1$ vector) signify the ensemble forecasts and corresponding observations of the evaluation data set, respectively. The fields of the return argument val are listed in table 5.

Table 5: Content of the argument `val` computed by the function MODELAVG_EVAL after the MODELAVG toolbox has returned the output arguments `x` and `output`

Field `val`	Description	Content
Ye	Deterministic point forecast, Equation (4)	$m \times 1$ vector
RMSE	Root mean square error point forecast	scalar
R	Cross-correlation of point forecasts and training data	scalar
RMSE_mod	Root mean square errors ensemble members	$1 \times K$ vector
RunTime	Elapsed time	scalar
	BMA method	
pred	Prediction intervals mixture distribution	$m \times 2$ matrix
coverage	Coverage of prediction intervals	scalar
spread	Average width of prediction intervals	scalar

Thus, the structure `val` stores metrics for the evaluation data set. This data set is used to evaluate the performance of each model averaging method for an independent data set. The structure `val` can be computed after the main function MODELAVG has returned its output (with/without screen output).

4. Numerical examples

I now illustrate the main functionalities of the MODEAVG package by application to multi-model forecast ensembles of river discharge, surface temperature and sea level pressure, respectively.

4.1. Case Study 1: The rainfall-runoff transformation

The first case study involves an eight-member ensemble of calibrated watershed models of the Leaf River, near Collins, Mississippi. This discharge ensemble was created by *Vrugt and Robinson* (2007) and used to evaluate the sharpness and coverage of the BMA forecast distribution. Figure 4 provides a snapshot of the model ensemble for a portion of the training data set.

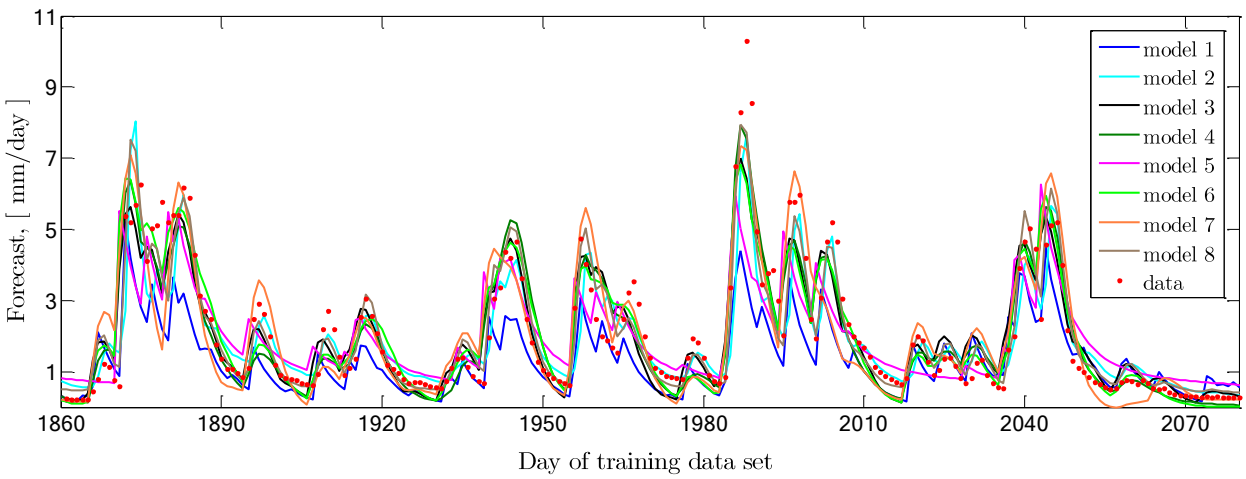


Figure 4: Streamflow predictions of the eight individual models of the ensemble for a representative portion of the calibration period. The red dots represent the verifying observations.

The spread of the ensemble is sufficient and generally brackets the observations (red dots). The calibrated models appear to provide somewhat different forecasts. This is a necessary requirement for an ensemble prediction system, otherwise model averaging cannot improve the forecast skill and distribution.

The following script file (.m) summarizes the setup of case study 1 in the MODELAVG toolbox.

MODELAVG INPUT FILE FOR CASE STUDY 1: The BMA method is used (`method = 'bma'`) with a gamma conditional pdf (`options.PDF` = 'gamma') for the members' forecast distribution. As the field `VAR` of options is set to `'3'`, the standard deviation of the gamma distribution is assumed to be heteroscedastic, or, $\sigma_{jk} = cD_{jk}$, where the value of the multiplier c applies to all models of the ensemble. The discharge forecasts of the ensemble members and verifying observations are stored in the ascii-file "discharge.txt". This data file is unpacked in the $n \times K$ matrix `D` and $n \times 1$ vector `Y`. Bias correction is used of the forecasts of each member using the linear regression model of Equation (2).

```
% ------------------------------------------------------------------------------------------ %
%                                                                                            %
% MM        MM   OOOOOOO   DDDDDDDD   EEEEEEEE  LL            AAA      VV        VV  GGGGGGGG %
% MMM       MM  OOOOOOOOO  DDDDDDDDD  EEEEEEEE  LL           AA AA     VV        VV  GGG   GGG %
% MMMM    MMMM  OO     OO  DD     DD  EE        LL           AA AA      VV      VV   GG     GG %
% MM MM MM MM  OO     OO  DD     DD  EEEEE     LL          AA   AA     VV      VV   GGG   GGG %
% MM  MMM  MM  OO     OO  DD     DD  EEEEE     LL         AAAAAAAAA     VV    VV    GGGGGGGG %
% MM        MM  OO     OO  DD     DD  EE        LL         AA     AA      VV VV           GG %
% MM        MM  OOOOOOOOO  DDDDDDDDD  EEEEEEEE  LLLLLLLL  AA       AA     VV VV      GGGGGGG %
% MM        MM   OOOOOOO   DDDDDDDD   EEEEEEEE  LLLLLLLL  AA       AA      VVV      GGGGGGGG %
%                                                                                            %
% ------------------------------------------------------------------------------------------ %

%% CASE STUDY I: RAINFALL-RUNOFF TRANSFORMION - ENSEMBLE OF CALIBRATED WATERSHED MODELS
%% PLEASE CHECK: J.A. VRUGT AND B.A. ROBINSON, WRR, 43, W01411, doi:10.1029/2005WR004838,
    2007

%% DEFINE MODEL AVERAGING METHOD
method = 'bma';             % 'ewa'/'bga'/'aica'/'bica'/'gra'/'bma'/'mma'/'mma-s'

%% BMA -> CONDITIONAL DISTRIBUTION NEEDS TO BE DEFINED
options.PDF = 'gamma';     % normal conditional pdf
options.VAR = '3';          % common constant variance
options.alpha = 0.95;       % prediction intervals of BMA model (0.90/0.95/0.99)
options.print = 'yes';      % print output (figures, tables) to screen

%% NOW LOAD DATA
S = load('discharge.txt');  % daily discharge forecasts (mm/day) of models and verifying data
T_idx = [ 1:1:3000 ];       % start/end training period

%% DEFINE ENSEMBLE AND VECTOR OF VERYFYING OBSERVATIONS
D = S(T_idx,1:8); Y = S(T_idx,9);

%% APPLY LINEAR BIAS CORRECTION TO ENSEMBLE ( UP TO USER )
[ D , a , b ] = Bias_correction ( D , Y );

%% NUMBER OF PARAMETERS OF EACH MODEL (ABC/GR4J/HYMOD/TOPMO/AWBM/NAM/HBV/SACSMA)
options.p = [ 3 4 5 8 8 9 9 13 ];   % ( only used for AICA, BICA, MMA, MMA-S)

%% NOW EXECUTE THE MODELAVG TOOLBOX
[ beta , output ] = MODELAVG ( method , D , Y , options );
```

The BMA method is used for postprocessing of the discharge forecast ensemble. A gamma forecast distribution is used for each ensemble members' conditional pdf. The standard deviation of this distribution is assumed to be heteroscedastic with coefficient c that applies to all models of the ensemble. This thus requires the inference of $d = K + 1$ parameters with DREAM, namely the values of the K weights of the watershed models, and multiplier c, or $\mathbf{x} = \{\beta_1, \ldots, \beta_K, c\}$ (see Table 3).

Figure 5 presents histograms of the marginal posterior distribution of the weights of each model of the ensemble. The maximum likelihood values are separately indicated with the "×" symbol.

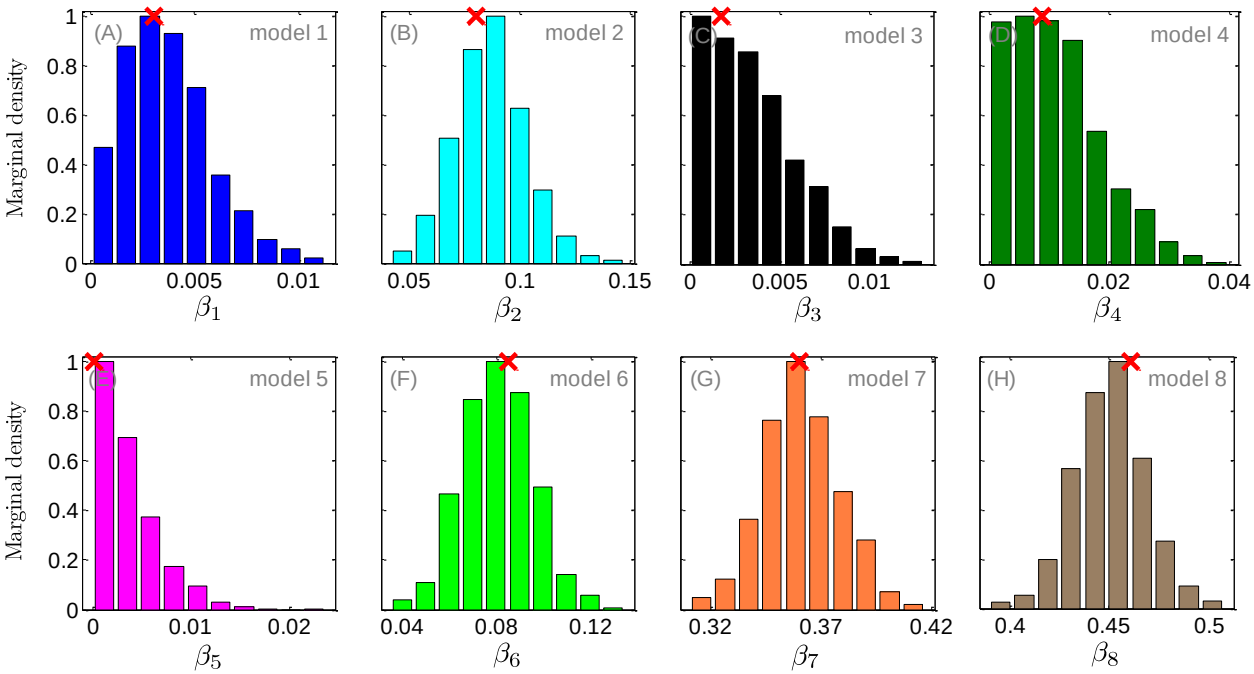


Figure 5: Histograms of the marginal posterior distribution of the weights of each watershed model of the discharge ensemble. The corresponding forecasts of each model can be found in Figure 4. The red cross symbol in each plot indicates the maximum likelihood solution.

Most histograms are well described with a normal distribution, except those of models with a very low weight truncated at zero to lie on the unit simplex. The posterior uncertainty of the models' weights appears rather small as most histograms are well defined. Models 1, 3, 4, and 5 receive negligible weights and can thus be removed from the ensemble without much loss of forecast skill. The marginal distributions of the weights are particularly useful to determine the importance of each members forecasts.

To understand how the BMA posterior parameter uncertainty translates into predictive uncertainty, please consider Figure 6 that presents the 95% hydrograph prediction uncertainty ranges (gray region) of the BMA mixture distribution for a portion of the training data period. The mean forecast of the BMA model is separately indicated with the black line, and the red dots denote the verifying observations.

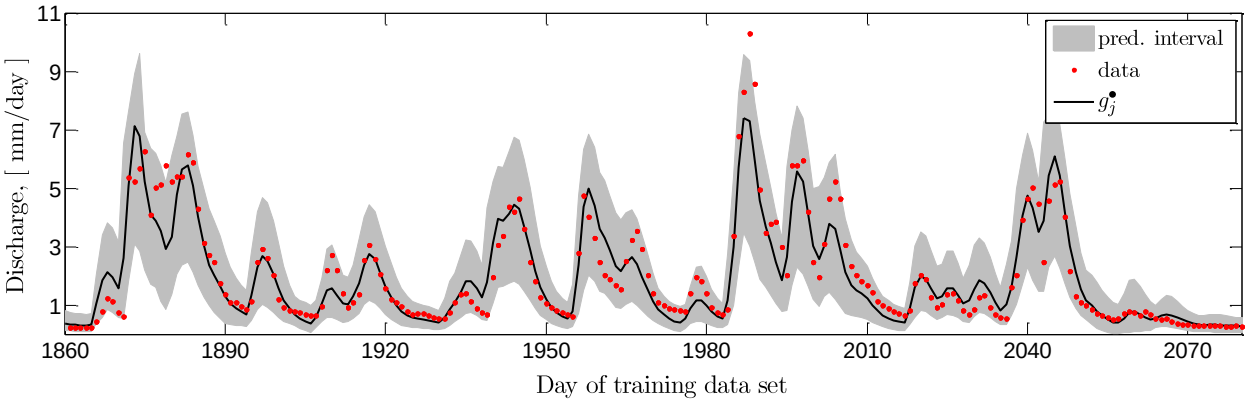


Figure 6: 95% prediction intervals (gray region) of the BMA predictive density for a representative portion of the 3000 days calibration period. The black line displays the mean point forecast of the BMA model derived from Equation 14), whereas the red dots signify the verifying observations.

The 95% prediction uncertainty ranges of the BMA model appear rather large, particularly at lower discharge values, yet envelop almost 95% observations (data appears in Table 6). The RMSE of the BMA point forecast (mean of the mixture density) is about 0.723 mm/day, and nearly similar as the value of 0.720 derived from the best model of the ensemble (= Sacramento soil moisture accounting model).

Table 6 summarizes the results of the BMA method for two different forecast distributions (`options`.PDF = 'normal' or 'gamma') and four different treatments of the variance of this distribution (`options`.VAR = '1', '2', '3' or '4'). For convenience, I list only the maximum likelihood values of the weights of each implementation. Values listed in parentheses denote the posterior standard deviation derived from the DREAM sample. I also report the maximum value of the log-likelihood of Equation (16), the RMSE (mm/day) of the averaged forecast, coverage (%) and spread (mm/day) of the 95% prediction intervals of the BMA model.

Table 6: BMA results for the discharge ensemble of the Leaf River watershed using the normal and gamma forecast distribution and four different treatments of the variance of these distributions (see section 2.5.2). The different columns list the maximum likelihood values of the BMA model parameters, corresponding value of the log-likelihood function of Equation (16), RMSE (mm/day), coverage (%) and spread (mm/day) of the 95% prediction intervals of the BMA model.

BMA	Normal distribution				Gamma distribution			
	'1'	'2'	'3'	'4'	'1'	'2'	'3'	'4'
β_1	0.0169	0.0382	0.0022	0.0038	0.1087	0.0369	0.0026	0.0027
β_2	0.2005	0.1631	0.0882	0.0891	0.1859	0.1431	0.0852	0.0857
β_3	0.1041	0.0080	0.0059	0.0109	0.0180	0.1107	0.0025	0.0010
β_4	0.0701	0.1129	0.0271	0.0246	0.0022	0.0433	0.0054	0.0134
β_5	0.0330	0.0317	0.0002	0.0004	0.0633	0.0443	0.0007	0.0037
β_6	0.0444	0.1303	0.1228	0.1582	0.0076	0.0987	0.0818	0.0841
β_7	0.0423	0.1401	0.3226	0.3024	0.3342	0.2297	0.3663	0.3674
β_8	0.4887	0.3757	0.4309	0.4108	0.2801	0.2932	0.4555	0.4422
σ	0.4724				0.4013			
c			0.317				0.3548	
σ_1		0.5277				0.3348		
σ_2		0.1587				0.0854		
σ_3		5.4986				0.7936		
σ_4		1.0512				2.7103		
σ_5		0.1827				0.0489		
σ_6		0.0722				0.0470		
σ_7		0.0830				0.0882		
σ_8		0.1125				0.0959		
c_1				0.3023				0.3360
c_2				0.2379				0.2910
c_3				0.3737				0.9083
c_4				0.1975				0.1234
c_5				0.2974				0.5990
c_6				0.4333				0.3868
c_7				0.3096				0.3610
c_8				0.2992				0.3495
$\mathcal{L}(\mathbf{x}\|\mathbf{D}, \tilde{\mathrm{Y}})$	-2416.3	-617.92	150.93	164.95	-2965.6	-831.63	244.75	249.28
RMSE	0.7072	0.7210	0.7234	0.7258	0.7499	0.7332	0.7237	0.7245
Coverage	96.133	94.767	95.600	95.667	95.867	94.633	96.033	95.967
Spread	2.2263	2.0809	1.4444	1.4589	2.2217	1.4665	1.4493	1.4465

The maximum likelihood values of the BMA weights depend somewhat on the assumed forecast distribution of the deterministic prediction of each model. The HBV (model 7) and SAC-SMA (model 8) models almost always receive the highest weights of the ensemble and their forecasts are thus of crucial importance for the BMA model. The TOPMO model (model 6) receives particularly low weights, despite it having the second lowest RMSE for the 3000-day raining data period. The TOPMO forecasts might be redundant as they are correlated with other models of the ensemble. Note that the performance of the BMA model appears to be much more affected by the standard deviation of the forecast distribution, than the shape of this distribution (normal or gamma). The best results are obtained if a heteroscedastic standard deviation is assumed for the members forecast distribution. This is perhaps not surprising as the measurement error

of the discharge data is known to increase with flow level. I refer interested readers to the paper of *Vrugt and Robinson* (2007) and *Rings et al.* (2012) for a more detailed analysis of the BMA results, and a comparison with data assimilation methods.

4.2. Case Study 2: 48 hour forecasting of sea level temperature

The second case study involves a five-member multianalysis ensemble of 48-h forecasts of surface temperature (in Kelvin) between January and June 2000 in the Pacific Northwest of the USA. The data set was created by the mesoscale short-range ensemble system of the University of Washington (UW) (*Grimit and Mass*, 2002) using different runs of the fifth-generation Pennsylvania State University - National Center for Atmospheric Research Mesoscale Model (MM5) and initial conditions from different operational centers.

I use a 25-day period between April 16 and 9 June 2000 for BMA model calibration. This involves a total of $n = 14668$ temperature forecasts at different locations in the Pacific Northwest. For some days the data were missing, so that the number of calendar days spanned by the training data set is larger than the number of days of training used.

Figure 7 displays the ensemble forecasts, $\mathbf{D}$ and verifying observations, $\tilde{\mathbf{Y}}$ for a small portion of the training data set.

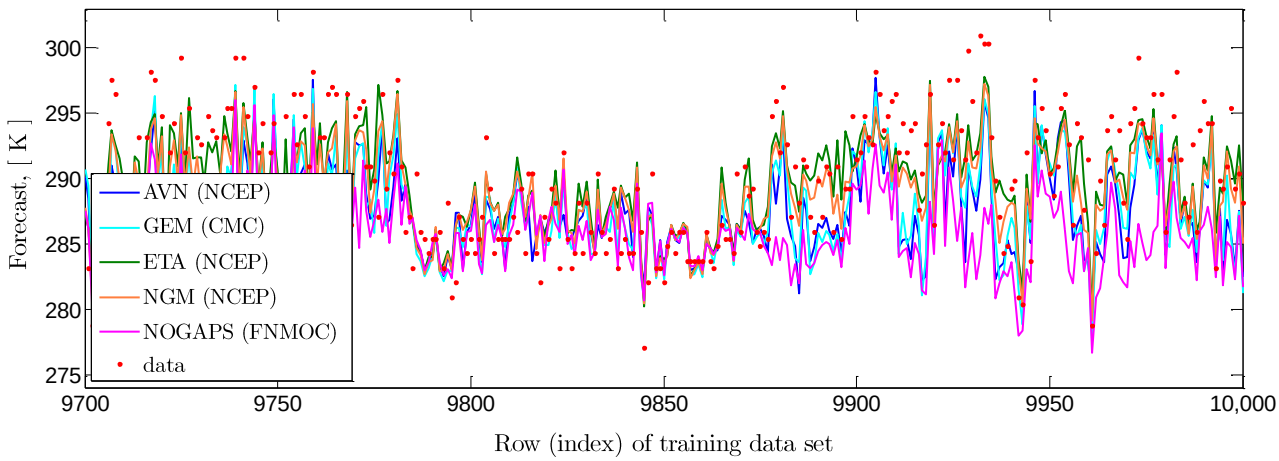


Figure 7: Temperature forecasts for the Pacific Northwest of the five-member UW ensemble for a short period of the training data set. The red dots represent the verifying observations.

