MODELAVG MANUAL

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*

Abstract

Multi-model averaging is currently receiving a surge of attention in the atmospheric, hydrologic, and statistical literature to explicitly handle conceptual model uncertainty in the analysis of environmental systems and derive predictive distributions of model output. Such density forecasts are necessary to help analyze which parts of the model are well resolved, and which parts are subject to considerable uncertainty. Yet, accurate point predictors are still desired in many practical applications as well. Here, I present a simple MATLAB toolbox for multimodel averaging of forecast ensembles. This toolbox, called MODELAVG implements a suite of different model averaging techniques, including (among others) equal weights averaging (EWA), Bates-Granger model averaging (BGA), Bayesian model averaging (BMA), Mallows model averaging (MMA), and Granger-Ramanathan averaging (GRA). The toolbox returns the posterior distribution of the weights and associated parameters of each model averaging method, along with graphical output of the results. Markov chain Monte Carlo simulation with the DREAM algorithm is used for Bayesian inference (*Vrugt*, 2015a). Three case studies involving forecast ensembles of hydrologic and meteorologic models are used to illustrate the main capabilities and functionalities of the MODELAVG toolbox.

Keywords: Forecast ensembles, Model averaging, Akaike's information criterion, Bayes information criterion, Equal weights averaging, Granger-Ramanathan averaging, Bates-Granger averaging, Mallows model

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)

averaging, Bayesian model averaging, Bayesian inference, likelihood function, Markov chain Monte Carlo simulation, DREAM

---

1. Introduction and Scope

Predictive uncertainty analyses is typically carried out using a single conceptual mathematical model of the system of interest, rejecting a priori valid alternative plausible models and possibly underestimating uncertainty in the model itself (*Raftery et al.*, 1999; *Hoeting et al.*, 1999; *Neuman*, 2003; *Raftery et al.*, 1999; *Vrugt et al.*, 2006). Model averaging is a statistical methodology that is frequently utilized in the statistical and meteorological literature to account explicitly for conceptual model uncertainty. The motivating idea behind model averaging is that, with various competing models at hand, each having its own strengths and weaknesses, it should be possible to combine the individual model forecasts into a single new forecast that, up to one's favorite standard, is at least as good as any of the individual forecasts. As usual in statistical model building, the aim is to use the available information efficiently, and to construct a predictive model with the right balance between model flexibility and overfitting. Viewed as such, model averaging is a natural generalization of the more traditional aim of model selection. Indeed, the model averaging literature has its roots in the model selection literature, which continues to be a very active area of research.

Figure 1 provides a simple overview of model averaging. Consider that at a given time we have available the output of multiple different models (calibrated or not). 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. Moreover, the density of the averaged model is hopefully a good estimator of the total predictive uncertainty.

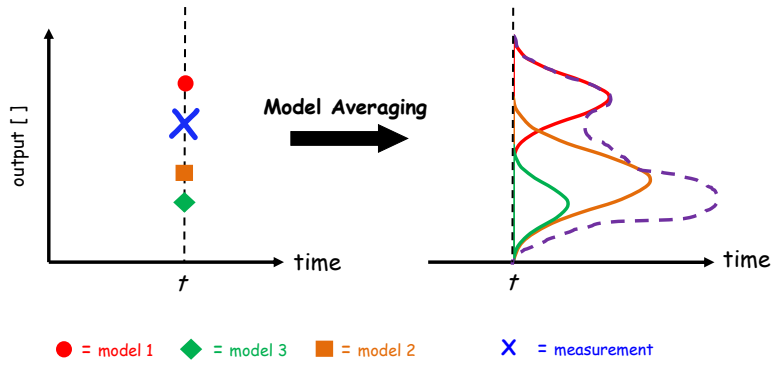

Figure 1: Schematic overview of model averaging using the outcome (simulations/forecasts/predictions) of three different (numerical) models. The premise is that a weighted average of these forecasts addresses explicitly conceptual model uncertainty, and hence is a better predictor of the observed data than any of the individual models themselves. This weighted average constitutes a point forecast (predictor). Some model averaging methods also estimate a predictive density, which allows for probabilistic forecasting and analysis of predictive uncertainty.

To formalize the various model averaging strategies considered herein, let us 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. Further assume that there is an ensemble of K different models available with associated point forecasts $D_{j,t}$, $k = \{1, \ldots, K\}$ and $j = \{1, \ldots, n\}$. A popular way to combine the point forecasts of the $n \times K$ matrix $\mathbf{D}$ is to consider the following linear model combining the individual predictions

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

where $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_K\}$ denotes the weight vector, and $\{\varepsilon_j\}$ is a white noise sequence, which will be assumed to have a normal distribution with zero mean and unknown variance.

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

$$\tilde{D}_{j,k} = a_k + b_k D_{j,k}, \tag{2}$$

will often suffice. The coefficients a_k and b_k for each of the models, $k = 1, \ldots, K$ are found by ordinary least squares using the simple regression

model

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

and the observations in the calibration set. Typically this bias correction leads to a small improvement of the predictive performance of the individual models, with a_k close to zero and b_k close to 1. If the calibration set is very small, the ordinary least squares estimates become unstable, and bias correction may distort the ensemble (*Vrugt and Robinson*, 2007a). 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 $D_{j,k}$ (rather than $\tilde{D}_{j,k}$) for the bias corrected predictors of $\tilde{y}_j$.

The point forecasts associated with model (1) are

$$y_j^e = \mathbf{D}_j^T \boldsymbol{\beta} = \sum_{k=1}^{K} \beta_k D_{j,k}, \tag{4}$$

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

In this manual, I discuss several model averaging methods for postprocessing of forecast ensembles. These methods are implemented in a MATLAB toolbox, and solved using Bayesian inference with DREAM (*Vrugt et al.*, 2008a, 2009; *Vrugt*, 2015a). The different utilities and functionalities of the toolbox are illustrated using two different case studies involving discharge data and surface temperature and sea level pressure. These example studies are easy to run and adapt and serve as templates for other data sets. The present manual has elements in common with the toolboxes of DREAM (*Vrugt*, 2015a), AMALGAM (*Vrugt*, 2015b) and FDCFIT (*Vrugt*, 2015c) and is specifically developed to help users with the postprocessing of forecast ensembles.

The remainder of this paper is organized as follows. Section 2 discusses the different model averaging that are available to the user. This is followed in section 3 with a description of the MATLAB toolbox MODELAVG. In this section we are especially concerned with the input and output arguments of MODELAVG and the various utilities and options available to the user. Section 4 discusses two case studies which illustrate how to use the toolbox. The penultimate section of this paper (section 5) highlights recent research efforts aimed at further improving model averaging results. Finally, section 6 concludes this manual with a summary of the main findings.

2. Model Averaging - Different Methods

The MATLAB toolbox MODELAVG implements the following model averaging techniques: equal weights averaging (EWA) where each of the available models is weighted equally, model averaging with Bates-Granger (BGA) (*Bates and Granger*, 1969), AIC and BIC-based model averaging (AICA and BICA, respectively) (*Buckland et al.*, 1997; *Burnham and Anderson*, 2002; *Hansen*, 2008), Bayesian model averaging (BMA) (*Raftery et al.*, 1999; *Hoeting et al.*, 1999; *Raftery et al.*, 1999), Mallows model averaging (MMA) (*Hansen*, 2007, 2008) and weights equal to the ordinary least squares estimates of the coefficients of a multiple linear regression model, as first suggested for forecasting by *Granger and Ramanathan* (1984), and referred to here as Granger-Ramanathan averaging (GRA). Note that some of these model averaging techniques only allow for positive weights of the constituent models of the ensemble and that sum to one, $\{\boldsymbol{\beta}|\beta_k \geq 0, k = 1, \ldots, K$ and $\sum_{k=1}^{K} \beta_k = 1\}$. This is also referred to as weights in the unit simplex $\Delta^{K-1} = \left\{\beta_k \in [0,1]^K : \sum_{k=1}^{K} \beta_k = 1\right\}$ in $\mathbb{R}^K$. Other methods relax this assumption and allow for positive and negative values of $\boldsymbol{\beta}$.

2.1. Equal weights averaging

Under equal weights averaging (EWA), the combined forecast is simply obtained by giving each of the models of the ensemble a similar weight, $\boldsymbol{\beta}_{\text{EWA}} = \left(\frac{1}{K}, \ldots, \ldots, \frac{1}{K}\right)$. These weights are independent of the training data, $\tilde{\mathbf{Y}}$ and results in the following forecast, $y_j^e = \frac{1}{K}\sum_{k=1}^{K} D_{j,t}$ 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, $\hat{\sigma}_k^2 = \frac{1}{n}\sum_{j=1}^{n}(\tilde{y}_j - D_{j,k})^2$. 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 add up to one (and thus lie on Δ^{K-1}) we normalize the weights using

$$\beta_{\text{BGA},k} = \frac{1/\hat{\sigma}_k^2}{\sum_{k=1}^{K} 1/\hat{\sigma}_k^2} \tag{5}$$

In the remainder of this paper, BGA is used as acronym 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

$$\beta_{\mathrm{ICA},k} = \frac{\exp\left(-\frac{1}{2}I_k\right)}{\sum_{k=1}^{K}\exp\left(-\frac{1}{2}I_k\right)}, \tag{6}$$

where I_k is an information criterion that depends on the model complexity and data fit

$$I_k = -2\log(L_k) + q(p_k), \tag{7}$$

where L_k is the maximum likelihood of model k, and $q(p_k)$ is a term that penalizes for the number of model parameters. We consider 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. We refer to the model averaging scheme (6) based on IC and BIC as AICA and BICA, respectively, and to their respective $\boldsymbol{\beta}$-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. To evaluate the information criteria numerically, it is convenient to assume, as we do here, that the errors of the individual models are normally distributed. In that case, the log-likelihood of the kth model of the ensemble, $\log(L_k)$ can be calculated from

$$-2\log(L_k) = n\log\hat{\sigma}_k^2 + n \tag{8}$$

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 the use of OLS estimators within the linear regression model.

