Dynamic Model Averaging for Practitioners in Economics and Finance: The eDMA Package

Leopoldo Catania
University of Rome, "Tor Vergata"

Nima Nonejad
Aalborg University and CREATES

Abstract

Raftery *et al.* (2010) introduce an estimation technique referred to as Dynamic Model Averaging (DMA). In their application, DMA is used to the problem of predicting the output strip thickness for a cold rolling mill, where the output is measured with a time delay. Recently, DMA has also shown to be very useful in macroeconomic and financial applications. In this paper, we present the **eDMA** package for DMA estimation in R, which is especially suited for practitioners in economics and finance. Our implementation proves to be up to 133 times faster then a standard implementation on a single–core CPU. With the help of this package, practitioners are able perform DMA on a standard PC without resorting to large clusters, which are not easily available to all researchers. We demonstrate the usefulness of this package through simulation experiments and an empirical application using quarterly U.S. inflation data.

Keywords: Dynamic model averaging, Multi core CPU, Parallel computing, `R`, `OpenMP`.

1. Introduction

Modeling and forecasting economic variables such as real GDP, inflation and equity premium is of clear importance for researchers in economics and finance. For instance, forecasting inflation is crucial for central banks with regards to conducting optimal monetary policy. Similarly, understanding and predicting equity premium is one of the most widely important topics discussed in financial economics as it has great implications on portfolio choice and risk management, see for instance Dangl and Halling (2012) among many others.

In order to obtain the best forecast as possible, practitioners often try to take advantage of the many potential predictors available and seek to combine the information from these predictors in an optimal way, see Stock and Watson (1999), Stock and Watson (2008) and Groen *et al.* (2013) just to mention a few references. In the context of economic applications, Koop and Korobilis (2011) and Koop and Korobilis (2012), implement a technique developed by Raftery *et al.* (2010), referred to as Dynamic Model Averaging (DMA). The original intent of the mentioned article is to predict the output strip thickness for a cold rolling mill, where the output is measured with a time delay. Typically, DMA consists of thousands of models based on all possible combinations of the predictors available. Furthermore, besides allowing for time–variation in the regression coefficients of each individual model, DMA also allows the relevant model set to change with time as well by introducing a forgetting factor. Koop and Korobilis (2011) and Koop and Korobilis (2012) argue (very elegantly) that by slightly adjusting the original framework, DMA can be very useful in economic contexts, especially

inflation forecasting.[1] Dangl and Halling (2012) provide further suggestions on how to improve DMA such that it better suits the nature of economic and financial data. The aforementioned authors, also provide a very useful variance decomposition scheme using the output from the estimation procedure to understand which source of uncertainty explains the degree of variation in the observed variable. Byrne *et al.* (2014), among others, use the modifications proposed in Dangl and Halling (2012) to model currency exchange–rate behavior. We must also emphasize that DMA is not solely limited to these series and can be used in a wide range of economic applications such as: Forecasting realized volatility as well as house, oil and commodity prices.

However, designing an efficient DMA algorithm remains a challenging issue. As previously mentioned, DMA considers all possible combinations from a set of predictors at each point in time. Typically, many candidate variables are available and, as a consequence, it poses a limit given the computational facilities at hand, which for many practitioners typically consists of a standard 8 core CPU. Thus, while handling a relatively small number of model combinations, usually between 1000 to 3000, allows one to perform DMA using standard loops and software, handling larger number of combinations becomes very burdensome, especially in the context of memory consumption, see also Koop and Korobilis (2012).

In order to deal with this issue, Onorante and Raftery (2016) suggest a strategy that considers not the whole model space, but rather a subset of models and dynamically optimizes the choice of models at each point in time. However, Onorante and Raftery (2016) have to assume that models do not change too fast over time, which is not an ideal assumption when dealing with financial and in some cases monthly economic data. Furthermore, the approach of Onorante and Raftery (2016) is not able to clearly state why certain predictors can be considered as less important than others or vice versa. Finally, incorporating the modifications suggested in Dangl and Halling (2012), which are very important with regards to interpretation of results, is not possible using the approach of Onorante and Raftery (2016).

In this paper, we introduce the **eDMA** package for R (R Core Team 2016), which efficiently implements a DMA procedure based on Raftery *et al.* (2010) and Dangl and Halling (2012). The routines in the **eDMA** package are principally written in C++ using the `armadillo` library of Sanderson (2010) and then made available in R exploiting the **Rcpp** and **RcppArmadillo** packages of Eddelbuettel *et al.* (2016a) and Eddelbuettel *et al.* (2016b), respectively. Furthermore, the `OpenMP` API (OpenMP 2008) is used to speedup the computations when a shared memory multiple processors hardware is available, which, nowadays, is standard for the majority of commercial laptops. However, if the hardware does not have multiple processors, the **eDMA** package can still be used with the classical sequential CPU implementation.

Our aim is to provide a package that can be used by a broad audience from different academic fields who are interested in implementing DMA in their research. Furthermore, our package enables practitioners, to perform DMA using a large number of predictors without needing to understand and possibly implement complex programming concepts such as "how to efficiently allocate memory", or "how to efficiently parallelize the computations".

It is worth noting that, within the R environment, the package **dma** of McCormick *et al.* (2016) downloadable from CRAN can be used to perform the DMA of Raftery *et al.* (2010).

[1] Specifically, Koop and Korobilis (2012) change the conditional volatility formula of Raftery *et al.* (2010) arguing that their original formula is not suited in the context of economic data. Their suggestion proves more compatible with the behavior of economic data such as inflation, see also Dangl and Halling (2012) for a similar approach.

However, **dma** has several weaknesses such as (i): Does not allow for the extensions of Dangl and Halling (2012), which are important in the context of interpreting the amount of time–variation in the regression coefficients and performing a variance decomposition analysis, (ii): It is remarkably slow compared to our alternative, (iii): Requires a very large amount of RAM when executed for moderately large applications, (iv): Does not allow for parallel computing. We refer the reader interested in these aspects to Section 4, where we report a comparative analysis between **dma** and **eDMA** using simulated data. **eDMA** permits us to also perform Bayesian Model Averaging (BMA) and Bayesian Model Selection (BMS) for linear regression models with constant coefficients implemented, for example, in the R packages **BMA** (Raftery *et al.* 2015) and **BMS** (Zeugner and Feldkircher 2015).

The empirical application in Section 6 presents different features of this package, provides illustrations on how practitioners can implement DMA and more importantly use the output from the estimation procedure to interpret results.

The structure of this paper is as follows: Sections 2 and 3 briefly introduce DMA and its extensions. Section 4 presents the technical aspects. Section 5 provides an intuitive description of the challenges that DMA posses from a computational point of view and proposes solutions. Section 6 provides an empirical application to demonstrate the advantages of **eDMA** from a practical point of view. Therefore, practitioners who are solely interested on how to implement DMA using the **eDMA** package can skip Sections 2 and 3. Finally, Section 7 concludes.

2. The model

In this section, we briefly introduce the DMA algorithm of Raftery *et al.* (2010). We then pinpoint some of the main challenges associated with applying DMA in economic and financial applications. Thereafter, we propose solutions in Section 5.

Let y_t denote the dependent variable at time t. Our goal is to model y_t using a pool of $i = 1, \ldots, k$, candidate Dynamic Linear Models (DLM) of the types introduced in West and Harrison (1999) and Raftery *et al.* (2010),

$$y_t = \mathbf{F}_t^{(i)\prime}\boldsymbol{\theta}_t^{(i)} + \varepsilon_t^{(i)}, \quad \varepsilon_t^{(i)} \sim N\left(0, V_t^{(i)}\right) \tag{1}$$

$$\boldsymbol{\theta}_t^{(i)} = \boldsymbol{\theta}_{t-1}^{(i)} + \boldsymbol{\eta}_t^{(i)}, \quad \boldsymbol{\eta}_t^{(i)} \sim N\left(0, \mathbf{W}_t^{(i)}\right), \tag{2}$$

where $\mathbf{F}_t^{(i)}$ is a subset from the total n predictors. Let p denote the numbers of predictors in $\mathbf{F}_t^{(i)}$ for model i. Then, $\boldsymbol{\theta}_t^{(i)}$ is a $p \times 1$ vector of time–varying regression coefficients, which evolve according to (2) and determine the impact of $\mathbf{F}_t^{(i)}$ on y_t. Note, we do not assume any systematic movements in $\boldsymbol{\theta}_t^{(i)}$. On the contrary, we consider changes in $\boldsymbol{\theta}_t^{(i)}$ as unpredictable.[2] The conditional variances, $V_t^{(i)}$ and $\mathbf{W}_t^{(i)}$, for model i are the unknown quantities associated with the observational equation, (1), and the state equation, (2). Obviously, when $\mathbf{W}_t^{(i)} = \mathbf{0}$ for $t = 1, \ldots, T$, then $\boldsymbol{\theta}_t^{(i)}$ is constant over time. Thus, our model can nest the specification of constant regression coefficients. As $\mathbf{W}_t^{(i)}$ increases, the variation in $\boldsymbol{\theta}_t^{(i)}$ varies according to Equation 2. However, this does not mean that $\boldsymbol{\theta}_t^{(i)}$ needs to change at every time period. For

[2] See Dangl and Halling (2012) and Koop and Korobilis (2012) for a similar model specification.

instance, we can simply have periods where $\mathbf{W}_t^{(i)} = 0$ in which $\boldsymbol{\theta}_t^{(i)} = \boldsymbol{\theta}_{t-1}^{(i)}$ and other periods when $\boldsymbol{\theta}_t^{(i)}$ is allowed to change according to Equation 2.[3]

In DMA, we consider a total of $k = 2^n - 1$ possible combinations of the predictors at each point in time while contemporaneously assuming that $\boldsymbol{\theta}_t^{(i)}$ changes according to Equation 2.[4] DMA then averages forecasts/predictions across these combinations using a recursive updating scheme based on the predictive likelihood. The predictive likelihood measures the ability of a model to predict y_t, thus making it the central quantity of interest for model evaluation. Apparently, models containing important combinations of predictors receive high predictive likelihood values, which means that these models obtain higher weights in the averagning process. Besides averaging, we can also use the prediction/forecasts of the model receiving the highest probability among all model combinations considered at each point in time. In this case, we are performing Dynamic Model Selection (DMS), see also Koop and Korobilis (2012).

In the context of estimation, DMA avoids the difficult task of specifying $\mathbf{W}_t^{(i)}$ for each individual model. Instead DMA relies on using a forgetting factor, $0 < \delta \leq 1$. This parameter describes the loss of information through time, which simplifies things greatly from a practical point of view. The forgetting factor avoids the need to estimate $\mathbf{W}_t^{(i)}$ for each individual model. Particularly, using the same notation as the Appendix in Dangl and Halling (2012), we define the variables in the Kalman recursions of the i–t model as: (i): $\mathbf{R}_t^{(i)}$, the unconditional variance of $\boldsymbol{\theta}_t^{(i)}$ (see Equation 14 in the Appendix of Dangl and Halling (2012)), (ii): $\mathbf{C}_t^{(i)}$, the estimator for the variance of $\boldsymbol{\theta}_t^{(i)}$, (see Equation 20 in the Appendix of Dangl and Halling (2012)), and (iii): $S_t^{(i)}$, the estimator of the observational variance (see Equation 17 in the Appendix of Dangl and Halling (2012)). Then, using δ, we can rewrite $\mathbf{R}_t^{(i)} = \delta^{-1}\mathbf{C}_{t-1}^{(i)}$, indicating that there is a relationship between $\mathbf{W}_t^{(i)}$ and δ, which is given as $\mathbf{W}_t^{(i)} = (1-\delta)/\delta\mathbf{C}_{t-1}^{(i)}$. In other words, the loss of information is proportional to the covariance of the state parameters. This way, we can control the magnitude of the shocks that affect $\boldsymbol{\theta}_t^{(i)}$ by adjusting δ instead of directly estimating $\mathbf{W}_t^{(i)}$. Accordingly, $\delta = 1$ corresponds to $\mathbf{W}_t^{(i)} = \mathbf{0}$, which means that $\boldsymbol{\theta}_t^{(i)}$ is constant over time. For $\delta < 1$, we introduce time–variation in $\boldsymbol{\theta}_t^{(i)}$. For instance, when $\delta = 0.99$, in the context quarterly data, observations five years ago receive approximately 80% as much weight as last period's observation, which corresponds to gradual time–variation in $\boldsymbol{\theta}_t^{(i)}$. When $\delta = 0.95$, observations 20 periods ago receive only about 35% as much weight as last period's observations, suggesting relatively more drastic change in $\boldsymbol{\theta}_t^{(i)}$ at each point in time. Evidently, while this renders the model more flexible to adapt to changes in y_t, the increased variability in $\boldsymbol{\theta}_t^{(i)}$ translates into high prediction variance. Thus, finding the correct value of δ at each point in time is extremely important with regards to the ability of the model to capture the correct magnitude of variations in $\boldsymbol{\theta}_t^{(i)}$ an thus produce optimal forecasts/predictions. The same line of argumentation holds for the parameter, α, which controls the forgetting for the entire model set, see Raftery *et al.* (2010) for more details. For instance, $\alpha = 0.95$ means that past model performance is less important than when $\alpha = 0.99$. We must

[3] As indicated below, we model time–variation in $\boldsymbol{\theta}_t^{(i)}$ through a forgetting factor, δ. The way we update δ, which determines the magnitude of the shocks that hit $\boldsymbol{\theta}_t^{(i)}$ at each t, avoids any unreasonable behavior of $\boldsymbol{\theta}_t^{(i)}$. Thus, we do not need to put any restrictions on $\boldsymbol{\theta}_t^{(i)}$ itself, see Dangl and Halling (2012) for a similar approach.