The different members of the ensemble issue different 48-h temperature forecasts, yet the spread is not always sufficient to bracket the observations (red dots). Note, that *Raftery et al.* (2005) used the exact same 25-day data set to introduce and benchmark the BMA method.

The MATLAB script (.m file) listed below details the setup of case study 2 in the MODELAVG toolbox.

MODELAVG INPUT FILE FOR CASE STUDY 2: The BMA method is used (`method = 'bma'`) with a normal conditional pdf (`options.PDF = 'normal'`) for the members' forecast distribution. As the field `VAR` of options is set to `'2'`, each model is assumed to have a different standard deviation of the forecast distribution, but this standard deviation is constant. The surface temperature forecasts and verifying observations are stored in the ascii-file "temp.txt". This data file is unpacked in the $n \times K$ matrix `D` and $n \times 1$ vector `Y`. Bias correction is used of the forecasts of each member using the linear regression model of Equation (2).

```
% ------------------------------------------------------------------------------------------ %
%                                                                                            %
% MM       MM   OOOOOOO   DDDDDDDD   EEEEEEEE  LL            AAA      VV       VV  GGGGGGGG %
% MMM      MM  OOOOOOOOO  DDDDDDDDD  EEEEEEEE  LL           AA AA     VV       VV  GGG  GGG %
% MMMM   MMMM  OO     OO  DD     DD  EE        LL           AA AA      VV     VV   GG    GG %
% MM MM MM MM  OO     OO  DD     DD  EEEEE     LL          AA   AA     VV     VV   GGG  GGG %
% MM  MMM  MM  OO     OO  DD     DD  EEEEE     LL         AAAAAAAAA     VV   VV    GGGGGGGG %
% MM       MM  OO     OO  DD     DD  EE        LL         AA     AA      VV VV           GG %
% MM       MM  OOOOOOOOO  DDDDDDDDD  EEEEEEEE  LLLLLLLL  AA       AA     VV VV      GGGGGGG %
% MM       MM   OOOOOOO   DDDDDDDD   EEEEEEEE  LLLLLLLL  AA       AA      VVV      GGGGGGGG %
%                                                                                            %
% ------------------------------------------------------------------------------------------ %

%% CASE STUDY II: 48-FORECASTS OF SEA SURFACE TEMPERATURE
%% PLEASE CHECK: A.E. RAFTERY ET AL., MWR, 133, pp. 1155-1174, 2005.

%% DEFINE MODEL AVERAGING METHOD
method = 'bma';             % 'ewa'/'bga'/'aica'/'bica'/'gra'/'bma'/'mma'/'mma-s'

%% BMA -> CONDITIONAL DISTRIBUTION NEEDS TO BE DEFINED
options.PDF = 'normal';     % pdf predictor: normal/heteroscedastic/gamma
options.VAR = '2';          % individual non-constant variance
options.alpha = 0.95;       % prediction intervals of BMA model (0.90/0.95/0.99)
options.print = 'yes';      % print output (figures, tables) to screen

%% NOW LOAD DATA
T = load('temp.txt');       % 48-hour forecasts temperature (Kelvin) and verifying
    observations

%% DEFINE ENSEMBLE AND VECTOR OF VERYFYING OBSERVATIONS ( APRIL 16 TO JUNE 9, 2000 )
idx = find(T(:,1) == 2000 & T(:,2) == 4 & T(:,3) == 16); start_idx = idx(1);
idx = find(T(:,1) == 2000 & T(:,2) == 6 & T(:,3) == 9); end_idx = idx(end);
D = T(start_idx:end_idx,5:9); Y = T(start_idx:end_idx,4);

%% APPLY LINEAR BIAS CORRECTION TO ENSEMBLE ( UP TO USER )
[ D , a , b ] = Bias_correction ( D , Y );

%% NOW EXECUTE THE MODELAVG TOOLBOX
[ beta , output ] = MODELAVG ( method , D , Y , options );
```

I implement the BMA for the 25-day temperature ensemble and assume a normal distribution for the members forecast distribution. The choice of this forecast distribution is supported by the frequency distribution of the temperature observations of the training data set (see also Figure 3A). As temperature

observations have a constant measurement error, we assume that the forecast distribution has a fixed variance, but allow this variance to vary among the ensemble members. This thus involves the inference of $d = 2K = 10$ parameters with DREAM, namely the weight and standard deviation of each modelsâĂŹ forecast distribution. Appendix E presents the screen output of the main function of the MODELAVG toolbox for the data set and BMA model defined in the MATLAB input file above.

Figure 8 presents trace plots of the $\hat{R}$-statistic of *Gelman and Rubin* (1992) for the DREAM sampled weights and standard deviations of each members' conditional distribution.

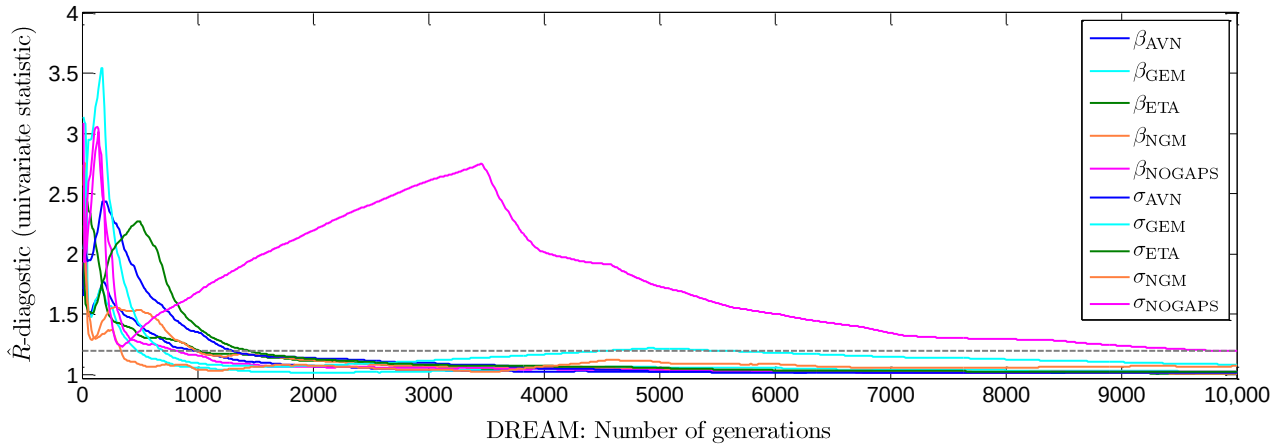


Figure 8: Evolution of the $\hat{R}$ scale reduction factor of *Gelman and Rubin* (1992) used to diagnose convergence of the sampled Markov chains of DREAM to a limiting distribution. The dashed grey horizontal line signifies the commonly used threshold for convergence.

The $\hat{R}$ diagnostic illustrates that about $10,000$ generations are required with DREAM to reach convergence to a stationary distribution $\hat{R} \leq 1.2; \forall i = \{1, \ldots, d\}$. This constitutes a relative large number of iterations and is explained by bimodality of the posterior distribution of the standard deviation of the forecast distribution of ETA. This is demonstrated graphically in Figures E17 and E18 in Appendix E.

Figure 9 presents marginal distributions of the posterior samples of the BMA weights (top panel) and standard deviations (bottom panel) of the forecast distribution. The maximum likelihood values are separately indicated with the red cross "×" symbol.

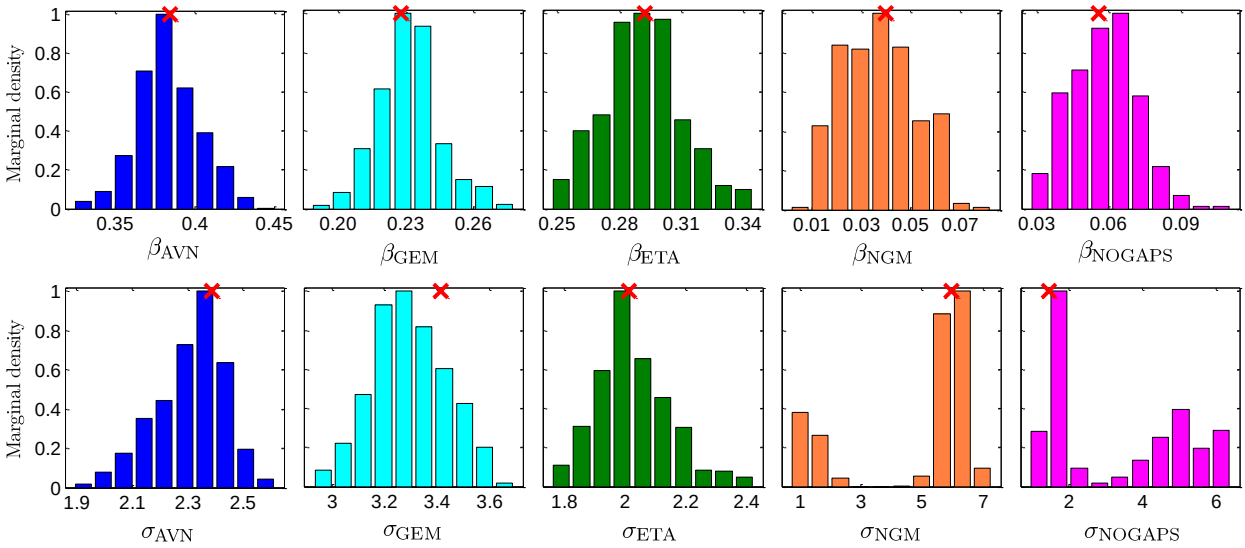


Figure 9: Histograms of the marginal posterior distribution of the weights and standard deviations of the five models of the ensemble. The maximum likelihood solution of each BMA model parameter is indicated with the red cross.

All histograms appear approximately Gaussian, and the median solution of each weight and standard deviation coincides almost perfectly with their respective maximum likelihood values. Note the presence of bimodality in the marginal distribution of the standard deviation of the forecast distribution of the NGM and NOGAPS models. This bimodality is particularly obvious for σ_{NGM} (two disconnected modes) and makes it much more difficult to MCMC methods to sample the target distribution (*Vrugt*, 2016). Hence, this explains why DREAM needs a relatively large number of function evaluations to convergence to the posterior distribution. In fact, NGM and NOGAPS, receive much lower weights than the order three models of the ensemble, and can perhaps be discarded without affecting too much the predictive skill and coverage of the BMA model.

Figure 10 presents the 95% prediction uncertainty of the maximum likelihood BMA mixture distribution. The deterministic point forecast of the BMA model is separately indicated with the solid black line. This point predictor is derived from Equation (14) using the forecast ensemble, $\mathbf{D}$ and maximum likelihood values of the weights, $\hat{\beta}_{\mathrm{BMA}}$.

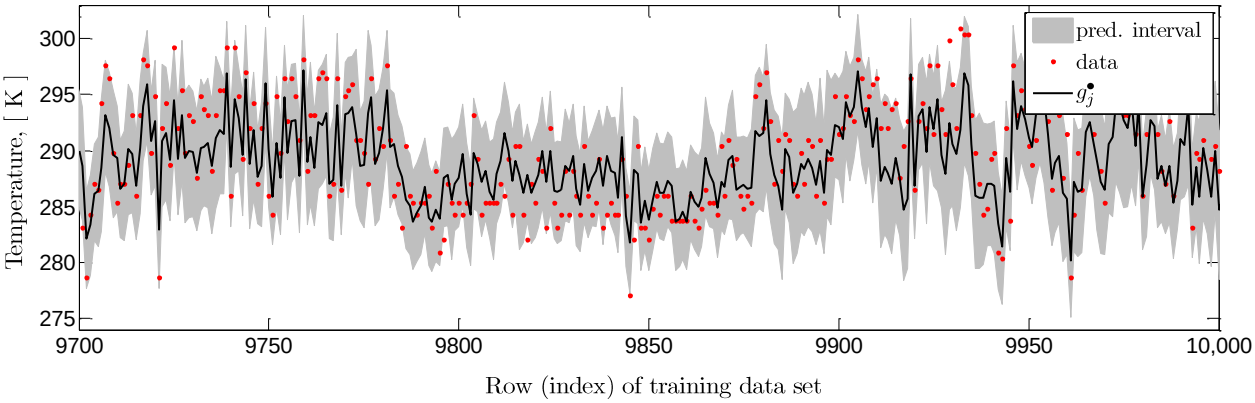


Figure 10: 95% prediction intervals (gray region) of the BMA predictive density for a representative portion of the 25-day training data set. The black line displays the mean point forecast of the BMA model derived from Equation (14), whereas the red dots signify the verifying observations.

The 95% prediction uncertainty ranges of the BMA model appear rather large with average spread of about 9.7 Kelvin and coverage of 90.7% of the observations. The RMSE of the BMA point forecast (mean of the mixture density) is approximately 2.96 K which is equivalent to the RMSE of the best model (AVN) of the ensemble (see Table 7)

Table 7 lists the (maximum likelihood) values of the weights for each of the model averaging methods of the MODELAVG toolbox.

Table 7: Results of different model averaging methods for the five-member multimodel ensemble of 48-h forecasts of surface temperature in the Pacific Northwest of the USA. The first two columns list the names of each model of the ensemble and their corresponding RMSE values (in Kelvin), respectively. Subsequent columns list the (maximum likelihood) values of the weights of each model averaging method of the toolbox. The bottom part of the Table lists the RMSE of the deterministic forecast of each method. This point forecast is calculated with Equation (4) using the ensemble forecasts of **D** and (maximum likelihood) weights.

Model	RMSE	EWA	BGA	AICA†	BICA†	GRA	BMA	MMA†	MMA$^{\Delta}$ †,‡
AVN	3.0578	0.2000	0.2208	1.0000	1.0000	0.4836	0.3896	0.4798	0.4730
GEM	3.3425	0.2000	0.1848	0.0000	0.0000	0.2028	0.2277	0.2002	0.1984
ETA	3.1143	0.2000	0.2129	0.0000	0.0000	0.3602	0.2877	0.3596	0.3119
NGM	3.1881	0.2000	0.2031	0.0000	0.0000	-0.0692	0.0442	-0.0712	0.0000
NOGAPS	3.4020	0.2000	0.1784	0.0000	0.0000	0.0227	0.0508	0.0316	0.0166
Point		2.9941	2.9886	3.0578	3.0578	2.9541	2.9587	2.9542	2.9548

†: I assume `p = 20*ones(1,K)` (twenty parameters assigned to each model)
‡: Presence of numerous local optima on likelihood surface

Information criterion averaging (AICA and BICA) assigns the AVN model a weight of unity, whereas all other models are given a zero weight. The BMA method distributes the weights more equally among the different ensemble members, yet assigns the NGM and NOGAPS members relatively low weights. Discarding these two models hardly affects the performance of each of the model averaging methods.

The point forecasts of the different model averaging methods exhibit a rather similar performance. The main advantage of the BMA method, however is that it provides a forecast distribution which can be used

for probabilistic analysis and prediction. Nevertheless, if point forecasting is of main concern, the Granger-Ramanathan averaging provides the lowest RMSE of the deterministic forecast at a negligible CPU-cost.

4.3. Case Study 3: 48 hour forecasting of sea surface pressure

I now do a similar analysis but using 48-h forecasts of sea surface pressure (in hPa) from the University of Washington mesoscale short-range ensemble prediction system (*Grimit and Mass*, 2002). The same 25-day training period as in case study 2 is used for BMA model calibration (April 16 - June 9, 2000). For some days the data were missing, so that the number of calendar days spanned by the training data set is larger than the number of days of training used. The training data set includes $n = 4013$ observations.

Figure 11 displays the ensemble forecasts of the sea level pressure, $\mathbf{D}$ and verifying observations, $\tilde{\mathbf{Y}}$ for a small portion of the training data set.

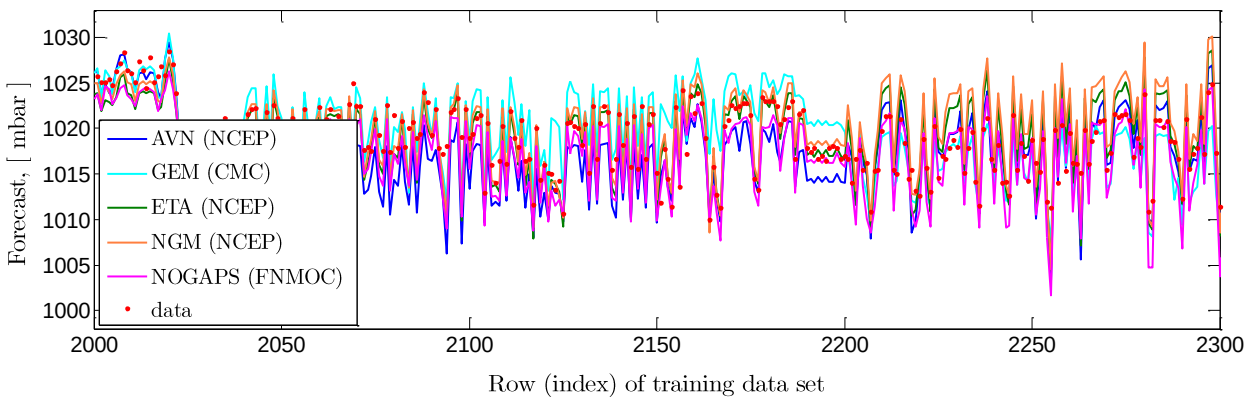


Figure 11: 48-h sea level pressure forecasts for the Pacific Northwest of the five member UW ensemble. I only display a small portion of the training data set. The red dots represent the verifying observations.

The models differ in their 48-h pressure forecasts, and the ensemble covers a large majority of the observed sea level pressure data (red dots). Note, that *Raftery et al.* (2005) used the exact same 25-day data set to introduce and benchmark the BMA method.

The MATLAB script (.m file) listed below details the setup of case study 3 in the MODELAVG toolbox.

MODELAVG INPUT FILE FOR CASE STUDY 3: The BMA method is used (`method = 'bma'`) with a normal conditional pdf (`options.PDF` = 'normal') for the members' forecast distribution. As the field `VAR` of options is set to `'2'`, each model is assumed to have a different standard deviation of the forecast distribution, but this standard deviation is constant. The sea pressure forecasts are verifying observations are stored in the ascii-file "pressure.txt". The built-in function `find` is used to extract the temperature forecasts of April 16 to June 9. This data is unpacked in the $n \times K$ matrix `D` and $n \times 1$ vector `Y`. Bias correction is used of the forecasts of each member using the linear regression model of Equation (2).

```
% ------------------------------------------------------------------------------------------ %
%                                                                                            %
% MM        MM   OOOOOOO   DDDDDDDD   EEEEEEEE  LL            AAA      VV        VV  GGGGGGGG %
% MMM       MM  OOOOOOOOO  DDDDDDDDD  EEEEEEEE  LL           AA AA     VV        VV  GGG   GGG %
% MMMM    MMMM  OO      OO  DD      DD  EE        LL           AA AA      VV      VV   GG     GG %
% MM MM MM MM  OO      OO  DD      DD  EEEEE     LL          AA   AA     VV      VV   GGG   GGG %
% MM  MMM  MM  OO      OO  DD      DD  EEEEE     LL         AAAAAAAAA     VV    VV    GGGGGGGG %
% MM        MM  OO      OO  DD      DD  EE        LL         AA     AA      VV VV           GG %
% MM        MM  OOOOOOOOO  DDDDDDDDD  EEEEEEEE  LLLLLLLL  AA       AA     VV VV      GGGGGGG %
% MM        MM   OOOOOOO   DDDDDDDD   EEEEEEEE  LLLLLLLL  AA       AA      VVV      GGGGGGGG %
%                                                                                            %
% ------------------------------------------------------------------------------------------ %

%% CASE STUDY III: 48-FORECASTS OF SEA SURFACE PRESSURE
%% PLEASE CHECK: A.E. RAFTERY ET AL., MWR, 133, pp. 1155-1174, 2005.

%% DEFINE MODEL AVERAGING METHOD
method = 'bma';             % 'ewa'/'bga'/'aica'/'bica'/'gra'/'bma'/'mma'/'mma-s'

%% BMA -> CONDITIONAL DISTRIBUTION NEEDS TO BE DEFINED
options.PDF = 'gamma';      % gamma distribution
options.VAR = '4';          % individual non-constant variance
options.alpha = 0.95;       % prediction intervals of BMA model (0.90/0.95/0.99)
options.print = 'yes';      % print output (figures, tables) to screen

%% NOW LOAD DATA
P = load('pressure.txt');   % 48-hour forecasts air-pressure and verifying observations (mbar
    )

%% DEFINE ENSEMBLE AND VECTOR OF VERYFYING OBSERVATIONS ( APRIL 16 TO JUNE 9, 2000 )
idx = find(P(:,1) == 2000 & P(:,2) == 4 & P(:,3) == 16); start_idx = idx(1);
idx = find(P(:,1) == 2000 & P(:,2) == 6 & P(:,3) == 9); end_idx = idx(end);
D = P(start_idx:end_idx,5:9); Y = P(start_idx:end_idx,4);

%% APPLY LINEAR BIAS CORRECTION TO ENSEMBLE ( UP TO USER )
[ D , a , b ] = Bias_correction ( D , Y );

%% NOW EXECUTE THE MODELAVG TOOLBOX
[ beta , output ] = MODELAVG ( method , D , Y , options );
```

The BMA method is used for postprocessing of the 25-day sea level pressure ensemble. A normal distribution is used for the ensemble members' forecast distribution. The choice of this forecast distribution

is supported by the frequency distribution of the pressure observations of the training data set (see also Figure 3B). As air pressure observations have a constant measurement error, we assume that the forecast distribution has a fixed variance, but allow this variance to vary among the ensemble members. This thus involves the inference of $d = 2K = 10$ parameters with DREAM, namely the weight and standard deviation of each models' forecast distribution. Prior to this BMA model training, the individuals members of the ensemble were bias corrected using simple linear regression of their forecasts on the verifying observations of the training data set.