Granger and Ramanathan (1984) suggest using OLS to estimate the unknown parameters (weights) of the linear regression model (1)

$$\boldsymbol{\beta}_{\mathrm{GRA}} = \left(\mathbf{D}^T\mathbf{D}\right)^{-1}\mathbf{D}^T\tilde{\mathbf{Y}}, \tag{9}$$

where $\mathbf{D}$ is the $n \times K$ matrix of ensemble forecasts and $\tilde{\mathbf{Y}}$ signifies the $n \times 1$ vector of observations of the calibration data set. The OLS estimator can be

shown to be the best linear unbiased estimator of $\boldsymbol{\beta}$. We 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 can be viewed as a possible way to deal with model uncertainty. It offers an alternative to the selection of a single model from a number of candidate models, by weighting each candidate model according its statistical evidence. Applications of BMA in hydrology and meteorology have been described by *Raftery et al.* (1999), *Gneiting et al.* (2005), *Vrugt and Robinson* (2007a) and *Vrugt et al.* (2008b). See *Bishop and Shanley* (1969) for a recent contribution that improves the performance of BMA for extreme weather forecasting (see also *Vrugt et al.* (2006)).

Depending on the type of application one has in mind, different flavors of BMA can be used. For instance, it makes a crucial difference whether one would like to average point forecasts (in which case some forecasts may be assigned negative weights) or density forecasts (in which negative weights could lead to negative forecast densities). We now describe the most popular BMA method which is particularly useful when dealing with the output of dynamic simulation models.

If we assume a conditional density, $f_{j,k}(\cdot)$ that is centered around the forecasts of each of the individual models of the ensemble, we can derive the combined forecast density as follows (see also right-hand-side of Figure 1)

$$g_j(\tilde{y}_j) = \sum_{k=1}^{K} \beta_k f_{j,k}(\tilde{y}_j) \tag{10}$$

A popular choice for $f_{j,k}(\cdot)$ is a normal distribution with mean $D_{j,k}$ and standard deviation, σ,

$$f_{j,k}(\tilde{y}_j) = \frac{1}{\sqrt{2\pi}\sigma} \exp\Big(-\frac{1}{2}\sigma_k^{-2}(\tilde{y}_j - D_{j,k})^2\Big), \tag{11}$$

and thus the BMA predictive density, $g_j(\tilde{y}_j)$ consists of a mixture of normal distributions, each centered around their individual point forecast $D_{j,k}$. To ensure that $g_j(\tilde{y}_j)$ represents a proper density, the BMA weights are assumed to lie on the unit simplex, Δ^{K-1} in $\mathbb{R}^K$. The BMA point predictor is simply

a weighted average of the individual models of the ensemble

$$y_j^e = \sum_{k=1}^{K} \beta_k D_{j,k} \tag{12}$$

To estimate the BMA weights of each member of the ensemble and the variance of the Gaussian forecast distributions, the following optimization problem needs to be solved

$$\left(\widehat{\boldsymbol{\beta}}_{\mathrm{BMA}}, \widehat{\boldsymbol{\sigma}}_{\mathrm{BMA}}\right) = \underset{\boldsymbol{\beta}\in\Delta^{K-1}, \sigma\in\mathbb{R}_+^K}{\arg\max} \sum_{j=1}^{n} \log\left\{\sum_{k=1}^{K} \beta_k f_{j,k}(\tilde{y}_j)\right\}, \tag{13}$$

which requires an iterative solution. In their seminal paper, *Raftery et al.* (1999) recommends using the Expectation-Maximization (EM) algorithm for BMA model training, even though global convergence of this algorithm cannot be guaranteed. What is more, algorithmic modifications are required to adapt the EM method for predictive distributions other than the normal distribution, considered in Equation (11). This will be necessary for variables whose probability density functions deviate considerably from normality (*Vrugt and Robinson*, 2007a). Examples include wind speed (*Sloughter et al.*, 2010), streamflow (*Vrugt and Robinson*, 2007a) and precipitation (*Sloughter et al.*, 2007).

Vrugt et al. (2008b) presents an alternative method for BMA model training using Markov chain Monte Carlo (MCMC) simulation with DREAM. This approach overcomes some of the limitations of the EM approach and has three distinct advantages. First, MCMC simulation does not require algorithmic modifications when using different conditional probability distributions for the individual ensemble members. Second, MCMC simulation provides a full view of the posterior distribution of the BMA weights and variances. This information is helpful to assess the usefulness of individual ensemble members. A small ensemble has important computational advantages, since it requires calibrating and running the smaller possible number of models. Finally, MCMC simulation with DREAM can handle large ensemble sizes with predictions of many different constituent models.

The MATLAB toolbox presented herein includes different options for the conditional distribution, $f_{j,k}(\cdot)$ of each member of the ensemble. These options include a Gaussian distribution with heteroscedastic error variance and the Gamma distribution, and increase flexibility of application of BMA to

variables that deviate considerably from a normal distribution. A simple extension of Equation (11) is to use a different variance for each member of the ensemble, and thus to estimate K different variances, $\sigma_k; k = 1, \ldots, K$ in Equations (11) and (13). A heteroscedastic error variance of Equation (11) is easily adopted if we assume

$$\sigma_{j,k}^2 = cD_{j,k}, \tag{14}$$

where c is a coefficient that applies to all models of the ensemble, and whose value is estimated along with the BMA weights using maximization of Equation (13). Alternatively, the gamma conditional distribution can be used

$$f_{j,k}(\tilde{y}_j) \sim \frac{1}{\kappa\Gamma(\alpha)}\tilde{y}_j^{(\alpha-1)}\exp(-\tilde{y}_j/\kappa), \tag{15}$$

where κ and α are a shape and scale parameter, respectively and $f_{j,k}(\tilde{y}_j) = 0$ if $\tilde{y}_j \leq 0$. The mean of this distribution is $\mu = \kappa\alpha$ and variance equivalent to $\sigma^2 = \kappa\alpha^2$. For each member of the ensemble we can set

$$\mu_{j,k} = D_{j,k} \qquad , \qquad \sigma_{t,k}^2 = c_{1,k}D_{j,k} + c_2, \tag{16}$$

and derive the $K+1$ coefficients, $c_{1,k}$ and c_2; $k = 1, \ldots, K$ from the training ensemble and verifying observations (*Vrugt and Robinson*, 2007a; *Sloughter et al.*, 2010).

The MATLAB function `BMA_calc` returns as output argument the log-likelihood of the BMA model in Equation (13) for a given vector, $\mathbf{x}$ (input argument) of weights and variances (or proxies thereof).

```
function [ log_L ] = BMA_calc ( x );
% This function calculates the log likelihood corresponding to the weights and sigma's

global BMA          % Request the BMA structure
L = 0;              % Set likelihood equivalent to zero
beta = x(1:BMA.K);  % Unpack weights

switch BMA.PDF      % Now check which BMA model is used

    case {'normal'}             % Normal distribution with homoscedastic error variance

        if strcmp(BMA.VAR,'single');                % One or multiple variances?
            sigma = x(BMA.K+1) * ones(1,BMA.K);
        elseif strcmp(BMA.VAR,'multiple');
            sigma = x(BMA.K + 1 :  end);
        else
            error('do not know this option for variance treatment')
        end

        for k = 1:BMA.K,                            % Mixture model
            L = L + beta(k)*exp(-1/2*((BMA.Y-BMA.D(:,k))./sigma(k)).^2)./ ...
                (sqrt(2*pi).*sigma(k));             % Now calculate likelihood
        end

    case {'heteroscedastic'}    % Normal distribution with heteroscedastic error variance

        c = x(BMA.K+1);                             % Unpack variance parameter
        for k = 1:BMA.K,                            % Mixture model
            sigma = abs(c*BMA.D(:,k));              % Calculate measurement error of data
            L = L + beta(k)*exp(-1/2*((BMA.Y-BMA.D(:,k))./sigma).^2)./ ...
                (sqrt(2*pi).*sigma);                % Calculate likelihood
        end

    case {'gamma'}              % Gamma distribution

        c1 = x(BMA.K+1:2*BMA.K); c2 = x(2*BMA.K+1); % Unpack variables gamma distribution
        for k = 1:BMA.K,                            % Mixture model
            mu = abs(BMA.D(:,k));                   % Derive mean of gamma distribution
            var = abs(c2 + c1(k) * BMA.D(:,k));     % Derive variance of gamma distribution
            A = mu.^2./var; B = var./mu;            % Derive A and B of gamma distribution
            z = BMA.Y./B;                           % Compute help variable
            u = (A - 1).*log(z) - z - gammaln(A);   % Compute help variable
            L = L + beta(k) * (exp(u)./B);          % Calculate likelihood
        end

end

L(L==0) = 1e-300;       % Replace zero likelihood with 1e-300
log_L = sum(log(L));    % Now compute the log-likelihood of the BMA model
```

Figure 2: MATLAB code that calculates the log-likelihood (return argument) of the BMA model for given values of the weights and variances of the conditional distribution, encapsulated in the vector (and input argument) `x`. Notation is consistent with main text. To minimize the CPU-time, vectorization is used of each member of the ensemble. Built-in functions are highlighted with a low dash. The different options (cases) for the conditional forecast distribution of each member of the ensemble are grouped under the `switch` statement. The function `global` request the values of the different fields of the BMA structure - those fields and their entries have been defined in the main function MODELAVG. The function `strcmp(S1,S2)` compares the strings S1 and S2 and returns logical 1 (true) if they are identical, and returns logical 0 (false) otherwise. `exp()` computes the exponential, and `sum()` calculates the sum of a vector of log-likelihood values.

The log-likelihood of the BMA model are thus computed as the log of the sum of the likelihoods of each of the different members of the ensemble. It is rather straightforward for the user to add additional options (cases) for the probability density function of the forecast distribution.