[4] The model $y_t = \varepsilon_t$ is **not** considered in the universe of models, see also Dangl and Halling (2012).

also determine a way to model the evolution of $V_t^{(i)}$. Here, we have several choices, which we go into more details below, see point (c).

To summarize, DMA depends on:

(a) The number of predictors to consider. Typically, in economic applications, the set of predictors contains exogenous predictors as well as lagged values of y_t. For instance, in the context of forecasting quarterly inflation, besides considering predictors such as unemployment rate and T–bill rates, Koop and Korobilis (2012) also consider the first three lags of y_t as explanatory variables. Of course, how to distinguish useful variables from noise remains an important issue. For instance, following a recession or revisions of data, the set of predictors to use probably changes.

(b) The choice of the forgetting factors, α and δ. In many applications $\alpha \in \{0.98, 0.99, 1\}$ work very well and results do not change drastically across different values of α. On the other hand, as previously mentioned, we often find that the choice of δ is more important. Koop and Korobilis (2012) fix δ at $\{0.95, 0.98, 0.99, 1.00\}$ and run DMA using each of these values. They find that results differ considerably in terms of out–of–sample forecasts, see Koop and Korobilis (2012) for more details. Evidently, in many economic applications, it is very plausible that δ would indeed be time–varying. For instance, it is plausible to expect that δ is relatively low in recessions or periods of market turmoil (as there is considerable time–variation in $\boldsymbol{\theta}_t^{(i)}$ in these periods). Conversely, δ is expected to be close to 1.00 during tranquil and periods of low volatility. Dangl and Halling (2012) propose a very elegant solution to this problem by considering a grid of values for δ and incorporate this in the DMA setting by averaging over all possible combinations of the predictors as well as over a grid value for δ. Furthermore, this procedure can also be used to obtain more information from the data through a variance decomposition scheme, see below for more details.

(c) Modeling $V_t^{(i)}$: In this paper, we make things easy for conjugate analysis by using the following assumptions, see also Dangl and Halling (2012) and Byrne *et al.* (2014) for the same approach: We assume that $V_t^{(i)} = V^{(i)}$ for all t. For time $t = 0$, we specify a Normal prior on $\boldsymbol{\theta}_0^{(i)}$ and a Gamma prior on $\phi^{(i)} = V^{-1(i)}$. The time t estimate of the variance of the error term in (1) is then given as Equation 17 in the Appendix of Dangl and Halling (2012). More importantly, by using these assumptions, we find that, when we integrate the conditional density of y_t over the values of $\boldsymbol{\theta}_t^{(i)}$ and $V^{(i)}$ to obtain the predictive density, $p\left(y_t|\mathcal{F}_{t-1}\right)$, the corresponding density has a closed–form solution, which is given as $p\left(y_t|\mathcal{F}_{t-1}\right) \sim t_{n_t^{(i)}}\left(\hat{y}_t^{(i)}, Q_t^{(i)}\right)$, where $t_{n_t^{(i)}}$ stands for the Student–t distribution with $n_t^{(i)}$ degrees–of–freedom and mean and scale given by $\hat{y}_t^{(i)}$ and $Q_t^{(i)}$, see Equations 13 and 16 of Dangl and Halling (2012), respectively.

We can also follow Koop and Korobilis (2012) and use an Exponentially Weighted Moving Average (EWMA) estimate of $V_t^{(i)}$. However, this requires the practitioner to also consider an additional parameter, namely, κ, which increases the computation burden. Furthermore, by experimenting with EWMA on a smaller model, we find that δ and κ in many ways are intertwined, in the sense that we obtain the same magnitudes of variation in $\boldsymbol{\theta}_t^{(i)}$ as for (c) when we allow κ and δ to vary over time. In this case, compared to (c), we obtain higher

estimates of δ over time whereas we estimate κ close to 0.96, which is the value suggested in Koop and Korobilis (2012). We observe the same phenomena when we allow α to vary with δ. Overall, our conclusion is that it is best to use (c) and fix α close to 0.99 for monthly and quarterly data. This way, we maintain a very parsimonious model structure and are never in doubt with regards to under (over) estimating the true magnitude of variation in δ (α).

3. Modified DMA

Below, we present the DMA algorithm modified to incorporate the extensions mentioned above. We refer the reader to Dangl and Halling (2012) for more details. Let M_i denote a model containing a specific set of predictors chosen from a set of $k = 2^n - 1$ candidates and $\delta_j \in \{\delta_1, ..., \delta_d\}$ denote a specific choice of the degree of time–variation in the regression coefficients at time t. The total posterior density of model M_i and forgetting δ_j at time t, $p(M_i, \delta_j|\mathcal{F}_t)$, is given as

$$p(M_i, \delta_j|\mathcal{F}_t) = p(M_i|\delta_j, \mathcal{F}_t)\, p(\delta_j|\mathcal{F}_t).$$

In order to obtain $p(M_i|\mathcal{F}_t)$ and $p(\delta_j|\mathcal{F}_t)$, we can use that

$$p(M_i|\mathcal{F}_t) = \sum_{j=1}^{d} p(M_i|\delta_j, \mathcal{F}_t)\, p(\delta_j|\mathcal{F}_t). \tag{3}$$

The term, $p(M_i|\delta_j, \mathcal{F}_t)$, in Equation 3 is given as

$$p(M_i|\delta_j, \mathcal{F}_t) = \frac{p(y_t|M_i, \delta_j, \mathcal{F}_{t-1})\, p(M_i|\delta_j, \mathcal{F}_{t-1})}{\sum_{l=1}^{k} p(y_t|M_l, \delta_j, \mathcal{F}_{t-1})\, p(M_l|\delta_j, \mathcal{F}_{t-1})} \tag{4}$$

where

$$p(M_i|\delta_j, \mathcal{F}_{t-1}) = \frac{p(M_i|\delta_j, \mathcal{F}_{t-1})^{\alpha}}{\sum_{l=1}^{k} p(M_l|\delta_j, \mathcal{F}_{t-1})^{\alpha}}. \tag{5}$$

The second term on the right–hand side of Equation 3 is given as

$$p(\delta_j|\mathcal{F}_t) = \frac{p(y_t|\delta_j, \mathcal{F}_{t-1})\, p(\delta_j|\mathcal{F}_{t-1})}{\sum_{l=1}^{d} p(y_t|\delta_l, \mathcal{F}_{t-1})\, p(\delta_l|\mathcal{F}_{t-1})}. \tag{6}$$

where

$$p(\delta_j|\mathcal{F}_{t-1}) = \frac{p(\delta_j|\mathcal{F}_{t-1})^{\alpha}}{\sum_{l=1}^{d} p(\delta_l|\mathcal{F}_{t-1})^{\alpha}}.$$

As previously mentioned, $p(y_t|M_i, \delta_j, \mathcal{F}_{t-1}) \sim t_{n_t}\left(\hat{y}_{i,t}^{(j)}, Q_{i,t}^{(j)}\right)$, where $\hat{y}_{i,t}^{(j)}$ and $Q_{i,t}^{(j)}$ are the quantities of model M_i, $i = 1, ..., k$, conditional on δ_j, $j = 1, ..., d$, and α. Typically, $p(M_i, \delta_j|\mathcal{F}_0) = 1/(d \cdot k)$ such that, initially, all model combinations and degrees of time–variation are equally likely. Thereafter, as a new observation arrives, model probabilities are updated using the above recursions.

3.1. Using the output from DMA

For practitioners, the most interesting output from DMA are:

(i) The predictive mean of y_{t+1} conditional on $\mathcal{F}_t, \hat{y}_{t+1}$. This is simply an average of each of the individual model predictive means. That is

$$\hat{y}_{t+1} = \sum_{j=1}^{d} \mathsf{E}\left[y_{t+1}^{(j)}|\mathcal{F}_t\right] p\left(\delta_j|\mathcal{F}_t\right), \tag{7}$$

where

$$\mathsf{E}\left[y_{t+1}^{(j)}|\mathcal{F}_t\right] = \sum_{i=1}^{k} \mathsf{E}\left[y_{i,t+1}^{(j)}|\mathcal{F}_t\right] p\left(M_i|\delta_j, \mathcal{F}_t\right).$$

The formulas for the predictive density are very similar. We only replace the predictive mean with the predictive density inside Equation 7, see Dangl and Halling (2012). Besides averaging over the individual predictive means/densities, we can simply choose the predictive mean/density associated with the model with the highest posterior probability, see Equation 3. Henceforth, we label this as Dynamic Model Selection (DMS). Furthermore, as mentioned in Koop and Korobilis (2012), for DMA and DMS, with δ and α fixed at 1, we have Bayesian model averaging (BMA) and Bayesian Model Selection (BMS) based on exact predictive likelihood, see for instance Zeugner and Feldkircher (2015).[5]

(ii) Quantities such as the expected size of the predictor, $\mathsf{E}\left[Size_t\right] = \Sigma_{i=1}^{k} Size^{(i)} p\left(M_i|\mathcal{F}_t\right)$, where $Size^{(i)}$ be the number of predictors in model i. This quantity reveals the average number of predictors in the DMA, see Koop and Korobilis (2012). Similarly, we can compute the number of predictors for the model with the highest posterior probability, (3), at each point in time, which give the optimal model size at time t.

(iii) Posterior inclusion probabilities for the predictor. That is, at each t we calculate $\sum_{i=1}^{k} 1_{(i \subset m)} p\left(M_i|\mathcal{F}_t\right)$, where $1_{(i \subset m)}$ is an indicator function taking the value of either 0 or 1 and m is the mth predictor. We can also report the highest posterior model probability or the sum of the top 10% model probabilities among all model combinations after the effect of δ is integrated out. This information can be used to determine if there is a group or an individual model that obtains relatively high posterior probability.

(iv) Posterior weighted average of δ at each point in time, see Equation 6.

(v) Posterior weighted average estimates of $\boldsymbol{\theta}_t$ for DMA,

$$\mathsf{E}\left[\boldsymbol{\theta}_t|\mathcal{F}_t\right] = \sum_{j=1}^{d} \mathsf{E}\left[\boldsymbol{\theta}_t^{(j)}|\mathcal{F}_t\right] p\left(\delta_j|\mathcal{F}_t\right), \tag{8}$$

where

$$\mathsf{E}\left[\boldsymbol{\theta}_t^{(j)}|\mathcal{F}_t\right] = \sum_{i=1}^{k} \mathsf{E}\left[\theta_{i,t}^{(j)}|\mathcal{F}_t\right] p\left(M_i|\delta_j, \mathcal{F}_t\right).$$

[5] Zeugner and Feldkircher (2015) also implement BMA using the MC^3 algorithm relying on Markov Chain Monte Carlo (MCMC) techniques. However, their framework does not allow for time–variation in the regression coefficients nor model size.

(vi) Variance decomposition of the data, $\text{Var}\left(y_{t+1}|\mathcal{F}_t\right)$, decomposed into

$$\begin{aligned}\text{VAR}\left(y_{t+1}|\mathcal{F}_t\right) = &\sum_{j=1}^{d}\left[\sum_{i=1}^{k}\left(S_t|M_i,\delta_j,\mathcal{F}_t\right)p\left(M_i|\delta_j,\mathcal{F}_t\right)\right]p\left(\delta_j|\mathcal{F}_t\right)\\ &+\sum_{j=1}^{d}\left[\sum_{i=1}^{k}\left(\mathbf{F}_t'\mathbf{R}_t\mathbf{F}_t|M_i,\delta_j,\mathcal{F}_t\right)p\left(M_i|\delta_j,\mathcal{F}_t\right)\right]p\left(\delta_j|\mathcal{F}_t\right)\\ &+\sum_{j=1}^{d}\left[\sum_{i=1}^{k}\left(\hat{y}_{t+1,i}^{(j)}-\hat{y}_{t+1}^{(j)}\right)^2 p\left(M_i|\delta_j,\mathcal{F}_t\right)\right]p\left(\delta_j|\mathcal{F}_t\right)\\ &+\sum_{j=1}^{d}\left(\hat{y}_{t+1}^{(j)}-\hat{y}_{t+1}\right)^2 p\left(\delta_j|\mathcal{F}_t\right). \qquad (9)\end{aligned}$$

The first term is the observational variance, Obs. The remaining terms are: Variance due to errors in the estimation of the coefficients, Coeff, variance due to uncertainty with respect to the choice of the predictors, Mod, and variance due to uncertainty with respect to the choice of the degree of time–variation in the regression coefficients, TVP, see Dangl and Halling (2012) for more details.