Figure 12 presents histograms of the DREAM derived marginal posterior distributions of the BMA weights and standard deviations of the different ensemble members. The optimal values derived with the EM algorithm are separately indicated in each panel with the black cross symbol. These EM values are computed using the following command

```
[beta,sigma,loglik] = EM_normal(D,Y,options)
```
(29)

where `beta` ($1 \times K$ vector) and `sigma` ($1 \times K$ vector) return the maximum likelihood values of the weights and standard deviations of the membersâĂŹ forecast distribution, respectively, and variable `loglik` (scalar) contains the maximized value of the log-likelihood function of Equation (16). The function EM_NORMAL is discussed in Appendix B and part of the MODELAVG toolbox.

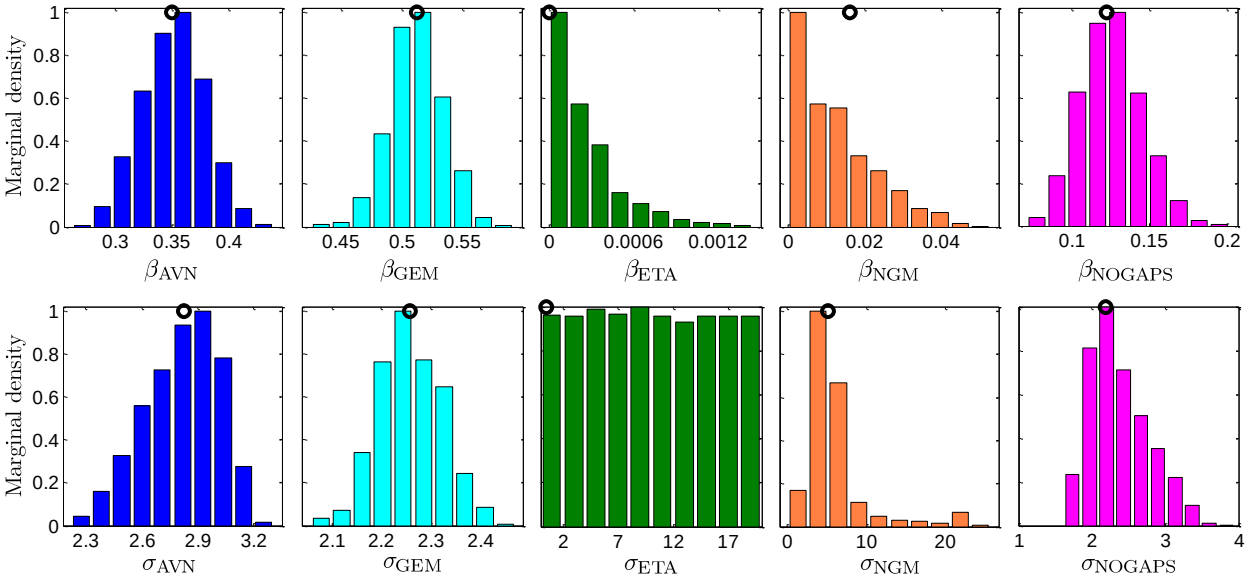


Figure 12: Histograms of the marginal posterior distributions of the BMA weights and standard deviations of the ensemble members' forecast distribution. The EM solution is separately indicated in each panel with the "o" symbol.

The results show an excellent agreement between the modes of the histograms derived from MCMC simulation with DREAM (`loglik = -9864.1`) and the maximum likelihood values of the EM algorithm (`loglik = -9863.7`). Hence, previous applications of the EM algorithm are likely to have yielded robust estimates for the BMA weights and standard deviations (variances) (see also *Vrugt et al.* (2008b)). However, DREAM has the desirable feature that it not only returns to the user the maximum likelihood values of

the BMA weights and standard deviations of the members' forecast distribution, but also their underlying posterior distribution and (posterior) correlation among ensemble members. This information is of great practical value as it helps determine each ensemble members' contribution to the predictive skill, and the dependencies among the forecasts of the K different members. This is crucial information to help determine which ensemble members to keep and which ones to discard as it takes time to setup and run each model of the ensemble. For instance, the ETA and NGM models receive very low weights in the present application. In fact, the forecast distribution of the ETA model is particularly ill-defined as its standard deviation has an almost uniform marginal distribution. The weights and standard deviations of the other models forecast distributions (AVN, GEM, and NOGAPS) appear much better defined, hence their forecasts play a key role in the construction of the BMA model.

I now turn attention to the DREAM algorithm. Figure 13 displays trace plots of the sampled weights of the AVN (top), GEM (middle) and NOGAPS (bottom) model. I use color coding for each different Markov chain sampled by DREAM. The maximum likelihood values are indicated at the right hand side with the "×" symbol. The EM solutions are also presented separately with the black "∘" symbol.

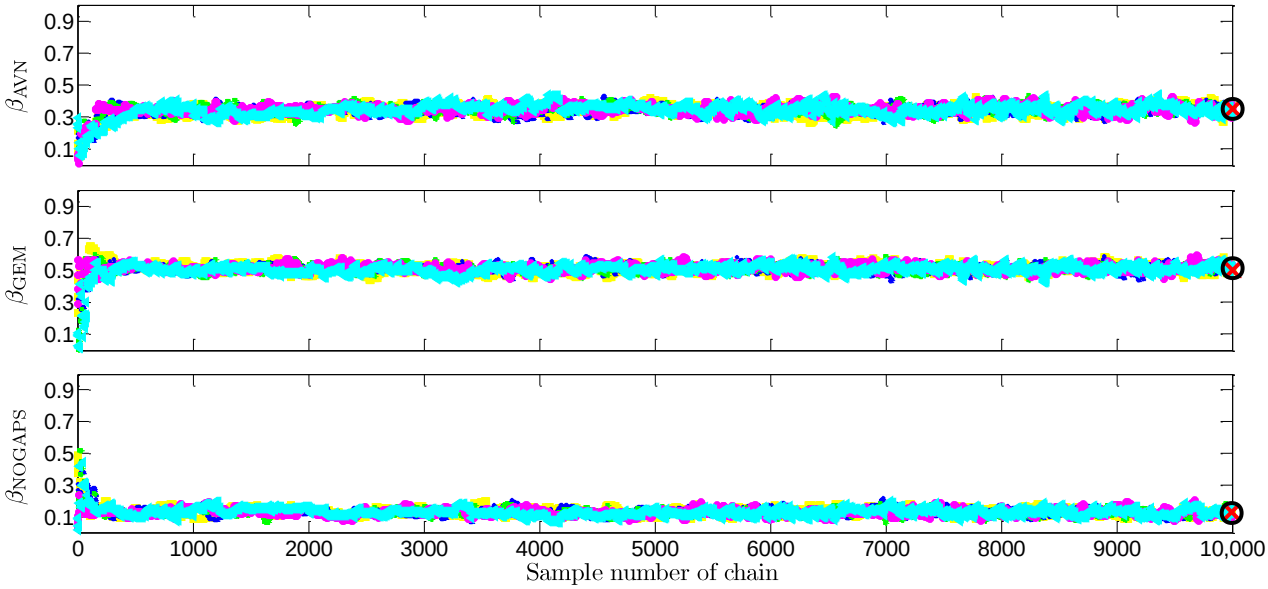


Figure 13: Trace plots of the sampled BMA weights of the AVN, GEM, and NOGAPS model in the Markov chains generated with DREAM. The maximum likelihood estimates computed of DREAM and the EM algorithm are separately indicated at the right hand side in each panel using the × and ∘ symbols.

During the initial stages of the search (first few hundred BMA_CALC evaluations, the chains occupy different parts of the weight space, resulting in a relatively high value for the $\hat{R}$-convergence diagnostic (not shown). After this, the five chains settle down in the approximate same region of the parameter space and successively visit solutions stemming from a stable distribution. This demonstrates convergence to a limiting distribution.

Figure 14 presents the 95% prediction uncertainty of the maximum likelihood BMA mixture distribution. The deterministic point forecast of the BMA model is separately indicated with the solid black line. This point predictor is derived from Equation (14) using the maximum likelihood values of the weights, $\hat{\beta}_{\text{BMA}}$,

and forecasts of the members of the ensemble stored in the $n \times K$ matrix $\mathbf{D}$.

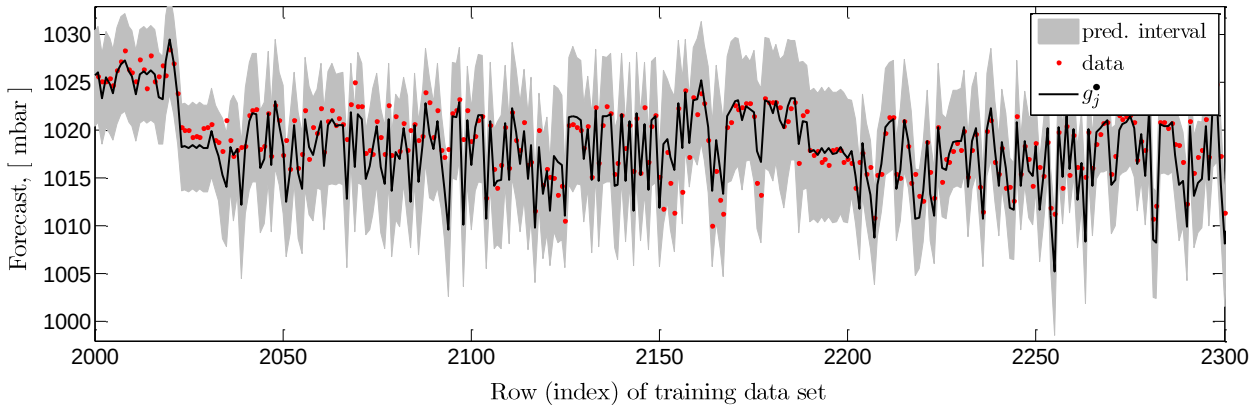


Figure 14: 95% prediction intervals (gray region) of the BMA predictive density for a representative portion of the 25-day training data set. The black line displays the mean point forecast of the BMA model derived from Equation (14), whereas the red dots signify the verifying observations.

The 95% prediction uncertainty ranges of the BMA model appear rather large with average spread of about 10.86 mbar and coverage of 94.36% of the observations. The RMSE of the BMA point forecast (mean of the mixture density) is approximately 2.80 mbar which is equivalent to the RMSE of the best model (NOGAPS) of the ensemble (see Table 8).

Table 8 lists the (maximum likelihood) values of the weights for each of the model averaging methods of the MODELAVG toolbox.

Table 8: Results of different model averaging methods for the five-member multimodel ensemble of 48-h forecasts of sea level pressure in the Pacific Northwest of the USA. The first two columns list the names of each model of the ensemble and their corresponding RMSE values (in mbar), respectively. Subsequent columns list the (maximum likelihood) values of the weights of each model averaging method of the toolbox. The bottom part of the Table lists the RMSE of the deterministic forecast of each method. This point forecast is calculated with Equation (4) using the ensemble forecasts of $\mathbf{D}$ and (maximum likelihood) values for the weights.

Model	RMSE	EWA	BGA	AICA†	BICA†	GRA	BMA	MMA	MMA$^{\Delta}$ †,‡
AVN	2.8507	0.2000	0.2304	0.0000	0.0000	0.5402	0.2538	0.5048	0.2774
GEM	2.9769	0.2000	0.2112	0.0000	0.0000	0.3461	0.1834	0.3457	0.2038
ETA	3.4640	0.2000	0.1560	0.0000	0.0000	0.0005	0.0003	0.0606	0.0000
NGM	3.5664	0.2000	0.1472	0.0000	0.0000	-0.4903	0.0009	-0.5305	0.0000
NOGAPS	2.7084	0.2000	0.2552	1.0000	1.0000	0.6035	0.5616	0.6194	0.5188
Point		2.8734	2.7875	2.7084	2.7084	2.4327	2.5731	2.4337	2.5714

†: I assume `p = 20*ones(1,K)` (twenty parameters assigned to each model)
‡: Presence of numerous local optima on likelihood surface

Information criterion averaging (AICA and BICA) assigns the NOGAPS model a weight of unity, whereas all other models are given a zero weight. The BMA method distributes the weights more equally among the different 882 ensemble members, yet assigns almost zero weights to ETA and NGM. Discarding these two models hardly affects the predictive skill of the averaged model and (in case of BMA) the forecast density.

The GRA and MMA methods receive the lowest RMSE for their deterministic (point) forecasts. These are the only two methods that do not restrict the weights to lie on the unit simplex.

Finally, I follow *Vrugt et al.* (2008b) and benchmark the performance of the DREAM algorithm by comparing the maximum likelihood estimates of this method against those derived separately using the EM algorithm. The function EM_NORMAL of Appendix A applies only to Gaussian forecast distributions, and hence we compare both methods using a normal forecast distribution for the members of the sea surface pressure ensemble. Figure 15 compares the log-likelihood estimates derived from the EM method (solid red line) and DREAM (blue squares) for different lengths of the training data set (x-axis). I also compare the corresponding estimates of the spread (B) and coverage (C) of the 95% BMA model prediction ranges. Bias correction was applied to each training data set using simple linear regression of D_k on $\tilde{\mathbf{Y}}$ of the training data set.

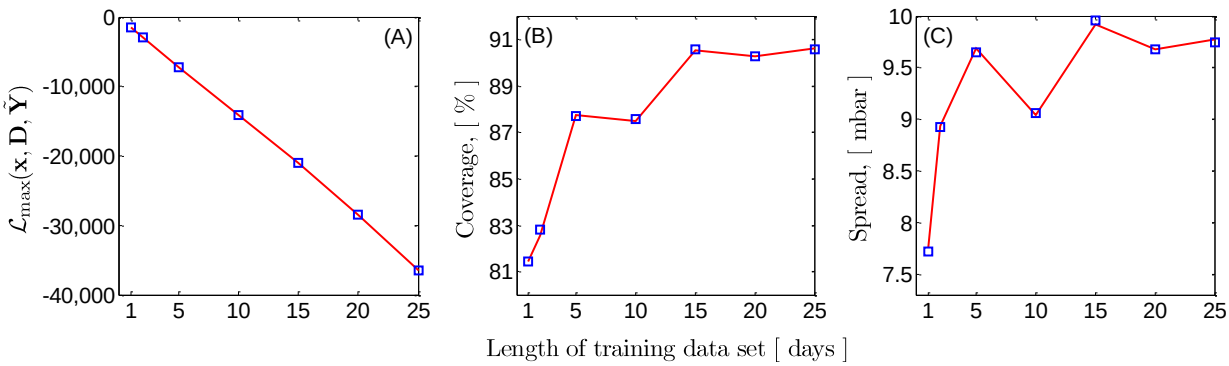


Figure 15: Maximum value of the log-likelihood of the EM (red) and DREAM (blue) method (A) and corresponding coverage (B) and spread (C) of the BMA prediction intervals.

The panels plot only the results for the training data set. The results presented here highlight several important observations. First, the DREAM algorithm and EM method derive exactly similar estimates of the log-likelihood value of the BMA model, irrespective of the length of the training data set. Secondly, the log-likelihood increases linearly with length of the training data set. This is simply the effect of the number of observations, n. Thirdly, the coverage of the 95% prediction intervals increases with length of the training data set. The same observation is made for the spread of the 95% prediction ranges. Both findings are not surprising as longer training data sets typically involve a larger diversity of weather events, requiring a larger standard deviation of the forecast distribution.

5. Recent developments

The original BMA approach presented by *Raftery et al.* (2005) assumes that the conditional pdf of each individual model is adequately described with a rather standard Gaussian or Gamma statistical distribution, possibly with a heteroscedastic variance. The work of *Rings et al.* (2012) has introduced a variant of BMA with a flexible representation of the conditional forecast distribution. A joint particle filtering and Gaussian mixture modeling framework was used to derive, as closely and consistently as possible, the evolving forecast density (conditional pdf) of each constituent ensemble member. These distributions are

subsequently combined with BMA and used to derive one overall predictive distribution. Benchmark studies demonstrate that this revised BMA method significantly receives lower-prediction errors than the original default BMA method (due to filtering) with predictive uncertainty intervals that are substantially smaller but still statistically coherent (due to the use of a time-variant conditional pdf)

6. Summary

In this manual I have introduced a MATLAB package, entitled MODELAVG, which provides interested users with a simple toolbox for postprocessing of forecast ensembles. This toolbox implements equal weight averaging, Bates-Granger averaging, information criterion averaging, Granger-Ramanathan averaging, Bayesian model averaging and Mallows model averaging. For those averaging methods for which an iterative solution is required to derive the weights and/or variance(s) of the conditional forecast distribution, MCMC simulation with DREAM is used, and a sample of the posterior distribution is generated. Three different case studies were used to illustrate the main capabilities and functionalities of the MATLAB toolbox. These example studies are easy to run and adapt and serve as templates for other modeling problems and watershed data sets.

The toolbox allows for different formulations of the BMA conditional forecast distribution. Forecast densities that differ from a normal or gamma distribution are readily implemented in the source code of the MODELAVG toolbox by adding a new "case" in the functions BMA_CALC.

Our current work involves new approaches to density forecasting using least-squares model averaging methods. Applications include precipitation estimation and forecasting using binomial conditional distributions.

7. Acknowledgements

The MATLAB toolbox of MODELAVG is available upon request from the first author, `jasper@uci.du`.

Appendix A. The Expectation-Maximization algorithm

The Expectation - Maximization (EM) algorithm is a broadly applicable approach to the iterative computation of maximum likelihood estimates, useful in a variety of incomplete-data problems (*Dempster et al.*, 1997; *McLachlan and Krihnan*, 2008). I use herein the index j to mean "$forall j \in \{1, \ldots, n\}$" and the index k to mean "$forall k \in \{1, \ldots, K\}$" and use the latent variable z_{jk} to help find the optimal values of the BMA weights and standard deviations. This unobserved quantity has a value of unity if ensemble member k is the best forecast of $\tilde{y}_j$ and zero otherwise. For each observation of the training data set, only one of the $\{z_{j1}, \ldots, z_{jK}\}$ values is thus equal to one, and all the other values are zero.

The EM algorithm alternates between an expectation (E) step and a maximization (M) step until convergence is achieved. In the expectation step, the values of z_{jk} are calculated given the current values of the BMA weights and variances. For the BMA model of Equation (11) and (12) the E step is given by

$$\hat{z}_{jk}^{(t)} = \frac{\beta_k \mathcal{N}(\tilde{y}_j | D_{jk}, \sigma_k^{(t-1)})}{\sum_{i=1}^{K} \beta_i \mathcal{N}(\tilde{y}_j | D_{ji}, \sigma_i^{(t-1)})} \qquad \text{(Expectation Step)}, \tag{A.1}$$

where the function $\mathcal{N}(a|b, c)$ returns the density at a of a normal distribution with mean b and standard deviation c, and the superscript t signifies iteration counter. In the subsequent maximization step, the values of β_k and σ_k^2 are updated using the current estimates of z_{jk}, i.e. $\hat{z}_{jk}^{(t)}$ as follows

$$\begin{aligned} \beta_k^{(t)} &= \frac{1}{n} \sum_{m=1}^{n} \hat{z}_{mk}^{(t)} \\ & \qquad\qquad\qquad\qquad \text{(Maximization Step)} \\ \sigma_k^{2(t)} &= \frac{\sum_{m=1}^{n} \hat{z}_{mk}^{(t)} (\tilde{y}_m - D_{mk})^2}{n \sum_{m=1}^{n} \hat{z}_{mk}^{(t)}}, \end{aligned} \tag{A.2}$$

where n denotes the number of observations of the training data set. By alternating between Equations (A.1) and (A.2) the EM algorithm improves iteratively the values of β_k and σ_k^2. Convergence is achieved when the values of the likelihood (= denominator of Equation (A.1)), weights, $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_K\}$, variances, $\boldsymbol{\sigma}^2 = \{\sigma_1^2, \ldots, \sigma_K^2\}$ and $\hat{z}_{jk}^{(t)}$s remain constant from one iteration to the next.

