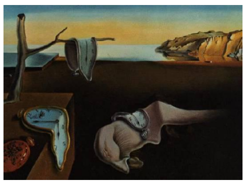

Journal of Statistical Software

April 2018, Volume 84, Issue 11. doi: 10.18637/jss.v084.i11

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

Leopoldo Catania
Aarhus University and CREATES

Nima Nonejad
Aalborg University and CREATES

Abstract

Raftery, Kárný, and Ettler (2010) introduce an estimation technique, which they refer to as dynamic model averaging (DMA). In their application, DMA is used to predict 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 useful in macroeconomic and financial applications. In this paper, we present the **eDMA** package for DMA estimation implemented in R. The **eDMA** package is especially suited for practitioners in economics and finance, where typically a large number of predictors are available. Our implementation is up to 133 times faster than a standard implementation using a single-core CPU. Thus, with the help of this package, practitioners are able to perform DMA on a standard PC without resorting to large computing 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 US 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 to 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 possible, practitioners often try to take advantage of the many 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, Paap, and Ravazzolo (2013) just to mention a few references. Recently, in the context of forecasting US and UK inflation, 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 purpose of DMA introduced in Raftery *et al.* (2010) is more oriented towards engineers. Particularly, their aim is to predict the output strip thickness for a cold rolling mill, where the output is measured with a time delay. DMA consists of many time-varying coefficient regression models formed from all possible combinations of the predictors available to a practitioner. Moreover, besides allowing for time-variation in the regression coefficients, interpreting inclusion probabilities of each individual model, DMA also allows the relevant model set to change with time as well through a forgetting factor. This way, past model performance receives relatively less weight than current model performance and the estimation procedure adapts better to the incoming data. Koop and Korobilis (2011) and Koop and Korobilis (2012) argue that by slightly adjusting the original framework of Raftery *et al.* (2010), DMA can be useful in economic applications, especially for inflation forecasting.[1] Dangl and Halling (2012) provide further suggestions on how to improve DMA such that it can better adapt to the patterns typically observed in economic and financial data. The aforementioned authors, also provide a useful variance decomposition scheme using the output from the estimation procedure. Byrne, Korobilis, and Ribeiro (2018), 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 this kind of data 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, from a practical point of view, designing an efficient DMA algorithm remains a challenging issue. As we demonstrate in Section 3, DMA considers all possible combinations of predictors and forgetting factor values at each time-period. Typically, many candidate variables are available and, as a consequence, this poses a challenge for the computational facilities at hand, which for many practitioners typically consist of a standard 8-core CPU. In most cases, averaging over a relatively small number of model combinations (usually between 1000 to 3000) allows one to perform DMA using standard loops and software. However, handling larger number of combinations can quickly become very cumbersome and imposes technical limits on the software at hand, especially with regards to memory consumption, see for example, 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, it is not clear to us how one can incorporate the modifications suggested in Dangl and Halling (2012) within the framework of Onorante and Raftery (2016).

In this paper, we introduce the **eDMA** (Catania and Nonejad 2018) package for R (R Core Team 2017) available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=eDMA`. The package 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 with the **Rcpp** and **RcppArmadillo** packages of Eddelbuettel,

[1]Specifically, Koop and Korobilis (2012) change the conditional volatility formula of Raftery *et al.* (2010) arguing that the original formula is not suited for the analysis of economic data.

François, Allaire, Ushey, Kou, Bates, and Chambers (2016a) and Eddelbuettel, François, and Bates (2016b), respectively. Furthermore, the **OpenMP** API (ARB **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 and obtain quantities such as inclusion probabilities and out-of-sample forecasts or to perform variance decomposition. 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 also worth noting that, within the R environment, the **dma** package of McCormick, Raftery, and Madigan (2016) available from CRAN can be used to perform the DMA of Raftery *et al.* (2010). However, **dma** has several weaknesses such as (i) it does not allow for the extensions mentioned in 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 slow compared to the package introduced in this paper, (iii) it requires a very large amount of RAM when executed for moderately large applications, and (iv) it does not allow for parallel computing. We refer the reader interested in these aspects to Section 5, where we report a comparative analysis between **dma** and **eDMA** using simulated data. Moreover, **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, Hoeting, Volinsky, Painter, and Yeung 2015) and **BMS** (Zeugner and Feldkircher 2015). At the same time, we obtain quantities such as: posterior inclusion probabilities and average model size, which allow us to compare DMA (as well as dynamic model selection; DMS) with BMA (BMS) with regards to model shrinkage and the magnitude of variation in the average model size.

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. Framework

Many forecasting applications are based on a model where the variable of interest at time t, y_t, depends on exogenous predictors and possibly lagged values of y_t itself. For instance, in panel (a) of Figure 1, we plot the quarterly US inflation rate, $100\triangle \ln P_t$, where P_t denotes the US gross domestic product implicit price deflator (GDPDEF) from 1968Q1 to 2011Q2. We then recursively (i.e., using data up to time t) estimate an autoregressive model of order 2, AR(2), of y_t and report the sum of the autoregressive coefficients, which can be considered as a basic measure of inflation persistence in panel (b). Our general conclusions from panels (a)–

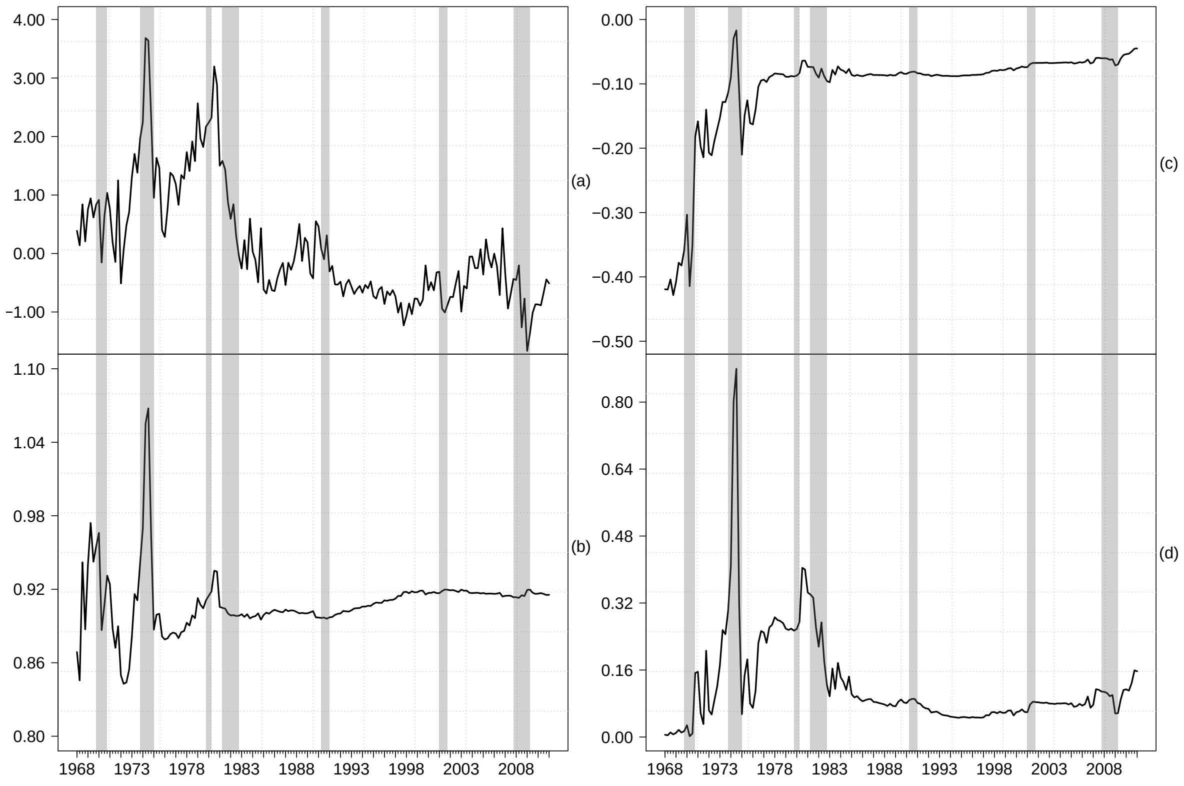

Figure 1: Panel (a): quarterly GDPDEF inflation from 1968Q1 to 2011Q2. Panel (b): inflation persistence estimates from an AR(2) model. Panel (c): recursive OLS estimates of, θ_4, in $y_t = \theta_1 + \theta_2 y_{t-1} + \theta_3 y_{t-2} + \theta_4 \text{UNEMP}_{t-1} + e_t$, where $e_t \sim N\left(0, \sigma^2\right)$. Panel (d): recursive p values for the null hypothesis of $\theta_4 = 0$ at the 5% level. 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.

(b) are: inflation is persistent and generally tends to be higher during recessions than tranquil periods. It does not appear to follow an identical cyclical pattern either. For example, inflation increases less aggressively towards the Great Recession of 2008 than the corresponding downturns in the 1970s, 1980s or the early 2000s. Furthermore, even in this simple model, we still observe some time-variation in the AR coefficients. We then extend the plain AR(2) model to also include the lagged unemployment rate (UNEMP) as a regressor. This way, we end up with a basic Philips curve. In panel (c), we report the recursive OLS estimates of UNEMP and in panel (d), we report the recursive p values associated to the null hypothesis that this estimate is equal to zero. Panel (d) shows that the unemployment rate in some periods seems to be a useful predictor of inflation.

Results from panels (a)–(d) of Figure 1 suggest that two changes can potentially help to improve the accuracy of inflation forecasts: (i) incorporating time-variation in the regression coefficients and (ii) augmenting the AR model with exogenous predictors that can capture information beyond that already contained in lagged values of y_t. Thus, in many economic applications, we eventually end up with a model such as:

$$y_t = \theta_{1t} + \theta_{2t} y_{t-1} + \theta_{3t} y_{t-2} + \theta_{4t} x_{t-1} + \ldots + \theta_{nt} z_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim N\left(0, V_t\right). \tag{1}$$

Obviously, n can be large and as a consequence, we may have to deal with a very large number of model combinations. For example, if our set of models is defined by whether each of the n potential predictors is included or excluded, then we can have as high as $k = 2^n$ model combinations to consider, which raises substantive challenges for model selection. This aspect is referred to as "model uncertainty", i.e., the uncertainty that a practitioner faces in choosing the correct combination of predictors. It is important to note, that discarding this aspect can have severe consequences on out-of-sample results. This is due to the fact that, simply adding additional predictors in our model without designing an optimal model selection strategy, can deteriorate out-of-sample performance due to the bias-variance trade-off (the additional reduction in bias afforded by including additional predictors does not offset the increased forecast variance related to the more heavily parameterized model). Besides model uncertainty, a practitioner also faces uncertainty regarding the nature of time-variation in the regression coefficients, i.e., "parameter uncertainty". Underestimating or overestimating the magnitude of time-variation in the regression coefficients also has important consequences as our model adapts either too slowly or too quickly to new data, generating either too rigid or too volatile forecasts. The DMA methodology provides an optimal way to deal with these sources of uncertainty. Moreover, it is simple, parsimonious and allows us to evaluate out-of-sample forecasts based on a large set of model combinations in real-time (no need to condition on the full sample at time t) without resorting to simulation.