Interested readers are referred to the DREAM manual and toolbox (*Vrugt*, 2015a) which demonstrates an application of the BMA methodology using several formulations for the condition forecast distribution, $f(\cdot)$. A multicriteria BMA optimization framework was introduced by (*Vrugt et al.*, 2006) and provides insights into the trade-offs of fitting of different objectives. The AMALGAM manual (*Vrugt*, 2015b) provides an example of this approach.

2.6. Mallows model averaging

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

$$C_n(\boldsymbol{\beta}) = \sum_{j=1}^{n} \left(\tilde{y}_j - \boldsymbol{\beta}^T \mathbf{D}_j\right)^2 + 2\hat{\sigma}^2 \sum_{k=1}^{K} \beta_k p_k \tag{17}$$

where, as before, p_k denotes the number of parameters of the kth model of the ensemble, and $\hat{\sigma}^2$ is an estimate of the variance σ^2 of ε_j in (1). This value is often conveniently taken to be the forecast error variance of the most complex model (largest number of parameters) of the ensemble.

The Mallows criterion is

$$\widehat{\boldsymbol{\beta}}_{\text{MMA}} = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^K} C_n(\boldsymbol{\beta}), \tag{18}$$

where $\mathbb{R}^K$ signifies the feasible space of the weights. The value of $\widehat{\boldsymbol{\beta}}_{\text{MMA}}$ in Equation (17) can be found by maximizing the following log-likelihood function

$$\mathcal{L}(\boldsymbol{\beta}) \simeq -\frac{1}{2} C_n(\boldsymbol{\beta}), \tag{19}$$

using a nonlinear optimization (maximization) method or MCMC simulation with DREAM (*Diks and Vrugt*, 2010). Indeed, the maximum likelihood of Equation (19) can be found by identifying the point in the MCMC sample for which Equation (19) is largest.

We can also restrict the MMA weights to be positive and add up to one, and thus to lie on Δ^{K-1}. This modification requires a change to the prior

distribution of the weights, $\beta_k \in [0,1]; \sum_{k=1}^{K} \beta_k = 1$. This second MMA model averaging method with weights restricted to the Simplex is hereafter conveniently referred to as MMA$^{\Delta}$.

The MATLAB function `MMA_calc` listed below in Figure 3 calculates the log-likelihood of MMA in Equation 19 for a given vector, **x** of weights.

```
function [ log_L ] = MMA_calc ( beta );
% This function calculates the log likelihood using Mallows model averaging

global MMA          % Request the MMA structure

% Calculate the Mallows criterion -> Equation (11) of B.C. Hansen, "Least
% Squares Model Averaging", Econometrica, vol. 75, no. 4, pp. 1175-1189, 2007.
Cn = sum ( (MMA.Y - MMA.D*beta').^2 ) + 2 * (beta * MMA.p') * MMA.S2;

% Now calculate an approximate log-likelihood (without normalization constants)
log_L = -1/2 * Cn;
```

Figure 3: MATLAB code that calculates the log-likelihood (return argument) of the MMA model for given values of the weights, encapsulated in the vector (and input argument) `x`. Notation is consistent with Equation (19) in main text. To minimize the CPU-time, vectorization is used of each member of the ensemble. The function `global` request the values of the different fields of the `MMA` structure. These fields and their values have been defined in the function MODELAVG. Built-in functions are highlighted with a low dash. `sum()` calculates the sum of the squared difference between the MMA mean forecast and the verifying observations. The field `p` of structure `MMA` stores the number of parameters of each model of the ensemble and will be discussed later.

3. MODELAVG

We have developed a MATLAB program called MODELAVG which implements each of the model averaging methods described in section 2 and returns the values of the weights, $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_K\}$ and properties of the conditional forecast distribution, $f(\cdot)$ (in case of BMA) for each constituent member of the ensemble. For EWA, BGA, IACA, BICA and GRA, we have available a direct solution for the values of the weights, whereas an iterative solution using MCMC simulation with DREAM is used for BMA, MMA and MMA$^{\Delta}$. In this section, we briefly describe the DREAM algorithm (in words, equations and code), and introduce the MATLAB toolbox of MODELAVG.

3.1. Markov Chain Monte Carlo simulation with DREAM

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 m dimensions of the original parameter space, $\mathbb{R}^m \subseteq \mathbb{R}^d$, then a jump in the ith chain, $i = \{1, \ldots, N\}$ at iteration $t = \{2, \ldots, T\}$ is calculated using different evolution (*Storn and Price*, 1997; *Price et al.*, 2005)

$$\begin{aligned} \mathrm{d}\mathbf{x}^{i,A} &= (\mathbf{1}_m + \boldsymbol{\lambda}_m)\gamma_{(\delta,m)} \sum_{j=1}^{\delta} \left(\mathbf{x}_{t-1}^{\mathbf{r}_j^1,A} - \mathbf{x}_{t-1}^{\mathbf{r}_j^2,A} \right) + \boldsymbol{\zeta}_m \\ \mathrm{d}\mathbf{x}^{i,\neq A} &= 0, \end{aligned} \tag{20}$$

where $\gamma = 2.38/\sqrt{2\delta m}$ is the jump rate, δ denotes the number of chain pairs used to generate the jump (default is 3), and $\mathbf{r}^1$ and $\mathbf{r}^2$ are vectors consisting of δ integer values drawn without replacement from $\{1, \ldots, i-1, i+1, \ldots, N\}$.

The values of $\boldsymbol{\lambda}$ and $\boldsymbol{\zeta}$ are sampled independently from $\mathcal{U}_m(-c, c)$ and $\mathcal{N}_m(0, c^*)$ with, typically, $c = 0.1$ and c^* small compared to the width of the target distribution, $c^* = 10^{-12}$ say. With a probability of 20% we set the jump rate to 1, or $p_{(\gamma=1)} = 0.2$ to enable jumping between disconnected posterior modes. The candidate point of chain i at iteration t then becomes

$$\mathbf{x}_p^i = \mathbf{x}_{t-1}^i + \mathrm{d}\mathbf{x}^i, \tag{21}$$

and the Metropolis ratio is used to determine whether to accept this proposal or not.

In DREAM a geometric series of n_{cr} different crossover values is used, $\mathrm{CR} = \{\frac{1}{n_{\mathrm{cr}}}, \frac{2}{n_{\mathrm{cr}}}, \ldots, 1\}$. The selection probability of each crossover value is assumed equal at the start of simulation and defines a vector $\mathbf{p}_{\mathrm{cr}}$ with n_{cr} copies of $\frac{1}{n_{\mathrm{cr}}}$. For each different proposal the crossover, cr is sampled randomly from a discrete multinomial distribution, $\mathrm{cr} = \mathfrak{F}(\mathrm{CR}, 1, \mathbf{p}_{\mathrm{cr}})$. Then, a vector $\mathbf{z} = \{z_1, \ldots, z_d\}$ with d standard uniform random labels is drawn from a standard multivariate uniform distribution, $\mathbf{z} \sim \mathcal{U}_d(0, 1)$. All those dimensions j for which $z_j \leq \mathrm{cr}$ are stored in A and span the subspace that will be sampled. In the case that A is empty, one dimension of $\mathbf{x}_{t-1}$ will be sampled at random.

The number of dimensions stored in A ranges between 1 and d and depends on the actual crossover value used. This randomized strategy, activated

when cr < 1, constantly introduces new directions that chains can take outside the subspace spanned by their current positions. This relatively simple randomized selection strategy enables single-site Metropolis sampling (one dimension at a time), Metropolis-within-Gibbs (one or a group of dimensions) and regular Metropolis sampling (all dimensions). In principle, this allows using $N < d$ in DREAM, an important advantage over DE-MC that requires $N = 2d$ chains to be run in parallel (*ter Braak*, 2006).

The core of the DREAM algorithm can be written in MATLAB in about 30 lines of code (see Figure 4). Based on input arguments, `prior`, `pdf`, N, T, and d, defined by the user `DREAM` returns a sample from the posterior distribution. `prior` is an anonymous function that draws N samples from a d-variate prior distribution, and similarly `pdf` is a function handle which computes the posterior density of a proposal (candidate point).

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

[delta,c,c_star,nCR,p_g] = deal(3,0.1,1e-12,3,0.2);  % Default values DREAM algorithmic parameters
x = nan(T,d,N); p_x = nan(T,N);                      % Preallocate memory for chains and density
X = prior(N,d); p_X = pdf(X);                        % Create initial population and compute density
x(1,1:d,1:N) = reshape(X',1,d,N); p_x(1,1:N) = p_X'; % Store initial position of chain and density
for i = 1:N, R(i,1:N-1) = setdiff(1:N,i); end        % R-matrix: ith chain, the index of chains for DE
CR = [1:nCR]/nCR; pCR = ones(1,nCR)/nCR;             % Crossover values and their selection probability