The function EM_NORMAL listed below implements the EM algorithm in MATLAB. This subroutine has three input arguments, including `D` (matrix with ensemble forecasts), `Y` (calibration data vector), and structure `options` (for field VAR with variance option) and returns the maximum likelihood values of the BMA weights, $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_K\}$ and standard deviation, σ (`options`.VAR = '1') or standard deviations, $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_K\}$ (`options`.VAR = '2') and corresponding log-likelihood, $\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\sigma} | \mathbf{D}, \tilde{\mathbf{Y}})$ in Equation (16).

MATLAB code of the Expectation-Maximization algorithm: This function calculates the maximum likelihood values of the BMA weights, `beta` and standard deviations, `sigma` of each members normal forecast distribution for a given $(n \times K)$ matrix of ensemble forecasts, and $(n \times 1)$ vector of training observations. The third input argument, `options` is a structure with field VAR that determines whether to use a single common variance for all forecast distributions of the members of the BMA ensemble (`options`.VAR = '1') or a member specific variance (`options`.VAR = '2'). Notation and variable names are consistent with main text and vectorization is used to minimize somewhat the number of lines of the code. Built-in functions are highlighted with a low dash. The binary singleton expansion function `bsxfun(FUNC,A,B)` applies an element-by-element binary operation to the $(n \times K)$ matrix `A` and $(n \times 1)$ vector `B` using either a right array divide (`FUNC = @rdivide`) or a minus operation (`FUNC = @minus`). The function `abs(X)` calculates the absolute value of the elements of `X`, and `sum(X,dim)` sums along the dimension `dim` (1: vertical, 2: horizontal).

```
function [ beta , sigma , loglik , t ] = EM_normal ( D , Y , options )
% Expectation-Maximization method for training of BMA model                %
% SYNOPSIS      [w,sigma,lik] = EM_normal(D,Y);                            %
%               [w,sigma,lik] = EM_normal(D,Y,options);                    %
%   INPUT       D           (n x K)-matrix of ensemble forecasts           %
%               Y           (n x 1)-vector with training data set          %
%               options     structure with settings                        %
%   OUTPUT      beta        (1 x K)-vector of BMA weights                  %
%               sigma       (1 x K)-vector of BMA standard deviations      %
%               lik         log-likelihood value                           %
% ------------------------------------------------------------------------
% Assumption: Gaussian forecast distribution

if nargin < 2,
    error('EM:TooFewInputs', 'Requires at least two input arguments.');
end
if nargin < 3, VAR = '1'; else VAR = lower(options.VAR); end

[n,K] = size(D);                                % Matrix of ensemble forecasts
beta = ones(1,K)/K;                             % Initial values weights
sigma = std(Y)*ones(1,K); sigma2 = sigma.^2;    % Initial values stds./var.
z_t = zeros(n,K);                               % Initial values latent variables
loglik_t = -Inf; err = 1; t = 0; max_t = 1e4;   % Settings/constraints while loop

while ( max(err) > 1e-6 ) && ( t < max_t ),       % Until ... do
    loglik = loglik_t; z = z_t;                           % Copy z and loglik
    for k = 1:K,                                          % EXPECTATION STEP
        z_t(:,k) = beta(k)*normpdf(Y,D(:,k),sigma(k));    % Update latent variables
    end
    loglik_t = sum(log(sum(z_t,2)));                      % Log-likelihood BMA model
    z_t = bsxfun(@rdivide,z_t,sum(z_t,2));                % Normalize latent variables

    beta_t = sum(z_t)/n;                                  % MAXIMIZATION STEP
    sigma2_t = sum(z_t.*bsxfun(@minus,D,Y).^2)./sum(z_t);
    if strcmp(VAR,'1'),                                   % If common constant variance
        sigma2_t = mean(sigma2_t)*ones(1,K);              % Use mean value
    end

    err(1) = max(abs(beta_t - beta));                     % Convergence: weights
    err(2) = max(abs(log(sigma2_t./sigma.^2)));           % Convergence: variance(s)
    err(3) = max(max(abs(z - z_t)));                      % Convergence: latent variables
    err(4) = max(abs(loglik - loglik_t));                 % Convergence: log-likelihood

    beta = beta_t; sigma = sqrt(sigma2_t);                % Update BMA weights and variances
    t = t + 1;                                            % Iteration counter
end                                                 % End while loop
```

Appendix B. The DREAM algorithm

The DREAM algorithm is an efficient multi-chain MCMC simulation method that uses differential evolution as genetic algorithm for population evolution with a Metropolis selection rule to decide whether candidate points should replace their parents or not. In DREAM, N different Markov chains are run simultaneously in parallel. If the state of a single chain is given by the d-vector $\mathbf{x}$, then at each generation $t-1$ the N chains in DREAM define a population $\mathbf{X}$, which corresponds to an $N \times d$ matrix, with each chain as a row. If A is a subset of d^*-dimensions of the original parameter space, $\mathbb{R}^{d^*} \subseteq \mathbb{R}^d$, then a jump, $\mathbf{dX}^i$ in the ith chain, $i = \{1, \ldots, N\}$ at iteration $t = \{2, \ldots, T\}$ is calculated from the collection of chains, $\mathbf{X} = \{\mathbf{x}^1_{t-1}, \ldots, \mathbf{x}^N_{t-1}\}$ using differential evolution (*Storn and Price*, 1997; *Price et al.*, 2005)

$$\begin{aligned} \mathbf{dX}^i_{\mathrm{A}} &= \boldsymbol{\zeta}_{d^*} + (\mathbf{1}_{d^*} + \boldsymbol{\lambda}_{d^*})\gamma_{(\delta,d^*)} \sum_{j=1}^{\delta} \left(\mathbf{X}^{\mathbf{a}_j}_{\mathrm{A}} - \mathbf{X}^{\mathbf{b}_j}_{\mathrm{A}}\right) \\ \mathbf{dX}^i_{\neq \mathrm{A}} &= 0, \end{aligned} \tag{B.1}$$

where $\gamma = 2.38/\sqrt{2\delta d^*}$ is the jump rate, δ denotes the number of chain pairs used to generate the jump, and $\mathbf{a}$ and $\mathbf{b}$ are vectors consisting of δ integers drawn without replacement from $\{1, \ldots, i-1, i+1, \ldots, N\}$. The default value of $\delta = 3$, and results, in practice, in one-third of the proposals being created with $\delta = 1$, another third with $\delta = 2$, and the remaining third using $\delta = 3$. The values of $\boldsymbol{\lambda}$ and $\boldsymbol{\zeta}$ are sampled independently from $\mathcal{U}_{d^*}(-c, c)$ and $\mathcal{N}_{d^*}(0, c_*)$, respectively, the multivariate uniform and normal distribution with, typically, $c = 0.1$ and c_* small compared to the width of the target distribution, $c_* = 10^{-6}$ say. In 20% of the proposals, I use a jump rate of unity, $p_{(\gamma=1)} = 0.2$, to enable the chains of DREAM to jump directly between disconnected posterior modes. The candidate point of chain i at iteration t then becomes

$$\mathbf{X}^i_{\mathrm{p}} = \mathbf{X}^i + \mathbf{dX}^i, \tag{B.2}$$

and the Metropolis ratio

$$p_{\mathrm{acc}}(\mathbf{X}^i \to \mathbf{X}^i_{\mathrm{p}}) = \min[1, p(\mathbf{X}^i_{\mathrm{p}})/p(\mathbf{X}^i)], \tag{B.3}$$

is used to determine whether to accept this proposal or not. If $p_{\mathrm{acc}}(\mathbf{X}^i \to \mathbf{X}^i_{\mathrm{p}}) \geq \mathcal{U}(0,1)$ the candidate point is accepted and the ith chain moves to the new position, that is $\mathbf{x}^i_t = \mathbf{X}^i_{\mathrm{p}}$, otherwise $\mathbf{x}^i_t = \mathbf{x}^i_{t-1}$. The default equation for γ should, for Gaussian and Student target distribution, result in optimal acceptance rates close to 0.44 for $d = 1$, 0.28 for $d = 5$, and 0.23 for large d (please refer to section 7.84 of *Roberts and Casella* (2004) for a cautionary note on these references acceptance rates).

The d^*-members of the subset A are sampled from the entries $\{1, \ldots, d\}$ (without replacement) and define the dimensions of the parameter space to be sampled by the proposal. This subspace spanned by A is construed in DREAM with the help of a crossover operator. This genetic operator is applied before each proposal is created and works as follows. First, a crossover value, cr is sampled from a geometric sequence of n_{CR} different crossover probabilities, $\mathrm{CR} = \{\frac{1}{n_{\mathrm{CR}}}, \frac{2}{n_{\mathrm{CR}}}, \ldots, 1\}$ using the discrete multinomial distribution, $\mathcal{M}(\mathrm{CR}, \mathbf{p}_{\mathrm{CR}})$ on CR with selection probabilities $\mathbf{p}_{\mathrm{CR}}$. Then, a d-vector $\mathbf{z} = \{z_1, \ldots, z_d\}$ is drawn from a standard multivariate normal distribution, $\mathbf{z} \overset{\mathcal{D}}{\sim} \mathcal{U}_d(0,1)$. All those values j which satisfy $z_j \leq \mathrm{cr}$ are stored in the subset A and span the subspace of the proposal that will be sampled using Equation (B.1). If A is empty, one dimension of $\{1, \ldots, d\}$ will be sampled at random to avoid the jump vector to have zero length.

The use of a vector of crossover probabilities enables single-site Metropolis (A has one element), Metropolis-within-Gibbs (A has one or more elements) and regular Metropolis sampling (A has d elements), and constantly introduces new directions in the parameter space that chains can take outside the subspace spanned by their current positions. What is more, the use of subspace sampling allows for $N < d$, thereby reducing as much as possible the total number of function evaluations required for burn-in. Subspace sampling as implemented in DREAM adds one extra algorithmic variable, n_{CR} to the algorithm. The default setting of $n_{\mathrm{CR}} = 3$ has shown to work well in practice, but larger values of this algorithmic variable might seem appropriate for high-dimensional target distributions, say $d > 50$, to preserve the frequency of low-dimensional jumps. Note, more intelligent subspace selection methods can be devised for target distributions involving many highly correlated parameters.

To enhance search efficiency the selection probability of each crossover value, stored in the n_{CR}-vector $\mathbf{p}_{\mathrm{CR}}$, is tuned adaptively during burn-in by maximizing the distance traveled by each of the N chains. This adaptation is described in detail in *Vrugt et al.* (2008a, 2009), and a numerical implementation of this approach appears in the MATLAB code of DREAM below.

The core of the DREAM algorithm can be written in about 30 lines of code (see algorithm) and include the function handles `prior` and `pdf` and the values of `N` (number of chains), `T` (number of iterations) and `d` (number of parameters).

MATLAB code of the DiffeRential Evolution Adaptive Metropolis (DREAM) algorithm: Built-in functions are highlighted with a low dash. The jump vector, `dX(i,1:d)` of the `i`th chain contains the desired information about the scale and orientation of the proposal distribution and is derived from the remaining `N-1` chains. `deal()` assigns default values to the algorithmic variables of DREAM, `std()` returns the standard deviation of each column of `X`, and `sum()` computes the sum of the columns `A` of the chain pairs `a` and `b`. The function `check()` is a critical patch for outlier chains that impair convergence to a limiting distribution.

```
function [x,p_x] = dream(prior,pdf,N,T,d)
% DiffeRential Evolution Adaptive Metropolis (DREAM) algorithm

[delta,c,c_star,n_CR,p_g] = deal(3,0.1,1e-12,3,0.2);      % Default of algorithmic parameters
x = nan(T,d,N); p_x = nan(T,N);                           % Preallocate chains and density
[J,n_id] = deal(zeros(1,n_CR));                           % Variables select. prob. crossover
for i = 1:N, R(i,1:N-1) = setdiff(1:N,i); end             % R-matrix: index of chains for DE
CR = [1:n_CR]/n_CR; p_CR = ones(1,n_CR)/n_CR;             % Crossover values and select.
    prob.

X = prior(N,d);                                           % Create initial population
for i = 1:N, p_X(i,1) = pdf(X(i,1:d)); end                % Compute density initial
    population
x(1,1:d,1:N) = reshape(X',1,d,N); p_x(1,1:N) = p_X';      % Store initial states and density

for t = 2:T,     % Dynamic part: Evolution of N chains
    [~,draw] = sort(rand(N-1,N));                         % Permute [1,...,N-1] N times
    dX = zeros(N,d);                                      % Set N jump vectors to zero
    lambda = unifrnd(-c,c,N,1);                           % Draw N lambda values
    std_X = std(X);                                       % Compute std each dimension
    for i = 1:N,                                          % Create proposals + accept/reject
        D = randsample([1:delta],1,'true');               % Select delta (equal probability)
        a = R(i,draw(1:D,i)); b = R(i,draw(D+1:2*D,i));   % Extract vectors a + b unequal i
        id = randsample([1:n_CR],1,'true',p_CR);          % Select index of crossover value
        z = rand(1,d);                                    % Draw d values from U[0,1]
        A = find(z < CR(id));                             % Subset A dimensions to update
        d_star = numel(A);                                % How many dimensions sampled?
        if d_star == 0, [~,A] = min(z); d_star = 1; end   % A must contain one dimension
        gamma_d = 2.38/sqrt(2*D*d_star);                  % Calculate jump rate
        g = randsample([gamma_d 1],1,'true',[1-p_g p_g]); % Select gamma: 80/20 mix [def: 1]
        dX(i,A) = c_star*randn(1,d_star) + ...
            (1+lambda(i))*g*sum(X(a,A)-X(b,A),1);         % Compute ith jump diff. evol.
        Xp(i,1:d) = X(i,1:d) + dX(i,1:d);                 % Compute ith proposal
        p_Xp(i,1) = pdf(Xp(i,1:d));                       % Calculate density ith proposal
        p_acc = min(1,p_Xp(i,1)./p_X(i,1));               % Compute acceptance probability
        if p_acc > rand,                                  % p_acc larger than U[0,1]?
            X(i,1:d) = Xp(i,1:d); p_X(i,1) = p_Xp(i,1);   % True: Accept proposal
        else
            dX(i,1:d) = 0;                                % Set jump back to zero for pCR
        end
        J(id) = J(id) + sum((dX(i,1:d)./std_X).^2);       % Update jump distance id crossover
        n_id(id) = n_id(id) + 1;                          % How many times id crossover used
    end
    x(t,1:d,1:N) = reshape(X',1,d,N); p_x(t,1:N) = p_X';  % Append current X and density
    if t<T/10,
        p_CR = J./n_id; p_CR = p_CR/sum(p_CR);            % Update selection prob. crossover
    end
    [X,p_X] = check(X,mean(log(p_x(ceil(t/2):t,1:N))));   % Outlier detection and correction
end             % End dynamic part
```

The MATLAB code listed above implements the different steps of the DREAM algorithm as detailed in the main text of this Appendix. Variable names correspond with their symbols used in Equations (B.1) and (B.2). Indents and comments are used to enhance readability and to convey the main intent of each line of code. The computational efficiency of this code can be improved considerably, for instance through vectorization of the inner for loop, but this will affect negatively readability. Note that this basic code of DREAM does not monitor convergence of the sampled chain trajectories.

The function `check` scans for dissident chains using as proxy for fitness the mean log density of the second half of the samples stored in each Markov chain. These N values are examined for anomalies using an outlier detection test. Those chains that are labeled as an outlier will relinquish their dissident state by moving their position to one of the other chains (chosen at random). Details of this procedure can be found in *Vrugt et al.* (2009).

For those proficient in statistics, computer coding and numerical computation, the DREAM code listed above will be sufficient to solve for the posterior distribution of the BMA, MMA, and MMAΔ parameters. Yet, for others this code might not suffice as it has very few built-in options and capabilities. I therefore refer to MATLAB toolbox of the DREAM algorithm described in *Vrugt* (2016).

Appendix C. Download and installation

The MODELAVG code can be downloaded from my website at the following link `http://faculty.sites.uci.edu/MODELAVG`. Please save this file called "MATLAB-pCode-MODELAVG-V1.0" to your hard disk, for instance, in the directory "D:\Downloads\Toolboxes \MATLAB\MODELAVG". Now open Windows explorer in this directory (see Figure C1).

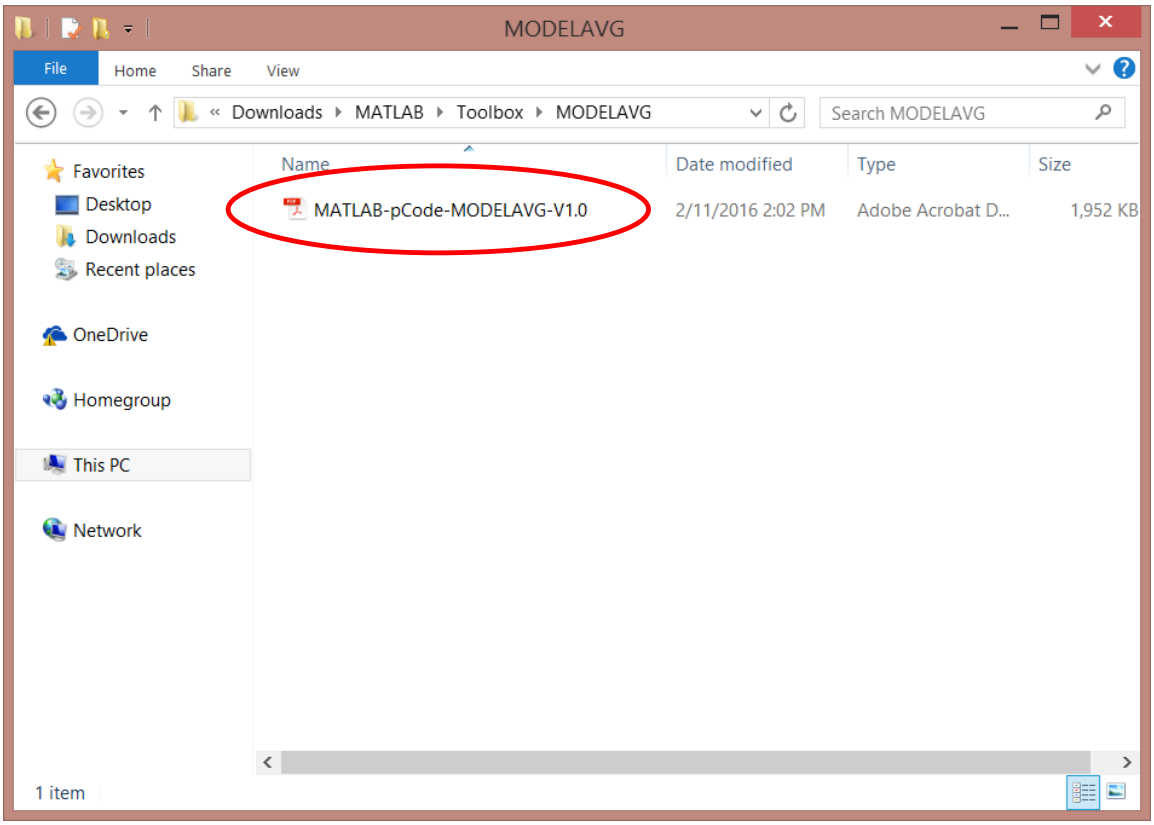


Figure C1

You will notice that the file does not have an extension - it is just called **MATLAB-pCode-MODELAVGV1.0**. That is because Windows typically hides extension names.

If you can already see file extensions on your computer, then please skip the next step. If you cannot see the file extension, please click the **View** tab. Then check the box titled "File name extensions" (see Figure C2).