To provide more details on the underlying mechanism of DMA, we start by assuming that any combination of the elements on the right-hand-side of (1) can be expressed as a dynamic linear model (DLM), see West and Harrison (1999) and Raftery *et al.* (2010). Particularly, let $\mathbf{F}_t^{(i)}$denote a $p \times 1$ vector based on a given combination of our total predictors, $\mathbf{F}_t = (1, y_{t-1}, y_{t-2}, x_{t-1}, \ldots, z_{t-1})^\top$. Then, we can express our ith DLM as:

$$y_t = \mathbf{F}_t^{(i)\top} \boldsymbol{\theta}_t^{(i)} + \varepsilon_t^{(i)}, \quad \varepsilon_t^{(i)} \sim N\left(0, V_t^{(i)}\right) \tag{2}$$

$$\boldsymbol{\theta}_t^{(i)} = \boldsymbol{\theta}_{t-1}^{(i)} + \boldsymbol{\eta}_t^{(i)}, \quad \boldsymbol{\eta}_t^{(i)} \sim N\left(0, \mathbf{W}_t^{(i)}\right), \tag{3}$$

where the $p \times 1$ vector of time-varying regression coefficients, $\boldsymbol{\theta}_t^{(i)} = \left(\theta_{1t}^{(i)}, \ldots, \theta_{pt}^{(i)}\right)^\top$, evolves according to (3) and determines 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 $\mathbf{W}_t^{(i)}$, are unknown quantities associated with the observational equation, (2), and the state equation, (3). Obviously, when $\mathbf{W}_t^{(i)} = \mathbf{0}$ for $t = 1, \ldots, T$, then $\boldsymbol{\theta}_t^{(i)}$ is constant over time. Thus, (2)–(3) nests the specification of constant regression coefficients. For $\mathbf{W}_t^{(i)} \neq \mathbf{0}$, $\boldsymbol{\theta}_t^{(i)}$ varies according to Equation 3. However, this does not mean that $\boldsymbol{\theta}_t^{(i)}$ needs to change at every time period. For instance, we can easily have periods where $\mathbf{W}_t^{(i)} = \mathbf{0}$ and thus $\boldsymbol{\theta}_t^{(i)} = \boldsymbol{\theta}_{t-1}^{(i)}$. Ultimately, the nature of time variation in the regression coefficients is dependent on the data at hand.[3]

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

[3]We model time-variation in $\boldsymbol{\theta}_t^{(i)}$ through a forgetting factor, δ, see below for more details. Moreover, we show that the recursive updating of the forgetting factor based on the predictive likelihood, avoids any unreasonable behavior of $\boldsymbol{\theta}_t^{(i)}$ even though, we do not specifically put any structure on $\boldsymbol{\theta}_t^{(i)}$. We also refer the reader to Appendix A.3 of Dangl and Halling (2012), where it is shown that (3) outperforms the autoregressive structured counterpart.

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 $\boldsymbol{\theta}_t^{(i)}$ can evolve over time.[4] DMA then averages forecasts across the different 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. Models containing important combinations of predictors receive high predictive likelihood values, which means that they obtain relatively higher posterior weights in the averaging process. Besides averaging, we can also use the 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).

As indicated in (3), we must specify $\mathbf{W}_t^{(i)}, i = 1, \ldots, k$. Obviously, this task can be very daunting if we were to specify $\mathbf{W}_t^{(i)}$ for each of the total k models. However, DMA avoids the difficult task of specifying $\mathbf{W}_t^{(i)}$ for each individual model relying on a forgetting factor, $0 < \delta \leq 1$. This in turn simplifies things greatly from a practical point of view as instead of working with many parameters, we only need to worry about δ. Now, we briefly explain how this mechanism works. We start by defining the variables of the Kalman recursions for the ith model as follows: (i) $\mathbf{R}_t^{(i)}$, the prediction variance of $\boldsymbol{\theta}_t^{(i)}$ (see Equation 17 in Appendix A at the end of paper), (ii) $\mathbf{C}_t^{(i)}$, the estimator for the covariance matrix of $\boldsymbol{\theta}_t^{(i)}$, (see Equation 19), and (iii) $S_t^{(i)}$, the estimator of the observational variance. Then, using δ, we can rewrite $\mathbf{R}_t^{(i)} = \mathbf{C}_{t-1}^{(i)} + \mathbf{W}_t^{(i)}$ in Appendix A as $\mathbf{R}_t^{(i)} = \delta^{-1}\mathbf{C}_{t-1}^{(i)}$, 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, $\mathbf{C}_t^{(i)}$. Clearly, we can control the magnitude of the shocks that impact $\boldsymbol{\theta}_t^{(i)}$ 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 $\boldsymbol{\theta}_t^{(i)}$ equals its value at time $t-1$. For $\delta < 1$, we introduce time-variation in $\boldsymbol{\theta}_t^{(i)}$. For instance, when $\delta = 0.99$, in the context of 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 observation, suggesting that a relatively larger shock hits the regression coefficients. Evidently, while this renders the model more flexible to adapt to changes in the data, the increased variability in $\boldsymbol{\theta}_t^{(i)}$ also results in higher prediction variance. Thus, estimating (2)–(3) depends not only on the choice of the predictors in $\mathbf{F}_t^{(i)}$ but also the choice of δ.

Conditional on δ, the DMA probability of model M_i conditional on the current information set at time t, $\mathcal{F}_t$, is then defined as:

$$p\left(M_i \mid \mathcal{F}_t\right) = \frac{p\left(y_t \mid M_i, \mathcal{F}_{t-1}\right) p\left(M_i \mid \mathcal{F}_{t-1}\right)}{\sum_{l=1}^{k} p\left(y_t \mid M_l, \mathcal{F}_{t-1}\right) p\left(M_l \mid \mathcal{F}_{t-1}\right)},$$

where $p\left(y_t \mid M_i, \mathcal{F}_{t-1}\right)$ is the predictive likelihood of model M_i evaluated at y_t, $p\left(M_i \mid \mathcal{F}_{t-1}\right) = p\left(M_i \mid \mathcal{F}_{t-1}\right)^{\alpha} / \Sigma_{l=1}^{k} p\left(M_l \mid \mathcal{F}_{t-1}\right)^{\alpha}$ where $0 < \alpha \leq 1$ is the forgetting factor for the entire model chosen by the practitioner and $p(M_i|\mathcal{F}_{t-1})$ is the model probability at time $t-1$. The forgetting factor parameter, α, induces time-variation in the entire model set. Clearly, the lower the value of α, the lesser weight is given to past performance. Raftery *et al.* (2010) and

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

Koop and Korobilis (2012) recommend setting α close to one. Dangl and Halling (2012), on the other hand, fix α at 1.

Finally, we must also determine a way to model the evolution of $V_t^{(i)}, i = 1, \dots, k$. Here, we have two options, which we go into more details below, see point (c). Thus in order to initialize the DMA recursions, a practitioner must:

(a) Consider the number of predictors. Typically, in economic applications, we use exogenous variables as well as lagged values of y_t as predictors. For instance, in the context of forecasting quarterly inflation, besides considering predictors such as unemployment and T-bill rates, Koop and Korobilis (2012) also consider the first three lags of y_t as predictors.

(b) Choose δ and α. In many applications $\alpha \in \{0.98, 0.99, 1\}$ works well and generally results do not change drastically across different values of α[5]. 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. Evidently, in many economic applications, it is 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 (where there is considerable time-variation in $\boldsymbol{\theta}_t^{(i)}$). Conversely, δ ought to be close to 1.00 during tranquil periods, where basically nothing changes. Dangl and Halling (2012) propose an 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 the corresponding grid of δ. At the same time, this strategy means that we avoid any unreasonable behavior of $\boldsymbol{\theta}_t^{(i)}$ as δ values incompatible with the data (and of course resulting in bad behavior on $\boldsymbol{\theta}_t^{(i)}$) do not receive a weight in the averaging process. 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) Evolution of $V_t^{(i)}$: We can make things easy for conjugate analysis by assuming that $V_t^{(i)} = V^{(i)}$ for all t. At time $t = 0$, we specify a Normal prior on $\boldsymbol{\theta}_0^{(i)}$ and an inverted-gamma prior on $V^{(i)}$, i.e., $V^{(i)}|\mathcal{F}_0 \sim \mathcal{IG}\left(\frac{1}{2}, \frac{1}{2}S_0^{(i)}\right)$, where $\mathcal{IG}\left(\frac{v}{2}, \frac{\kappa}{2}\right)$ stands for the inverted-gamma distribution with scale, v, and shape κ, see also Prado and West (2010). Then, the posterior of $V^{(i)}$ follows an $\mathcal{IG}$ distribution with parameters, $S_t^{(i)}$, $n_t^{(i)}$, where the time t point estimate of $V^{(i)}$, $S_t^{(i)}$, is given as

$$S_t^{(i)} = S_{t-1}^{(i)} + \frac{S_{t-1}^{(i)}}{n_t^{(i)}} \left(\frac{e_t^{2(i)}}{Q_t^{(i)}} - 1 \right),$$

$n_t^{(i)} = n_{t-1}^{(i)} + 1$, $e_t^{(i)}$ and $Q_t^{(i)}$ are given in Appendix A and Prado and West (2010). Clearly, $S_t^{(i)}$ approaches to a constant level as $n_t^{(i)}$ increases. More importantly, under these assumptions, we find that, when we integrate the conditional density of y_t over the values of $\boldsymbol{\theta}_t^{(i)}$ and $V^{(i)}$, the corresponding predictive density has a closed-form solution

[5]Recently, in the context of binary regressions, McCormick, Raftery, Madigan, and Burd (2012) suggest a technique where one can model α as time-varying.

given by, $\mathcal{T}_{n_t^{(i)}}\left(\hat{y}_t^{(i)}, Q_t^{(i)}\right)$, where $\mathcal{T}_{n_t^{(i)}}$ stands for the Student's t-distribution with $n_t^{(i)}$ degrees-of-freedom, mean and scale given by $\hat{y}_t^{(i)} = \mathbf{F}_t^{(i)\top}\mathbf{a}_{t-1}^{(i)}$ and $Q_t^{(i)}$, see Appendix A for more details.