4. The eDMA package for R

The **eDMA** package for R offers an integrated environment for practitioners in economics and finance to perform our DMA algorithm. It is principally written in C++ exploiting the **armadillo** library of Sanderson (2010) to speed up computations. The relevant functions are then made available in R through the **Rcpp** and **RcppArmadillo** packages of Eddelbuettel *et al.* (2016a) and Eddelbuettel *et al.* (2016b), respectively. It also makes use of the `OpenMP` API (OpenMP 2008) to parallelize part of the routines needed to perform DMA. Furthermore, multiple processors are automatically used if supported by the hardware, however, as will be discussed later, the user is free to manage the level of resources used by the program.

The **eDMA** package is written using the S4 object oriented language, meaning that classes and methods are available in the code. Specifically, R users will find common methods such as `plot()`, `show()`, `as.data.frame()`,`coef()` and `residuals()`, among others, in order to visualise the output of DMA and extract estimated quantities.

Package **eDMA** is available from GitHub at https://github.com/LeopoldoCatania/eDMA and can be installed using the command:[6]

```
R> library("devtools")
R> install_github("LeopoldoCatania/eDMA")
```

Once the package is correctly installed and loaded, the user faces one function named `DMA()` to perform DMA over a series of possible models. The `DMA()` function accepts a series of

[6]Note that, to build the **eDMA** package under Windows it is advised to install Rtools from https://cran.r-project.org/bin/windows/Rtools/ before preceding to the next steps.

arguments and returns an object of the class `DMA` which comes with several methods, see Section 4.2. The arguments the `DMA()` function accepts are:

- `formula`: An object of class `formula` (or one that can be coerced to that class): A symbolic description of the model to be fitted. The formula should include all the predictors one chooses to include. The inclusion of the constant term follows the usual R practice, *i.e.*, it is included by default and can be removed if necessary. For instance, in order to model `y ~ x`, however, without the constant, we can write for example, `y ~ x - 1`, see `help(formula)`. This implementation follows the common practice for R users, see *e.g.*, the **plm** package of Croissant and Millo (2008).

- `data`: A `data.frame` (or object coercible by `as.data.frame()` to a `data.frame`) containing the variables in the model. If `data` is an object of the class `ts`, `zoo` or `xts`, then the time information is used in the graphical representation of the results as well as for the estimated quantities. The dimension of `data` is $T \times (1 + n)$, containing at each row, the dependent variables y_t and the predictors $\mathbf{F}_t$, that is $(y_t, \mathbf{F}_t')$, for all $t = 1, \ldots, T$.

- `vDelta`: A $d \times 1$ numeric vector representing a grid of δ. Typically we choose the following grid: $\{0.90,\ 0.91,\ \ldots,\ 1.00\}$. By default `vDelta = c(0.90, 0.95, 0.99)`.

- `dAlpha`: A numeric variable representing α in Equation 5. By default `dAlpha = 0.99`.

- `vKeep`: A numeric vector of indices representing the predictors that must be always included in the models. The models that do not include the variables declared in `vKeep` are automatically discarded. The indices must be consistent with the model description given in `formula`. For instance, if the first and fourth variables always have to be included, then we must set vKeep=c(1, 4). Notice that, the intercept (if not removed from `formula`) is always in the first position. `vKeep` can also be a character vector indicating the names of the predictors if these are consistent with the provided `formula`. Furthermore, if `vKeep = "KS"` the "Kitchen Sink" formulation is adopted, *i.e.*, all the predictors are always included, see, *e.g.*, Paye (2012). By default all the combinations are considered, `vKeep = NULL`.

- `bZellnerPrior`: A boolean variable indicating whether the Zellner's prior (see Dangl and Halling 2012) should be used for the coefficients at time $t = 0$. By default `bZellnerPrior = FALSE`.

- `dG`: A numeric variable equal to 100 by default. If `bZellnerPrior = TRUE`, this represents the prior hyperparameter value, g, in Equation 4 of Dangl and Halling (2012). Otherwise, if `bZellnerPrior = FALSE`, it represents the scaling factor for the variance covariance matrix of the Normal prior for $\boldsymbol{\theta}_0$, *i.e.*, $\boldsymbol{\theta}_0 \sim N(0, dG \times \mathbb{I})$, where $\mathbb{I}$ is the identity matrix. We generally recommend practitioners to use the default prior, especially in the context of quarterly data, where we typically have 200 to 300 observations. For longer time–series, the differences between priors is of relatively less importance.

- `bParallelize`: A boolean variable indicating wether to use multiple processors to speed up the computations. By default `bParallelize = TRUE`. Since the use of multiple processors is basically effortless for the user, we suggest to not change this value. Furthermore, if the hardware does not permit parallel computations, the program will automatically adapt to run on a single core.

- `iCores`: An integer indicating the number of cores to use if `bParallelize = TRUE`. By default, all but one cores are used. The number of cores is guessed using the `detectCores()` function from the **parallel** package. The choice of the number of cores depends somehow from the specific application, namely the length of the time–series T and the number of the predictors n. However, as detailed in Chapman *et al.* (2008), the level of parallelization of the code should be traded off with the increase in computational time due to threads communications. Consequently, the user can fine tune its application depending on its hardware changing this parameter.

The `DMA()` function returns an object of the *formal* class `DMA`.[7] This object contains model information and the estimated quantities. It is organized in three slots: `model`, `Est`, `data`. The slot, `model`, contains information about the specification used to perform DMA. Examples are: The number of considered models and the computational time in seconds. The slot, `Est`, contains the estimated quantities such as: Point forecasts, Predictive likelihood, Posterior inclusion probabilities of the predictors, Filtered estimates of the regression coefficients, $\boldsymbol{\theta}_t$, and so on. Finally, the slot, `data`, includes the data passed to the `DMA()` function, organised in the vector of responses `vY` and a design matrix `mF`.

4.1. Using eDMA

After having installed **eDMA**, it can be easily loaded using:

```
R> library("eDMA")
```

Thereafter, model estimation can be performed using the R commands reported below.

In order to illustrate how **eDMA** works in practice, we present the following guidelines using simulated data. We also provide an application using quarterly inflation data in Section 6.

We simulate a time–series of $T = 500$ observations from

$$y_t = \mathbf{F}_t'\boldsymbol{\theta}_t + \sqrt{0.1}\varepsilon_t, \quad \varepsilon_t \overset{iid}{\sim} \mathcal{N}(0,1)\,. \tag{10}$$

The first four elements of $\boldsymbol{\theta}_t$ vary according to random–walks, whereas the remaining elements in $\boldsymbol{\theta}_t$ are equal to zero at all time periods. In other words, $\boldsymbol{\theta}_t = (\theta_{1,t}, \theta_{2,t}, \theta_{3,t}, \theta_{4,t}, \theta_{5,t}, \theta_{6,t})'$ with

$$\theta_{k,t} = \theta_{k,t-1} + \sqrt{0.01}\eta_{k,t}, \quad \eta_{k,t} \overset{iid}{\sim} \mathcal{N}(0,1)\,, \tag{11}$$

for $k = 1, 2, 3, 4$, and $\eta_{k,t} \perp\!\!\!\perp \eta_{j,t}$, for all $k \neq j$. The last two elements of $\boldsymbol{\theta}_t$ are equal to zero, that is, $\theta_{5,t} = \theta_{6,t} = 0$ for $t = 1, \ldots, T$. The first element of the 6×1 vector, $\mathbf{F}_t$, is one, representing the constant term. The remaining elements are generated from a standard Gaussian distribution, *i.e.*, $\mathbf{F}_t = (1.0, x_{2,t}, x_{3,t}, x_{4,t}, x_{5,t}, x_{6,t})'$, where $x_{k,t} \overset{iid}{\sim} \mathcal{N}(0,1)$ and $x_{k,t} \perp\!\!\!\perp x_{j,t}$ for all $k \neq j$. We simulate the data in the following way (that is $\theta_{5,t} = \theta_{6,t} = 0$) to illustrate that DMA is indeed able to identify the correct variables. In other words, the inclusion probabilities of the last two predictors ought to be zero as they do not impact y_t

[7] see, `help("class")` and `help("DMA-class")`.

through $\mathbf{F}_t$. Conversely, inclusion probabilities of the first four predictors ought to converge to 1.

This data is simulated using the `SimulateDLM()` function available in **eDMA**, details are reported in the R documentation, see `help("SimulateDLM")`. We organize the data in a `data.frame` named `SimData`, which is included in **eDMA** and can be loaded into the workspace by executing:

```
R> data("SimData", package = "eDMA")
```

DMA is then performed using the function `DMA()` as:

```
R> Fit <- DMA(y ~ x2 + x3 + x4 + x5 + x6 , data = SimData,
          vDelta = seq(0.9, 1.0, 0.01))
```

Information on the DMA procedure is available by typing:

```
R> Fit

-----------------------------------------
-        Dynamic Model Ageraging        -
-----------------------------------------

Model Specification
T     = 500
n     = 6
d     = 11
Alpha = 0.99
Model combinations = 63
Model combinations including averaging over delta = 693
-----------------------------------------
Prior : Multivariate Gaussian with mean vector 0
        and covariance matrix equal to: 100 x diag(6)
-----------------------------------------
The grid for delta:

Delta =  0.90, 0.91, 0.92, 0.93, 0.94, 0.95,
         0.96, 0.97, 0.98, 0.99, 1.00
-----------------------------------------

Elapsed time        : 0.57 secs
```

Note, we specify a grid of eleven equally spaced values for δ ($d = 11$) ranging from 0.90 to 1.00. Furthermore, since we do not specify any value for `bZellnerPrior` and `bParallelize`, their default values, `bZellnerPrior = FALSE` and `bParallelize = TRUE` have been used.

In order to extract the quantities estimated by DMA, the user can relay on the `as.data.frame()` method. `as.data.frame()` accepts two arguments: (i): An object of the class `DMA` and (ii): A character string, `which`, indicating the quantity to extract. Possible values for `which` are:

- `"vyhat"`: Point forecasts of DMA, see Equation 7. `"vyhat_DMS"` for point forecast according to DMS.
- `"mincpmt"`: Posterior inclusion probabilities of the predictors at each point in time, see Koop and Korobilis (2012) for more details.
- `"vsize"`: Expected number of predictors (average size), see Koop and Korobilis (2012) and point (ii) at page 7.
- `"vsize_DMS"`: Number of predictors in the model with the highest posterior model probability, at each point in time, see Equation 3.
- `"mtheta"`: Filtered estimates of the regression coefficients for DMA, see Equation 8.
- `"mpmt"`: Posterior probability of the degrees of instability, see Equation 6.
- `"vLpdfhat"`: Predictive (log–)likelihood of DMA, see Dangl and Halling (2012).
- `"vLpdfhat_DMS"`: Predictive (log–)likelihood according to DMS.
- `"vdeltahat"`: Posterior weighted average of δ, that is $\hat{\delta}_t = \sum_{j=1}^{d} \delta_j p\left(\delta_j | \mathcal{F}_t\right)$.
- `"mvdec"`: Individual components of Equation 9, see point (vi) in page 8 and Dangl and Halling (2012) for more details. The function returns a $T \times 5$ matrix whose columns contain the variables.
  - `vobs`: Observational variance, Obs.
  - `vcoeff`: Variance due to errors in the estimation of the coefficients, Coeff.
  - `vmod`: Variance due to model uncertainty, Mod.
  - `vtvp`: Variance due to uncertainty with respect to the choice of the degrees of time–variation in the regression coefficients, TVP.
  - `vtotal`: Total variance, that is `vtotal` = `vobs` + `vcoeff` + `vmod` + `vtvp` .
- `"vhighmp_DMS"`: Highest posterior model probability, *i.e.*, $\max_i P\left(M_i | \mathcal{F}_t\right), \quad t = 1, \ldots, T.$
- `"vhighmpTop01_DMS"`: Sum of the 10% highest posterior model probabilities.