for t = 2:T,                                         % Dynamic part: Evolution of N chains
    [~,draw] = sort(rand(N-1,N));                        % Randomly permute [1,...,N-1] N times
    dx = zeros(N,d);                                     % Set N jump vectors equal to zero
    lambda = unifrnd(-c,c,N,1);                          % Draw N lambda values
    for i = 1:N,                                         % Create proposal each chain and accept/reject
        r1 = R(i,draw(1:delta,i));                       % Derive vector r1
        r2 = R(i,draw(delta+1:2*delta,i));               % Derive vector r2
        cr = randsample(CR,1,true,pCR);                  % Select crossover value
        A = find(rand(1,d) < cr);                        % Derive subset A with dimensions to sample
        m = numel(A);                                    % How many dimensions are sampled?
        gamma_m = 2.38/sqrt(2*delta*m);                  % Calculate jump rate
        g = randsample([gamma_m 1],1,true,[1-p_g p_g]);  % Select gamma: 80/20 ratio (default)
        dx(i,A) = (1+lambda(i))*g*sum(X(r1,A)-...
            X(r2,A),1) + c_star*randn(1,m);              % Compute ith jump with differential evolution
        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 of ith proposal
        alpha = min(p_Xp(i,1)./p_X(i,1),1);              % Compute Metropolis ratio
        idx = alpha > rand;                              % Alpha larger than U[0,1] or not?
        if idx,
            X(i,1:d) = Xp(i,1:d); p_X(i,1) = p_Xp(i,1);  % True: Accept proposal
        end
    end
    [X,p_X] = outlier(X,log(p_x(ceil(t/2):t,1:N)));      % Outlier detection and correction
    x(t,1:d,1:N) = reshape(X',1,d,N); p_x(t,1:N) = p_X'; % Add current position and density to chain
end
```

Figure 4: MATLAB code of the differential evolution adaptive Metropolis (DREAM) algorithm. Notation is consistent with main text. Based on input arguments `prior`, `pdf`, `N`, `T` and `d`, the DREAM algorithm evolves `N` different trajectories simultaneously to produce a sample of the posterior target distribution. The output arguments `x` and `p_x` store the sampled Markov chain trajectories and corresponding density values, respectively. 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. The function `outlier()` computes the mean of the log posterior density of the samples in the second half of each of the Markov chains. These N values make up a distribution and can be checked for outliers using common statistical tests such as the interquartile range (among others). The states of aberrant trajectories are subsequently reset and samples discarded through burn-in. We refer to introductory textbooks and/or the MATLAB "help" utility for the remaining functions `deal()`, `nan()`, `reshape()`, `setdiff()`, `ones()`, `sort()`, `rand()`, `zeros()`, `unifrnd()`, `randsample()`, `find()`, `numel()`, `sqrt()`, `sum()`, `randn()`, `min()`, `log()`, `ceil()`, and `reshape()`, respectively

.

The performance of DREAM can suffer from one critical deficiency. If one of the N chains is trapped in a local minimum sufficiently far removed from the target distribution then the search can stagnate because the outlier chain is unable to reach the posterior and join the other $N-1$ chains (*ter Braak and Vrugt*, 2008; *Vrugt et al.*, 2009). This happens if the differences between the states of the chains that sample the target distribution are too small to enable the aberrant chain to jump outside the space spanned by the local optimum and move in the direction of the posterior distribution. This deteriorates search efficiency and prohibits convergence to a limiting distribution. The function `outlier` corrects the state of aberrant trajectories by comparing the mean log-density values of the last 50% of the samples of the `N` different Markov chains. Details can be found in (*Vrugt et al.*, 2009).

To enhance search efficiency, the probability, $\mathbf{p}_{\text{cr}}$ of each of the n_{cr} crossover values is tuned adaptively during burn-in by maximizing the normalized Euclidean distance between successively sampled states of the N different chains. Details of this approach can be found in *Vrugt et al.* (2008a, 2009).

The basic code of DREAM listed in Figure 4 was written in 2006 but many new functionalities and options have been added to the source code in recent years due to continued research developments and to support the needs of a growing group of users. A toolbox of DREAM has been developed recently and we use this code in our present work. A detailed description of DREAM appears in various publications (*Vrugt et al.*, 2008a, 2009, 2011; *Laloy and Vrugt*, 2012; *Sadegh and Vrugt*, 2014) and a manual of the MATLAB toolbox (*Vrugt*, 2015a), and interested readers are referred to these publications for benchmark studies and real-world applications. In the present MODELAVG toolbx we assume default values for the algorithmic parameters of DREAM.

3.2. MODELAVG: MATLAB implementation

The basic code of MODEAVG was written in 2014 and some changes have been made recently to support the needs of users. The MODELAVG code can be executed from the MATLAB prompt by the command

```
[beta] = MODELAVG(method,Meas_info,options)
```

where `method` (string), `Meas_info` (structure array) and `options` (structure array) are input arguments defined by the user, and `beta` (vector), is the main output variable computed by MODELAVG and returned to the user. To minimize the number of input and output arguments in the MODELAVG

function call and related primary and secondary functions called by this program, we use MATLAB structure arrays and group related variables in one main element using data containers called fields, more of which later. The third input variable, `options` is optional, and deemed necessary for information criterion averaging, Bayesian model averaging and Mallows model averaging.

A summary of the different functions of the MODELAVG toolbox is given in Appendix A. We will now discuss the content and usage of each variable.

3.3. Input argument 1: `method`

The string `method` defines the model averaging method to be used. The user can select among the eight different methods and their acronyms presented in section 2 including 'ewa', 'bga', 'aica', 'bica', 'gra', 'bma', 'mma', and 'mma-simplex' (case insensitive). The first five model averaging methods are solved using a direct solution for the weights, whereas the last three methods ('bma'/'mma'/'mma-simplex') are solved using MCMC simulation with DREAM.

3.4. Input argument 2: Meas_info

The second input argument `Meas_info` of the MODELAVG function has two fields that summarize the data of the ensemble and veryfying observations. Table 2 summarizes these two different fields of `Meas_info`, their content and corresponding variable type.

Table 1: Content of input structure `Meas_info`.

Field of `Meas_info`	Description	Type
D	Ensemble forecasts	$n \times K$-matrix
Y	Verifying measurements	$n \times 1$-vector

The field D of `Meas_info` stores the $n \times K$ matrix of ensemble forecasts that are used with the different model averaging methods. Each of the K predictors is bias corrected using the linear correction of Equation 3) (more advanced bias-correction methods can be used as well - more of which later). The field Y of `Meas_info` stores the observations against which the forecast ensemble is evaluated, and thus the weights and or other coefficients of the averaged model are derived from. The number of elements of Y and G should

match exactly, otherwise a warning is given and the MODELAVG code terminates prematurely.

3.5. (Optional) input argument 3: options

The structure `options` is optional and passed as third input argument to MODELAVG. This structure needs to be defined if information criterion averaging, Bayesian model averaging or Mallows model averaging is used. Table 2 summarizes the different fields of `options` and their content.

Table 2: Content of (optional) input structure `options`. This third input argument of MODELAVG is required if information criterion averaging, Bayesian model averaging or Mallows model averaging is activated.

Field of `options`	Description	Options	Type
PDF	BMA: Conditional distribution	'normal'/'heteroscedastic'/'gamma'	string
VAR	BMA: Variance of 'normal'	'single'/'multiple'	string
alpha	BMA: prediction interval	e.g. 0.68/0.90/0.95/0.99	scalar
p	Number of parameters of models		$1 \times K$ vector

The field p of `options` stores in a K-vector the number of parameters of each model and is required if `method` is equivalent to 'aica', 'bica' ,'mma' or 'mma-simplex'. The fields PDF and VAR (only for 'normal') of `options` are required for Bayesian model averaging and define the name of the conditional distribution, and whether this distribution has a fixed variance for all models of the ensemble or whether each model has its own variance. The field alpha defines the confidence interval of the forecast distribution of the BMA model, the results of which are stored in field pred of `output`. The default option for alpha is 0.95 and thus a 95% confidence interval of the BMA mixture distribution.

3.6. Output arguments

The output argument, `beta` of MODELAVG is a $1 \times K$ vector of weights that correspond to the model averaging `method` selected by the user. If Bayesian model averaging and Mallows model averaging is selected as the `method` of choice then `beta` is equivalent to an array of weight values produced by the different Markov chains of DREAM. This array is a matrix of size $T \times d + 2 \times N$, where T denotes the number of samples in each Markov chain (is equivalent to maximum number of generations with DREAM), and

N denotes the number of chains. The first d columns of `beta` store the sampled parameter values (state), whereas the subsequent two columns list the associated log-prior (zero) and log-likelihood values respectively.

If Mallows model averaging or Bayesian model averaging is used then MODELAVG returns a second output argument, called `output` which is a structure with different fields that store the prediction ranges, log-likelihood and coverage of the BMA model, and diagnostic information about the progress and performance of the DREAM algorithm. The field pred of `output` is a matrix of size $n \times 2$ with lower and upper ranges of the BMA prediction intervals. The field log_L lists the maximum log-likelihood of the BMA model, and the field contained stores the coverage of the prediction intervals.

Diagnostic information of DREAM appears in fields RunTime (scalar) which stores the wall-time (seconds), and fields R_stat (matrix), AR (matrix) and CR (matrix) that list for a given number of generations the $\hat{R}$ convergence diagnostic of *Gelman and Rubin* (1992) for each of the d parameters of the target distribution, the average acceptance rate and selection probabilities of each of the n_{cr} crossover values used by DREAM, respectively. Finally, the field outlier (vector) contains the index of all outlier chains (often empty). Note that at the end of each DREAM trial, the autocorrelation function and *Geweke* (1992) and *Raftery and Lewis* (1992, 1995) convergence diagnostics of each of the N chains and printed to the screen in the MATLAB editor.

The MATLAB command