However, in many applications, allowing for time-variation in the conditional error variance better suits our underlying economic assumptions. Therefore, we follow Prado and West (2010) and in a similar fashion as for $\mathbf{W}_t^{(i)}$ adopt a discount factor to induce time-variation in $V_t^{(i)}$. Particularly, we do this by imposing a forgetting factor, $0 < \beta \leq 1$, which enters the scale and the shape parameters of the inverted-gamma distribution, such that $n_t^{(i)} = \beta n_{t-1}^{(i)} + 1$. This way, $V_t^{(i)}$ is updated according to new data and forgetting past information to reflect changes in volatility. This approach means that, if $\beta < 1$, the time t estimate of $V_t^{(i)}$ is given as:

$$S_t^{(i)} = (1-\beta)\sum_{s=0}^{t-1}\beta^s\left(\frac{e_{t-s}^{2(i)}S_{t-s-1}^{(i)}}{Q_{t-s}^{(i)}}\right). \tag{4}$$

In other words, $V_t^{(i)}$ has the form of an exponentially weighted moving average (EWMA) and older data are further discounted as time progresses. When $\beta = 1$, then we recover $V_t^{(i)} = V^{(i)}$.[6] This extension obviously requires the practitioner to also consider a value for β. By experimenting with smaller models using simulated data, we observe that in some instances certain combinations of δ and β result in similar estimates of the regression coefficients. For example, in a model with a moderate degree of variation in $\boldsymbol{\theta}_t^{(i)}$, the values $\beta = 0.99$ and $\delta = 0.94$, imply similar dynamics in the regression coefficients as $\beta = 0.95$ and $\delta = 0.98$. This is understandable as allowing for variation in the conditional variance takes always some dynamics from the regression coefficients, whereas more dynamics in $\boldsymbol{\theta}_t^{(i)}$ are required in order to compensate for the possible lack of time-variation in $V_t^{(i)}$. Overall, our conclusion is that if a practitioner chooses to fix $\beta < 1$, then it is best to fix δ close to 1, say at 0.96, which is also the value used by Riskmetrics (1996). This way, we maintain a parsimonious model structure and allow for time-variation in $V_t^{(i)}$. More importantly, we reduce the risk of under-(over-)estimating the true magnitude of variation in $\boldsymbol{\theta}_t^{(i)}$.[7]

3. Modified DMA

Below, we present the DMA algorithm modified to incorporate the extensions mentioned in Section 2. Let M_i denote a model containing a specific set of predictors chosen from a set of $k = 2^n - 1$ candidates and δ_j denotes a specific forgetting factor value chosen from a pre-specified grid of values, $\{\delta_1, \ldots, \delta_d\}$. The total posterior density of model M_i and forgetting factor value δ_j at time t, $p(M_i, \delta_j|\mathcal{F}_t)$, is then given as

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

[6] We would like to thank an anonymous reviewer for this suggestion.

[7] 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. However, if a practitioner wishes to set $\beta < 1$, then we generally recommend $\beta > 0.96$ and $\alpha = 0.99$.

In order to obtain $p\left(M_i|\mathcal{F}_t\right)$ we can use the relation

$$p\left(M_i|\mathcal{F}_t\right)=\sum_{j=1}^{d} p\left(M_i|\delta_j,\mathcal{F}_t\right)p\left(\delta_j|\mathcal{F}_t\right). \tag{5}$$

The term, $p\left(M_i|\delta_j,\mathcal{F}_t\right)$, in Equation 5 is given as

$$p\left(M_i|\delta_j,\mathcal{F}_t\right)=\frac{p\left(y_t|M_i,\delta_j,\mathcal{F}_{t-1}\right)p\left(M_i|\delta_j,\mathcal{F}_{t-1}\right)}{\sum_{l=1}^{k}p\left(y_t|M_l,\delta_j,\mathcal{F}_{t-1}\right)p\left(M_l|\delta_j,\mathcal{F}_{t-1}\right)}, \tag{6}$$

where

$$p\left(M_i|\delta_j,\mathcal{F}_{t-1}\right)=\frac{p\left(M_i|\delta_j,\mathcal{F}_{t-1}\right)^{\alpha}}{\sum_{l=1}^{k}p\left(M_l|\delta_j,\mathcal{F}_{t-1}\right)^{\alpha}}. \tag{7}$$

The second term on the right-hand side of Equation 5 is given as

$$p\left(\delta_j|\mathcal{F}_t\right)=\frac{p\left(y_t|\delta_j,\mathcal{F}_{t-1}\right)p\left(\delta_j|\mathcal{F}_{t-1}\right)}{\sum_{l=1}^{d}p\left(y_t|\delta_l,\mathcal{F}_{t-1}\right)p\left(\delta_l|\mathcal{F}_{t-1}\right)}, \tag{8}$$

where

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

Typically, $p\left(M_i,\delta_j|\mathcal{F}_0\right)=1/(d\cdot k)$ 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 consists of:

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

$$\hat{y}_{t+1}=\sum_{j=1}^{d}\mathsf{E}\left[y_{t+1}^{(j)}|\mathcal{F}_t\right]p\left(\delta_j|\mathcal{F}_t\right), \tag{9}$$

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 given as

$$p\left(y_{t+1}|\mathcal{F}_t\right)=\sum_{j=1}^{d}p(y_{t+1}^{(j)}|\mathcal{F}_t)p(\delta_j|\mathcal{F}_t), \tag{10}$$

where

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

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. Henceforth, we label this as dynamic model selection (DMS), see also Koop and Korobilis (2012). When, δ, β and α are all fixed at 1, we have Bayesian model averaging (BMA, see Raftery 1995) and Bayesian model selection (BMS) based on exact predictive likelihood, see for instance Zeugner and Feldkircher (2015).[8]

(ii) Quantities such as the expected size, $\mathsf{E}\left[\text{Size}_t\right] = \Sigma_{i=1}^{k}\text{Size}^{(i)}p\left(M_i|\mathcal{F}_t\right)$, where $\text{Size}^{(i)}$ corresponds to 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, (5), at each point in time, which give the optimal model size at time t.

(iii) Posterior inclusion probabilities for the predictors. 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, $m = 1,\ldots,n$, 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 integrating out the effect of δ. 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 that is $\sum_{j=1}^{d}\delta_j p\left(\delta_j|\mathcal{F}_t\right)$, for $t = 1,\ldots,T$.

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

$$\mathsf{E}[\boldsymbol{\theta}_t|\mathcal{F}_t] = \sum_{j=1}^{d} \mathsf{E}[\boldsymbol{\theta}_t^{(j)}|\mathcal{F}_t]p(\delta_j|\mathcal{F}_t), \tag{11}$$

where

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

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

$$\mathsf{VAR}\left(y_{t+1}|\mathcal{F}_t\right) = \text{Obs}_{t+1} + \text{Coeff}_{t+1} + \text{Mod}_{t+1} + \text{TVP}_{t+1}, \tag{12}$$

[8] 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.

where

$$\begin{aligned}
\mathrm{Obs}_{t+1} &= \sum_{j=1}^{d}\left[\sum_{i=1}^{k}\left(S_t|M_i,\delta_j,\mathcal{F}_t\right)p\left(M_i|\delta_j,\mathcal{F}_t\right)\right]p\left(\delta_j|\mathcal{F}_t\right),\\
\mathrm{Coeff}_{t+1} &= \sum_{j=1}^{d}\left[\sum_{i=1}^{k}\left(\mathbf{F}_t^\top\mathbf{R}_t\mathbf{F}_t|M_i,\delta_j,\mathcal{F}_t\right)p\left(M_i|\delta_j,\mathcal{F}_t\right)\right]p\left(\delta_j|\mathcal{F}_t\right),\\
\mathrm{Mod}_{t+1} &= \sum_{j=1}^{d}\left[\sum_{i=1}^{k}\left(\hat{y}_{i,t+1}^{(j)}-\hat{y}_{t+1}^{(j)}\right)^2p\left(M_i|\delta_j,\mathcal{F}_t\right)\right]p\left(\delta_j|\mathcal{F}_t\right),\\
\mathrm{TVP}_{t+1} &= \sum_{j=1}^{d}\left(\hat{y}_{t+1}^{(j)}-\hat{y}_{t+1}\right)^2p\left(\delta_j|\mathcal{F}_t\right).
\end{aligned} \tag{13}$$

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 (ARB **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 also free to manage the level of resources used by the program.

The **eDMA** package is written using the S4 framework for object-oriented programming (Chambers 1998), meaning that classes and methods are defined. Specifically, R users will find common methods are available, such as `plot()`, `show()`, `as.data.frame()`, `coef()` and `residuals()`, among others, in order to visualize the output of DMA and extract estimated quantities.

The **eDMA** package is available from CRAN at `https://CRAN.R-project.org/package=eDMA` and can be installed using the command:

```
R> install.packages("eDMA")
```

Once the package is correctly installed and loaded, the user has available the function `DMA()` to perform DMA. The `DMA()` function then accepts a series of 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 use. 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 $\left(y_t, \mathbf{F}_t^\top\right)$, for all $t = 1, \ldots, T$.[9]

- `vDelta`: a $d \times 1$ numeric vector representing a grid for 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 7. By default `dAlpha = 0.99`.

- `dBeta`: a numeric variable indicating the forgetting factor for the measurement variance, see Equation 4 and Prado and West (2010, p. 132) and Beckmann and Schüssler (2014). By default `dBeta = 1.0`, i.e., the observational variance is constant.