The additional `numeric` argument, `iBurnPeriod`, determines the length of the burn–in period, *i.e.*, results before t=`iBurnPeriod` are discarded. By default, `iBurnPeriod = NULL`, meaning that no burn–in period is considered. For instance, in order to extract the posterior inclusion probabilities of the predictors, with a burn–in period of 50 observations, we can easily run the following command

```
R> PostProb = as.data.frame(Fit, which = "mincpmt", iBurnPeriod = 50)
```

which returns a ($T-$`iBurnPeriod`) $\times$ 7 matrix of inclusion probabilities for the predictors at each point in time. Final values of `PostProb` are printed as:

```
R> round(tail(PostProb), 2)
```

```
       (Intercept) x2 x3 x4   x5   x6
[445,]           1  1  1  1 0.06 0.03
[446,]           1  1  1  1 0.06 0.03
[447,]           1  1  1  1 0.07 0.03
[448,]           1  1  1  1 0.07 0.03
[449,]           1  1  1  1 0.07 0.03
[450,]           1  1  1  1 0.09 0.04
```

Furthermore, if the supplied data is a `ts`, `zoo` or `xts` object, the class membership is automatically transferred to the output of the `as.data.frame()` method.

The `plot()` method is also available for the class `DMA`. Specifically, this method prints an interactive menu in the console permitting the user to chose between a series of interesting graphical representation of the estimated quantities. It can be straightforwardly executed running:

```
R> plot(Fit)
```

```
Print 1-11 or 0 to exit
 1: Point forecasts
 2: Predictive likelihood
 3: Posterior weighted average of delta
 4: Posterior inclusion probabilities of the predictors
 5: Posterior probability of the degree of instability
 6: Filtered estimates of the regression coefficients, theta
 7: Observational variance
 8: Variance due to errors in the estimation of the coefficients
 9: Variance due to model uncertainty
10: Variance due to uncertainty with respect to the choice of
    the degrees of time-variation in the regression coefficients
11: Expected number of predictors (average size)
12: Number of predictors (highest posterior model probability) (DMS)
13: Highest posterior model probability (DMS)
14: Point forecasts (highest posterior model probability) (DMS)
15: Predictive likelihood (highest posterior model probability) (DMS)
```

and selecting the desiderated options. The additional character argument, `which`, can be supplied in order to directly plot one particular quantity. Possible values for `which` are the same of the `as.data.frame()` method. Similar to `as.data.frame()`, the additional `numeric` argument `iBurnPeriod` determines the length of the burn–in period. Typically, it takes around 30 to 50 for the model to adapt to the time–series given the prior. Therefore, in almost all applications, the first 30 to 50 observations are discarded.

The code:

```
R> plot(Fit, which = "mincpmt", iBurnPeriod = 50)
```

plots the inclusion probabilities for the predictors discarding the first 50 observations. The outcome is reported in Figure 1. As expected, x_1 to x_4 quickly converge to 1 after few

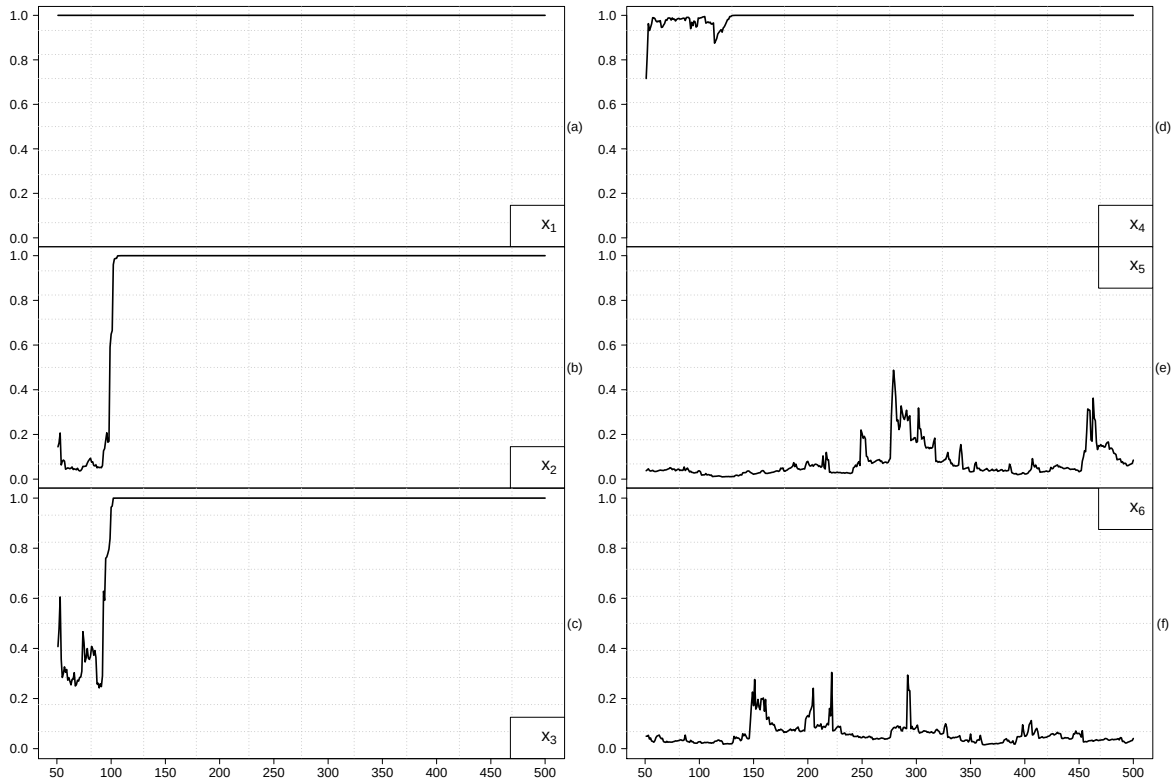

Figure 1: Posterior inclusion probabilities of the predictors using simulated data.

observations. Conversely, the inclusion probabilities of the last two predictors with loading factor equal to zero, quickly converge to 0.

4.2. Additional methods for the `DMA` class

The `DMA` class comes with several methods for extracting and representing estimated quantities. The `plot()`, `as.data.frame()` and `show()` methods have been previously introduced, additional methods are: `summary()`, `coef()`, `residuals()`, `inclusion.prob()`, and `pred.like()`.

For instance, the `summary` method prints a summary of the estimated model directly in the console. The code:

```
R> summary(Fit, iBurnPeriod = 50)
```

produces the output:

```
Call:
 DMA(formula =  vY ~ x1 + x2 + x3 + x4 + x5 )

Residuals:
    Min      1Q  Median      3Q     Max
-2.0408 -0.3833  0.0421  0.4427  2.3140
```

```
Coefficients:
            E[theta_t] SD[theta_t] E[P(theta_t)] SD[P(theta_t)]
(Intercept)       0.51        0.68          1.00           0.00
x1               -0.64        0.65          0.90           0.29
x2                2.11        1.74          0.93           0.20
x3               -1.43        1.02          0.99           0.02
x4                0.01        0.03          0.07           0.07
x5                0.00        0.01          0.06           0.04

Variance contribution (in percentage points):
  vobs vcoeff   vmod   vtvp
  3.18   1.73   0.08  95.01

Top 10% included predictors:        (Intercept)

Forecast Performance:
                            DMA      DMS
MSE                       0.486    0.482
MAD                       0.537    0.532
Predictive Likehood -462.360 -463.083
```

where the quantities, `E[theta_t], SD[theta_t], E[P(theta_t)]` and `SD[P(theta_t)]` represent the means and standard deviations across the time dimension of the filtered estimates of $\boldsymbol{\theta}_t^{(i)}$, and the inclusion probabilities after burn-in.

The last part of the summary, (`Forecast Performance`), prints the output of the `BacktestDMA()` function implemented in **eDMA**. `BacktestDMA()` accepts a `DMA` object and returns a `matrix` with out–of–sample Mean Squared Error (MSE), Mean Absolute Deviation (MAD) and Predictive Likelihood, computed according to DMA and DMS, see `help("BacktestDMA")`.

The additional methods: `coef()`, `residuals()`, `inclusion.prob()`, and `pred.like()` are wrapper to the `as.data.frame()` method and focus on particular estimated quantities, for instance:

- `coef()`: Returns a $T \times n$ `matrix` with the filtered regressor coefficients, $\boldsymbol{\theta}_t, \quad t = 1, \ldots, T$.
- `residuals()`: Extract the residuals of the model, *i.e.*, $y_t - \hat{y}_t, \quad t = 1, \ldots, T$. The additional `boolean` argument `standardize` controls if the standardize residuals should be returned. By default `standardize = FALSE`. The additional `character` argument, `Type`, permits to choose between residuals evaluated using DMA (`"DMA"`) or DMS (`"DMS"`). By default `Type = "DMA"`.
- `inclusion.prob()`: Extract the inclusion probabilities of the predictors. Analogous to `as.data.frame(object, which = "mincpmt", iBurnPeriod)`.
- `pred.like()`: Extract the predictive log likelihood series. The additional argument `Type` permits to choose between predictive likelihoods evaluated using DMA and DMS. By default `Type = "DMA"`. Similar to the above variables, `pred.like()` accepts `iBurnPeriod`.

5. Computational challenges

Although estimation of DMA does not require resorting to simulation methods, in many economic applications, performing DMA can become computationally very cumbersome. As it can be seen from the set of recursions from the Section 3, DMA consists of considering a large number of model combinations. In many cases, DMA tends to occupy a large chunk of Random–Access Memory (RAM). Often on a standard PC, the system basically runs out of memory due to the large number of combinations and the amount of information that must be saved. Therefore, it limits the use of DMA to middle–sized data sets. For instance, in their seminal paper, Koop and Korobilis (2012) use DMA to forecast quarterly inflation. Thus, y_t in Equation 1 is the percentage changes in the quarterly U.S. GDP price deflator and $\mathbf{F}_t$ consists of 14 exogenous predictors and three lags of y_t for a total of 17 variables. However, handling 2^{17} combinations reveals to be very burdensome in their programming framework. Therefore, Koop and Korobilis (2012) choose to include three lags of inflation in all model combinations and thus reduce the model space to 2^{14} model combinations.

We can argue that DMA can impose a very substantial challenge for the practitioner when dealing with a large number of predictors, namely that, besides dealing with the task of transforming mathematical equations from paper to codes, handling data and estimation issues, a practitioners also has to overcome "technical/computer science" challenges such as how to deal with extensive memory consumption and how to use multiple cores instead of a single core to speed up computation time. Although one can always improve the computational procedure by "coding smarter" or discovering ways to optimize memory allocation, it seems unreasonable to expect that practitioners in economics should have extensive knowledge of computer science concepts such as those stated above.

In this paper, we provide practical solutions to these problems. First, reduction in computation time is implemented by writing all the code in C++ using the `armadillo` library of Sanderson (2010). Second, we exploit multiple processors through the `OpenMP` API whenever the hardware is suited for that. The combination of C++ routines and parallel processing permits to dramatically speed up the computations over the same code written in plain R.

In order to provide an intuitive example of the advantages of our package, we report a comparison between our code and the available **dma** package of McCormick *et al.* (2016). For this experiment, since the **dma** package cannot operate over a grid value of δ, we fix δ at 0.95. We simulate $T = \{100, 500, 1000\}$ observations from a DLM with $n = \{4, 6, 8, 10, 12, 14, 16\}$ predictors and evaluate the differences in the computational time of the `dma()` function in the **dma** package and the `DMA()` function in the presented **eDMA** package. The experiment is performed on a standard Intel Core i7–4790 processor with 8 threads and Ubuntu 12.04 server edition.