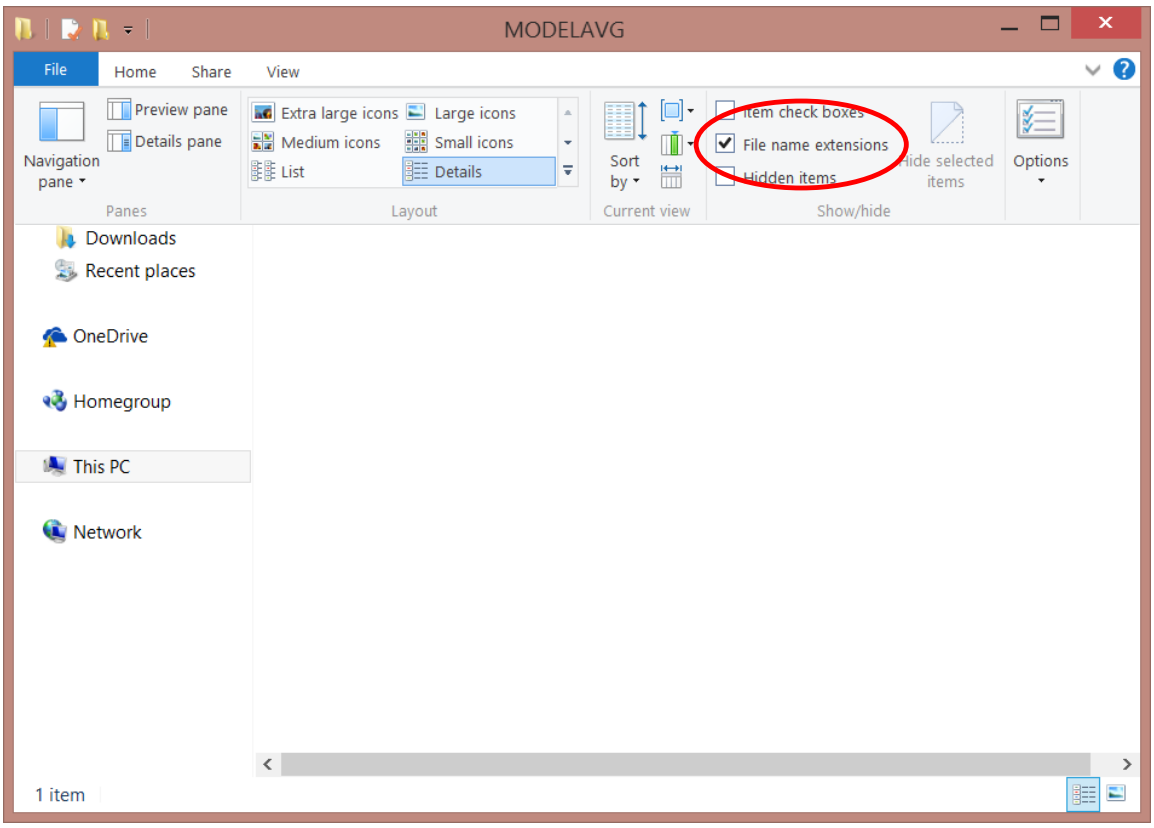


Figure C2

Now you should be able to see the file extension. Right-click the file name and select **Rename** (see Figure C3).

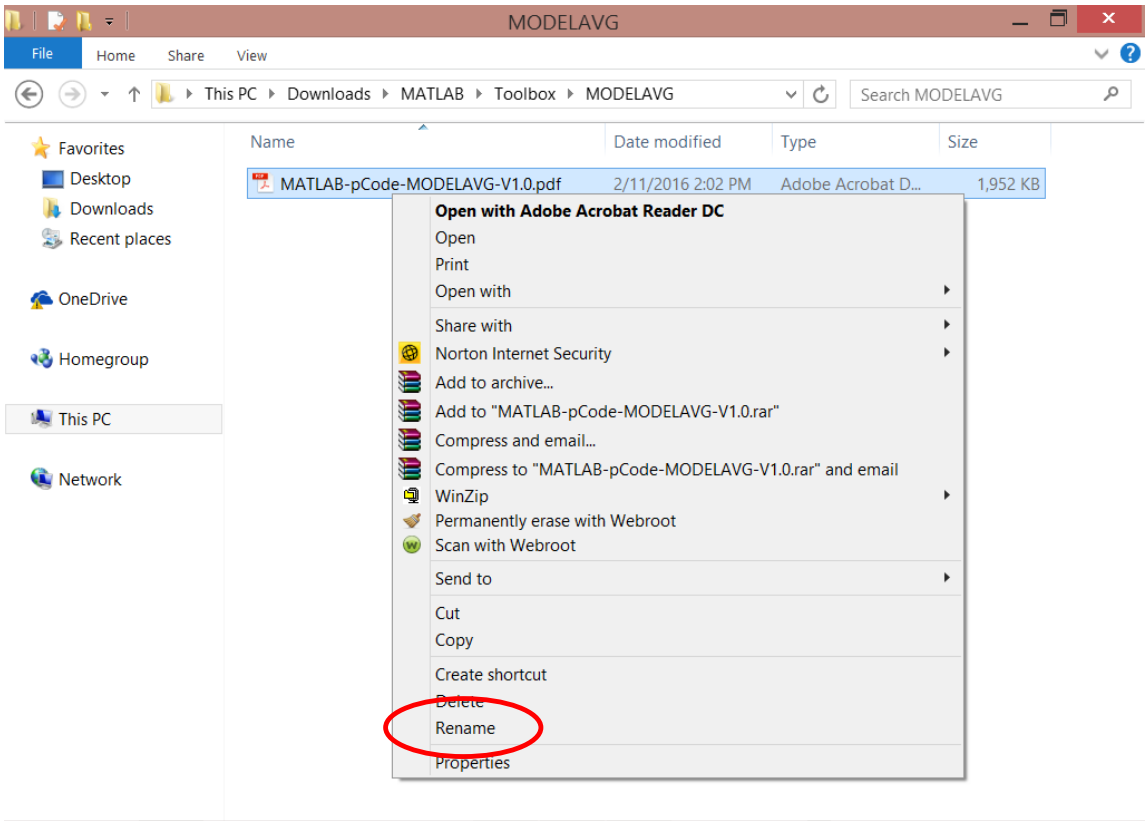


Figure C3

Now change the extension of "MATLAB-pCode-MODELAVG-V1.0" from ".pdf" to ".rar" (see Figure C4).

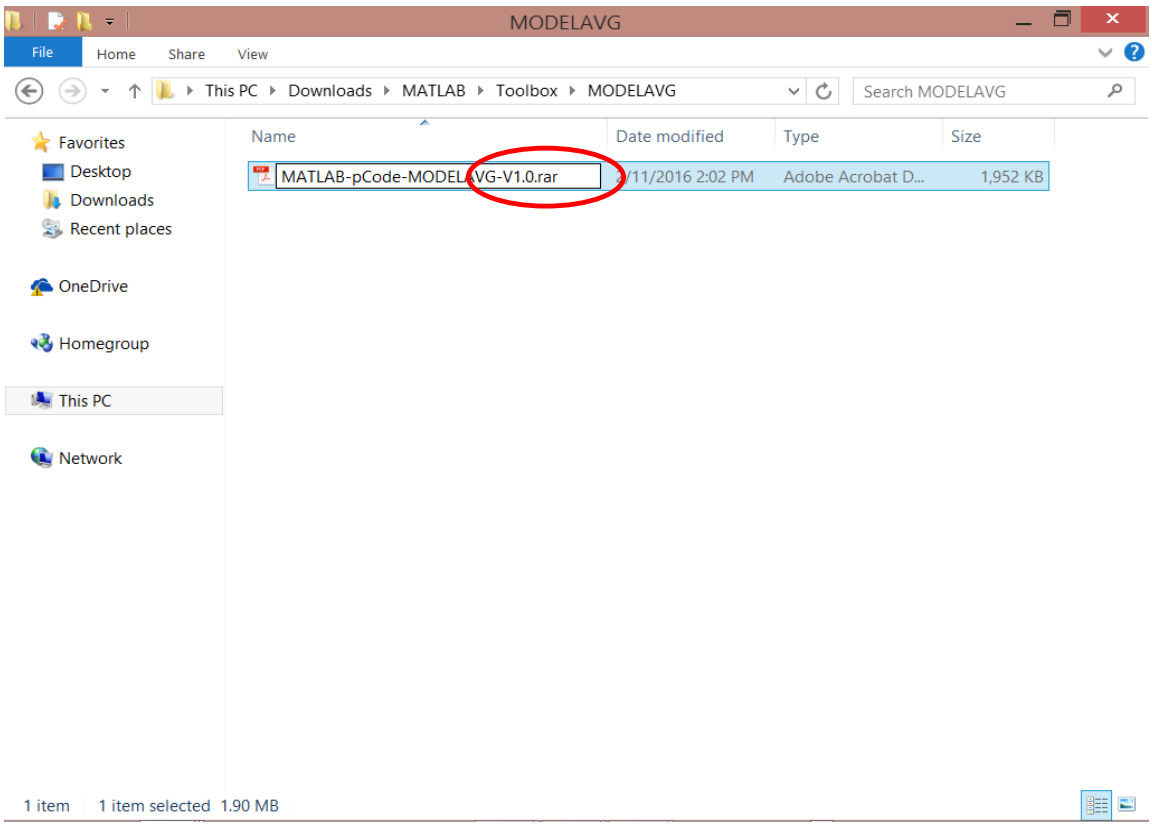


Figure C4

After entering the new extension, hit the **Enter** (return) key. Windows will give you a warning that the file may not work properly (see Figure C5). This is quite safe - remember that you can restore the original extension if anything goes wrong.

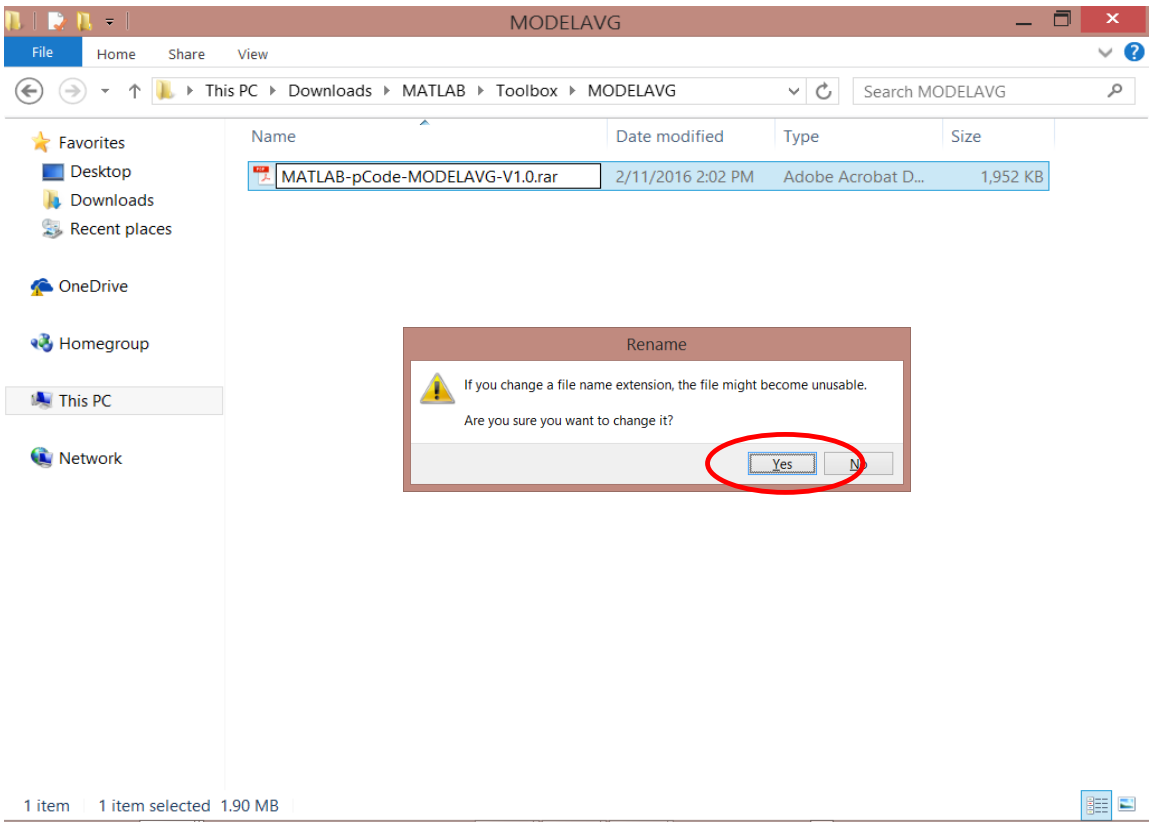


Figure C5

It is also possible that you might get another message telling you that the file is "read-only". In this case either say yes to turning off read-only, or right-click the file, select **Properties** and uncheck the **Read-only** box.

If you do not have permission to change the file extension, you may have to login as Administrator. Another option is to make a copy of the file, rename the copy and then delete the original.

Now you have changed the extension of the file to ".rar" you can use the program WinRAR to extract the files to whatever folder your desire, for instance "D:\Downloads\Toolboxes\MATLAB\MODELAVG". Right-click the file name and select **Extract Here** (see Figure C6).

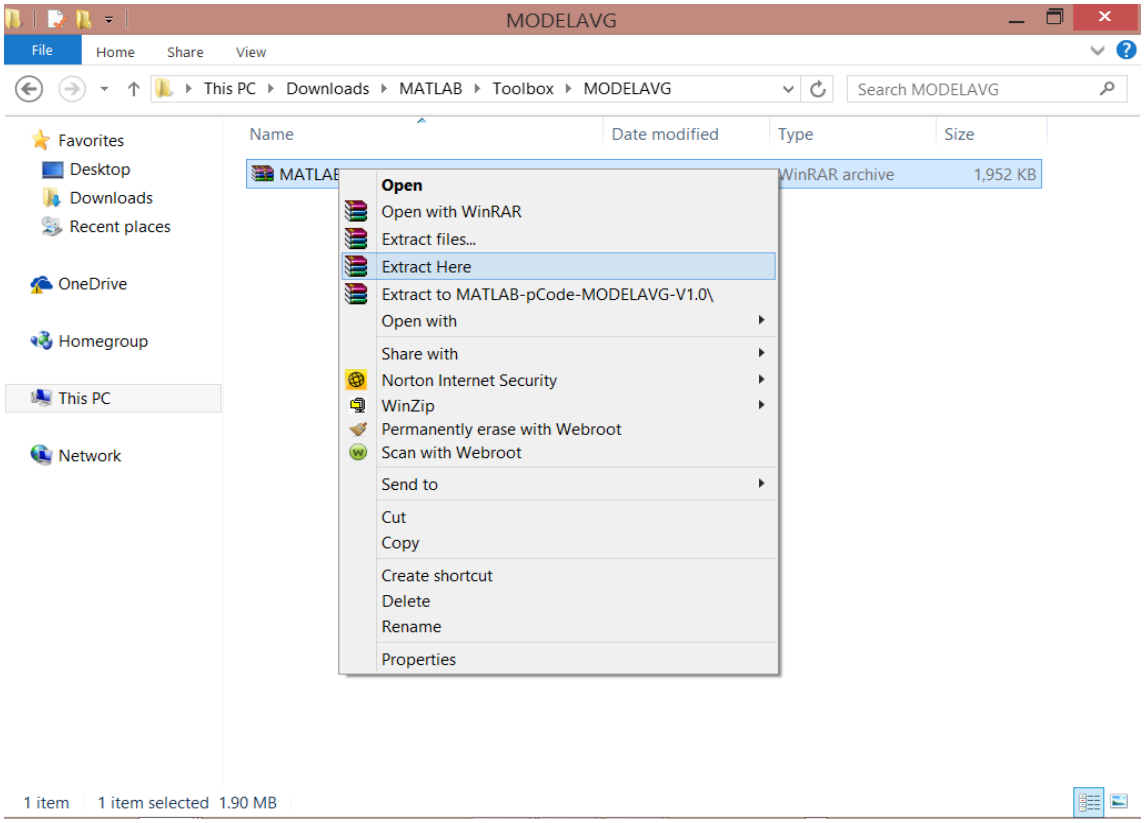


Figure C6

Now WinRAR should extract the files to your folder. The end result should look as in Figure C7.

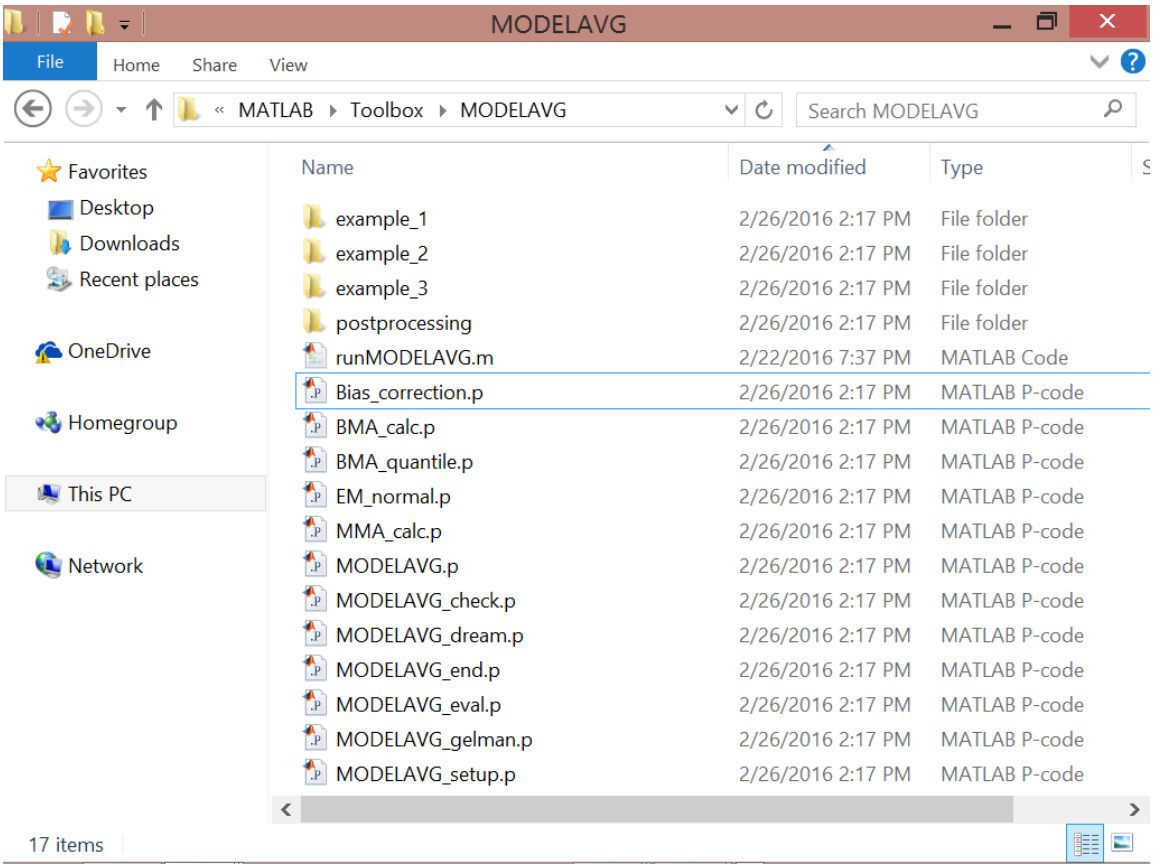


Figure C7

The MODELAVG toolbox is now ready for use in MATLAB.

Appendix D. Main functions of MODELAVG toolbox

Table D1 summarizes, in alphabetic order, the different function/program files of the MODELAVG package in MATLAB.

The main program RUNMODELAVG calls the input file of each case study. Template input files were given on Pages 24, 29 and 34 for each of the three case studies considered herein. These templates can be used for other data sets and/or forecast ensembles. The last line of each template file involves a call to the function MODELAVG, which computes the values of the weights (all methods except BMA) or weights and standard deviations (or proxies thereof) of the forecast density (if BMA is used). What is more, a structure `output` with results of each model averaging method is produced, and tables and figures are printed to the screen if the print option in structure `options` is activated ('yes'). The data of each case study are stored in their respective folders in the root directory of the MODELAVG toolbox (see Figure C7 for a screen shot of these directories).

The function MODELAVG_DREAM contains a basic implementation of the DREAM algorithm. This function is used to derive the posterior distribution of the BMA, MMA, and MMA$^{\Delta}$ weights and standard deviation(s) of the members' forecast distribution (BMA). Users are referred to the DREAM toolbox of *Vrugt* (2016).

The directory "postprocessing" in the root of the MODELAVG toolbox contains the script MODELAVG_POSTPROC which is called by the main function of the toolbox, MODELAV and prints to the screen a table with the main results of each model averaging method, and many different figures. The tables are displayed in the MATLAB editor (see Appendix E), whereas figures are printed directly to the screen, including a time series plot of the ensemble members, the verifying observations and the averaged forecast, an autocorrelation function and a quantile-quantile graph of the error residuals of this point predictor. If BMA, MMA or MMA$^{\Delta}$ are used, many more figures are created using the DREAM output including trace plots of the sampled chain trajectories and $\hat{R}$-convergence diagnostic, and histograms of the marginal distributions of the parameters sampled by DREAM posterior samples (among others). Appendix E presents the screen output produced by MODELAVG_postproc for case study 2.