- `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 the fourth variable 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 (g) equal to 100 by default. If `bZellnerPrior = TRUE`, then

$$p\left(\boldsymbol{\theta}_0^{(i)} | \mathcal{F}_0\right) \sim \mathcal{N}\left(0, g S_0^{(i)} \left(\mathbf{F}_{1:T}^{(i)\top} \mathbf{F}_{1:T}^{(i)}\right)^{-1}\right), \tag{14}$$

  where

$$S_0^{(i)} = \frac{1}{T-1} \mathbf{y}_{1:T}^\top \left(I_T - \mathbf{F}_{1:T}^{(i)} \left(\mathbf{F}_{1:T}^{(i)\top} \mathbf{F}_{1:T}^{(i)}\right)^{-1} \mathbf{F}_{1:T}^{(i)\top}\right) \mathbf{y}_{1:T},$$

[9] Recall that the inclusion of the constant term should be managed via the `formula` argument.

and $\mathbf{y}_{1:T} = (y_1, \ldots, y_T)^\top$ and $\mathbf{F}_{1:T}^{(i)}$ indicating the $T \times p$ design matrix according to model i. If `bZellnerPrior = FALSE`, it represents the scaling factor for the covariance matrix of the Normal prior for $\boldsymbol{\theta}_0^{(i)}$, i.e., $\boldsymbol{\theta}_0^{(i)} \sim N(0, g \times \mathbb{I})$, $i = 1, \ldots, k$, where $\mathbb{I}$ is the identity matrix. We generally recommend practitioners to use the default prior, i.e., `bZellnerPrior = FALSE`, especially in the context of quarterly data, where we typically have 200 to 300 observations. For longer time series, results tend to be similar after 100 observations.

- `bParallelize`: a Boolean variable indicating whether 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. If complexity is judged "low" (that is, less than 100 models), `iCores` has not been selected by the user (`iCores = NULL`), and `bParallelize = TRUE`, the number of cores is subsequently reduced to 2. The choice of the number of cores depends on the specific application, namely the length of the time series T and the number of the predictors n. However, as detailed in Chapman, Jost, and Van Der Pas (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 their application depending on their hardware by changing this parameter. Section 5 reports details about code parallelization.

The `DMA()` function returns an object of the *formal* class '`DMA`'.[10] 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[11] of the regression coefficients, $\boldsymbol{\theta}_t$ (as in Equation 11), and so on. Finally, the slot, `data`, includes the data passed to the `DMA()` function, organized 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 provide an example based on simulated data. We also provide an application using quarterly inflation data in Section 6. We simulate a time series of $T = 500$ observations from

$$y_t = \mathbf{F}_t^\top \boldsymbol{\theta}_t + \sqrt{0.1}\varepsilon_t, \quad \varepsilon_t \overset{\text{iid}}{\sim} \mathcal{N}(0, 1). \tag{15}$$

[10]See `help("class")` and `help("DMA-class")`.

[11]With the term "filtered estimates" we indicate estimates at time t conditional on information up to time t.

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, $\boldsymbol{\theta}_t = (\theta_{1,t}, \theta_{2,t}, \theta_{3,t}, \theta_{4,t}, \theta_{5,t}, \theta_{6,t})^\top$ with

$$\theta_{k,t} = \theta_{k,t-1} + \sqrt{0.01}\eta_{k,t}, \quad \eta_{k,t} \overset{\text{iid}}{\sim} \mathcal{N}(0,1), \tag{16}$$

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 $t = 1, \dots, T$. 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., $\mathbf{F}_t = (1.0, x_{2,t}, x_{3,t}, x_{4,t}, x_{5,t}, x_{6,t})^\top$, where $x_{k,t} \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$ and $x_{k,t} \perp\!\!\!\perp x_{j,t}$ for all $k \neq j$. We simulate the data in this 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 through $\mathbf{F}_t$. Conversely, the 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 Averaging        -
------------------------------------------

Model Specification
T     = 500
n     = 6
d     = 11
Alpha = 0.99
Beta  = 1
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 =  0.9, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1
```

```
------------------------------------------

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 rely on the implemented `as.data.frame()` method. The `as.data.frame()` method 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 9. `"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) on page 10.
- `"vsize_DMS"`: number of predictors in the model with the highest posterior model probability, at each point in time, see Equation 5.
- `"mtheta"`: filtered estimates of the regression coefficients for DMA, see Equation 11.
- `"mpmt"`: posterior probability of the forgetting factors, see Equation 8.
- `"vdeltahat"`: posterior weighted average of δ, see point (iv) on page 10 of this paper.
- `"vLpdfhat"`: predictive log-likelihood of DMA, see Equation 10.
- `"vLpdfhat_DMS"`: predictive log-likelihood of DMS. That is instead of averaging over the individual predictive likelihoods, we select the predictive likelihood of the model combination with the highest posterior probability (i.e., Equation 5) at each time period.
- `"mvdec"`: individual components of Equation 12, see point (vi) on page 10 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., $\max_i p(M_i|\mathcal{F}_t)$, $t = 1, \dots, T$.
- `"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`) $\times$ 6 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 0.06 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 0.08 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.

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

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

```
Type 1-16 or 0 to exit
 1: Point forecast
 2: Predictive likelihood
 3: Posterior weighted average of delta
 4: Posterior inclusion probabilities of the predictors
 5: Posterior probabilities of the forgetting factors
 6: Filtered estimates of the regression coefficients
 7: Variance decomposition
 8: Observational variance
 9: Variance due to errors in the estimation of the coefficients, theta
10: Variance due to model uncertainty
11: Variance due to uncertainty with respect to the choice of
    the degrees of time-variation in the regression coefficients
12: Expected number of predictors (average size)
13: Number of predictors (highest posterior model probability) (DMS)
14: Highest posterior model probability (DMS)
15: Point forecasts (highest posterior model probability) (DMS)
16: Predictive likelihood (highest posterior model probability) (DMS)
```

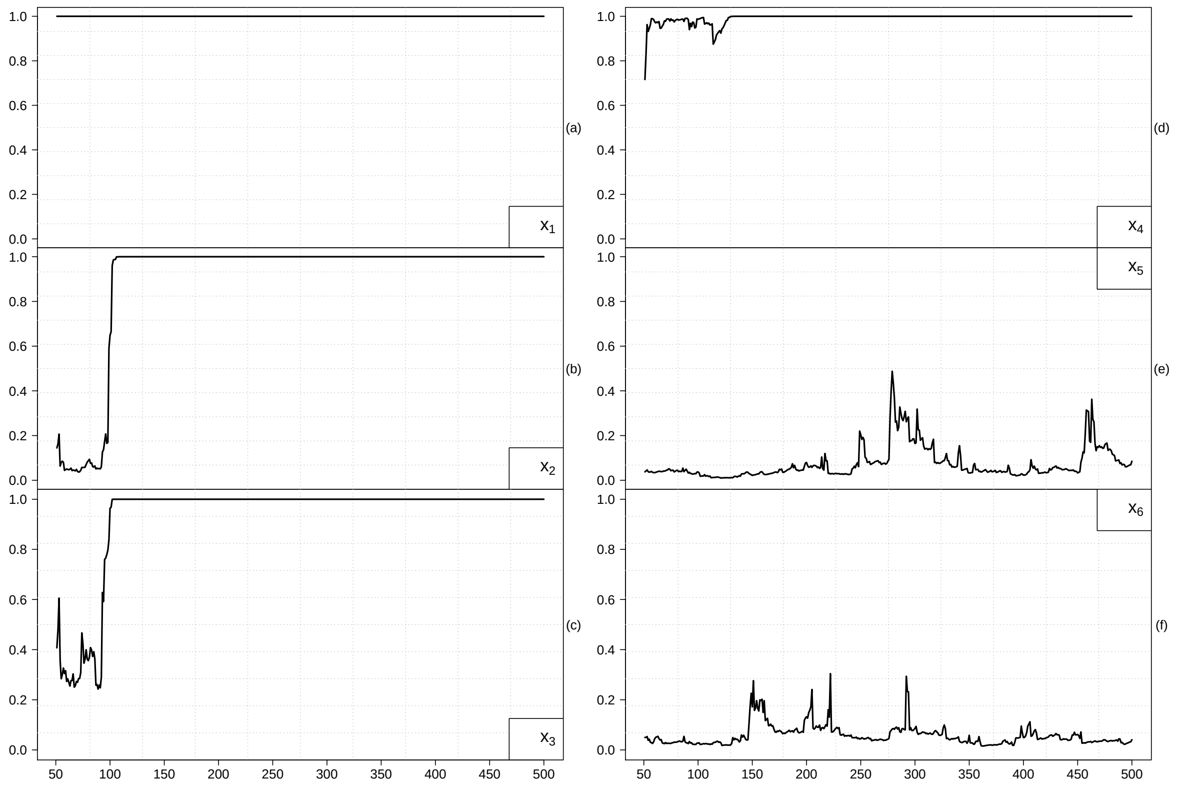

Figure 2: Posterior inclusion probabilities of the predictors using simulated data.

and selecting the desired options. The additional character argument, `which`, can be supplied in order to directly plot one particular quantity. Possible values for `which` are the same as for 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 iterations for the model to adapt to the time series given the prior. Therefore, in almost all applications, the first 30 to 50 observations should be 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 2. As expected, x_1 to x_4 quickly converge to 1 after a few 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.

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

Call:
 DMA(formula =  y ~ x2 + x3 + x4 + x5 + x6 )

Residuals:
    Min      1Q  Median      3Q     Max
-2.0445 -0.3844  0.0414  0.4398  2.3759

Coefficients:
            E[theta_t] SD[theta_t] E[P(theta_t)] SD[P(theta_t)]
(Intercept)       0.51        0.68          1.00           0.00
x2               -0.64        0.65          0.90           0.29
x3                2.10        1.74          0.92           0.23
x4               -1.43        1.02          0.99           0.03
x5                0.01        0.03          0.07           0.07
x6                0.00        0.01          0.06           0.04

Variance contribution (in percentage points):
  vobs vcoeff   vmod   vtvp
 64.12  34.24   1.50   0.15

Top 10% included regressors:        (Intercept)