Table 1 reports the ratio of the CPU time for different values of T and n between `dma()` and `DMA()`. As one can note, the decrease in computational time in favor of our package is huge. For example, in the case $T = 500$ and $n = 16$, `dma()` takes 37.5 minutes while `DMA()` only 1.8. It is also worth stressing that, the benefit of using **eDMA** does not only concern the possibility of running moderately large applications in a reasonable time using a commercial hardware, but also enables practitioners to run application with a large number of exogenous variables. To give an idea of the computational time a **eDMA** user faces, we report a second simulation study. We simulate from a DLM with $T = \{100, 200, ..., 900, 1000\}$, $n = \{2, 3, .., 18\}$ and run `DMA()` using a grid of values for δ between 0.9 and 1.0 with different spaces d, namely

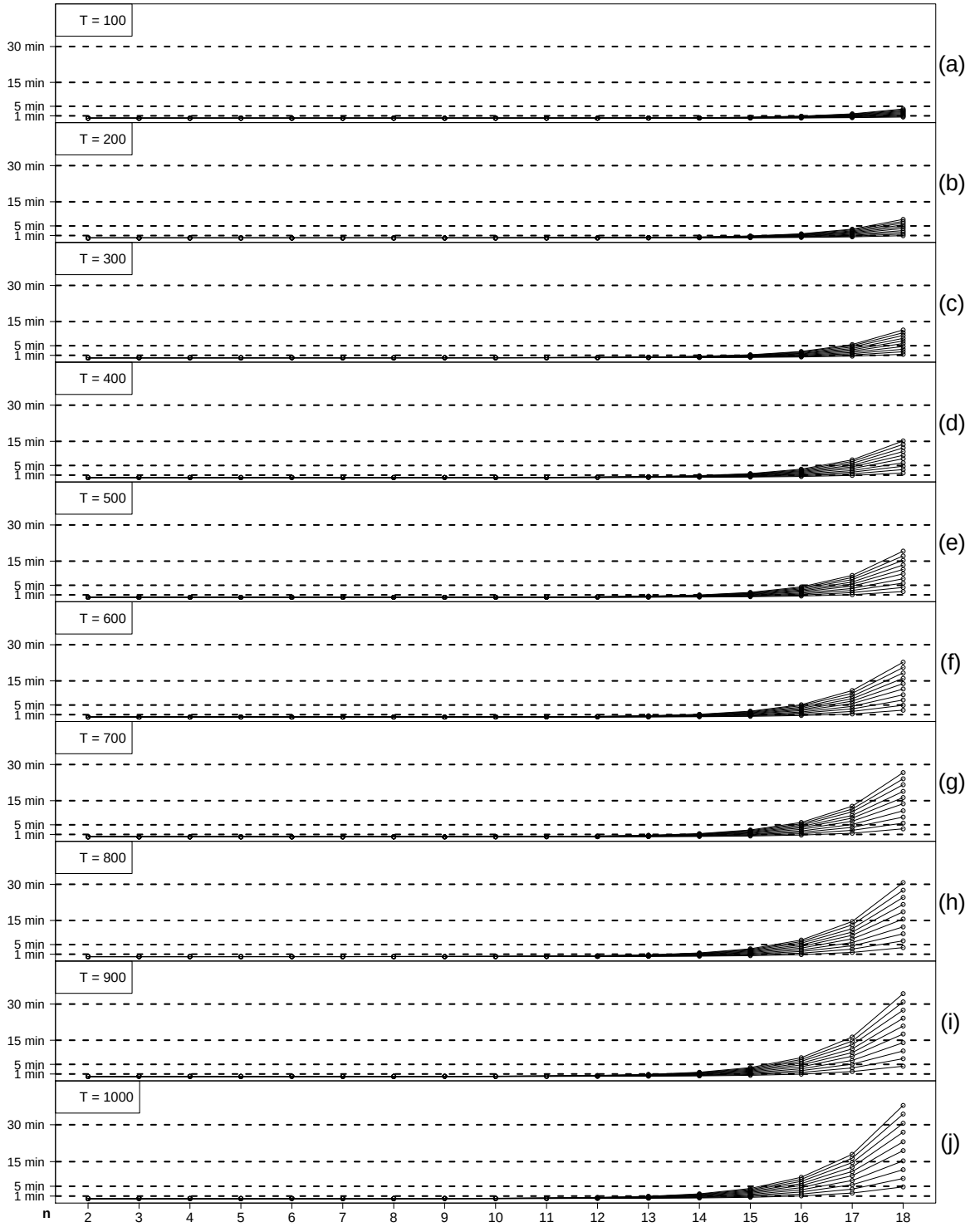

Figure 2: Computational time for `DMA()` using simulated data. Each panel represents computation time in minutes for `DMA()` using different sample sizes, T, number of predictors, n, and values of d, the number of points in the grid of δ. The values for d range between 2 and 10, the line at the bottom of each subfigure is for $d = 2$, the one immediately above is $d = 3$ and so on until the last which is for $d = 10$. Computations are performed on a standard Intel Core i7–4790 processor with 8 threads and 8 GB of RAM with Ubuntu 12.04 server edition.

T/n	4	6	8	10	12	14	16
100	10.9	92.6	133.2	88.4	69.4	70.5	58.4
500	37.5	29.4	31.5	30.7	25.7	26.5	25.4
1000	13.0	15.0	13.8	12.9	12.7	13.5	13.8

Table 1: Ratio of computation time between the `dma()` function from the **dma** package of McCormick *et al.* (2016) and the `DMA()` function of the **eDMA** package for different values of T and n.

$d = \{2, 3, \ldots, 10\}$. Figure 2 displays the computational time in minutes for all the combination of T, n, d. The lines reported in each subfigure represent the computational time for a specific choice of d. The line at the bottom of each subfigure is for $d = 2$,[8] the one immediately above is for $d = 3$ and so on until $d = 10$. From the Figure, we can see that, when $T \leq 400$, even for $n = 18$ and $d = 10$, the computational time is less then 15 minutes. Such sample sizes are relatively common in economic applications. When T increases, computational time increases linearly. For example, when $T = 800, n = 18$ and $d = 10$, computational time is 30 minutes, which is the double of the same case with $T = 400$.

The other relevant problem with DMA is the RAM usage. Specifically, if we want to store the quantities defined in Equations 1 and 4, we need to define two arrays of dimension $T \times d \times (2^n - 1)$. These kind of objects are not present in the **eDMA** package since we rely on the markovian nature of the model clearly evident from Equation 1. In this respect, we keep track of the quantities coming from Equation 4 and $p\left(y_t | M_i, \delta_j, \mathcal{F}_{t-1}\right)$ only for two consecutive periods during the loop over T. RAM usage is still efficiently performed in the **eDMA** package. Indeed, the computer where we run all our simulations has only 8GB of RAM.

6. A DMA example: Inflation data

This section shows how to run DMA using the R (R Core Team 2016) package **DMA** (Catania and Nonejad 2016). We use a time–series of quarterly U.S. inflation rate for illustration, show how to obtain posterior output and briefly interpret results. The example can be thought of as a typical assignment for a researcher at a central bank who is interested in forecasting inflation several–quarters ahead and understand the relationship between inflation and business cycles.

6.1. Data

In this analysis, we use the data of Groen *et al.* (2013).[9] Particularly, as a measure of inflation, y_t, we consider quarterly log changes in the Gross Domestic Product implicit price deflator (GDPDEF) observed at a quarterly frequency ranging from 1960q1 to 2011q2. The number of exogenous predictors are fifteen. This number is in accordance with typical "real–world" application, see Dangl and Halling (2012), Koop and Korobilis (2012) and Groen *et al.* (2013).

The predictors are: Real GDP in volume terms (ROUTP), real durable personal consumption expenditures in volume terms (RCONS), real residential investment in volume terms (RINVR), the import deflator (PIMP), the unemployment ratio (UNEMP), non–farm pay-

[8] In this case δ can takes values $\delta = 0.9$ and $\delta = 1.0$.

[9] The data is downloadable from http://www.tandfonline.com/doi/suppl/10.1080/07350015.2012.727718.

rolls data on employment (NFPR), housing starts (HSTS), the real spot price of oil (OIL), the real food commodities price index (FOOD) the real raw material commodities price index (RAW), and the M2 monetary aggregate (M2), which can reflect information on the current stance of monetary policy and liquidity in the economy as well as spending in households. In addition, we also use data on the term structure of interest rates approximated by means of: The level factor (YL), the slope factor (TS) and curvature factor (CS). Finally, we proxy inflation expectations through the one–year ahead inflation expectations that come from the Reuters/Michigan Survey of Consumers (MS). We include the data in the **eDMA** package as a `xts` object of dimension 206×16 named `USData`.

For most series, we use the percentage change of the original series in order to remove possible stochastic and deterministic trends. Exceptions are HSTS, for which we use the logarithm of the respective levels, as well as UNEMP, YL,TS, CS and MS, where we use the "raw" levels, see also Groen *et al.* (2013) for more details. Finally, since inflation very persistence, besides these 15 predictors, we follow Groen *et al.* (2013) and also include four inflation lags, $y_{t-1}, ..., y_{t-4}$, as predictors. In **eDMA**, we implement the function, `Lag()`, which allows us to lag variables delivered in the form of vector or matrices. For instance, to lag the `numeric` vector `GDPDEF` of length T by one period, we simply run

```
R> Lag(GDPDEF, 1)
```

which returns a `numeric` vector of length T containing the lagged inflation. Values that are not available (*i.e.*, the value y_0 in this example) are replaced by `NA`.

6.2. Model estimation

We have a total of $2^{19} = 524288$ model combinations.[10] Furthermore, we let $\delta = \{0.9, 0.91, ..., 1\}$ such that we have a total of $\left(2^{19}\right) \cdot 11 = 5767168$ combinations. We set $\alpha = 0.99$ and specify a noninformative prior over the model combinations, $p\left(M_s \mid \mathcal{F}_0\right) = 1/\left(d \cdot k\right), \quad s = 1, \ldots, d \cdot k$, such that initially, all models are equally likely. We then update these model probabilities as new information arrives. As previously mentioned, we include the constant in all models, see also Groen *et al.* (2013).

In terms of implementation, we start by loading the **eDMA** package and the data set by typing:

```
R> library("eDMA")
R> data("USData")
```

In order to perform DMA using the `DMA()` function, we write:

```
R> Fit <- DMA(GDPDEF ~  Lag(GDPDEF, 1) + Lag(GDPDEF, 2) +
                        Lag(GDPDEF, 3) + Lag(GDPDEF, 4) +
                        Lag(ROUTP, 1)  + Lag(RCONS, 1)  +
                        Lag(RINVR, 1)  + Lag(PIMP, 1)   +
                        Lag(UNEMP, 1)  + Lag(NFPR, 1)   +
```

[10] The model which include only the constant is also considered. Indeed, when `vKeep = NULL`, the number of models is $2^n - 1$, however, when `vKeep != NULL`, the number of models is 2^b, where `b = n - length(vKeep)`.

```
    Lag(HSTS, 1)   + Lag(M2, 1)     +
    Lag(OIL, 1)    + Lag(RAW, 1)    +
    Lag(FOOD, 1)   + Lag(YL, 1)     +
    Lag(TS, 1)     + Lag(CS, 1)     +
    Lag(MS, 1), data = USData,
    vDelta = seq(0.90, 1.00, 0.01), vKeep = 1)
```

In terms of implementation, we suggest using the non–informative prior, `bZellnerPrior = FALSE`, which is the default, see Section 6.6 for a prior sensitivity analysis. More details on the model can be made available by typing `Fit`:

```
R> Fit

-----------------------------------------
-        Dynamic Model Ageraging         -
-----------------------------------------

Model Specification
T     = 202
n     = 20
d     = 11
Alpha = 0.99
Model combinations = 524288
Model combinations including averaging over delta = 5767168
-----------------------------------------
Prior : Multivariate Gaussian with mean vector 0
        and covariance matrix equal to: 100 x diag(20)

Variables always included : (Intercept)
-----------------------------------------
The grid for delta:

Delta =  0.90, 0.91, 0.92, 0.93, 0.94, 0.95,
         0.96, 0.97, 0.98, 0.99, 1.00
-----------------------------------------

Elapsed time : 1686.14 secs
```

As it can be seen, the total estimation time of our DMA is 1686.14 seconds corresponding to around 28 minutes on an Intel Core i7-3630QM processor. A complete summary of the estimation is available as:

```
R> summary(Fit, iBurnPeriod = 32)

Call:
 DMA(formula =  Lag(GDPDEF, 1) + Lag(GDPDEF, 2) +
```

```
                Lag(GDPDEF, 3) + Lag(GDPDEF, 4) +
                Lag(ROUTP, 1)  + Lag(RCONS, 1)  +
                Lag(RINVR, 1)  + Lag(PIMP, 1)   +
                Lag(UNEMP, 1)  + Lag(NFPR, 1)   +
                Lag(HSTS, 1)   + Lag(M2, 1)     +
                Lag(OIL, 1)    + Lag(RAW, 1)    +
                Lag(FOOD, 1)   + Lag(YL, 1)     +
                Lag(TS, 1)     + Lag(CS, 1)     +
                Lag(MS, 1) )

Residuals:
    Min      1Q  Median      3Q     Max
-1.3494 -0.2990 -0.0138  0.2218  1.5653

Coefficients:
               E[theta_t] SD[theta_t] E[P(theta_t)] SD[P(theta_t)]
(Intercept)          0.11        0.16          1.00           0.00
Lag(GDPDEF, 1)       0.41        0.18          0.83           0.30
Lag(GDPDEF, 2)       0.02        0.02          0.19           0.11
Lag(GDPDEF, 3)       0.09        0.06          0.38           0.22
Lag(GDPDEF, 4)       0.12        0.06          0.52           0.23
Lag(ROUTP, 1)        0.00        0.01          0.15           0.09
Lag(RCONS, 1)        0.00        0.00          0.14           0.07
Lag(RINVR, 1)        0.02        0.02          0.21           0.11
Lag(PIMP, 1)         0.21        0.09          0.83           0.27
Lag(UNEMP, 1)       -0.03        0.06          0.22           0.12
Lag(NFPR, 1)         0.02        0.01          0.22           0.14
Lag(HSTS, 1)         0.02        0.03          0.20           0.09
Lag(M2, 1)           0.01        0.01          0.16           0.06
Lag(OIL, 1)         -0.03        0.09          0.36           0.22
Lag(RAW, 1)          0.00        0.01          0.16           0.07
Lag(FOOD, 1)         0.01        0.01          0.20           0.12
Lag(YL, 1)           0.27        0.44          0.36           0.32
Lag(TS, 1)           0.01        0.04          0.15           0.07
Lag(CS, 1)          -0.03        0.07          0.19           0.12
Lag(MS, 1)           0.02        0.03          0.18           0.09

Variance contribution (in percentage points):
  vobs vcoeff   vmod   vtvp
 43.59  11.71  12.84  31.86

Top 10% included predictors:        (Intercept), Lag(PIMP, 1)

Forecast Performance:
                         DMA      DMS
MSE                    0.235    0.251
MAD                    0.361    0.369
```

```
Predictive Likehood -106.352 -122.943
```

Below, we go into more details with regards to how to use the output from the estimation procedure to interpret our results.