```
plot(output.R_stat(1:end,2:DREAMPar.N+1))
```
(22)

plots with different colors the evolution of the $\hat{R}$-convergence diagnostic of each parameter. This information can be used to determine the burn-in and hence which of the samples of the array `beta` to use for posterior inference.

The directory '../postprocessing' (under main directory) contains the function MODELAVG_POSTPROC that can be used to visualize the different output arguments of MODELAVG. This script can be executed from the MATLAB prompt after the main program of MODELAVG has terminated. Appendix B provides a screen shot of the MATLAB command window after MODELAVG has terminated. Graphical output is generated as well, for instance a time series plot of the ensemble forecasts, observations, and averaged forecast, and a quantile-quantile plot of the residuals of this predictor. If MMA, MMA$^{\Delta}$ and BMA are used, then many additional figures are created using the output of DREAM. This includes trace plots of the sampled

weights and/or variance(s) (or proxies thereof) in each of the Markov chains (color coded), bivariate scatter plots of the posterior samples, histograms of the marginal posterior weights/variance distributions, and a plot of the evolution of the $\hat{R}$-convergence diagnostic (*Gelman and Rubin*, 1992). This latter plot is particularly important as it will help the user judge how many generations, T are required to guarantee converge to a limiting distribution. Otherwise the inferences from the DREAM sample can be misleading.

4. Numerical examples

We now demonstrate the application of the MATLAB MODEAVG package to two three different ensemble data sets. These three studies involve the fields of watershed hydrology, and meteorology, respectively.

4.1. Case Study I: The rainfall-runoff transformation

We now test the different model averaging methods (with/without density forecast) by applying them to streamflow simulations of eight different watershed models using historical data from the Leaf River watershed in Mississippi. Figure 5 provides a snapshot of the model ensemble for a portion of the water year 1953 (Oct. 1 - Sept. 30).

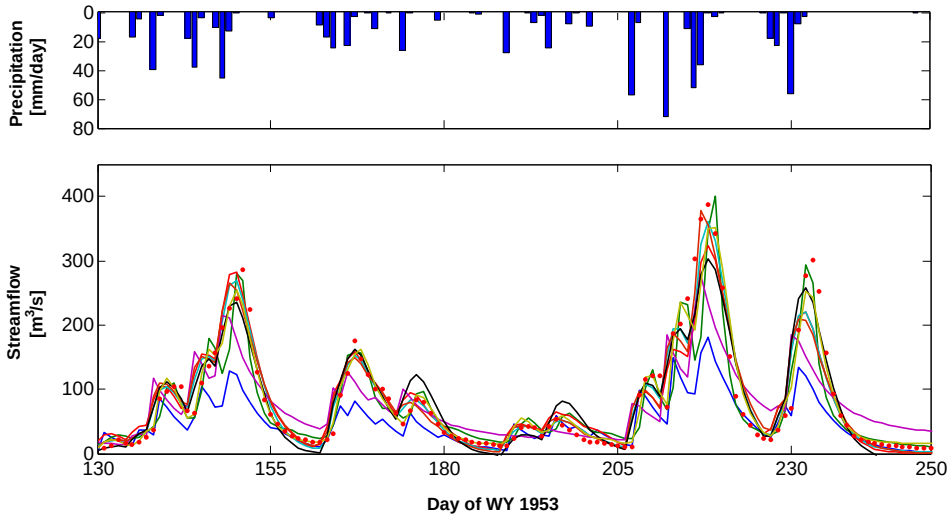

Figure 5: Streamflow predictions of the eight individual models of the ensemble for a representative portion of the calibration period. The circles represent the verifying observations.

The spread of the ensemble is sufficient and generally brackets the observations (the circles). The calibrated models appear to provide different forecasts. This is a desirable characteristic and prerequisite for accurate forecasting, and model averaging. We only consider BMA and MMA in the present study.

The following script (Figure 6) is used in MATLAB to run the MODELAVG package.

```
% ------------------------------------------------------------------------------------------%
%                                                                                           %
%   MM       MM   OOOOOOO   DDDDDDDD   EEEEEEEE  LL            AAA      VV       VV  GGGGGGGG   %
%   MMM      MM  OOOOOOOOO  DDDDDDDDD  EEEEEEEE  LL           AA AA     VV       VV  GG    GG   %
%   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   GGGGGGGG   %
%   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         GGG   %
%   MM       MM   OOOOOOO   DDDDDDDD   EEEEEEEE  LLLLLLLL  AA       AA      VVV      GGGGGGGG   %
%                                                                                           %
% ------------------------------------------------------------------------------------------%

% Check:  http://faculty.sites.uci.edu/jasper
% Papers: http://faculty.sites.uci.edu/jasper/publications/
% Google Scholar: https://scholar.google.com/citations?user=zkNXecUAAAAJ&hl=nl

% --> USER: Define problem to be solved
example = 1;

%% Define example directory to be global variable and add to path
global EXAMPLE_dir
%% Store subdirectory containing the files needed to run this example
EXAMPLE_dir = [pwd '\example_' num2str(example)]; addpath(EXAMPLE_dir);

%% USER: Which method are we using
method = 'bma';                 % 'ewa'/'bga'/'aica'/'bica'/'gra'/'bma'/'mma'/'mma-simplex'

%% USER: If BMA is used then pdf and var need to be defined
options.PDF = 'gamma';          % pdf predictor: normal/heteroscedastic/gamma
options.VAR = 'multiple';       % variance pdf: single/multiple (multiple for 'normal')
options.alpha = 0.95;           % prediction intervals of BMA model

%% USER: Load data from Vrugt and Robinson, WRR, 43, W01411, doi:10.1029/2005WR004838, 2007
load data.txt;                  % Daily streamflow simulations eight watershed models
load Y.txt;                     % Daily streamflow observations
start_idx = 1; end_idx = 3000;  % Start/end day training period

%% USER: Define the ensemble of simulations and the vector of verifying observations
Meas_info.D = data(start_idx:end_idx,1:8); Meas_info.Y = Y(start_idx:end_idx,1);

%% Apply linear bias correction to ensemble
[ Meas_info ] = Bias_correction ( Meas_info );

%% Run MODELAVG toolbox and return two output arguments
[beta,output] = MODELAVG(method,Meas_info,options);

%% Create graphical output and tables
MODELAVG_postproc
```

Figure 6: Case study I: Ensemble of streamflow forecasts of eight different watershed models

The predictive distribution of each constituent member of the ensemble is assumed to follow a gamma distribution with unknown heteroscedastic vari-

ance.

Figure 7 presents histograms of the marginal posterior distribution of the BMA weights for each of the models of the ensemble. The MAP values of the weights are separately indicated with a blue cross.

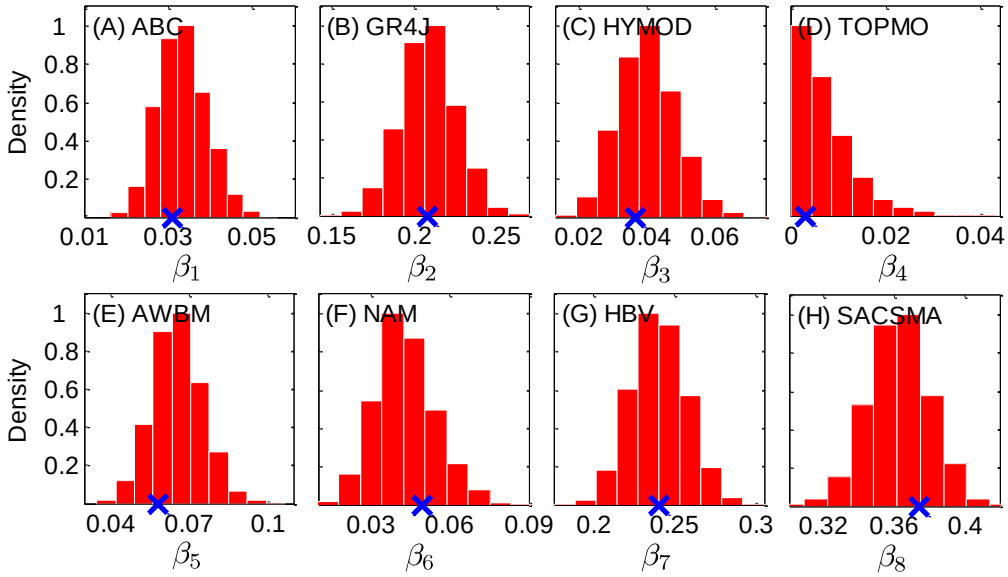

Figure 7: Histograms of the marginal posterior distribution of the weights and variances of each individual model of the ensemble. The MAP values of the weights are denoted with a blue cross.

The distributions appear rather well-defined and exhibit an approximate Gaussian shape. Analysis of the posterior weights helps to understand which models of the ensemble are parameter uncertainty can be used to assess BMA model uncertainty.

To understand how the BMA posterior parameter uncertainty translates into predictive uncertainty, please consider Figure 14 that presents the 95% hydrograph prediction uncertainty ranges of the BMA model for a representative period of the training data period. The total uncertainty is indicated by the shaded region, and the mean ensemble forecast is displayed with the red line. The observations are separately indicated with the blue circles.

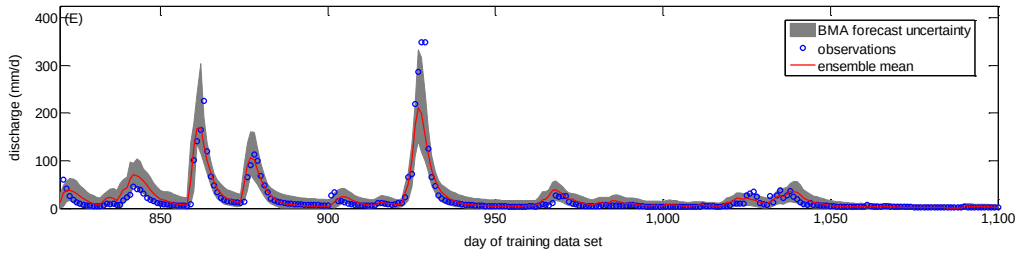

Figure 8: 95% prediction intervals (gray region) of the BMA model for a representative portion of the 3000 days calibration period. The black line signifies the ensemble mean, whereas the blue circles represent the verifying observations.

The prediction uncertainty ranges of the BMA model envelop almost 95% of the observations, but appear rather large, particularly at lower flows. The RMSE of the mean BMA forecast is equivalent to 15.98 m^3/s which is somewhat smaller than its counter part of 16.45 m^3/s for the best model of the ensemble. These results of the BMA model can further be enhanced by using a sliding window training approach. By allowing the weights and variance of each conditional distribution to vary over time we can endow the BMA method with an ability to evolve in a manner analogues to data assimilation approaches (*Vrugt and Robinson*, 2007a). Another possibility is to use a Box-Cox (*Box and Cox*, 1964) or normal quantile transformation of the ensemble and verifying observations prior to BMA model training. These transformations stabilize the variance and make the ensemble and corresponding measurements to be more normal distribution like.

Table 3 summarizes the results of the BMA method and presents (in column "Gamma") the maximum a-posteriori (MAP) values of the BMA weights for the eight different models of the ensemble. Values listed in parentheses denote the posterior standard deviation derived from the DREAM sample. We also summarize the MAP values of the weights for a Gaussian (conditional) distribution (columns "Normal") with homoscedastic (left) or heteroscedastic (right) error variance, and report the average RMSE (m^3/s), coverage (%) and spread (m^3/s) of the resulting BMA model during the 26-year evaluation period.

Table 3: Results of BMA by application to eight different watershed models using daily discharge data from the Leaf River in Mississippi, USA. We list the individual forecast errors (RMSE, m^3/s) of the models for the training data period, the corresponding MAP values of the weights for a Gamma (default) and Gaussian forecast distribution, and present the results of the BMA model (bottom panel) during the evaluation period. The spread (m^3/s) and coverage (%) correspond to a 95% prediction interval.

Model	RMSE	Gamma†	Normal ‡	Normal §	MMA¶	MMA$^{\Delta}$ ‖
ABC	31.67	0.02 *(0.006)*	0.03 *(0.010)*	0.00 *(0.002)*	0.03 *(0.001)*	0.00 *(0.000*
GR4J	19.21	0.21 *(0.016)*	0.14 *(0.013)*	0.10 *(0.013)*	0.56 *(0.002)*	0.14 *(0.001)*
HYMOD	19.03	0.03 *(0.008)*	0.13 *(0.046)*	0.00 *(0.005)*	0.51 *(0.003)*	0.00 *(0.000)*
TOPMO	17.68	0.03 *(0.006)*	0.08 *(0.047)*	0.03 *(0.010)*	-0.23 *(0.003)*	0.30 *(0.002)*
AWBM	26.31	0.05 *(0.009)*	0.01 *(0.010)*	0.00 *(0.002)*	0.20 *(0.001)*	0.00 *(0.000)*
NAM	20.22	0.05 *(0.011)*	0.14 *(0.048)*	0.11 *(0.014)*	-0.59 *(0.002)*	0.00 *(0.000)*
HBV	19.44	0.24 *(0.017)*	0.13 *(0.034)*	0.31 *(0.016)*	-0.00 *(0.002)*	0.00 *(0.000)*
SACSMA	16.45	0.37 *(0.017)*	0.34 *(0.022)*	0.43 *(0.017)*	0.70 *(0.002)*	0.55 *(0.002)*
BMA/MMA: log-likelihood		-9,775.1	-9,950.5	-9,189.4	-349,760.0	-367,011.9
BMA/MMA: RMSE		22.54	23.22	23.16	24.66	21.65
BMA: Spread		39.74	46.98	46.54		
BMA: Coverage		93.65%	92.59%	95.71%		

† `method` = 'bma'; `options.PDF` = 'normal'; `options.VAR` = 'single'
‡ `method` = 'bma'; `options.PDF` = 'heteroscedastic'
§ `method` = 'bma'; `options.PDF` = 'gamma'
¶ `method` = 'mma'
‖ `method` = 'mma-simplex'

The values of the BMA weights depend somewhat on the assumed conditional distribution of the deterministic model forecasts of the ensemble. The GR4J, HBV and SACSMA models consistently receive the highest BMA weights and are thus most important in BMA model construction for this data set. Note also that TOPMO receives a very low BMA weight, despite it having the second lowest RMSE value of the training data period. Correlation between the individual forecasts of the watershed models affects strongly the posterior distribution of the BMA weights. The gamma distribution is preferred for probabilistic streamflow forecasting with 95% simulation uncertainty ranges that, on average, are noticeably smaller than their counterparts derived from a normal distribution. We refer interested readers to *Vrugt and Robinson* (2007a) and *Rings et al.* (2012) for a more detailed analysis of the BMA results, and a comparison with data assimilation methods.

The point forecast of MMA$^{\Delta}$ is better than that of the three BMA models. However, if the MMA weights are allowed to vary freely in DREAM and take on any value, the performance of this method degrades markedly with forecast errors during the evaluation period that are considerably larger than their counterparts of the other model averaging methods. This demonstrates that regular MMA is prone to overfitting. Indeed, the restriction in MMA$^{\Delta}$

to only choose weights that lie on the unit simplex stabilizes the inverse solution of Equation (19) with DREAM. What is more, experience suggests that the log-likelihood of MMA$^{\Delta}$ is maximized at many different locations in the weight space (different trials of DREAM provides widely varying solutions, yet with approximately similar log-likelihoods). This makes MMA training rather difficult and the final posterior distribution of the weights (negligible uncertainty of each optimum) rather meaningless. Thus, MMA$^{\Delta}$ should be used with great care, particularly if this method is to be used for postprocessing of forecast ensembles outside the training data period.

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

Our second case study uses 48-h forecasts of surface temperature (in K) in the North American Pacific Northwest in January-June 2000 from the University of Washington (UW) mesoscale short-range ensemble system (*Grimit and Mass*, 2002). This is a five member multianalysis ensemble (hereafter referred to as the UW ensemble) consisting of different runs of the fifth-generation Pennsylvania State University - National Center for Atmospheric Research Mesoscale Model (MM5), in which initial conditions are taken from different operational centers. Following *Raftery et al.* (1999) a 25-day training period between April 16 and 9 June 2000 is used for BMA model calibration. 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 individuals members of the ensemble were bias corrected using simple linear regression for the training data set. We assume a Gaussian conditional distribution, $f(\cdot)$, with a fixed (homoscedastic) variance for each member of the ensemble.

The following setup is used in MATLAB.