Forecast Performance:
                               DMA      DMS
MSE                          0.489    0.483
MAD                          0.539    0.532
Log-predictive Likelihood -463.820 -463.076
```

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 $\boldsymbol{\theta}_t$, and the inclusion probabilities after burn-in.

The last part of the summary, i.e., the part titled `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 log-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, $\boldsymbol{\theta}_t$, $t = 1, \ldots, T$.
- `residuals()`: Extracts the residuals of the model, i.e., $y_t - \hat{y}_t$, $t = 1, \ldots, T$. The additional Boolean argument `standardize` controls if the standardized 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()`: Extracts the inclusion probabilities of the predictors. Analogous to `as.data.frame(object, which = "mincpmt", iBurnPeriod)`.

- `pred.like()`: Extracts 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 an argument `iBurnPeriod`.

- `getLastForecast`: If we extend the time series of the dependent variable of length T (i.e., observations that we actually observe until time T) with an `NA` value, resulting in a series of length $T + 1$, then the `DMA()` function computes the point forecast and the associated variance decomposition for the future observation at time $T + 1$, see Appendix B for further details. In this case, the `getLastForecast` can be used to extract the "true" out-of-sample[12] forecast at time $T + 1$.

5. Computational challenges

Although estimation of DMA does not require resorting to simulation, in many economic applications, performing DMA can become computationally cumbersome. As can be seen from the set of recursions in Section 3, DMA consists of a large number of model combinations, where a lot of the quantities must be saved for subsequent analysis. Therefore, 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 2 is the percentage changes in the quarterly US 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 even in the context of quarterly data, which at most consists of around 300 observations, reveals to be cumbersome 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. Furthermore, they do not consider a grid for different values of δ, which would result in $2^{14} \times d$ combinations, making inference even more challenging.

We can argue that DMA can impose a substantial challenge for the practitioner when dealing with a large number of predictors and a high number of observations. Besides dealing with the task of transforming mathematical equations to code, handling data and estimation issues, practitioners also have to overcome "technical/computer science related" 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

[12]We use the term "true" out-of-sample to distinguish from the case of "pseudo" out-of-sample which corresponds to the usual recursive out-of-sample forecast, where one compares the forecasts with the actual observed values.

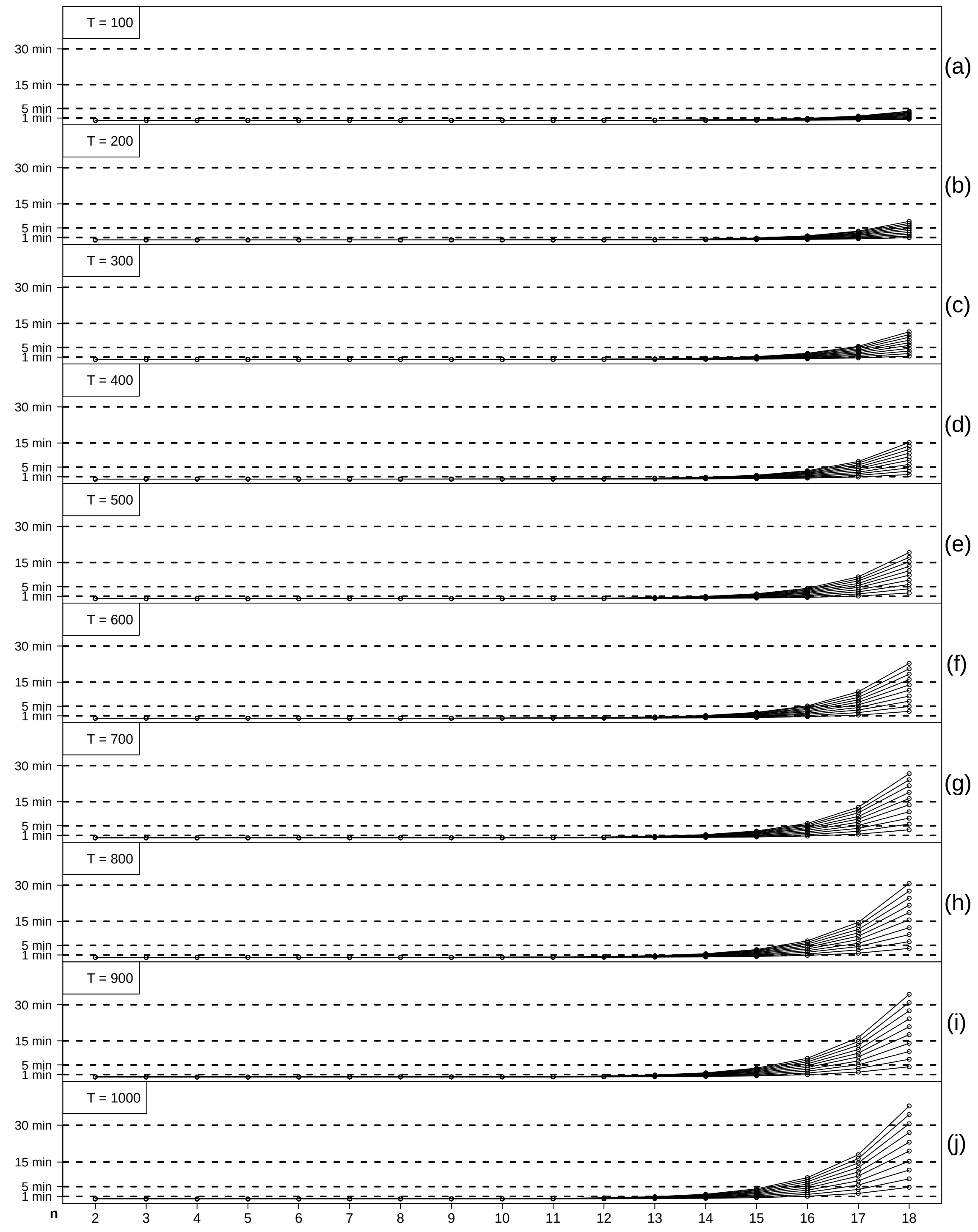

Figure 3: Computational time for `DMA()` using simulated data. Each panel represents computation time in minutes for `DMA()` using different sample sizes, T, number of predictors, n, and values of d, the number of points in the grid of δ. The values for d range between 2 and 10, the solid line at the 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.

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 achieved by writing all 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).[13] Note that, the **dma** package is entirely written in plain R and cannot be run in parallel, consequently, even if the algorithm we implement is slightly different from those of **dma** (recall that we follow the implementation of Dangl and Halling 2012), improvement in computational time should be principally attributed to the two aforementioned reasons.

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, for $T = 500$ and $n = 16$, `dma()` takes on average 20.48 minutes while `DMA()` only 25.08 seconds.[14] 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 standard hardware, but also enables practitioners to run applications with a large number of exogenous variables. To give an idea of the computational time a user of **eDMA** faces, we report a second simulation study. We simulate from a DLM with $T = \{100, 200, \dots, 900, 1000\}$, $n = \{2, 3, \dots, 18\}$ and run `DMA()` using a grid of values for δ between 0.9 and 1.0 with different spaces d, namely $d = \{2, 3, \dots, 10\}$. Figure 3 displays the computational time in minutes for all the combinations of T, n, and 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 $d = 2$,[15] the one immediately above is for $d = 3$ and so on until $d = 10$. From Figure 3, we can see that, when $T \leq 400$, even for $n = 18$ and $d = 10$, the computational time is less than 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

[13] Clearly, results reported in this section depend on the given hardware configuration, which we have detailed in this section as well as in the section "Computational details " at the end of the paper. However, even if the computational times change with the hardware configuration, the results presented in Table 1 and Figure 3 will remain qualitatively similar.

[14] Also note that this cannot be considered a one-to-one comparison because `DMA()` performs additional operations (such as DMS and variance decomposition) which are not considered by `dma()`. It is worth mentioning that a wrong specification of the number of cores for computational inexpensive tasks (for example, computing the first column of Table 1 where $n = 4$), can result in a slower execution time of `DMA()` relative to `dma()`. This is due to the additional computational time required by parallelization, see the description of the `iCores` argument at page 13.

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

T/n	4	6	8	10	12	14	16
100	34.5	41.4	60.7	81.6	69.5	54.5	49.8
500	47.3	54.3	92.9	82.4	70.5	49.0	54.1
1000	59.0	58.3	81.6	84.2	71.3	50.6	51.9

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 using different values of T and n. The ratio is computed using the average computational time taken after 10 code evaluations using the **microbenchmark** package of Mersmann (2015). Computing this table on a different configuration (Intel Xeon CPU E3-1535M v6 @ 3.10GHz with 8GB of RAM) gives qualitatively similar results.

$d = 10$, the 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 2 and 6, we need to define two arrays of dimension $T \times d \times k$. These kind of objects are not included in the **eDMA** package since we rely on the Markovian nature of the model clearly evident from Equation 2. In this respect, we keep track of the quantities coming from Equation 6 and $p\left(y_t|M_i, \delta_j, \mathcal{F}_{t-1}\right)$ only for two consecutive periods during the loop over T.[16] RAM usage is still efficiently performed in the **eDMA** package. Indeed, the computer where we run all our simulations has only 8 GB of RAM. A formal analysis of RAM usage with the **eDMA** package is hard to implement given that RAM profiling for C++ functions wrapped in R cannot be easily performed.[17] However, we find that **eDMA** on a Windows 10 based system equipped with 16 GB of RAM fixing $T = 300$ is able to handle 4'194'304 model combinations while, for example, **dma** can only handle 2'097'157, i.e., half of **eDMA**.

6. A DMA example: Inflation data

We use a time series of quarterly US inflation rate with exogenous predictors for illustration and then step by step show how to obtain the posterior output. 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, business cycles and perform variance decomposition.

6.1. Data

We rely on the data-set of Groen *et al.* (2013).[18] As a measure of inflation, y_t, we consider quarterly log-changes in the gross domestic product implicit price deflator (GDPDEF) ranging from 1960q1 to 2011q2. The number of exogenous predictors is fifteen. This number is in accordance with typical "real-world" applications, see also Dangl and Halling (2012) and Koop and Korobilis (2012).

[16]Differently, in the **dma** package a full $T \times k$ matrix is stored.

[17]This is the case also for contributed packages such as **profvis** (Chang and Luraschi 2017).

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

We start by loading the **eDMA** package and the data-set by typing:

```
R> library("eDMA")
R> data("USData", package = "eDMA")
```

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 payrolls 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 (the GDPDEF series along with the fifteen predictors) in the **eDMA** package as a '`xts`' object of dimension 206×16 named `USData`. A glance at the GDPDEF series and the first five predictors is obtained by typing:

```
R> head(round(USData[, 1:6], 2))
```

```
           GDPDEF ROUTP RCONS RINVR  PIMP UNEMP
1960-01-01  -1.14  1.66  0.62  0.55 -0.48 -0.56
1960-04-01  -0.77 -1.39  0.33 -1.84 -0.37 -0.49
1960-07-01  -0.71 -0.68 -0.71 -0.69 -0.16 -0.30
1960-10-01  -0.76 -2.32 -1.27 -0.07 -0.47  0.16
1961-01-01  -1.27 -0.19 -2.25  0.04 -0.32  0.49
1961-04-01  -1.16  1.24  0.23  0.03 -0.40  0.62
```

For most series, we follow Groen *et al.* (2013) and 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 Groen *et al.* (2013) for more details. Finally, since inflation is very persistent, besides these 15 predictors, we follow Groen *et al.* (2013) and also include four inflation lags, $y_{t-1}, \dots, y_{t-4}$, as predictors. In **eDMA**, we implemented the function, `Lag()`, which allows us to lag variables delivered in the form of vector or matrices. For instance, to lag the numeric vector `X` of length T by one period, we simply run

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

which returns a numeric vector of length T containing the lagged values of `X`. Values that are not available are replaced by `NA`.

6.2. Model estimation

We have a total of $2^{19} = 524'288$ model combinations.[19] Furthermore, we let $\delta = \{0.9, 0.91, \dots,$

[19]Models which do not include the constant term are not considered. Note that, when `vKeep = NULL`, the number of models is $2^n - 1$, however, when `vKeep != NULL`, the number of models is $2^b - 1$, where `b = n - length(vKeep)`.

1} such that we have a total of $(2^{19}) \cdot 11 = 5'767'168$ combinations. We set $\beta = 0.96$, a value we generally suggest in the context of working with quarterly data, $\alpha = 0.99$, $g = 100$, $p(M_s \mid \mathcal{F}_0) = 1/(d \cdot k)$, $s = 1, \ldots, d \cdot k$, such that initially, all models are equally likely. We then update these model probabilities as new information arrives. As previously mentioned, we include a constant term in all models, see also Groen *et al.* (2013).

In order to perform DMA using the `DMA()` function, we write[20]:

```
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) +
+    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,
+    dBeta = 0.96, dAlpha = 0.99)
```

We suggest using the non-informative prior, `bZellnerPrior = FALSE`, which is the default. In this way the regression coefficients are centered at 0 with a flat prior and adapt quickly in the averaging process as new information arrives. More details on the model can be made available by typing `Fit`.