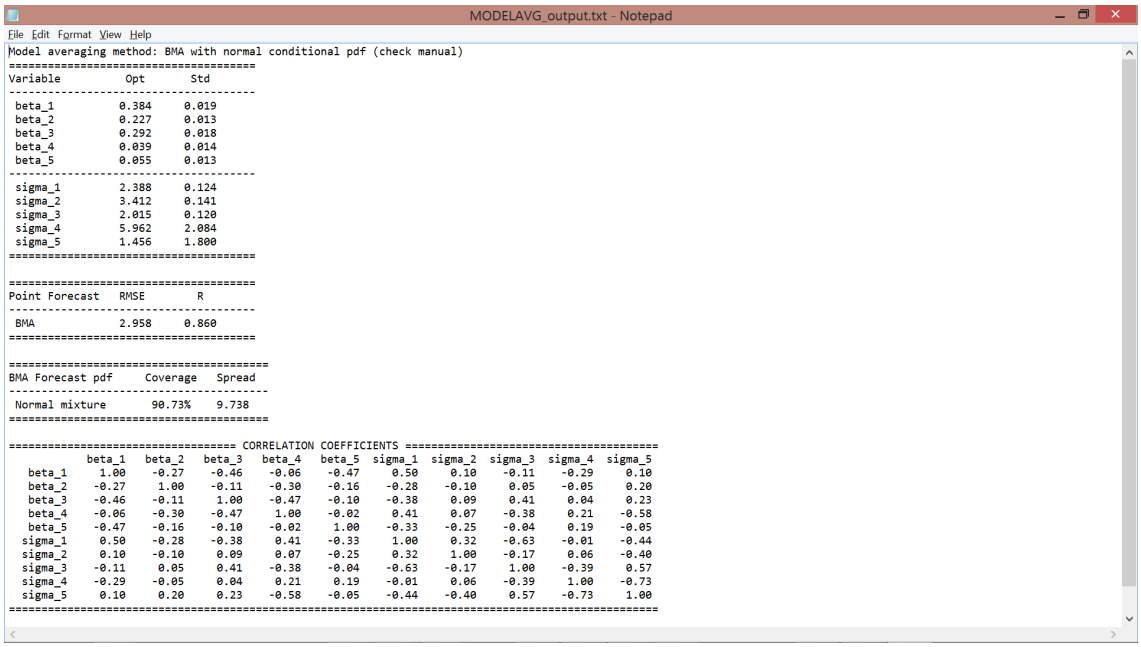
Table D1: Description of the MATLAB functions and scripts (.m files) used by MODELAVG, version 1.0.

Name of function	Description
BIAS_CORRECTION	Applies linear bias correction of each member of ensemble
BMA_CALC	Calculates the log-likelihood of BMA model parameters
BMA_QUANTILE	Computes the prediction intervals of BMA mixture distribution
EM_NORMAL	Expectation maximization algorithm for BMA model training
MMA_CALC	Calculates the log-likelihood of the MMA weights
MODELAVG	Main function of the toolbox - returns output arguments `x` and structure `output`
MODELAVG_CHECK	Verifies input arguments of MODELAVG toolbox
MODELAVG_DREAM	Basic implementation of DREAM algorithm for BMA and MMA model training
MODELAVG_END	Prepares graphical output and return arguments of MODELAVG toolbox
MODELAVG_EVAL	Calculates statistics of independent evaluation data set
MODELAVG_GELMAN	Calculates the $\hat{R}^d$ and $\hat{R}$-convergence diagnostics
MODELAVG_SETUP	Setup of computational framework of MODELAVG toolbox
RUNMODELAVG	Main program of the MODELAVG toolbox which executes the different case studies

Appendix E. Screen output

The MODELAVG toolbox presented herein returns to the user tables and figures which jointly summarize the results of the toolbox. This appendix displays all this output for the second case study involving application of the BMA method to the five-member ensemble of temperature forecasts in Pacific Northwest of the USA). A normal forecast distribution was assumed for each model of the ensemble. The standard deviation of this distribution was assumed to be constant, yet member-dependent.

Figure E1 displays the ascii file "MODELAVG_output.txt" which is created by the main function MODELAVG of the toolbox and printed to the screen in the MATLAB editor.



```
Model averaging method: BMA with normal conditional pdf (check manual)
=====================================
Variable          Opt       Std
-------------------------------------
 beta_1          0.384     0.019
 beta_2          0.227     0.013
 beta_3          0.292     0.018
 beta_4          0.039     0.014
 beta_5          0.055     0.013
-------------------------------------
 sigma_1         2.388     0.124
 sigma_2         3.412     0.141
 sigma_3         2.015     0.120
 sigma_4         5.962     2.084
 sigma_5         1.456     1.800
=====================================

=====================================
Point Forecast   RMSE        R
-------------------------------------
 BMA             2.958     0.860
=====================================

=========================================
BMA Forecast pdf     Coverage   Spread
-----------------------------------------
 Normal mixture       90.73%    9.738
=========================================

=================================== CORRELATION COEFFICIENTS ====================================
            beta_1   beta_2   beta_3   beta_4   beta_5  sigma_1  sigma_2  sigma_3  sigma_4  sigma_5
   beta_1     1.00    -0.27    -0.46    -0.06    -0.47     0.50     0.10    -0.11    -0.29     0.10
   beta_2    -0.27     1.00    -0.11    -0.30    -0.16    -0.28    -0.10     0.05    -0.05     0.20
   beta_3    -0.46    -0.11     1.00    -0.47    -0.10    -0.38     0.09     0.41     0.04     0.23
   beta_4    -0.06    -0.30    -0.47     1.00    -0.02     0.41     0.07    -0.38     0.21    -0.58
   beta_5    -0.47    -0.16    -0.10    -0.02     1.00    -0.33    -0.25    -0.04     0.19    -0.05
  sigma_1     0.50    -0.28    -0.38     0.41    -0.33     1.00     0.32    -0.63    -0.01    -0.44
  sigma_2     0.10    -0.10     0.09     0.07    -0.25     0.32     1.00    -0.17     0.06    -0.40
  sigma_3    -0.11     0.05     0.41    -0.38    -0.04    -0.63    -0.17     1.00    -0.39     0.57
  sigma_4    -0.29    -0.05     0.04     0.21     0.19    -0.01     0.06    -0.39     1.00    -0.73
  sigma_5     0.10     0.20     0.23    -0.58    -0.05    -0.44    -0.40     0.57    -0.73     1.00
=================================================================================================
```

Figure E1: Screen print of ascii file "MODELAVG_output.txt". This file is created by the toolbox and printed to the screen in the MATLAB editor. The notation that is used in this Table matches exactly the names of the variables used in the Equations and main text.

The toolbox also presents to the user a large number of figures that visualize the results. I now display all these figures, two per page.