```
% ------------------------------------------------------------------------------------------%
%                                                                                           %
%   MM       MM   OOOOOOO   DDDDDDDD   EEEEEEEE  LL            AAA      VV       VV  GGGGGGGG   %
%   MMM      MM  OOOOOOOOO  DDDDDDDDD  EEEEEEEE  LL           AA AA     VV       VV  GG    GG   %
%   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   GGGGGGGG   %
%   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          GGG   %
%   MM       MM   OOOOOOO   DDDDDDDD   EEEEEEEE  LLLLLLLL AA       AA      VVV      GGGGGGGG   %
%                                                                                           %
% ------------------------------------------------------------------------------------------%

% Check:  http://faculty.sites.uci.edu/jasper
% Papers: http://faculty.sites.uci.edu/jasper/publications/
% Google Scholar: https://scholar.google.com/citations?user=zkNXecUAAAAJ&hl=nl

% --> USER: Define problem to be solved
example = 2;

%% Define example directory to be global variable and add to path
global EXAMPLE_dir
%% Store subdirectory containing the files needed to run this example
EXAMPLE_dir = [pwd '\example_' num2str(example)]; addpath(EXAMPLE_dir);

%% --> USER: Which method are we using
method = 'bma';                 % 'ewa'/'bga'/'aica'/'bica'/'gra'/'bma'/'mma'/'mma-simplex'

%% --> USER: If BMA is used then pdf and var need to be defined
options.PDF = 'normal';         % pdf predictor: normal/heteroscedastic/gamma
options.VAR = 'single';         % variance pdf: single/multiple (only for 'normal')
options.alpha = 0.95;           % prediction intervals of BMA model

%% --> USER: Load data from Raftery et al., MWR, 133, pp. 1155-1174, 2005.
load temperature.mat;           % 48-hour forecasts of surface temperature, in Kelvin

%% --> USER: Training data period between April 16 and June 9, 2000
idx = find ( data(:,1) == 2000 & data(:,2) == 4 & data(:,3) == 16 ); start_idx = idx(1);
idx = find ( data(:,1) == 2000 & data(:,2) == 6 & data(:,3) == 9 ); end_idx = idx(1);

%% --> USER: Define the ensemble of simulations and the vector of verifying observations
Meas_info.D = data(start_idx:end_idx,5:9); Meas_info.Y = data(start_idx:end_idx,4);

%% Apply linear bias correction to ensemble
[ Meas_info ] = Bias_correction ( Meas_info );

%% Run MODELAVG toolbox and return two output arguments
[beta,output] = MODELAVG(method,Meas_info,options);

%% Create graphical output and tables
MODELAVG_postproc
```

Figure 9: Case study II: 48 hour forecasts of sea surface temperature in the Northwestern USA.

Table 4 lists the (maximum likelihood) values of the weights for each of the model averaging methods considered herein, except MMA with unrestricted values for the weights. This method provides rather inferior results and is discarded.

Table 4: Results of BMA by application to the five model temperature forecasts of the UW mesoscale short-range ensemble system. We list the individual forecast errors (RMSE, m^3/s) of the models for the training data period, the corresponding MAP values of the weights for the different model averaging methods discussed herein, with the exception of MMA. The spread (m^3/s) and coverage (%) of the BMA model correspond to a 95% prediction interval.

Model	RMSE	EWA	BGA	AICA	BICA	GRA	BMA†	MMA$^{\Delta}$ ‡
AVN (NCEP)	2.96	0.20	0.22	1.00	1.00	0.49	0.42 *(0.02)*	0.48 *(0.01)*
GEM (CMC)	3.05	0.20	0.19	0.00	0.00	0.20	0.21 *(0.01)*	0.19 *(0.00)*
ETA (NCEP)	3.01	0.20	0.21	0.00	0.00	0.36	0.27 *(0.02)*	0.31 *(0.01)*
NGM (NCEP)	2.99	0.20	0.20	0.00	0.00	-0.07	0.04 *(0.01)*	0.00 *(0.00)*
NOGAPS (FNMOC)	2.97	0.20	0.18	0.00	0.00	0.02	0.06 *(0.01)*	0.02 *(0.01)*
Averaged forecast: RMSE		2.99	2.99	3.06	3.06	2.95	2.96	2.96
BMA: Spread							10.12	
BMA: Coverage							91.60%	

† `method` = 'bma'; `options.PDF` = 'normal'; `options.VAR` = 'single'
‡ `method` = 'mma-simplex'

The point forecasts of the different model averaging methods exhibit a rather similar performance. Yet, the main advantage of the BMA method is that it provides a forecast distribution which can be used for probabilistic analysis and prediction. The use of a single variance for all models of the ensemble results in a reliable spread of the 95% prediction intervals of the BMA model. About 92% of the 14,043 temperature observations is contained in this prediction interval. A heteroscedastic variance (`options.PDF` = 'heteroscedastic') or individual variance (`options.VAR` = 'multiple') for each member of the ensemble would further improve the statistical adequacy of the forecast density (not shown). Nevertheless, if point forecasting is of main concern, the Granger-Ramanathan averaging provides the best results at negligible computation cost).

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

We now do a similar analysis but using 48-h forecasts of sea surface pressure (in hPa) from the University of Washington (UW) mesoscale short-range ensemble system (*Grimit and Mass*, 2002). Again, a 25-day training period between April 16 and 9 June 2000 is used for BMA model calibration. 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 individuals members of the ensemble were bias corrected using simple linear regression for the training data set. We assume a Gaussian conditional distribution, $f(\cdot)$, with a fixed variance for each member of the ensemble.

The following setup is used in MATLAB.