```
R> Fit

------------------------------------------
-        Dynamic Model Averaging         -
------------------------------------------

Model Specification
T     = 202
n     = 20
d     = 11
Alpha = 0.99
Beta  = 0.96
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 =  0.9, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1
------------------------------------------

Elapsed time : 1429.13 secs
```

[20]Note that this command can be computational expensive for non-**OpenMP** ready systems.

As can be seen, the total estimation time of our DMA when working with more than 5’700’000 model combinations at each time period is 1429.13 seconds corresponding to around 23.8 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
-1.3948 -0.3169 -0.0073  0.2309  1.6503

Coefficients:
               E[theta_t] SD[theta_t] E[P(theta_t)] SD[P(theta_t)]
(Intercept)          0.08        0.16          1.00           0.00
Lag(GDPDEF, 1)       0.43        0.17          0.84           0.29
Lag(GDPDEF, 2)       0.03        0.02          0.20           0.13
Lag(GDPDEF, 3)       0.10        0.08          0.38           0.25
Lag(GDPDEF, 4)       0.10        0.05          0.42           0.21
Lag(ROUTP, 1)        0.00        0.01          0.13           0.09
Lag(RCONS, 1)        0.00        0.00          0.12           0.07
Lag(RINVR, 1)        0.01        0.02          0.13           0.07
Lag(PIMP, 1)         0.19        0.08          0.77           0.29
Lag(UNEMP, 1)       -0.03        0.09          0.12           0.10
Lag(NFPR, 1)         0.02        0.02          0.20           0.16
Lag(HSTS, 1)         0.02        0.02          0.16           0.08
Lag(M2, 1)           0.01        0.01          0.16           0.08
Lag(OIL, 1)         -0.02        0.05          0.22           0.23
Lag(RAW, 1)          0.00        0.01          0.11           0.07
Lag(FOOD, 1)         0.01        0.01          0.17           0.12
Lag(YL, 1)           0.20        0.34          0.25           0.29
Lag(TS, 1)           0.00        0.01          0.11           0.05
Lag(CS, 1)          -0.02        0.04          0.14           0.07
Lag(MS, 1)           0.02        0.03          0.15           0.07

Variance contribution (in percentage points):
  vobs vcoeff   vmod   vtvp
 65.70  13.21  19.93   1.16

Top 10% included regressors:        (Intercept), Lag(GDPDEF, 1)

Forecast Performance:
                               DMA      DMS
MSE                          0.226    0.278
MAD                          0.355    0.386
Log-predictive Likelihood -98.490 -121.752
```

Note that, we set burn-in to 32 (`iBurnPeriod = 32`) such that the start of the evaluation period corresponds to 1969q1, see also Koop and Korobilis (2012). Below, we go into more details with regards to how to use the output from the estimation procedure.

6.3. Using the output from eDMA

The output can be divided into two main parts: (a) full-sample, and (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           0.48           0.71
2010-04-01            1           0.99           0.49           0.72
2010-07-01            1           0.99           0.51           0.73
2010-10-01            1           0.99           0.51           0.73
2011-01-01            1           0.99           0.51           0.73
2011-04-01            1           0.99           0.51           0.73
```

The above matrix shows the inclusion probabilities of: the constant and $y_{t-1}, \ldots, y_{t-3}$, 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) on page 10 of this paper. The interested reader can examine these estimates more carefully.

In Figure 4, we report the inclusion probabilities for the more important predictors. To be precise, any predictor where the inclusion probabilities are never above 0.2 is excluded. In these plots, we also make evident NBER recorded recessions (shaded gray bars).[21] Overall, we observe a good amount of time-variation in 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. Real spot price of oil (OIL) receives high inclusion probabilities

[21] Recession dates are included in the **eDMA** package and can be loaded with `data("USRecessions", package = "eDMA")`, see `help("USRecessions")`.

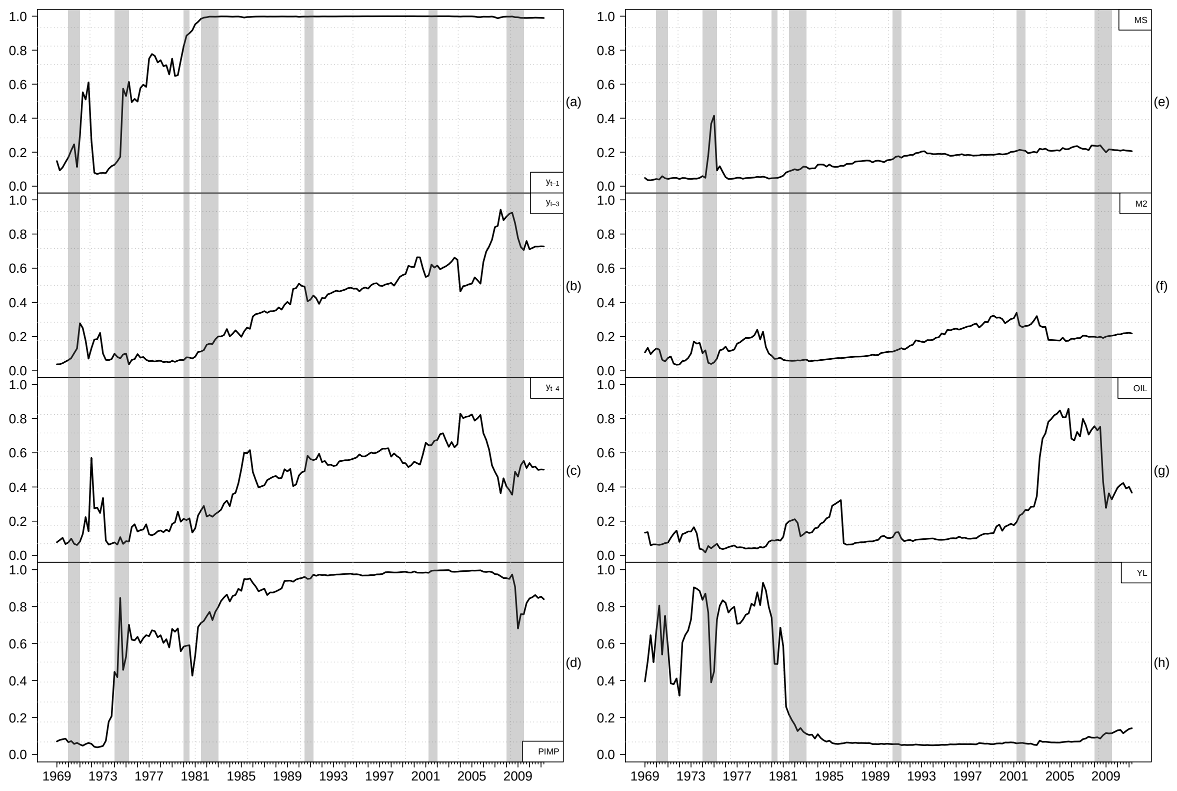

Figure 4: Posterior inclusion probabilities for the most important predictors of DMA. Panels (a), (b) and (c): first, third and fourth lags of inflation. Panel (d): import deflator (PIMP). Panel (e): inflation expectations (MS). Panel (f): M2 monetary aggregate (M2). Panel (g): real spot price of oil (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.

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 5. These quantities are extracted from `Fit` by 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 regression coefficients, `"vdeltahat"`, which is the posterior weighted average of δ at each point in time. We report this estimate in panel (a) of Figure 6. The analogous plot in R can be obtained using:

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

There is a very intuitive relationship between δ and the business cycles. Typically, δ falls at the onset of recessions, which fares well with the notion that relatively larger shocks hit $\boldsymbol{\theta}_t$ in these

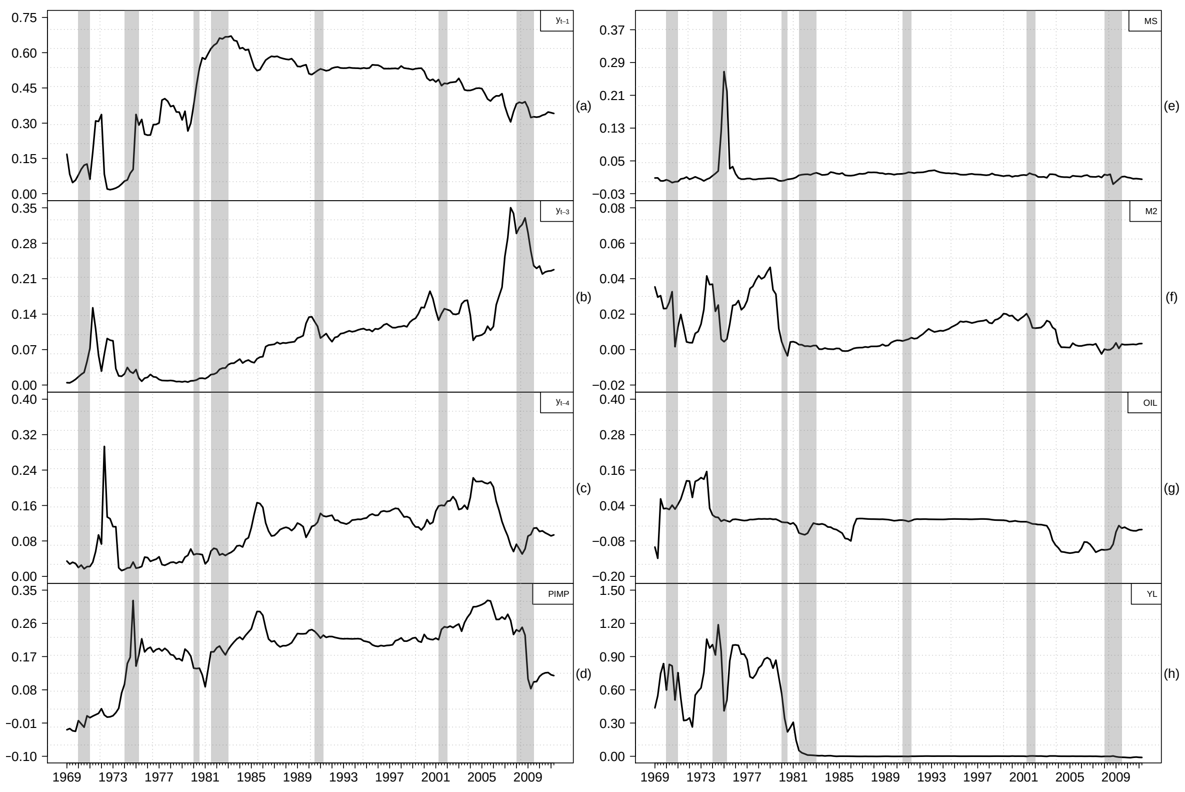

Figure 5: Filtered estimates of the regression coefficients for the most important predictors of DMA. Panels (a), (b) and (c): first, third and fourth lags of inflation. Panel (d): import deflator (PIMP). Panel (e): inflation expectations (MS). Panel (f): M2 monetary aggregate (M2). Panel (g): real spot 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.

periods. Thereafter, δ tends to rise again. Conversely, δ remains high and close to 1 during the Great Moderation, which again fares well with the notion of relatively minor variation in the regression coefficients in expansion periods. We can also use `as.data.frame()` to extract the posterior probability of each value of δ and print them using:

```
R> InclusionProbDelta <- as.data.frame(Fit, which = "mpmt", 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 0.01 0.01 0.01 0.02 0.05 0.10 0.21 0.31 0.27
2010-04-01   0    0 0.01 0.01 0.01 0.03 0.05 0.10 0.21 0.31 0.26
2010-07-01   0    0 0.00 0.00 0.01 0.02 0.04 0.10 0.22 0.33 0.27
2010-10-01   0    0 0.00 0.00 0.01 0.02 0.05 0.12 0.23 0.32 0.24
2011-01-01   0    0 0.00 0.00 0.01 0.02 0.05 0.12 0.23 0.31 0.24
2011-04-01   0    0 0.00 0.01 0.01 0.02 0.06 0.13 0.25 0.31 0.21
```

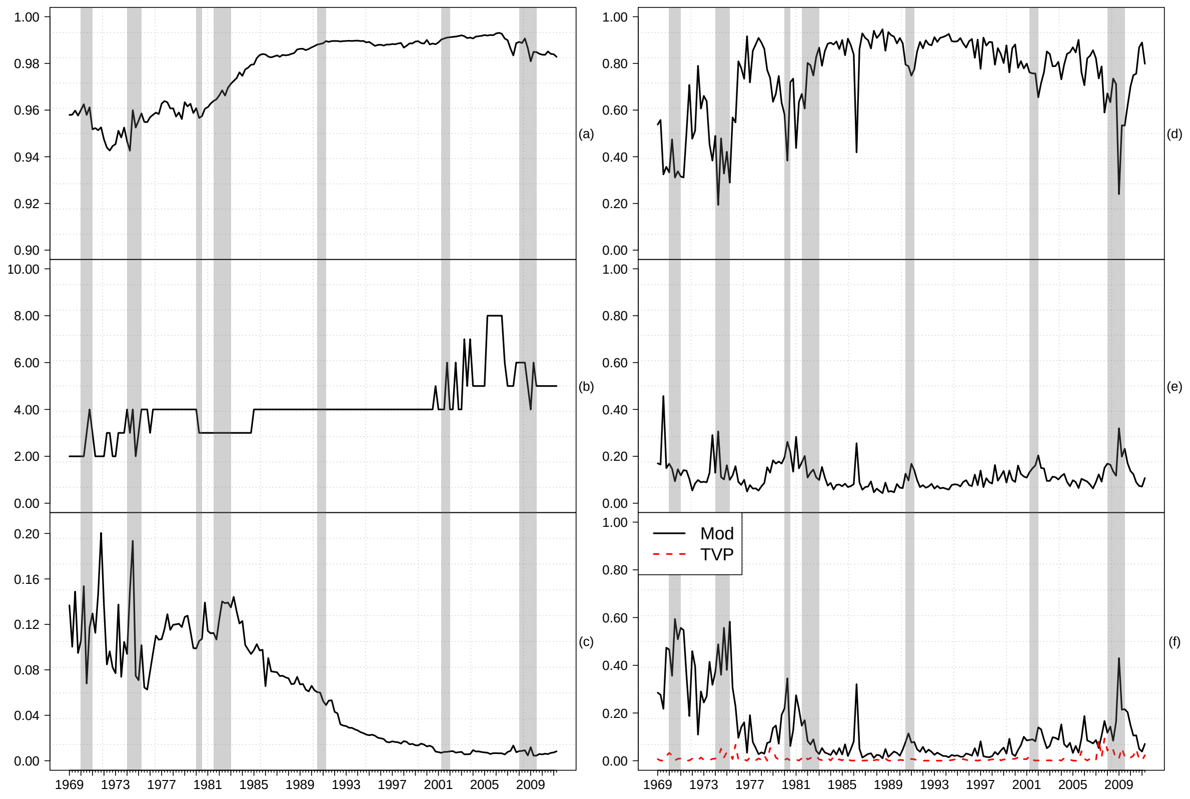

Figure 6: Posterior output for DMA. Panel (a): posterior weighted average estimate of δ. Panel (b): number of predictors for the model with the highest posterior probability. Panel (c): sum of top 10% inclusion probabilities. Panel (d): observational variance. Panel (e): variance due to errors in the 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.

where the column names are the values of δ.

In panel (b) of Figure 6, 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)
```

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

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 $\alpha = 1$, $\delta = 1$ and $\beta = 1$.
$\mathcal{M}_1$	Time-varying AR(4) model: The constant term and $y_{t-1}, \ldots, y_{t-4}$ are always included. We set $\alpha = 0.99$, $\beta = 0.96$ 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 $\alpha = 0.99$, $\beta = 0.96$ and average over the combinations of $y_{t-1}, \ldots, y_{t-4}$ 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 $\alpha = 0.99$, $\beta = 0.96$ 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 $\alpha = 0.99$, $\beta = 0.96$ and select the model with the highest posterior probability at each t and use it to forecast.
$\mathcal{M}_5$	BMA: DMA with $\alpha = 1$, $\delta = 1$ and $\beta = 1$.
$\mathcal{M}_6$	BMS: DMS with $\alpha = 1$, $\delta = 1$ and $\beta = 1$.
$\mathcal{M}_7$	Kitchen sink: The constant term, $y_{t-1}, \ldots, y_{t-4}$ and all exogenous predictors are always included. We set $\alpha = 0.99$, $\beta = 0.96$ 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.

Finally, in panels (d), (e) and (f) of Figure 6, 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. Furthermore, uncertainty regarding the degree of time-variation in the regression (TVP) is lower in comparison. However, this is understandable as posterior probabilities of δ (see above) favor $\delta = 0.98$, 0.99 and 1.

6.4. Out-of-sample forecasts

An 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 generate forecasts.

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

where `vKeep = c(1, 2, 3, 4, 5)` indicates that all predictors are included.[22] The same

[22]This is equivalent to `vKeep = "KS"`.

Model	$h = 1$		$h = 5$	
	MSE	PLD	MSE	PLD
$\mathcal{M}_1$	0.998	12.444	0.815	48.445
$\mathcal{M}_2$	0.964	14.416	0.728	64.008
$\mathcal{M}_3$	0.938	20.561	0.704	94.399
$\mathcal{M}_4$	1.155	−2.701	0.844	62.227
$\mathcal{M}_5$	0.985	7.234	1.138	19.368
$\mathcal{M}_6$	1.096	−7.543	1.308	−25.294
$\mathcal{M}_7$	1.839	−9.899	0.965	47.832

Table 3: Mean squared error (MSE) and log-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.

holds for Bayesian model averaging (BMA, $\mathcal{M}_5$) and Bayesian model selection (BMS, $\mathcal{M}_6$) by setting $\delta = 1.0$, $\alpha = 1.0$ and $\beta = 1.0$. Thus, **eDMA** also relates to the **BMS** package of Zeugner and Feldkircher (2015) and the **BMA** package of Raftery *et al.* (2015).

We use the models to obtain one ($h = 1$) and five ($h = 5$) quarter ahead forecasts through direct forecasting, see Marcellino, Stock, and Watson (2006).

Table 3 reports the mean squared error (MSE) and the log-predictive likelihood difference (PLD) of $\mathcal{M}_i$, $i = 1, \ldots, 7$, over $\mathcal{M}_0$ (the benchmark) at $h = 1$ and $h = 5$.[23]

Compared to the benchmark, $\mathcal{M}_1$ provides gains both in terms of MSE and PLD relative to the benchmark, especially at $h = 5$. By averaging over $y_{t-1}, \ldots, y_{t-4}$ and accounting for parameter instability, we obtain even more gains. DMA using lags of inflation as well as 15 additional predictors is the top performer, regardless of h. Similar to Groen *et al.* (2013) the exogenous predictors contain enough information besides the lags to improve forecast accuracy. Conversely, DMS is outperformed by the benchmark at $h = 1$. This result is understandable as panel (c) in Figure 6 demonstrates that no individual model or group of model combinations performs overwhelmingly better than the other specifications. By looking more carefully at DMS results, we find that at $h = 1$, DMS produces volatile forecasts at the start and towards the end of the sample, which explains why it is outperformed by the benchmark. This is evident from panel (b) of Figure 6, where we observe notable changes in the number of predictors in the optimal model at the start of the sample, towards and during the Great Recession of 2008.