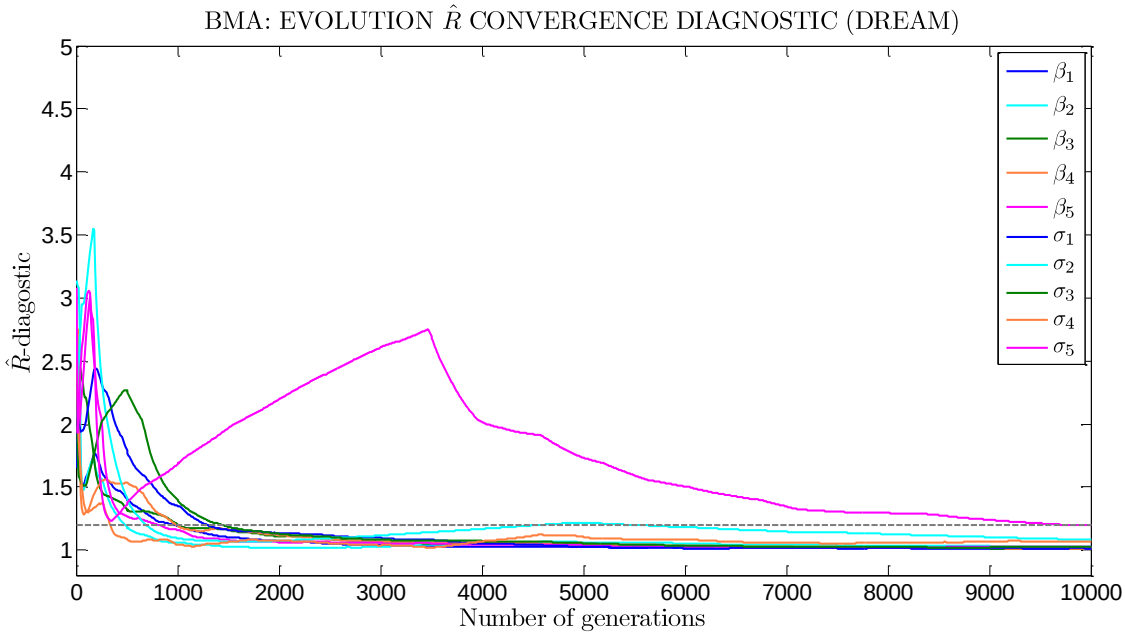


Figure E2

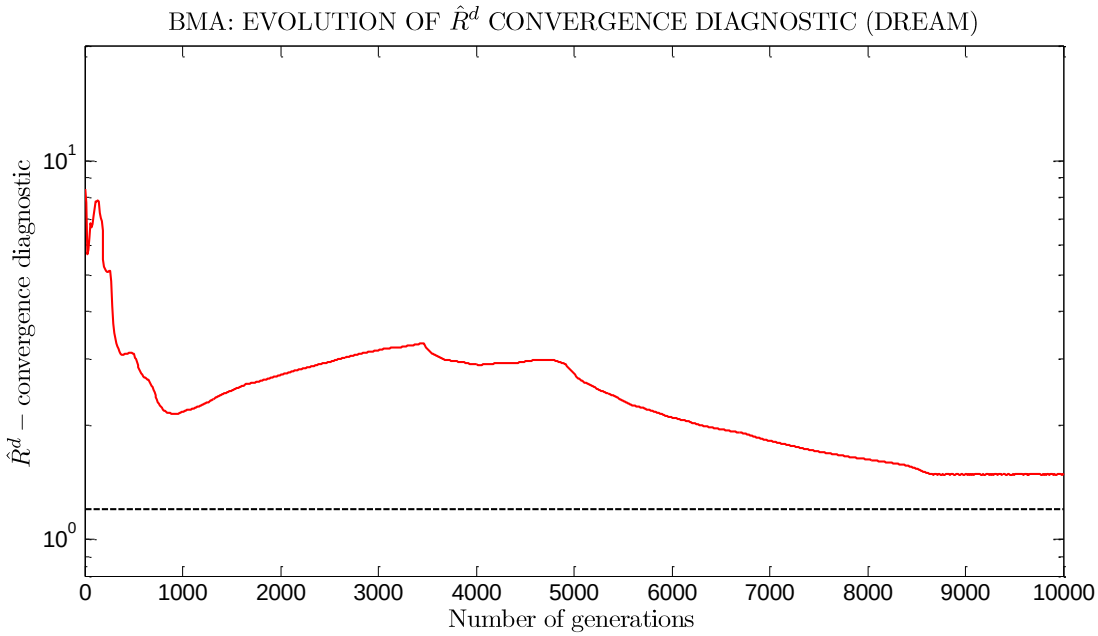


Figure E3

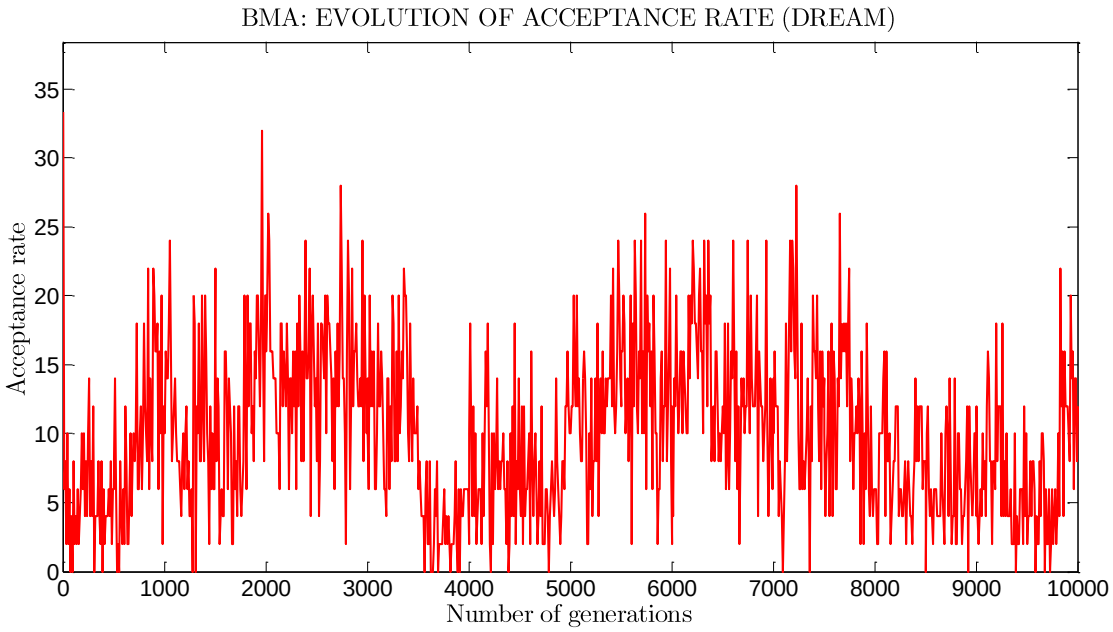


Figure E4

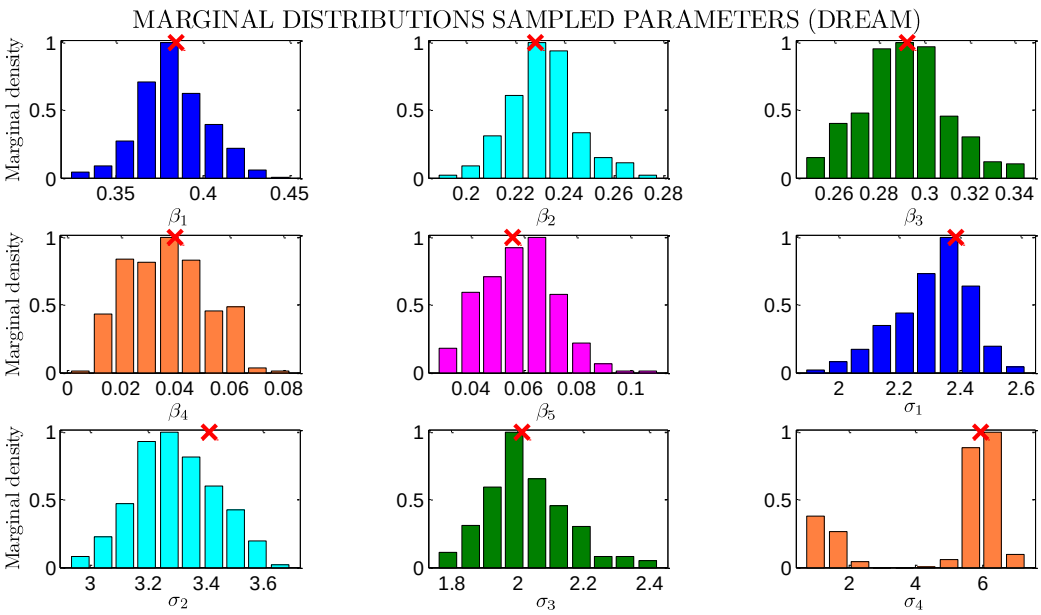


Figure E5

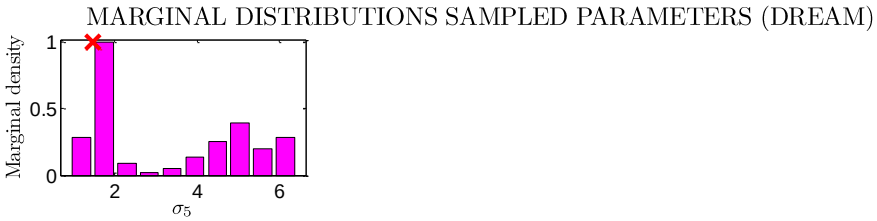


Figure E6

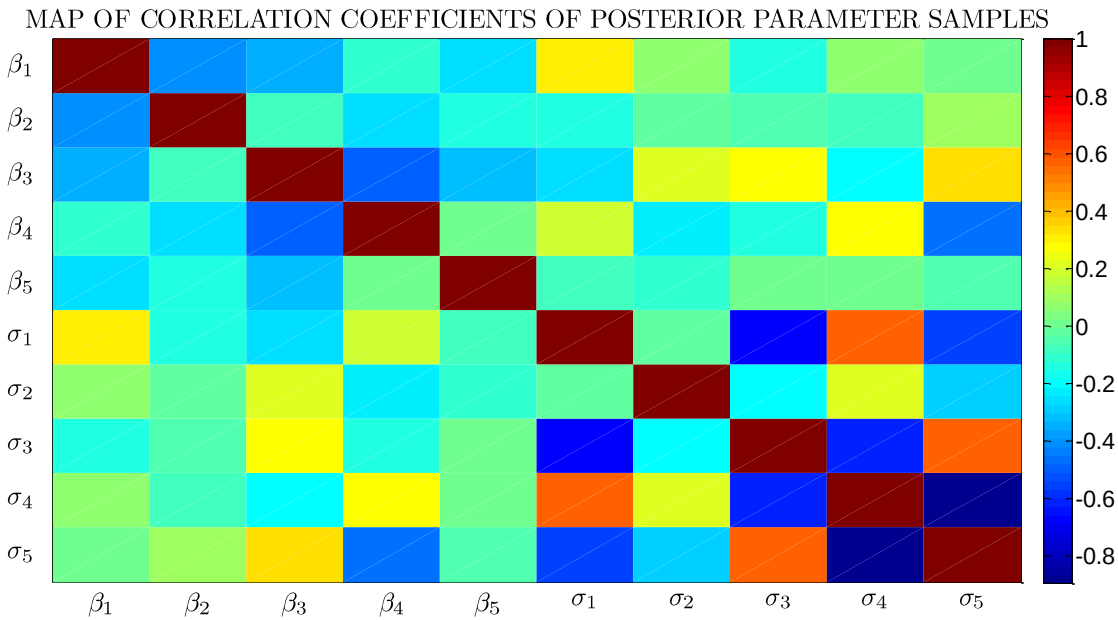


Figure E7

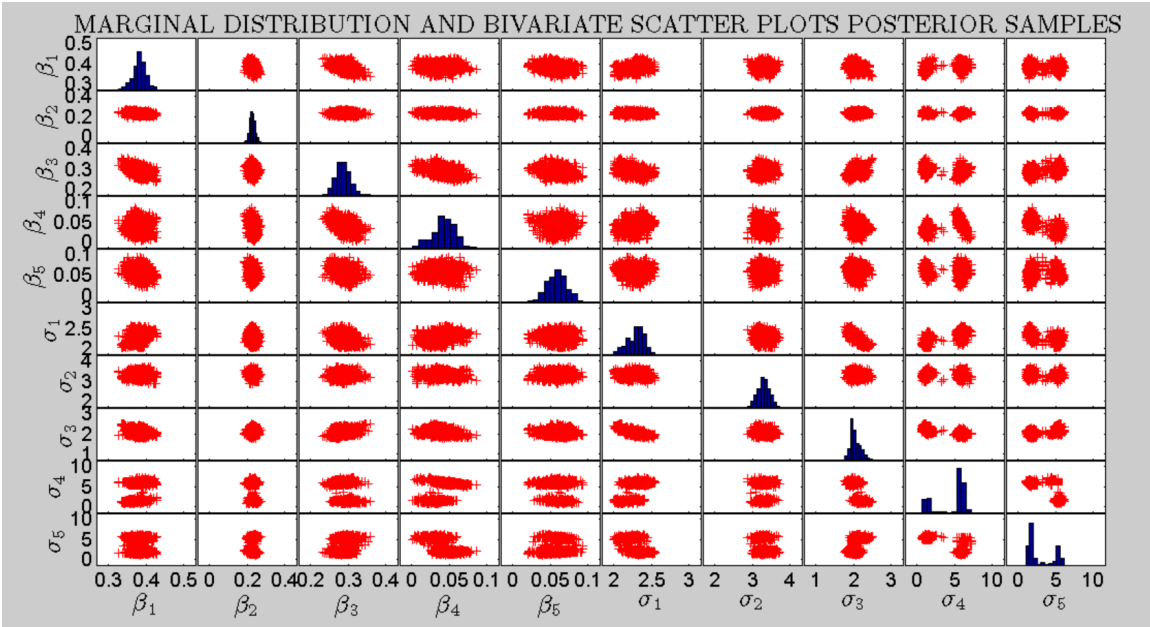


Figure E8

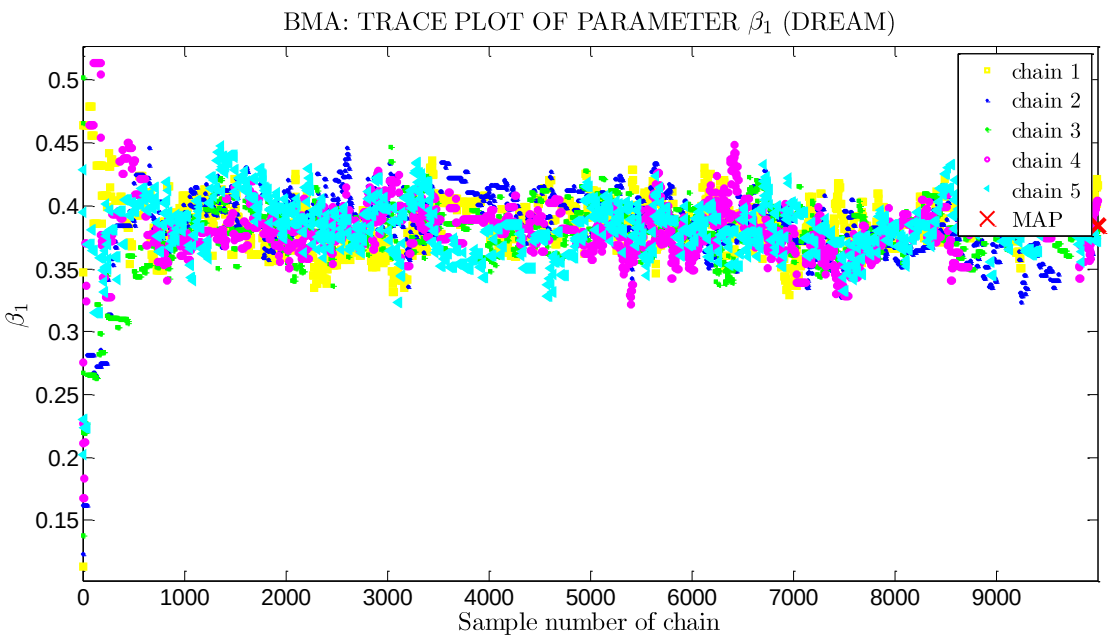


Figure E9

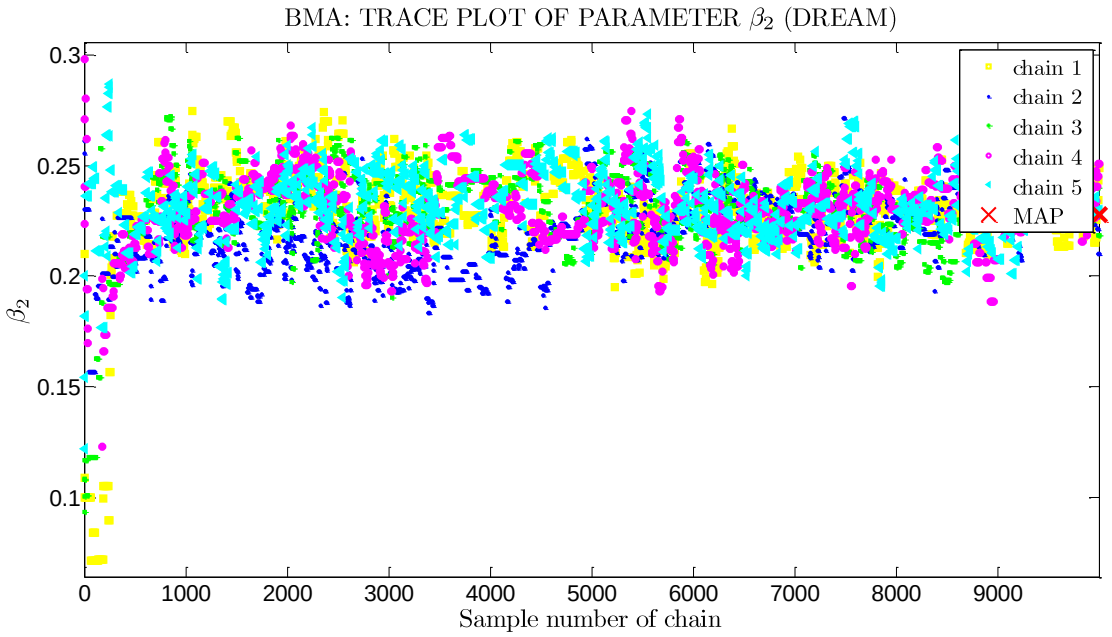


Figure E10

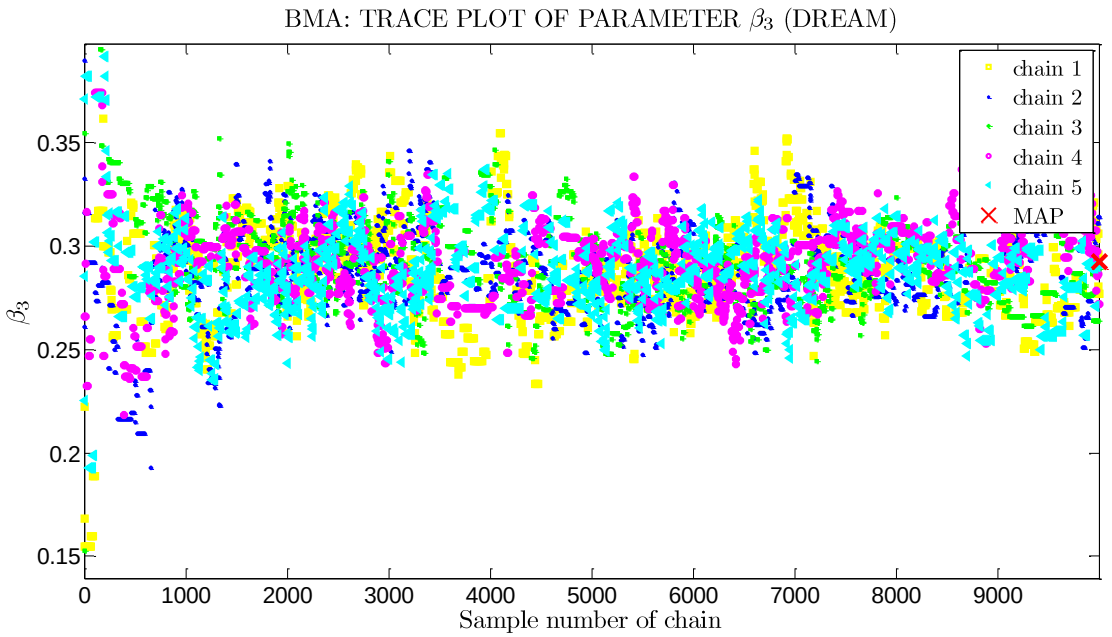


Figure E11

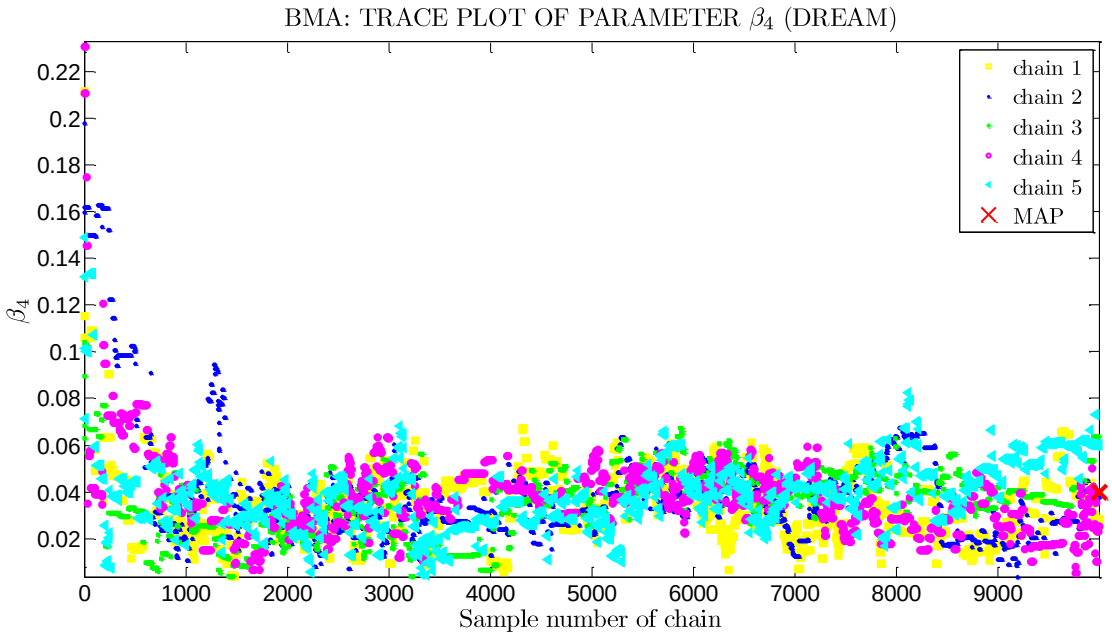


Figure E12

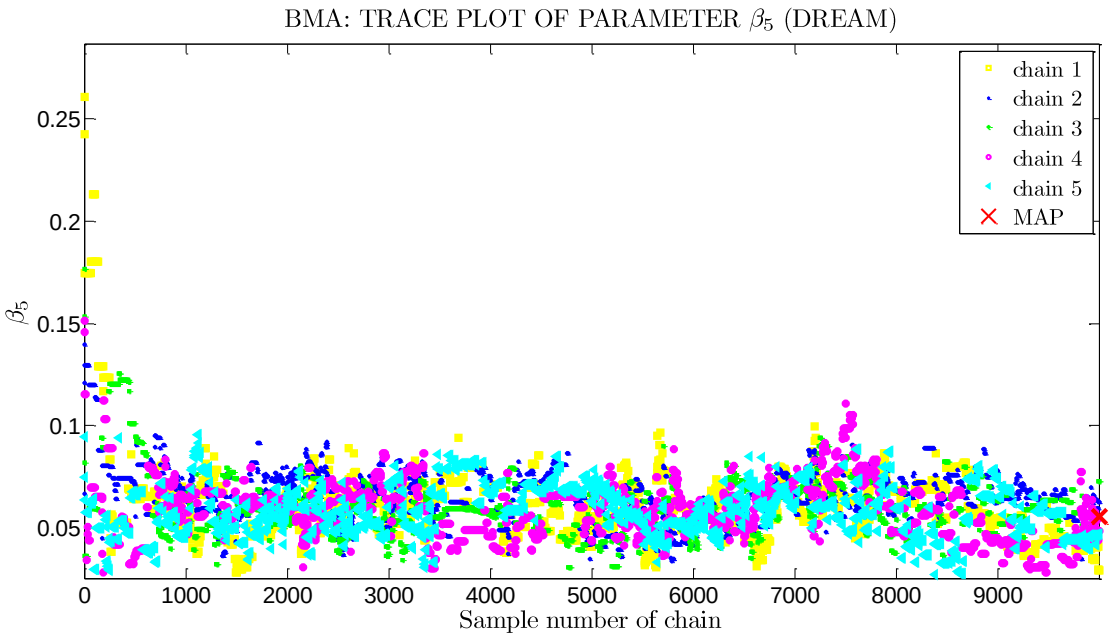


Figure E13

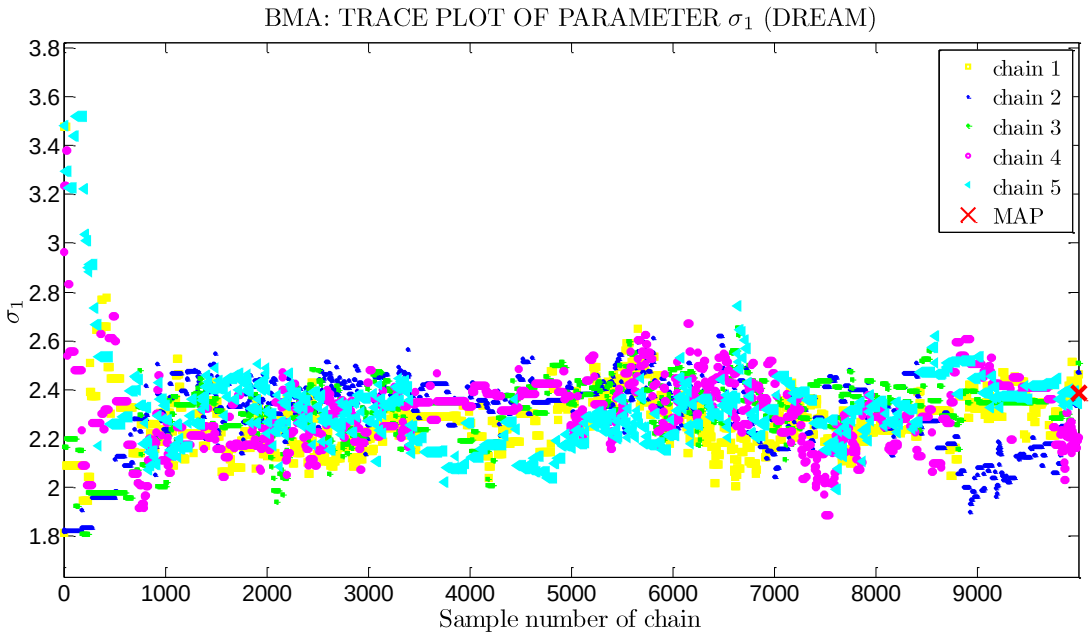


Figure E14

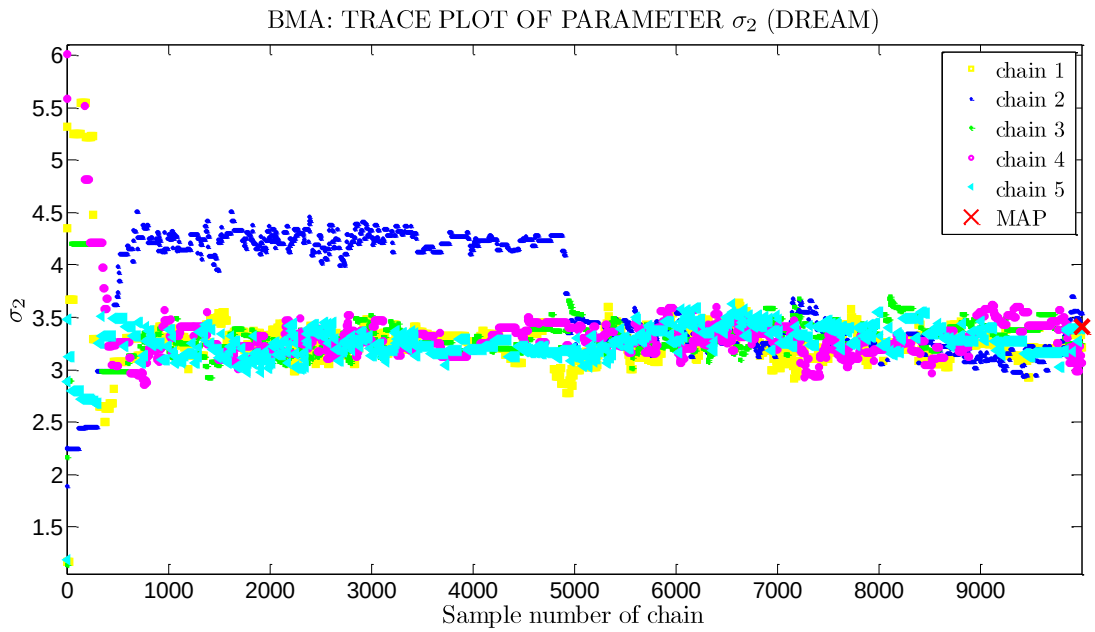


Figure E15

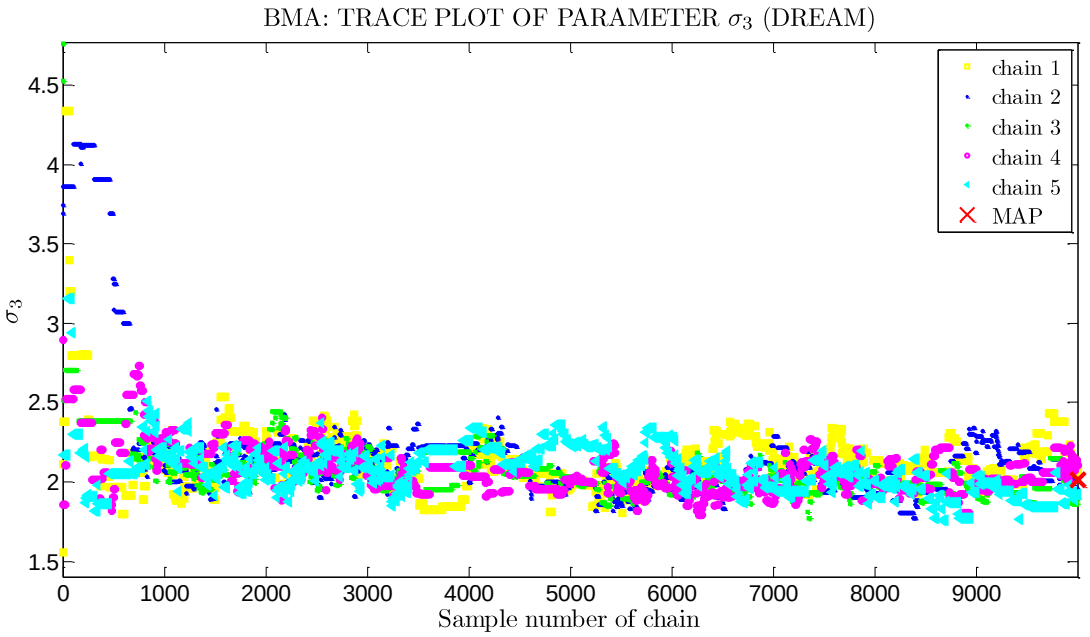


Figure E16

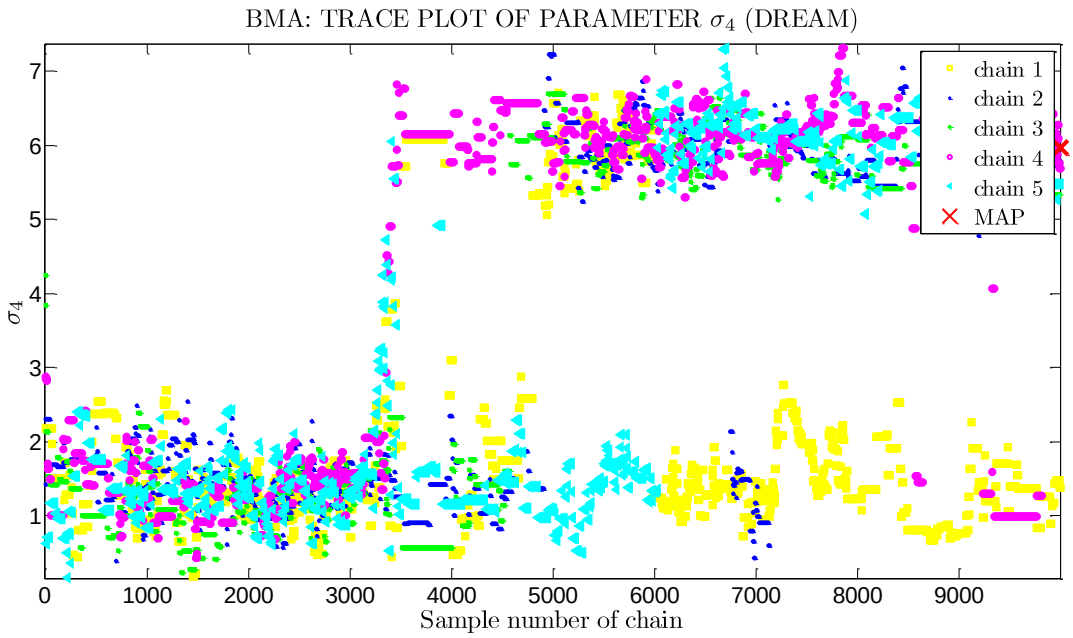


Figure E17

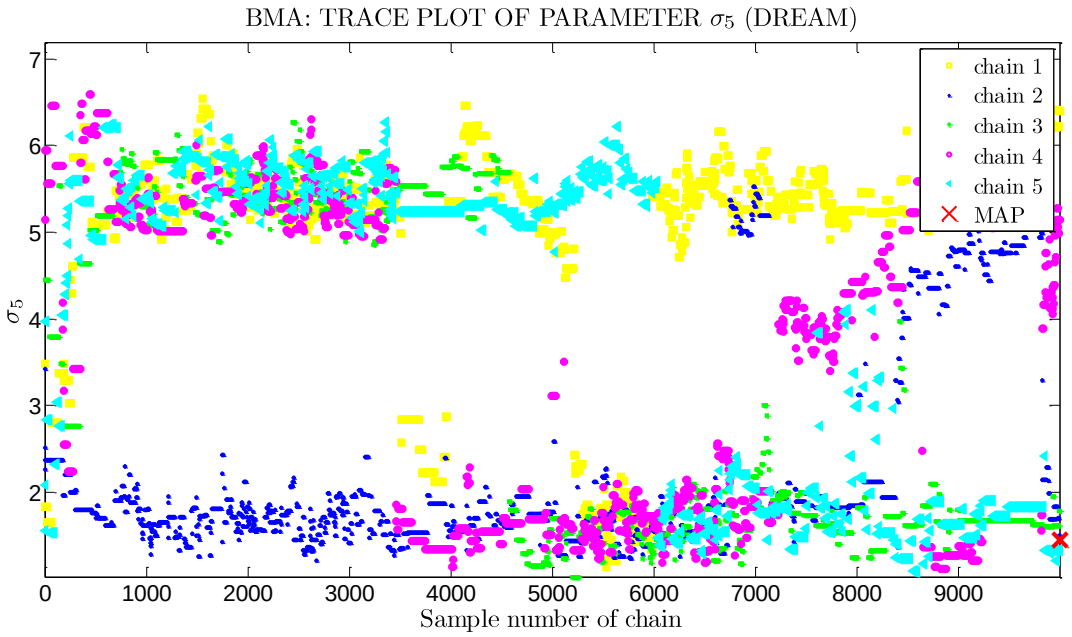


Figure E18

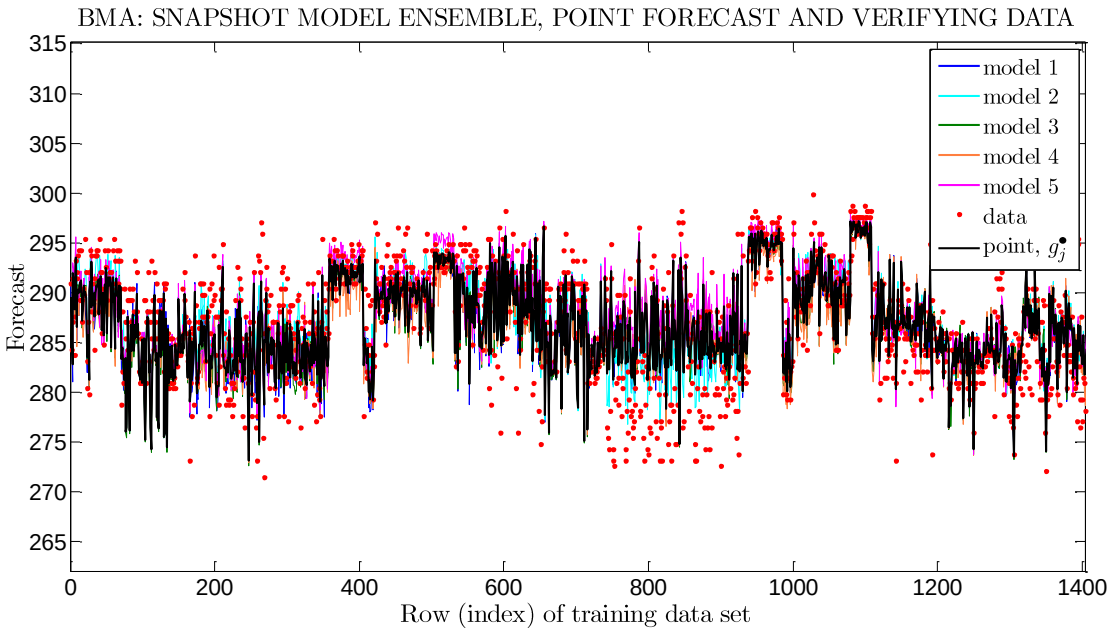


Figure E19

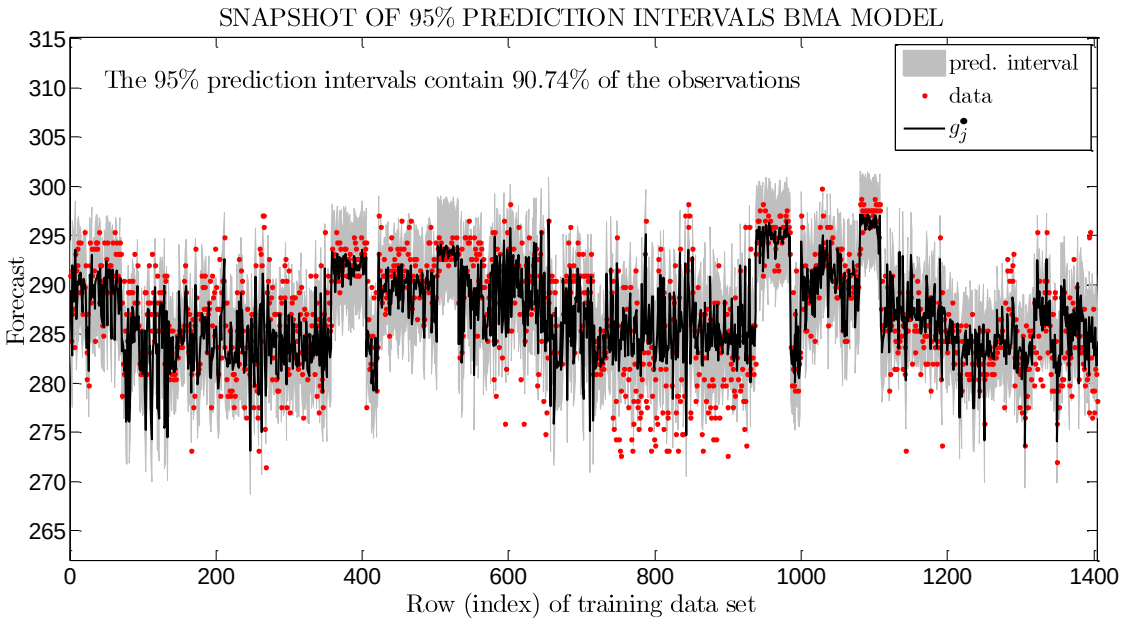


Figure E20

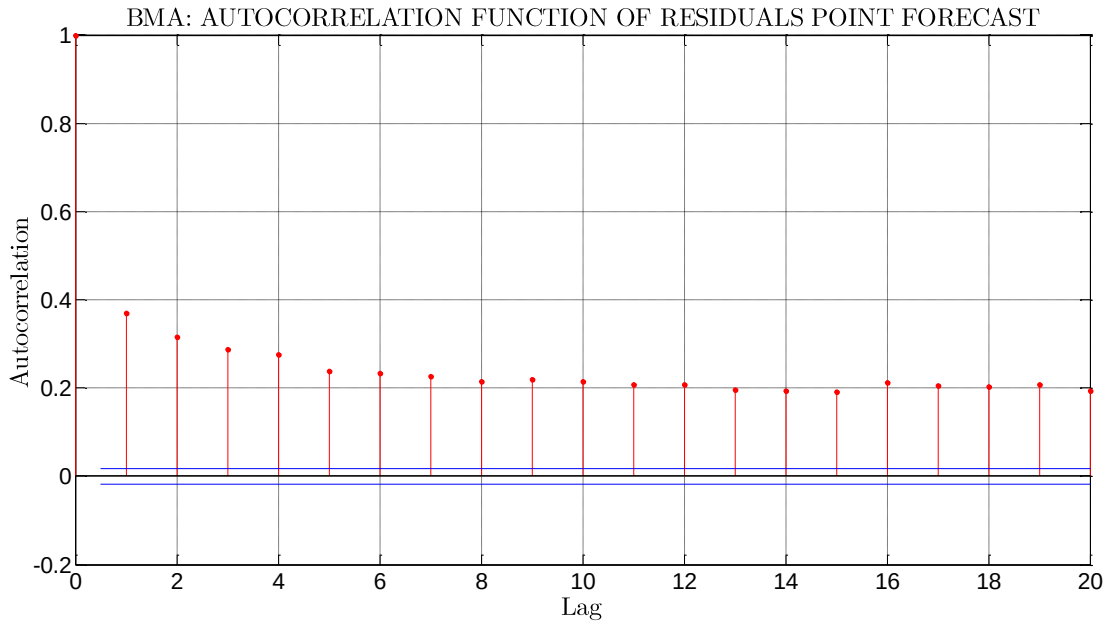


Figure E21

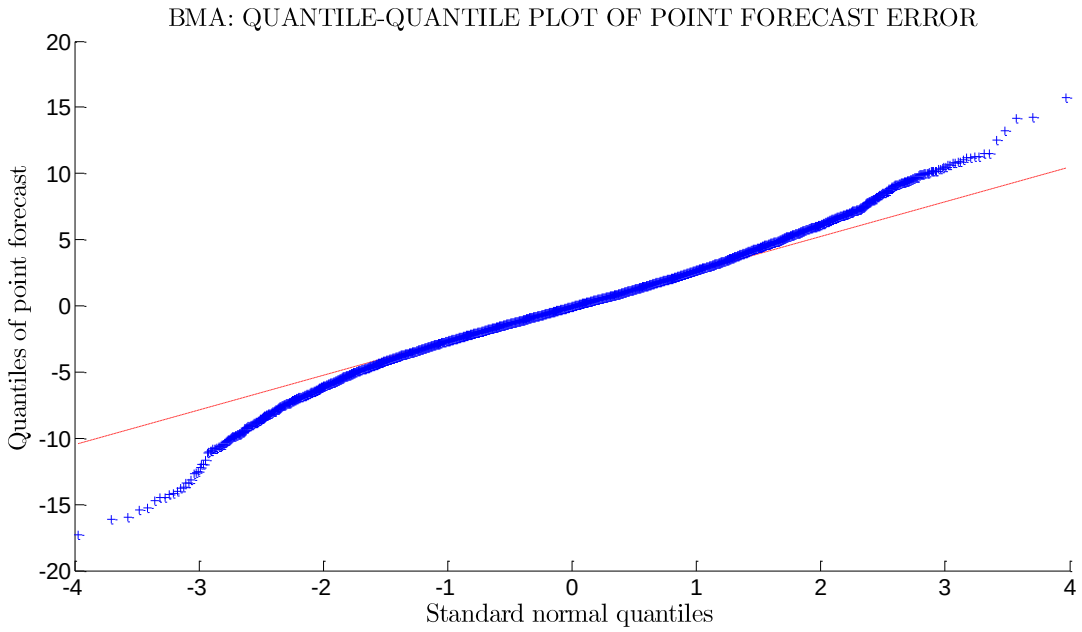


Figure E22

Appendix F. References

J.M. Bates and C.M.W. Granger, "The combination of forecasts," *Operations Research Quarterly*, vol. 20, pp. 451-468, 1969.

C.H. Bishop and K.T. Shanley, "Bayesian modeling averaging's problematic treatment of extreme weather and a paradigm shift that fixes it," *Monthly Weather Review*, vol. 136, pp. 4641-4652, 2008.

G.E.P. Box, and D.R. Cox, "An analysis of transformations," *Journal of the Royal Statistical Society*, Series B, vol. 26 (2), pp. 211-252, 1964.

C.J.F. ter Braak, "A Markov chain Monte Carlo version of the genetic algorithm differential evolution: easy Bayesian computing for real parameter spaces," *Statistics & Computing*, vol. 16, pp. 239-249, 2006.

C.J.F. ter Braak, and J.A. Vrugt, "Differential evolution Markov chain with snooker updater and fewer chains," *Statistics & Computing*, vol. 18 (4), pp. 435-446, doi:10.1007/s11222-008-9104-9, 2008.

S.P. Brooks, and A. Gelman, "General methods for monitoring convergence of iterative simulations," *Journal of Computational and Graphical Statistics*, vol. 7, pp. 434-455, 1998.

S.T. Buckland, K.P. Burnham, and N.H. Augustin, "Model selection: An integral part of inference," *Biometrics*, vol. 53, pp. 603-618, 1997.

K.P. Burnham, and D.R. Anderson, "Model selection and multimodel inference: A practical information-theoretic approach," 2nd edition, Springer, New York, 2002.

A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. 39(B), pp. 1-39, 1977.

C.G.H. Diks, and J.A. Vrugt, "Comparison of point forecast accuracy of model averaging methods in hydrologic applications," *Stochastic Environmental Research and Risk Assessment*, 24(6), pp. 809-820, doi:10.1007/s00477-010-0378-z, 2010.

Q.Y. Duan, S. Sorooshian, and V.K. Gupta, "Effective and efficient global optimization for conceptual rainfall-runoff models," *Water Resources Research*, 28 (4), pp. 1015-1031, doi:10.1029/91WR02985, 1992.

A.G. Gelman, and D.B. Rubin, "Inference from iterative simulation using multiple sequences," *Statistical Sciences*, vol. 7, pp. 457-472, 1992.

T. Gneiting, A.E. Raftery, A.H. Westveld, and T. Goldman, "Calibrated probabilistic forecasting using ensemble model output statistics and CRPS estimation," *Monthly Weather Review*, vol. 133, pp. 1098-1118, 2005.

C.W.J. Granger and R. Ramanathan, "Improved methods of combining forecast accuracy," *Journal of Forecasting*, vol. 3, pp. 197-204, 1984.

E.P. Grimit, and C.F. Mass, "Initial results of a mesoscale shortrange ensemble forecasting system over the Pacific Northwest", *Weather Forecasting*, vol. 17, pp. 192-205, 2002.

B.E. Hansen, "Least-squares model averaging," *Econometrica*, vol. 75, pp. 1175-1189, 2007.

B.E. Hansen, "Least-squares forecast averaging," *Journal of Econometrics*, vol. 146, pp. 342-350, 2008.

J.A. Hoeting, D. Madigan, A.E. Raftery, and C.T. Volinsky, "Bayesian model averaging: A tutorial," *Statistical Science*, vol. 14, pp. 382-417, 1999.

J. Geweke, "Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments," in Bayesian Statistics 4, edited by J.M. Bernardo, J.O. Berger, A.P. Dawid, and A.F.M. Smith, pp. 169-193, Oxford University Press, 1992.

E. Laloy, and J.A. Vrugt, "High-dimensional posterior exploration of hydrologic models using multiple-try DREAM$_{(ZS)}$ and high-performance computing," *Water Resources Research*, vol. 48, W01526, doi:10.1029/2011WR010608, 2012.

G. McLachlan, and T. Krishnan, The EM algorithm and extensions: 2nd Edition, 400 pages, Wiley, Apr. 2008.

S.P. Neuman "Maximum likelihood Bayesian averaging of uncertain model predictions," *Stochastic Environmental Research and Risk Assessment*, vol. 17, pp. 291-305, 2003.

K.V. Price, R.M. Storn, and J.A. Lampinen, Differential evolution, A practical approach to global optimization, Springer, Berlin, 2005.