6.3. Using the output from eDMA

The output can be divided into two main parts: (a): Full–sample analysis, (b): Out–of–sample analysis. With regards to (a), the most interesting quantities are: `mincpmt`, `vsize`, `mtheta`, `vdeltahat`, and `mvdec`, see Section 4.

For instance, the inclusion probabilities of the predictors for the last part of the sample can be printed by:

```
R> InclusionProb <- inclusion.prob(Fit, iBurnPeriod = 32)
R> tail(round(InclusionProb[, 1:4], 2))
```

```
           (Intercept) Lag(GDPDEF, 1) Lag(GDPDEF, 2) Lag(GDPDEF, 3)
2010-01-01           1           0.99           0.43           0.71
2010-04-01           1           1.00           0.44           0.72
2010-07-01           1           1.00           0.45           0.73
2010-10-01           1           1.00           0.45           0.73
2011-01-01           1           1.00           0.45           0.73
2011-04-01           1           1.00           0.45           0.73
```

The above matrix shows the inclusion probabilities of: The constant and $y_{t-1}, ..., y_{t-3}$, from $2010q1$ to $2011q2$. Notice that, the inclusion probabilities of the constant term, `(Intercept)`, are always equal to 1 as every model contains this term (since we set `vKeep = 1`), see (iii) in page 7 of this paper.

In Figure 3, we report the inclusion probabilities for predictors that are important at least one point in time. To be precise, any predictor where the inclusion probability is never above 0.2 is excluded. In these plots, we also make evident NBER recorded recessions (shaded gray bars). Overall, we observe a good amount of time–variation these plots. The lags of inflation, except for y_{t-2} all seem important. The import deflator (PIMP) also receives high posterior probability throughout the sample. Inflation expectation (MS) and M2 receive higher probabilities towards the end of the sample. However, the inclusion probability of these predictors at any point in time is relatively low. Real spot price of oil (OIL) receives high inclusion probability during the post Great Moderation era, whereas we observe the opposite trend for YL. In addition to the inclusion probabilities, we also report filtered estimates of the regression coefficients for these predictors in Figure 4. These quantities are extracted from `Fit` simply using

```
R> mTheta <- coef(Fit, iBurnPeriod = 32)
```

Besides these variables, the output from DMA can be used to analyze:

The magnitude of time–variation in the model parameters, `"vdeltahat"`, which is the posterior weighted average of δ at each point in time. We report this estimate in panel (a) of Figure 5. The analogous plot in R can be obtained using:

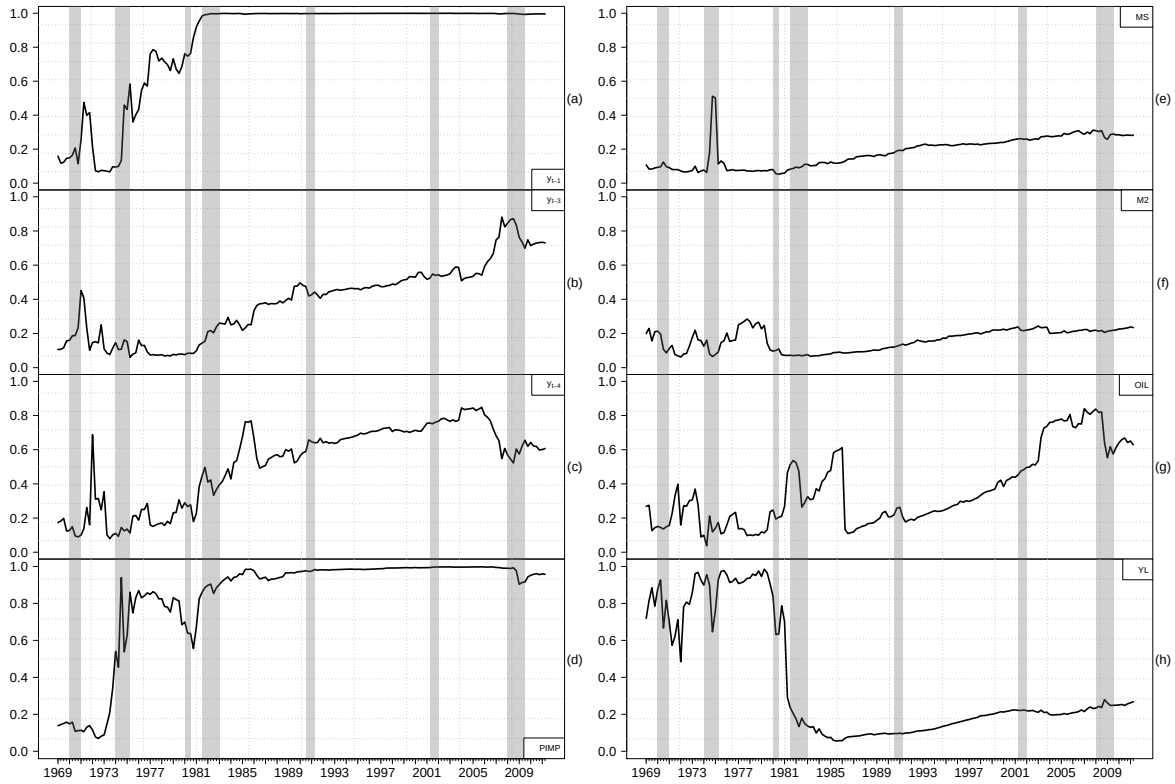

Figure 3: Posterior inclusion probabilities for the most important predictors of DMA. Panels (a), (b) and (c): First, third and fourth lags of inflation. Panel (d): Import deflator (PIMP). Panel (e): Inflation expectations (MS). Panel (f): M2 monetary aggregate (M2). Panel (g): Real spot price of oil (OIL). Panel (h): Level factor of the term structure (YL). We refer the reader to Groen *et al.* (2013) for more details regarding the variables. The gray vertical bars indicate business cycle peaks, *i.e.*, the point at which an economic expansion transitions to a recession, based on National Bureau of Economic Research (NBER) business cycle dating.

```
R> plot(Fit, which = "vdeltahat", iBurnPeriod = 32)
```

Except for the recessions in 1990 and 2001, there is a very intuitive relationship between δ and recession periods. Typically, δ falls during recessions, which fares well with the notion that $\boldsymbol{\theta}_t$ changes rapidly as the model needs to adopt to the changes in the data. Conversely, δ remains high and close to 1 during the Great Moderation. We can also use `inclusion.prob()` to extract the posterior probability of each value of δ. We can print these quantities for the final part of the sample using:

```
R> InclusionProbDelta <- inclusion.prob(Fit, iBurnPeriod = 32)
R> round(tail(InclusionProbDelta), 2)
```

```
           0.9 0.91 0.92 0.93 0.94 0.95 0.96 0.97 0.98 0.99    1
2010-01-01   0    0    0    0 0.01 0.02 0.06 0.16 0.30 0.28 0.17
2010-04-01   0    0    0    0 0.01 0.02 0.06 0.15 0.29 0.29 0.17
2010-07-01   0    0    0    0 0.01 0.02 0.05 0.15 0.30 0.29 0.18
```

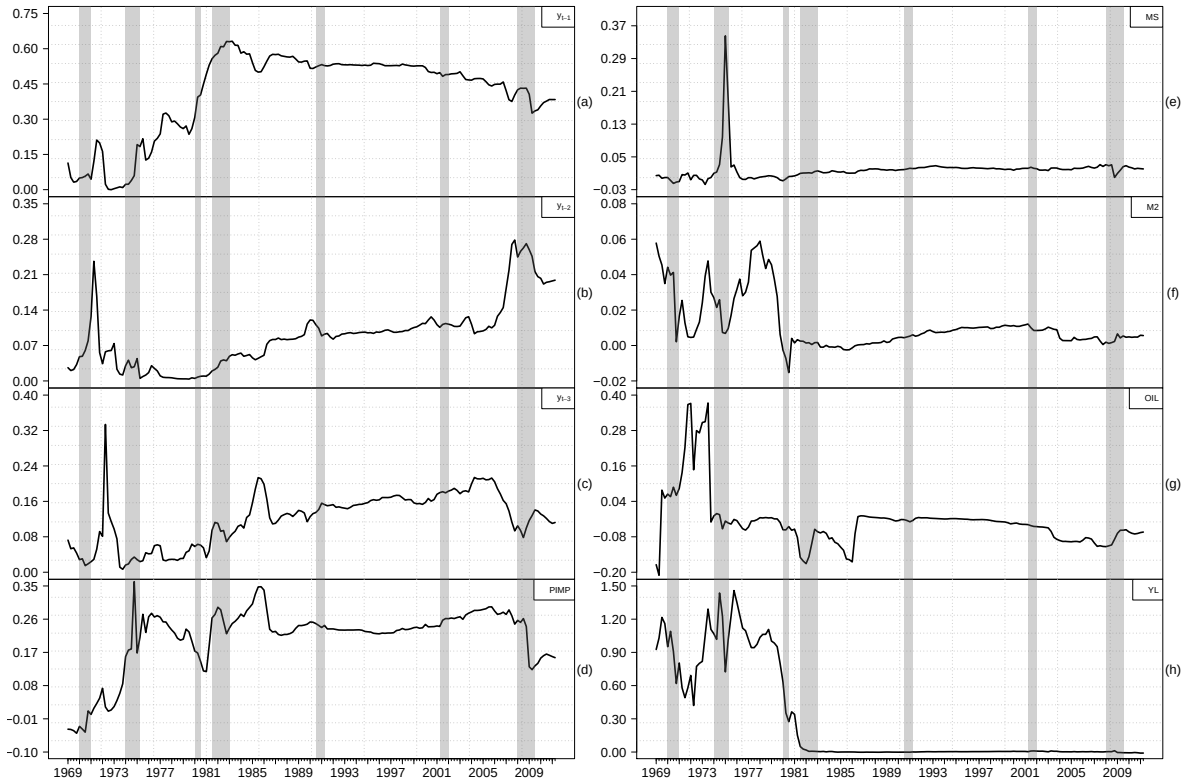

Figure 4: Filtered estimates of the regression coefficients for the most important predictors of DMA. Panels (a), (b) and (c): First, third and fourth lags of inflation. Panel (d): Import deflator (PIMP). Panel (e): Inflation expectations (MS). Panel (f): M2 monetary aggregate (M2). Panel (g): Real spot price of oil (OIL). Panel (h): Level factor for the terms structure (YL). We refer the reader to Groen *et al.* (2013) for more details regarding the variables. The gray vertical bars indicate business cycle peaks, *i.e.*, the point at which an economic expansion transitions to a recession, based on National Bureau of Economic Research (NBER) business cycle dating.

```
2010-10-01    0    0    0    0 0.01 0.02 0.06 0.16 0.30 0.29 0.16
2011-01-01    0    0    0    0 0.01 0.02 0.06 0.16 0.30 0.29 0.16
2011-04-01    0    0    0    0 0.01 0.02 0.06 0.17 0.32 0.28 0.14
```

where the column names are the values of δ.

In panel (b), we report the number of predictors contained in the model with the highest posterior probability , $p\left(M_i|\mathcal{F}_t\right)$,at each point in time. This can be achieved by:

```
R> plot(Fit, which = "vsize_DMS", iBurnPeriod = 32)
```

Alternatively, we can also plot the expected number of predictors at each point in time replacing `which = "vsize_DMS"` by `which = "vsize"`. A very interesting result from panel (b) is that, although we have 19 predictors, at each point in time the best model contains only a few predictors. Furthermore, there is also evidence of time–variation in the number of predictors as we have as many as eight and as few as two predictors over the sample.