```
% ------------------------------------------------------------------------------------------%
%                                                                                           %
%   MM       MM   OOOOOOO   DDDDDDDD   EEEEEEEE  LL           AAA      VV       VV  GGGGGGGG   %
%   MMM      MM  OOOOOOOOO  DDDDDDDDD  EEEEEEEE  LL          AA AA     VV       VV  GG     GG   %
%   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   GGGGGGGG   %
%   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          GGG   %
%   MM       MM   OOOOOOO   DDDDDDDD   EEEEEEEE  LLLLLLLL  AA       AA      VVV      GGGGGGGG   %
%                                                                                           %
% ------------------------------------------------------------------------------------------%

% Check:  http://faculty.sites.uci.edu/jasper
% Papers: http://faculty.sites.uci.edu/jasper/publications/
% Google Scholar: https://scholar.google.com/citations?user=zkNXecUAAAAJ&hl=nl

% --> USER: Define problem to be solved
example = 3;

%% Define example directory to be global variable and add to path
global EXAMPLE_dir
%% Store subdirectory containing the files needed to run this example
EXAMPLE_dir = [pwd '\example_' num2str(example)]; addpath(EXAMPLE_dir);

%% --> USER: Which method are we using
method = 'bma';                 % 'ewa'/'bga'/'aica'/'bica'/'gra'/'bma'/'mma'/'mma_simplex'

%% --> USER: If BMA is used then pdf and var need to be defined
options.PDF = 'normal';         % pdf predictor: normal/heteroscedastic/gamma
options.VAR = 'single';         % variance pdf: single/multiple (only for 'normal')
options.alpha = 0.95;           % prediction intervals of BMA model

%% --> USER: Load data from Raftery et al., MWR, 133, pp. 1155-1174, 2005.
load pressure.mat;              % 48-hour forecasts of surface pressure, in hPa

%% --> USER: Training data period between April 16 and June 9, 2000
idx = find ( data(:,1) == 2000 & data(:,2) == 4 & data(:,3) == 16 ); start_idx = idx(1);
idx = find ( data(:,1) == 2000 & data(:,2) == 6 & data(:,3) == 9 ); end_idx = idx(1);

%% --> USER: Define the ensemble of simulations and the vector of verifying observations
Meas_info.D = data(start_idx:end_idx,5:9); Meas_info.Y = data(start_idx:end_idx,4);

%% Apply linear bias correction to ensemble
[ Meas_info ] = Bias_correction ( Meas_info );

%% Run MODELAVG toolbox and return two output arguments
[beta,output] = MODELAVG(method,Meas_info,options);