A.E. Raftery, and S.M. Lewis, "One long run with diagnostics: Implementation strategies for Markov chain Monte Carlo," *Statistical Science*, vol. 7, pp. 493-497, 1992.

A.E. Raftery, and S.M. Lewis, "The number of iterations, convergence diagnostics and generic Metropolis algorithms," in Practical Markov chain Monte Carlo, edited by W.R. Gilks, D.J. Spiegelhalter and S. Richardson, London, U.K., Chapman and Hall, 1995.

A.E. Raftery, D. Madigan, and J.A. Hoeting, "Bayesian model averaging for linear regression models," *Journal of the American Statistical Association*, vol. 92, pp. 179-191, 1997.

A.E. Raftery, T. Gneiting, F. Balabdaoui, and M. Polakowski, "Using Bayesian model averaging to calibrate forecast ensembles," *Monthly Weather Review*, vol. 133, pp. 1155-1174, 2005.

J. Rings, J.A. Vrugt, G. Schoups, J.A. Huisman, and H. Vereecken, "Bayesian model averaging using particle filtering and Gaussian mixture modeling: Theory, concepts, and simulation experiments," *Water Resources Research*, 48, W05520, doi:10.1029/2011WR011607, 2012.

C.P. Roberts, and G. Casella, "Monte Carlo statistical methods," 2nd edition, Springer, New York, 2004.

M. Sadegh, and J.A. Vrugt, "Approximate Bayesian computation using Markov chain monte Carlo simulation: DREAM$_{(ABC)}$," *Water Resources Research*, vol. 50, doi:10.1002/2014WR015386, 2014.

J.M. Sloughter, A.E. Raftery, T. Gneiting, and C. Fraley, "Probabilistic quantitative precipitation forecasting using Bayesian model averaging," *Monthly Weather Review*, vol. 135, pp. 3209-3220, 2007.

J.M. Sloughter, T. Gneiting, and A.E. Raftery, "Probabilistic wind speed forecasting using ensembles and Bayesian model averaging," *Monthly Weather Review*, vol. 105, no. 489, pp. 25-35, doi:10.1198/jasa.2009.ap08615, 2010.

R. Storn, and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341-359, 1997.

J.A. Vrugt, H.V. Gupta, W. Bouten, and S. Sorooshian, "A Shuffled Complex Evolution Metropolis algorithm for optimization and uncertainty assessment of hydrologic model parameters," *Water Resources Research*, vol. 39 (8), 1201, doi:10.1029/2002WR001642, 2003.

J.A. Vrugt, M.P. Clark, C.G.H. Diks, Q. Duan, and B.A. Robinson, "Multi-objective calibration of forecast ensembles using Bayesian model averaging," *Geophysical Research Letters*, vol. 33, L19817, doi:10.1029/2006GL027126.

J.A. Vrugt, and B.A. Robinson, "Treatment of uncertainty using ensemble methods: Comparison of sequential data assimilation and Bayesian model averaging," *Water Resources Research*, vol. 43, W01411, doi:10.1029/2005WR004838, 2007.

J.A. Vrugt, C.J.F. ter Braak, M.P. Clark, J.M. Hyman, and B.A. Robinson, "Treatment of input uncertainty in hydrologic modeling: Doing hydrology backward with Markov chain Monte Carlo simulation," *Water Resources Research*, vol. 44, W00B09, doi:10.1029/2007WR006720, 2008a.

J.A. Vrugt, C.G.H. Diks, and M.P. Clark, "Ensemble Bayesian model averaging using Markov chain Monte Carlo sampling," *Environmental Fluid Dynamics*, vol 8, pp. 579-595, 2008b.

J.A. Vrugt, C.J.F. ter Braak, C.G.H. Diks, D. Higdon, B.A. Robinson, and J.M. Hyman, "Accelerating Markov chain Monte Carlo simulation by differential evolution with self-adaptive randomized subspace sampling," *International Journal of Nonlinear Sciences and Numerical Simulation*, vol. 10, no. 3, pp. 273-290, 2009.

J.A. Vrugt, and C.J.F. ter Braak, "DREAM$_{(D)}$: an adaptive Markov chain Monte Carlo simulation algorithm to solve discrete, noncontinuous, and combinatorial posterior parameter estimation problems," *Hydrology and Earth System Sciences*, vol. 15, pp. 3701-3713, doi:10.5194/hess-15-3701-2011, 2011.

J.A. Vrugt, and M. Sadegh, "Toward diagnostic model calibration and evaluation: Approximate Bayesian computation," *Water Resources Research*, vol. 49, doi:10.1002/wrcr.20354, 2013.

J.A. Vrugt, "Multi-criteria Optimization using the AMALGAM software package: Theory, concepts, and MATLAB Implementation, "*Manual, Version 1.0*, pp. 1-53, 2015a.

J.A. Vrugt, "FDCFIT: A MATLAB Toolbox of parametric expressions of the flow duration curve," *Manual, Version 1.0*, pp. 1-35, 2015b.

J.A. Vrugt, "Markov chain Monte Carlo simulation using the DREAM software package: Theory, concepts, and MATLAB Implementation," *Environmental Modeling & Software*, vol. 75, pp. 273-316, 10.1016/j.envsoft.2015.08.013, 2016.

T. Wöhling, and J.A. Vrugt, "Combining multiobjective optimization and Bayesian model averaging to calibrate forecast ensembles of soil hydraulic models," *Water Resources Research*, vol. 44, W12432, pp. 1-18, 2008.

M. Ye, P.D. Meyer and S.P. Neumann, "On model selection criteria in multimodel analysis," *Water Resources Research*, vol. 44, W03428, pp. 1-12, 2008.

X. Zhang, A.T.K. Wan, and G. Zou, "Least squares model combining by Mallows criterion," *SSRN working paper*, 2008.