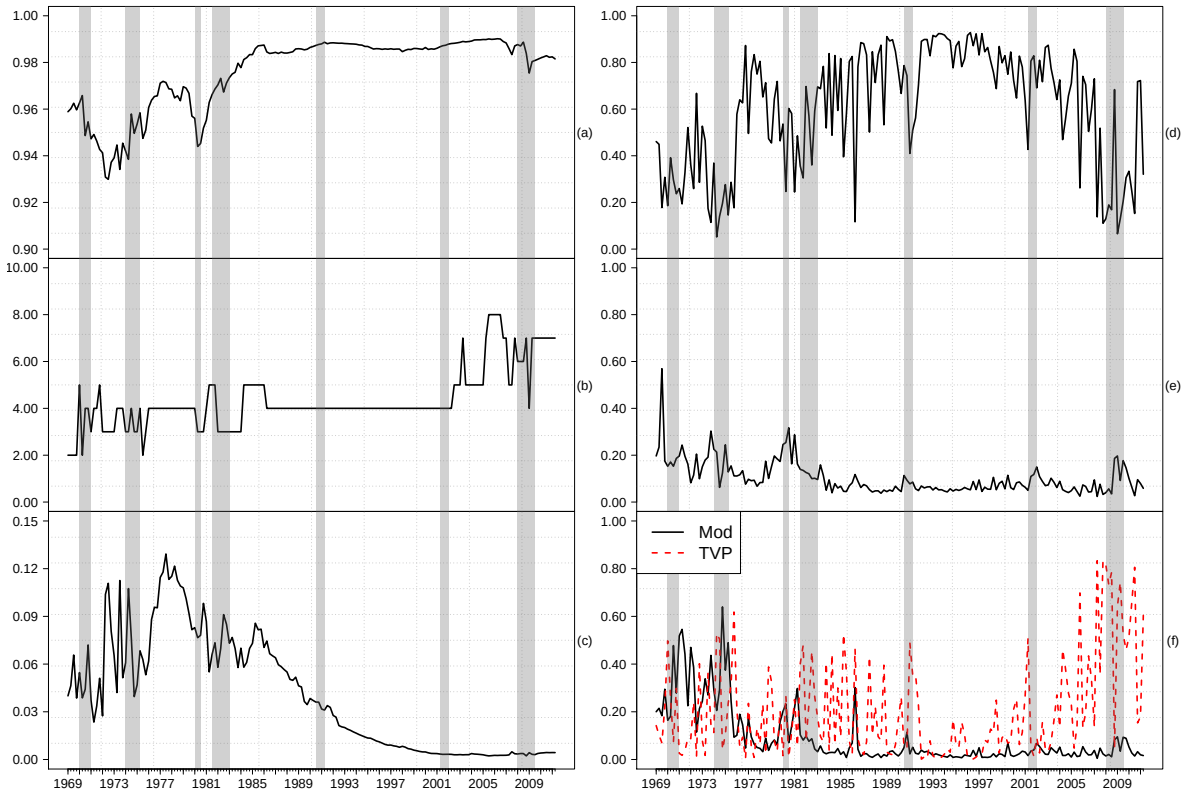

Figure 5: Posterior output for DMA. Panel (a): Posterior weighted average estimate of δ. Panel (b): Number of predictors for the model with the highest posterior probability. Panel (c): Sum of top 10% inclusion probabilities. Panel (d): Observational variance. Panel (e): Variance due to errors in the estimation of the coefficients. Panel (f) Variance due to model uncertainty (Mod, *solid*) and variance due to uncertainty with respect to the choice of the degrees of time-variation in the regression coefficients (TVP, *red-dotted*). The gray vertical bars indicate business cycle peaks, *i.e.*, the point at which an economic expansion transitions to a recession, based on National Bureau of Economic Research (NBER) business cycle dating.

We can also use posterior model probabilities to obtain an idea of how important is model averaging. In panel (c), we report the sum of the posterior inclusion probabilities for the 10% of models (`which = "vhighmpTop01_DMS"`). If this number is high, then it means that relatively few models dominate, obtaining relatively high posterior probabilities. Conversely, if this number is low, then no individual (or group of) models receive high probabilities, which provides evidence in favor of averaging over predictors instead of selecting.

Finally, in panels (d), (e) and (f), we report the variance decomposition analysis (`which = "mvdec"`). Evidently, the dominant source of uncertainty is the observational variance. This is not surprising as random fluctuation are expected to dominate uncertainty. However, uncertainty about the correct magnitude of time–variation in $\boldsymbol{\theta}_t$ is very high during recessions such as the Great recession of 2008, see the red–dashed line in panel (f). We believe that practitioners can build on the variance decomposition results to better understand the sources of variation in inflation. Indeed, as our preliminary results indicate, model uncertainty and estimation errors in the coefficients play a minor role compared to uncertainty due to time–

Model	Description
$\mathcal{M}_0$	Plain AR(4) model: The constant term and $y_{t-1}, \ldots, y_{t-4}$ are always included. We set $\alpha = 1$ and $\delta = 1$.
$\mathcal{M}_1$	Time-varying AR(4) model: The constant term and $y_{t-1}, ..., y_{t-4}$ are always included. We set $\alpha = 0.99$ and average over $\delta_1, \ldots, \delta_d$.
$\mathcal{M}_2$	DMA using $y_{t-1}, \ldots, y_{t-4}$: The constant term is always included. We set $\alpha = 0.99$ and average over the combinations of $y_{t-1}, ..., y_{t-4}$ and $\delta_1, \ldots, \delta_d$.
$\mathcal{M}_3$	DMA using $y_{t-1}, \ldots, y_{t-4}$ and the exogenous predictors: The constant term is always included. We set $\alpha = 0.99$ and average over the combinations of predictors as well as $\delta_1, \ldots, \delta_d$.
$\mathcal{M}_4$	DMS using $y_{t-1}, \ldots, y_{t-4}$ and the exogenous predictors: The constant term is always included. We set $\alpha = 0.99$ and select the model with the highest posterior probability at each t and use it to forecasts.
$\mathcal{M}_5$	BMA: DMA with $\alpha = 1$ and $\delta = 1$.
$\mathcal{M}_6$	BMS: DMS with $\alpha = 1$ and $\delta = 1$.
$\mathcal{M}_7$	Kitchen Sink: The constant term, $y_{t-1}, \ldots, y_{t-4}$ and all exogenous predictors are always included. We set $\alpha = 0.99$ and average only over $\delta_1, \ldots, \delta_d$.

Table 2: Model specifications. The first column is the model index. The second column provides a brief description of each individual model.

variation in $\boldsymbol{\theta}_t$.

6.4. Forecasts

A very important feature of DMA is out–of–sample forecasting, see Koop and Korobilis (2011) and Koop and Korobilis (2012). In this section, we illustrate how our package can be used to perform forecasting.

In Table 2, we provide an overview of several alternative models. Notice that, all models can be estimate using our package. For instance, the plain AR(4) model, ($\mathcal{M}_0$), can be estimated by setting $\delta = 1.0$, $\alpha = 1.0$, using the code:

```
R> Fit_M0 <- DMA(GDPDEF ~  Lag(GDPDEF, 1) + Lag(GDPDEF, 2) +
                           Lag(GDPDEF, 3) + Lag(GDPDEF, 4),
                           data = USData, vDelta = 1.00,
                           dAlpha = 1.00, vKeep = c(1, 2, 3, 4, 5))
```

The same holds for Bayesian Model Averaging (BMA, $\mathcal{M}_5$) and Bayesian Model Selection (BMS, $\mathcal{M}_6$) by setting $\delta = 1.0$, $\alpha = 1.0$. Thus, **eDMA** also relates to the **BMS** package of Zeugner and Feldkircher (2015).

We use the models to obtain one and five quarter ahead forecasts through direct forecasting, see Marcellino *et al.* (2006).

Model	$h = 1$		$h = 5$	
	MSE	PLD	MSE	PLD
$\mathcal{M}_1$	1.020	0.244	0.835	20.522
$\mathcal{M}_2$	0.982	1.544	0.694	37.192
$\mathcal{M}_3$	0.969	11.700	0.645	73.819
$\mathcal{M}_4$	1.038	-4.890	0.780	26.996
$\mathcal{M}_5$	0.976	6.809	1.127	9.234
$\mathcal{M}_6$	1.187	-18.358	1.310	-33.352
$\mathcal{M}_7$	1.762	-7.125	1.242	32.293

Table 3: Mean Squared Error (MSE) and Predictive Likelihood Difference (PLD) of $\mathcal{M}_i$, $i = 1, \ldots, 7$ compared to $\mathcal{M}_0$ for $h = 1$ and $h = 5$ quarters ahead out–of–sample forecasts.

Table 3 reports the Mean Squared Error (MSE) and the Predictive Likelihood Difference (PLD) of $\mathcal{M}_i$, $i = 1, \ldots, 7$, over $\mathcal{M}_0$ (the benchmark) at $h = 1$ and $h = 5$. Compared to the benchmark, $\mathcal{M}_1$ provides relatively small improvements at $h = 1$, whereas at $h = 5$, we obtain reductions in MSE by about 17% over the benchmark. The improvement in PLD at $h = 5$ is also notable. By averaging over $y_{t-1}, ...y_{t-4}$, we obtain more gains, which indicates that allowing for time–variation in the number of predictors is important. DMA using lags of inflation as well as 15 additional predictors is the top performer, regardless of h. Conversely, the improvements of DMS over $\mathcal{M}_0$ are much more modest. This result is understandable as panel (c) in Figure 5 demonstrates that there is no individual model that performs overwhelmingly better than other specifications. As a result, DMS is outperformed by DMA.

As previously mentioned, DMA (DMS) with $\alpha = \delta = 1$ correspond to BMA (BMS). Overall, compared to the benchmark model, BMA provides some improvements in density forecasts, whereas we do not observe any improvements in terms of point forecasts.

Finally, as an alternative to these models, we can consider the Kitchen Sink model (the model with all predictors, $\mathcal{M}_7$) where we only average over δ. Compared to $\mathcal{M}_0$, the kitchen sink model does not provide any improvements at $h = 1$. At $h = 5$, we observe improvements in density forecasts. However, the kitchen sink model is outperformed by DMA.

6.5. Why does DMA performs so well ?

To investigate how quickly our techniques adapt to changes in data, we report the accumulated log–PLD for several models over the benchmark in panels (a)–(d) of Figure 6. These can be obtained using the `pred.like()` method available for `DMA` objects. For instance, we create the two vectors `vPL_M0` and `vPL_DMA` containing the log–Predictive Likelihood of $\mathcal{M}_0$ and $\mathcal{M}_3$ using:

```
R> vPL_M0 <- pred.like(Fit_M0, iBurnPeriod = 32)
R> vPL_M3 <- pred.like(Fit, iBurnPeriod = 32)
```

and compute the accumulated log-PLD of $\mathcal{M}_3$ over $\mathcal{M}_0$ as:

```
R> vPLD_M3.M0  <- vPL_M3 - vPL_M0
```