%% Create graphical output and tables
MODELAVG_postproc
```

Figure 10: Case study III: 48 hour forecasts of sea surface pressure in the Northwestern USA.

Figures 11-13 are taken from *Vrugt et al.* (2008b) and compare the results of case study II with those of the current case study. The following text is

in verbatim copy of the cited paper from 2008 - with some corrections to the text.

Figure 11 presents histograms of the DREAM derived marginal posterior distributions of the BMA weights and variance of the individual ensemble members for the 25 day training period for surface temperature (panels 11A-11F) and sea level pressure (panels 11G-11L). The DREAM algorithm has generated 10 Markov chains, each with $5,000$ samples. The first $4,000$ samples in each chain are used for burn-in, leaving a total of 5,000 samples to draw inferences from. The optimal values derived with the EM algorithm are separately indicated in each panel with the 'x' symbol.

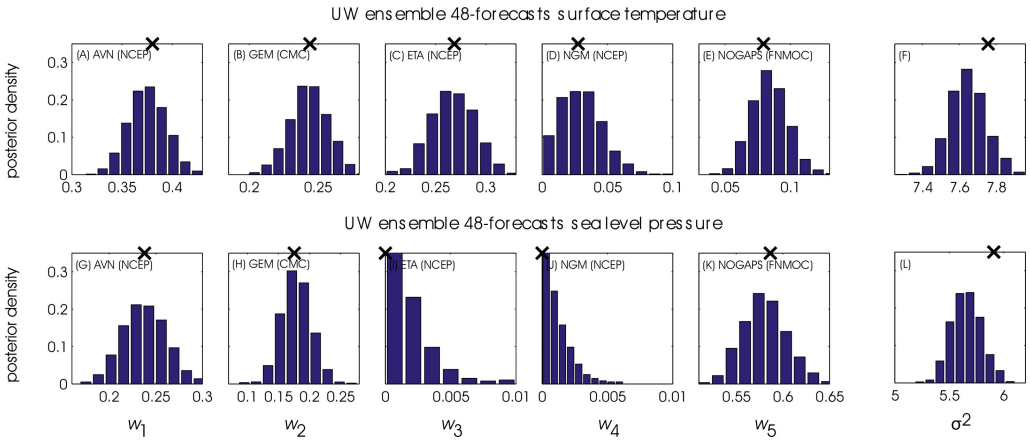

Figure 11: Histograms of the marginal posterior distributions of the DREAM derived BMA weights and variance for the surface temperature (A-F) and sea level pressure (G-L) training data sets. The EM derived solution is separately indicated in each panel with symbol '×'. Note the notation $w_.$ is used for the BMA weights.

The results generally show an excellent agreement between the modes of the histograms derived from MCMC simulation and the maximum likelihood estimates of the EM solution within this high-density region. Previous applications of the EM algorithm are likely to have yielded robust parameters for the UW ensemble data set. However, DREAM has as desirable feature that it not only correctly identifies the maximum likelihood values of the BMA weights and variances (or proxies thereof), but simultaneously also samples the underlying probability distribution. This is helpful information to assess the usefulness of individual ensemble members in the BMA prediction, and

the correlation among ensemble members. For instance, most of the histograms exhibit an approximate Gaussian distribution with relatively small dispersion around their modes. This indicates that there is high confidence in the weights applied to each of the individual models.

The evolution of the sampled BMA weights of each ensemble member and associated variance, σ_2 of the conditional normal forecast distribution, $f(\cdot)$ are shown in Figure 12. I randomly selected three different Markov chains, and coded each of them with a different color and symbol. The trace plots illustrate that during the initial stages of MCMC simulation with DREAM the different chains occupy different parts of the parameter space, resulting in a relatively high value for the $\hat{R}$-convergence diagnostic (not shown). After about 250 draws in each chain, the three trajectories 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.

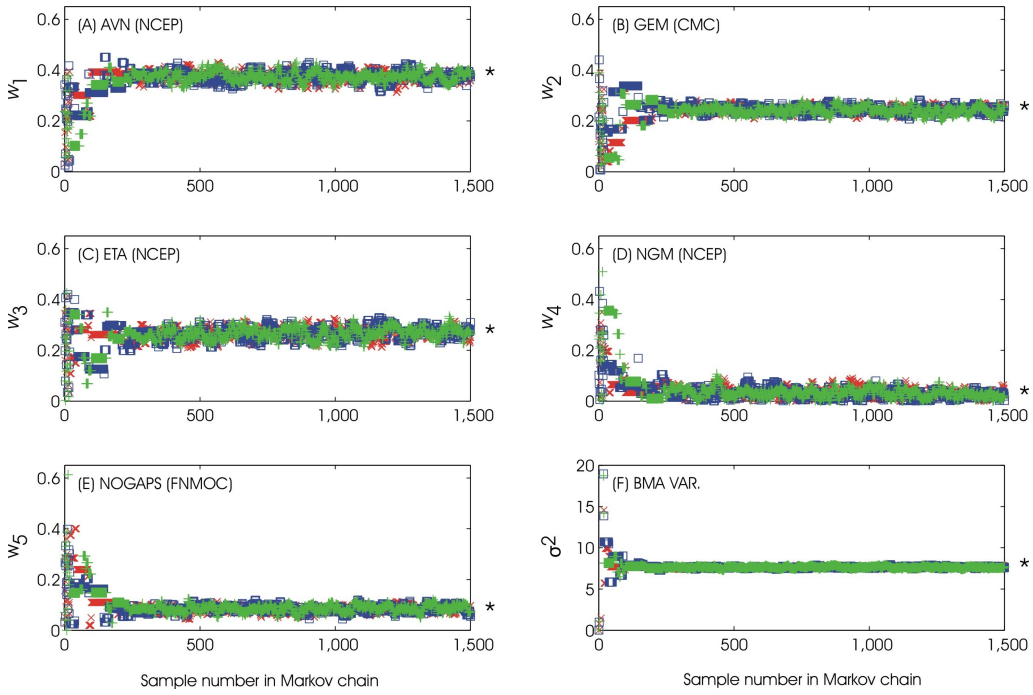

Figure 12: Trace plots of the sampled BMA weights of ensemble member AVN (A), ETA (B), NGM (C), GEM (D), NOGAPS (E), and the BMA variance (F) in three of the Markov chains generated with DREAM. The maximum likelihood estimates computed with the EM algorithm are separately indicated at the right hand side in each panel with the symbol '$*$'. Note the notation $w.$ is used for the BMA weights.

To explore whether the performance of the EM and DREAM sampling methods are affected by the length of the training data set, we sequentially increased the length of the training set. We consider training periods of 5, 10, 15, 20 and 25 days. For each length, we ran both methods thirty different times, using a randomly sampled (without replacement) calibration period from the original 25 day training data set. Each time, a bias correction was first applied to the ensemble using simple linear regression of $\mathbf{D}_k$ on $\tilde{\mathbf{Y}}$ of the training data set. Figure 13 presents the outcome of the 30 individual trials as a function of the length of the training set.

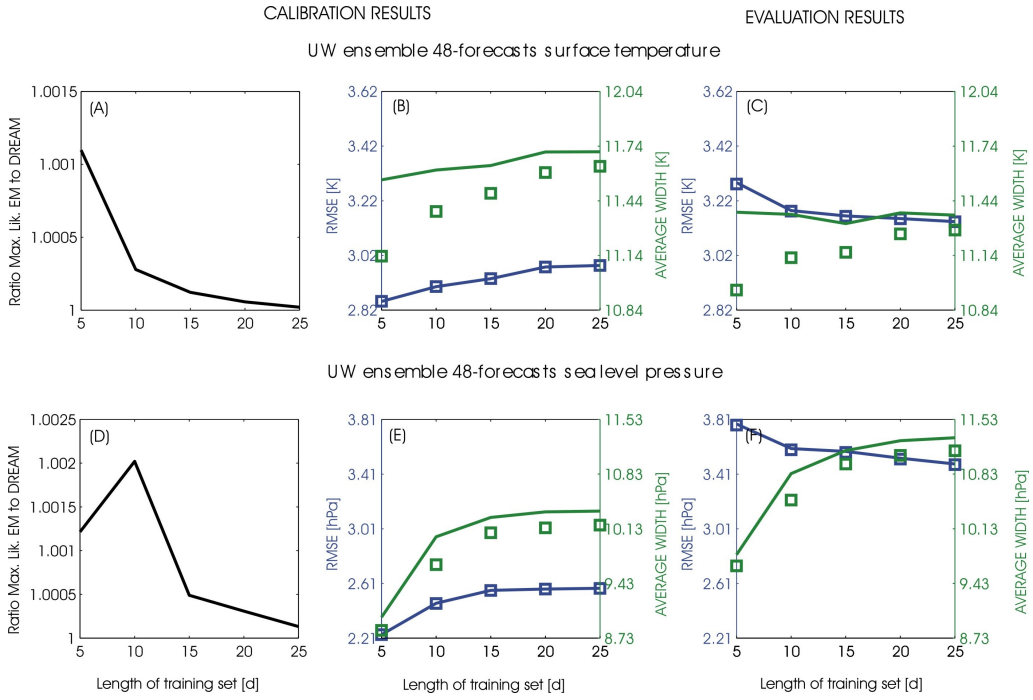

Figure 13: Comparison of training period lengths for surface temperature and sea level pressure. The reported results represent averages of 30 independent trials with randomly selected training data sets from the original 25 days data set. Solid lines in panels B-C and E-F denote the EM derived forecast error of the BMA predictive mean (blue), and associated width of the prediction intervals (green) for the calibration (left side) and evaluation period (right side) respectively; squared symbols are DREAM derived counterparts.

The top panels (Fig. 13A-C) display the results for the surface temperature data set, while the bottom panels (Fig. 13D-F) depict the results for sea level pressure. The blue lines denote the RMSE of the forecast error of the BMA predictive mean. The green lines plot the average width of the associated 95% prediction uncertainty intervals (green) obtained with the EM algorithm, and the squared symbols represent their DREAM derived counterparts. The panels differentiate between the calibration and evaluation data period.

The results presented here highlight several important observations. First, the ratio between the maximum likelihood values derived with the EM and DREAM algorithm closely approximates 1, and appears to be rather unaffected by the length of the training set. This provides strong empirical

evidence that the performance of the EM and MCMC sampling methods is quite similar, and not affected by the length of the training data set. Secondly, the RMSE of the BMA deterministic (mean) forecast (indicated in blue) generally increases with increasing length of the calibration period. This is to be expected as longer calibration time series are likely to exhibit more dynamics due to a larger variety of weather events. Thirdly, notice that the average width of the BMA 95% prediction uncertainty intervals (indicated in green) increases with length of the training period. This again has to do with a larger diversity of weather events in longer calibration time series. Finally, the BMA forecast pdf derived through MCMC simulation with DREAM is sharper (less spread) than its counterpart estimated with the EM method. This finding is consistent with the results depicted in Fig. 1 which shows larger maximum likelihood values of the BMA variance, σ^2 for the EM method.

5. Recent developments

The original BMA approach presented by *Raftery et al.* (1999) 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 paper we 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. The user is free to implement additional distributions - this necessitates a few changes to the functions BMA_calc and BMA_setup. 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.

8. Appendix A

Table 5 summarizes, in alphabetic order, the different function/program files of the MODELAVG package in MATLAB.

The main program runMODELAVG contains three different prototype studies which involve hydrologic and meteorologic forecasting. These example studies have been published in the literature and provide a template for users to setup their own case study. The last line of each example study involves a function call to MODELAVG, which computes the values of the weights and or variances of the conditional forecast distribution (in case of MMA and BMA). Each example problem of runMODELAVG has its own directory which stores the respective data.

The functions of the DREAM algorithm are stored in the directory '../DREAM' - those are used by BMA, MMA, and MMA$^{\Delta}$ to derive the posterior distribution of the their respective weights and/or variances (or proxies thereof). A description of these functions and the source code of DREAM appears in *Vrugt* (2015a), and I refer interested readers to this publication for further details.

The directory '../postprocessing' of the MODELAVG toolbox contains the script MODELAVG_postproc that summarizes the output of MODELAVG using a variety of different tables and figures. The tables are printed in the MATLAB command window (see Appendix B), whereas figures are printed directly to the screen. The tables list the (maximum likelihood) values of the weights, their posterior standard deviation and correlation (if this information is available), whereas figures include a time series plot of the ensemble members, the verifying observations and the averaged forecast, 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 from the DREAM output including trace plots of the sampled chain trajectories and $\hat{R}$-convergence diagnostic, and bivariate scatter plots and histograms of the posterior samples (among others). Some figures generated by MODELAVG_postproc appear in case study I and III.

Table 5: 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 the BMA model for vector of weights and variances (or proxies thereof)
BMA_SETUP	Setup of the BMA model including parameter ranges and settings for DREAM
BMA_QUANTILE	Derives the desired quantiles of the BMA model forecast (simulation)
MMA_CALC	Calculates the log-likelihood of the MMA model for vector of weights
MODELAVG	Main script that calculates the weights and/or variances (or proxies thereof) of given model averaging method
RUNMODELAVG	Setup of three different example problems and calls the main MODEALVG function

9. Appendix B

The MODELAVG toolbox presented herein returns to the user (in the MATLAB command window) the (maximum likelihood) values of the coefficients, their posterior standard deviation and correlation (if DREAM is executed). Figure 14 displays a screen shot of the MATLAB command window after the main function MODELAVG has terminated its calculations.

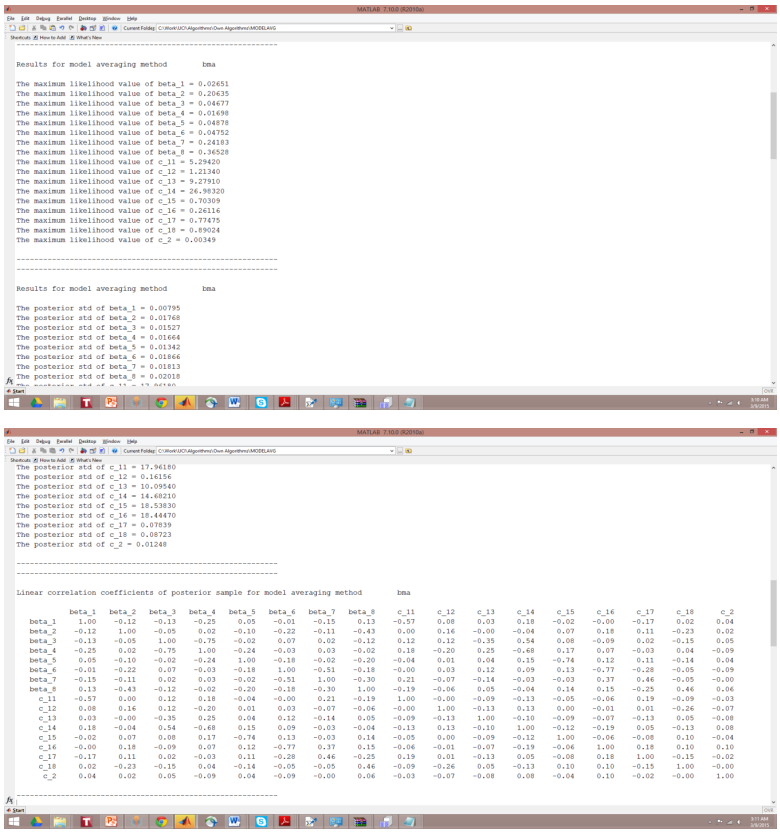

Figure 14: Different screen prints of the MATLAB command window after case study I has terminated. The BMA approach is used with a Gamma conditional distribution for each constituent member of the ensemble. The notation in the command window corresponds to the Equations presented in the main text. The postprocessing script has created 27 different figures - those are not shown herein.

Once the weights and or variance(s) have been determined, the user can proceed with analysis of the averaged forecast using the graphical output of

the script MODELAVG_POSTPROC. Alternatively, if DREAM has been executed, the user can interpret the marginal distributions of the weights and/or variance(s), the bivariate scatter plots of the posterior sample, and trace plots of the sampled chain trajectories and $\hat{R}$-convergence diagnostic.

10. 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.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.

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.

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.

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.

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, 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, 2007a.

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, "Markov chain Monte Carlo simulation using the DREAM software package: Theory, concepts, and MATLAB Implementation," *Environmental Modeling & Software*, vol. XX, no. XX, pp. XX-XX, doi:10.1016/j.envsoft.2014.XX.XXX, 2015a.

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

J.A. Vrugt, "FDCFIT: A MATLAB toolbox of closed-form parametric expressions of the flow duration curve, *Manual, Version 1.0*, pp. 1 - 36, 2015c.

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.