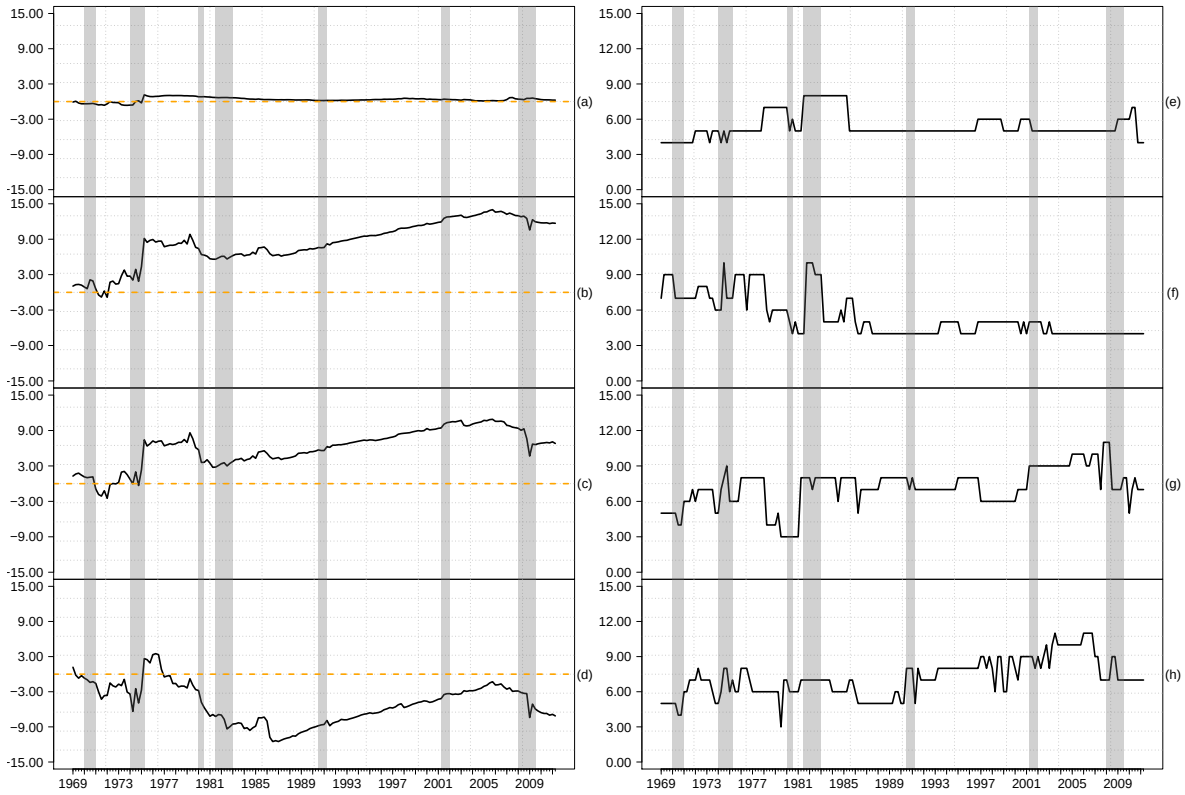

Figure 6: Accumulated PDL and the optimal number of predictors for prior sensitivity analysis. Panel (a): $\mathcal{M}_1$ over $\mathcal{M}_0$, Panel (b): $\mathcal{M}_3$ over $\mathcal{M}_0$, Panel (c): $\mathcal{M}_5$ over $\mathcal{M}_0$, Panel (d): $\mathcal{M}_7$ over $\mathcal{M}_0$. Panels (e)–(h): Number of predictors for the model with the highest posterior probability using Zellenr's prior with $g = 0.5, 20, 100, T$. The gray vertical bars indicate business cycle peaks, *i.e.*, the point at which an economic expansion transitions to a recession, based on National Bureau of Economic Research (NBER) business cycle dating.

which is reported in panel (b) of Figure 6.

In panels (a), (b), (c) and (d) of Figure 6 a value of zero corresponds to equal support of both models, positive values are in support of the model of choice over $\mathcal{M}_0$ and negative values show support of $\mathcal{M}_0$ over the model of choice at time t. In these panels, we decompose the effects of (i): Allowing for time–variation in the regression coefficients, (ii): Allowing for model uncertainty but no time–variation in the regression coefficients and (iii): Allowing for time–variation in the regression coefficients and model uncertainty.

In panel (a), we see that the time–varying AR(4) ($\mathcal{M}_1$) model outperforms the benchmark during the recession in the early 1970s. On the other hand, both models provide basically same results. The Great Recession is associated with a small improvements in favor of $\mathcal{M}_1$. Compared to the plain AR(4) model, it takes about 30 observations to provide compelling evidence for DMA. These improvements increase till the end of the sample. Furthermore, we also see that DMA improves model performance in recession as well as expansion periods. Compared to BMA, the improvements of DMA are mostly concentrated around recession periods, which indicates the importance of allowing for time–variation of $\boldsymbol{\theta}_t$ in periods of economic turmoil.

Prior	MSE	PLD
$g = 0.1$	4.875	-175.145
$g = 20$	1.255	-27.141
$g = 100$	1.072	-0.573
$g = T$	1.014	6.920

Table 4: Mean Squared Error (MSE) and Predictive Likelihood Difference (PLD) of DMA using the following values of g: $0.1, 20, 100, T$ and $\mathcal{M}_0$ for $h = 1$.

6.6. Prior sensitivity analysis

In this section, we evaluate the robustness of our results with regards to Zellner's prior shrinkage parameter, g, see (4) and (5) in Dangl and Halling (2012). Intuitively, a smaller value of g means more shrinkage around the prior mean of $\boldsymbol{\theta}_0$, *i.e.*, $\mathbf{0}$. Furthermore, if $g = 0$, then $p(M_i \mid \mathcal{F}_t)$ is equal for all models. The larger is g, the more we are willing to move away from the model priors in response to what we observe in the data. Thus, the choice of g can have important implications with respect to out–of–sample forecasting results.

In order to shed light on this issue, we re–estimate DMA with g equals to 0.1, 20, 100 and T (using `bZellnerPrior = TRUE`) and see to which extent it influences out–of–sample results, see Table 4 and Figure 6. Compared to our default priors, we see that very informative values such as $g = 0.1$ and 20 worsen results considerably as DMA tends to favor models with higher number of predictors, which obviously does not improve forecast performance, see panels (e) and (f) of Figure 6. As we increase g, we obtain closer results to the default prior. Therefore, we generally recommend practitioners to use the default prior.

7. Conclusion

In this paper, we present the **eDMA** package for R. The purpose of **eDMA** is to offer an integrated environment to perform DMA using the available `DMA()` function, which enables practitioners to perform DMA exploiting multiple processors without major efforts. Furthermore, R users will find common methods to represent and extract estimated quantities such as `plot()`, `as.data.frame()`, `coef()` and `residuals()`.

Overall, **eDMA** is able to: (i): Incorporate the extensions introduced in Dangl and Halling (2012), which is particularly relevant for economic and financial applications, (ii): Compared to other approaches, our package is much faster, (iii): It requires a smaller amount of RAM even in cases of moderately large applications, and (iv): It allows for parallel computing.

In Section 5, we also detail the expected time the program takes to perform DMA under different sample sizes, number of predictors and number of grid points. For typical economic applications, estimation time is around 30 min using a commercial laptop. Very large applications can still benefit from the use of **eDMA** when performed on desktop or even clusters, without additional effort from the user.

Computational details

The results in this paper were obtained using R 3.2.3 (R Core Team 2016) with the pack-

ages: **eDMA** version 1.0 (Catania and Nonejad 2016), **Rcpp** version 0.12.5 (Eddelbuettel and François 2011; Eddelbuettel *et al.* 2016a), **RcppArmadillo** version 0.7.100.3.1 (Eddelbuettel and Sanderson 2014; Eddelbuettel *et al.* 2016b), **xts** version 0.9-7 (Ryan and Ulrich 2015) and **devtools** version 1.1.1 (Wickham and Chang 2016). R itself and all packages used are available from CRAN at `http://CRAN.R-project.org/`. The package **eDMA** is available from GitHub at `https://github.com/LeopoldoCatania/eDMA`. Computations were performed on a Genuine Intel® quad core CPU i7–3630QM 2.40Ghz processor.

References

Byrne JP, Korobilis D, Ribeiro PJ (2014). "On the Sources of Uncertainty in Exchange Rate Predictability." *Available at SSRN 2502586*. URL http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2502586.

Catania L, Nonejad N (2016). ***eDMA**: Efficient Dynamic Model Averaging*. R package version 1.0.0, URL https://github.com/LeopoldoCatania/eDMA.

Chapman B, Jost G, Van Der Pas R (2008). *Using OpenMP: Portable Shared Memory Parallel Programming*, volume 10. MIT press, Cambridge, US.

Croissant Y, Millo G (2008). "Panel Data Econometrics in R: The **plm** Package." *Journal of Statistical Software*, **27**(2). URL http://www.jstatsoft.org/v27/i02/.

Dangl T, Halling M (2012). "Predictive Regressions with Time–Varying Coefficients." *Journal of Financial Economics*, **106**(1), 157–181. doi:doi:10.1016/j.jfineco.2012.04.003.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Eddelbuettel D, François R, Allaire J, Ushey K, Kou Q, Bates D, Chambers J (2016a). ***Rcpp**: Seamless R and C++ Integration*. R package version 0.12.5, URL https://cran.r-project.org/package=Rcpp.

Eddelbuettel D, François R, Bates D (2016b). ***RcppArmadillo**: **Rcpp** Integration for the **Armadillo** Templated Linear Algebra Library*. R package version 0.7.100.3.1, URL https://cran.r-project.org/package=RcppArmadillo.

Eddelbuettel D, Sanderson C (2014). "**RcppArmadillo**: Accelerating R with High–Performance C++ Linear Algebra." *Computational Statistics & Data Analysis*, **71**, 1054–1063. doi:10.1016/j.csda.2013.02.005.

Groen JJ, Paap R, Ravazzolo F (2013). "Real–Time Inflation Forecasting in a Changing World." *Journal of Business & Economic Statistics*, **31**(1), 29–44. doi:10.1080/07350015.2012.727718.

Koop G, Korobilis D (2011). "UK Macroeconomic Forecasting with Many Predictors: Which Models Forecast Best and When Do They Do So?" *Economic Modelling*, **28**(5), 2307–2318. doi:10.1016/j.econmod.2011.04.008.

Koop G, Korobilis D (2012). "Forecasting Inflation Using Dynamic Model Averaging." *International Economic Review*, **53**(3), 867–886. doi:10.1111/j.1468-2354.2012.00704.x.

Marcellino M, Stock JH, Watson MW (2006). "A comparison of direct and iterated multistep AR methods for forecasting macroeconomic time series." *Journal of Econometrics*, **135**(1-2), 499 – 526. ISSN 0304-4076. doi:10.1016/j.jeconom.2005.07.020. URL http://www.sciencedirect.com/science/article/pii/S030440760500165X.

McCormick TH, Raftery A, Madigan D (2016). ***dma**: Dynamic Model Averaging*. R package version 1.2-3, URL https://CRAN.R-project.org/package=dma.

Onorante L, Raftery AE (2016). "Dynamic Model Averaging in Large Model Spaces Using Dynamic Occam's Window." *European Economic Review*, **81**, 2–14. doi:10.1016/j.euroecorev.2015.07.013.

OpenMP A (2008). OpenMP *Application Program Interface, v. 3.0*. URL http://www.openmp.org/mp-documents/spec30.pdf.

Paye BS (2012). "'Déjà vol': Predictive Regressions for Aggregate Stock Market Volatility Using Macroeconomic Variables." *Journal of Financial Economics*, **106**(3), 527 – 546. ISSN 0304-405X. doi:10.1016/j.jfineco.2012.06.005. URL http://www.sciencedirect.com/science/article/pii/S0304405X12001316.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. R version 3.2.3, URL https://www.R-project.org/.

Raftery A, Hoeting J, Volinsky C, Painter I, Yeung KY (2015). ***BMA**: Bayesian Model Averaging*. R package version 3.18.6, URL https://CRAN.R-project.org/package=BMA.

Raftery AE, Kárnỳ M, Ettler P (2010). "Online Prediction Under Model Uncertainty via Dynamic Model Averaging: Application to a Cold Rolling Mill." *Technometrics*, **52**(1), 52–66. doi:10.1198/TECH.2009.08104.

Ryan JA, Ulrich JM (2015). ***xts**: Extensible Time Series*. R package version 0.9-7, URL https://CRAN.R-project.org/package=xts.

Sanderson C (2010). "**Armadillo**: An Open Source C++ Linear Algebra Library for Fast Prototyping and Computationally Intensive Experiments." *Technical report*, NICTA. URL http://arma.sourceforge.net/.

Stock JH, Watson MW (1999). "Forecasting Inflation." *Journal of Monetary Economics*, **44**(2), 293–335. doi:10.1016/S0304-3932(99)00027-6.

Stock JH, Watson MW (2008). "Phillips Curve Inflation Forecasts." *Technical report*, National Bureau of Economic Research. URL http://www.nber.org/papers/w14322.

West M, Harrison J (1999). *Bayesian Forecasting & Dynamic Models*. Springer–Verlag, Berlin.

Wickham H, Chang W (2016). ***devtools**: Tools to Make Developing R Packages Easier*. R package version 1.11.1, URL https://CRAN.R-project.org/package=devtools.

Zeugner S, Feldkircher M (2015). "Bayesian Model Averaging Employing Fixed and Flexible Priors: The BMS Package for R." *Journal of Statistical Software*, **68**(1), 1–37. doi:10.18637/jss.v068.i04.

Affiliation:

Leopoldo Catania
Department of Economics and Finance
Faculty of Economics
University of Rome, "Tor Vergata"
Via Columbia, 2
00133 Rome, Italy
E-mail: leopoldo.catania@uniroma2.it

Nima Nonejad
Department of Mathematical Sciences
Aalborg University and CREATES
Fredrik Bajers Vej 7G
9220, Aalborg East, Denmark
E-mail: nimanonejad@gmail.com