As previously mentioned, DMA (DMS) with $\alpha = \delta = \beta = 1$ corresponds to BMA (BMS). At $h = 1$, compared to the benchmark model, BMA provides improvements in density and point forecasts. Similar to DMS, BMS is outperformed by the benchmark at $h = 1$. At both horizons, results confirm that accounting for model uncertainty and parameter instability lead to out-of-sample gains.

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 compared to $\mathcal{M}_0$. However, the kitchen sink model is always outperformed by DMA.

[23] We recall that multi-step-ahead forecast is performed via direct forecasting as in Koop and Korobilis (2012). For instance, the `formula` used for model $\mathcal{M}_0$ when $h = 5$ is `GDPDEF ~ Lag(GDPDEF, 5) + Lag(GDPDEF, 6) + Lag(GDPDEF, 7) + Lag(GDPDEF, 8)`.

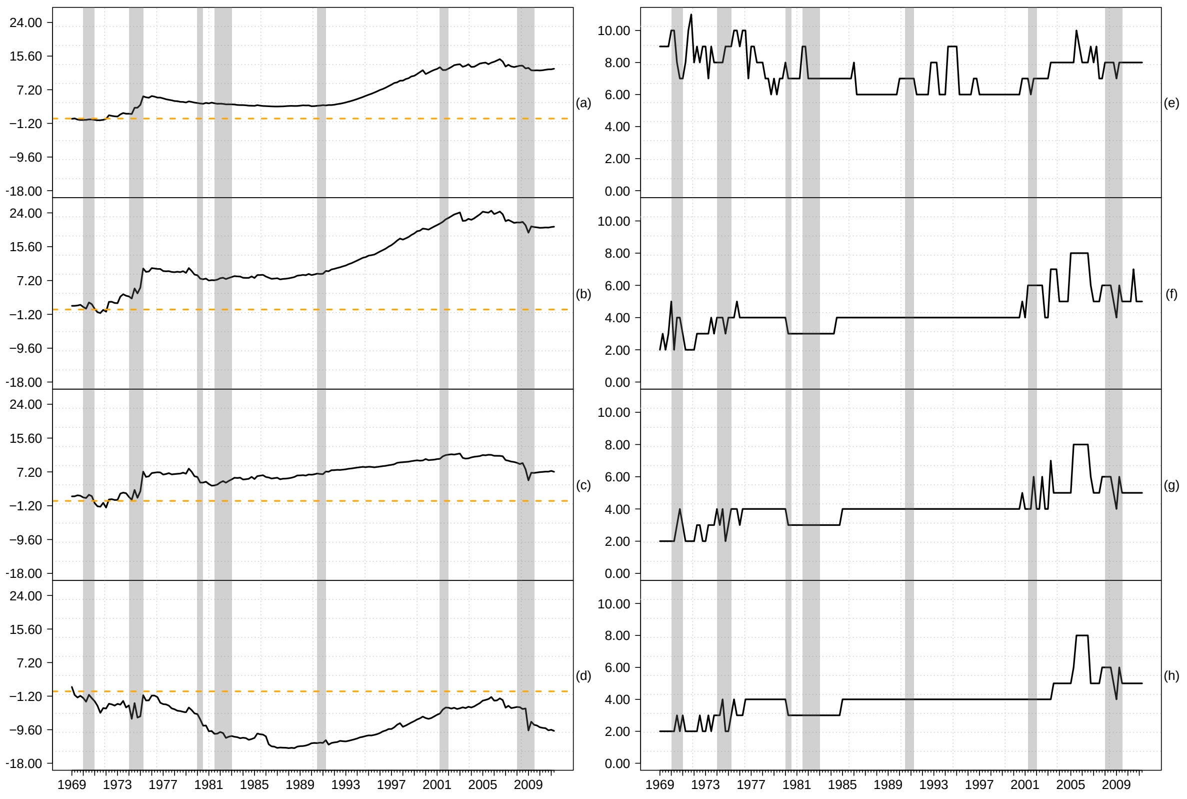

Figure 7: Accumulated PDL and the optimal number of predictors for prior sensitivity analysis. Panel (a): $\mathcal{M}_1$ over $\mathcal{M}_0$. Panel (b): $\mathcal{M}_3$ over $\mathcal{M}_0$. Panel (c): $\mathcal{M}_5$ over $\mathcal{M}_0$. Panel (d): $\mathcal{M}_7$ over $\mathcal{M}_0$. Panels (e)–(h): number of predictors for the model with the highest posterior probability using $g = 0.1, 20, T/2, T$. 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.

6.5. Why does DMA perform well?

To investigate how quickly our techniques adapt to changes in the data, we report the accumulated log-PLD for several models over the benchmark in panels (a)–(d) of Figure 7. These can be obtained using the `pred.like()` method available for ‘`DMA`’ objects. For instance, we create the two vectors `vPL_M0` and `vPL_M3` containing the log-predictive likelihoods 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 <- cumsum(vPL_M3 - vPL_M0)
```

which is reported in panel (b) of Figure 7.

In panels (a), (b), (c) and (d) of Figure 7 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) model outperforms the benchmark throughout the out-of-sample period. Compared to the plain AR(4) model, it takes about twenty observations to provide compelling evidence in favor of DMA. Furthermore, we also observe that DMA performs well in recession as well as expansion periods. Compared to BMA, the improvements of DMA are mostly concentrated on the onset of recessions. However, DMA also outperforms BMA during expansion periods. Conversely the kitchen sink model is generally outperformed by the benchmark throughout the out-of-sample, see panel (d) of Figure 7.

6.6. The choice of g

In the context of DMA, the prior hyperparameter value, g, must be specified by the practitioner. Intuitively, a smaller value of g means more shrinkage around the prior mean of $\boldsymbol{\theta}_0^{(i)}$, i.e., $\mathbf{0}$. 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. In other words, the larger the g, the more we allow the data to speak freely. This way, we ensure that the estimation procedure quickly adapts to the data, even at quarterly frequency, which typically consist of around 300 observations. On the other hand, for some data-sets, it can take the estimation procedure a longer time to adapt if we set g to relatively lower values. Thus, in such cases, DMA can initially overfit as the average model size becomes larger than it ought to be. This effect becomes evident by examining the average number of predictors in DMA and in most cases is also heavily reflected in the generated forecasts, where DMA is outperformed by the benchmark.

We re-estimate DMA with g equal to 0.1, 20, $T/2$ and T (using `bZellnerPrior = FALSE`) and observe to which extent different values of g influence out-of-sample results. Results are reported in Table 4 and panels (e)–(h) of Figure 7. Overall, we find that results are robust to different values of g. All g values lead to similar MSE and PLD estimates and the number of predictors in the model with the highest posterior probabilities are also similar, see panels (e)–(h) of Figure 7. However, we must mention that this is mainly due to the properties of our data and the fact that `bZellnerPrior = FALSE` such that, contrary to `bZellnerPrior = TRUE`, the observations do not affect the prior covariance matrix of $\boldsymbol{\theta}_t^{(i)}$, see Equation 14. In fact, when we repeat the analysis with `bZellnerPrior = TRUE`, we find that DMA using $g = 0.1$ and $g = 20$ perform much worse and are outperformed by the benchmark model. On the other hand, as we increase g to $T/2$ and T, we obtain similar results to those reported in Table 4. This result is understandable as given the scale of the prior covariance matrix under `bZellnerPrior = TRUE`, prior shrinkage is much greater under $g = 0.1$ and $g = 20$.

Ultimately, it is up to the practitioner to choose g. However, our general recommendation is to fix $g = T$ regardless of `bZellnerPrior = TRUE` or `FALSE` and the number of observations as it allows the data to speak freely about the underlying relations between the regressors and

Prior	MSE	PLD
$g = 0.1$	0.967	20.186
$g = 20$	0.937	20.664
$g = T/2$	0.938	20.556
$g = T$	0.941	20.431

Table 4: Mean squared error (MSE) and log-predictive likelihood difference (PLD) of DMA using the following values of g: $0.1, 20, T/2, T$ and $\mathcal{M}_0$ for $h = 1$.

the dependent variable. However, as previously mentioned, we recommend `bZellnerPrior = FALSE`, for small data-sets.[24]

7. Conclusion

In this paper, we present the **eDMA** package for R. The purpose of **eDMA** is to offer an integrated environment to easily perform DMA using the available `DMA()` function, which enables practitioners to perform DMA exploiting multiple processors. Furthermore, R users will find common methods to represent and extract estimated quantities such as `plot()`, `as.data.frame()`, `coef()` and `residuals()`.

Overall, **eDMA** has the following advantages: (i) it incorporates the extensions introduced in Prado and West (2010) and Dangl and Halling (2012), which are relevant for economic and financial applications, (ii) compared to other approaches, it performs the computations 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 minutes using a standard laptop. Large applications can still benefit from the use of **eDMA** even when performed on desktop or clusters, without additional effort from the user.

Computational details

The results in this paper are obtained using R 3.2.3 with the packages: **eDMA** version 1.5-0 (Catania and Nonejad 2018), **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), **devtools** version 1.1.1

[24]An anonymous reviewer also made a very good point regarding choosing g, which can be summarized as follows:

(i) Choose b values of g, say $g = \{0.1, 20, T/2, T\}$. Then run DMA for each of these values and save the predictive likelihoods $p\left(y_t^{(i)} \mid \mathcal{F}_{t-1}\right)$, $t = 1, \ldots, T$ for $i = 1, \ldots, b$.

(ii) Compute $p\left(y_t^{(i)} | \mathcal{F}_{t-1}\right) / \Sigma_{i=1}^{b} p\left(y_t^{(i)} | \mathcal{F}_{t-1}\right)$, $t = 1, \ldots, T$ for $i = 1, \ldots, b$.

Thus, we can observe which value of g obtains high posterior probabilities, especially at the start of the sample. We can then use the associated g value in the estimation procedure.

(Wickham and Chang 2016), and **microbenchmark** version 1.4-2.1 (Mersmann 2015). R itself and all packages used are available from CRAN at `https://CRAN.R-project.org/`. The package **eDMA** is available from CRAN at `https://CRAN.R-project.org/package=eDMA/`. Computations were performed on a Genuine Intel quad core CPU i7-3630QM 2.40Ghz processor.

Acknowledgments

We are grateful to the Editor (Bettina Grün), the Associate Editor (Rob J Hyndman), and two anonymous referees for their useful comments which improved the paper significantly. We thank Henning Bunzel and Søren Staunsager at CREATES for computational support during the testing of the **eDMA** package.

References

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

Beckmann J, Schüssler R (2014). "Forecasting Equity Premia Using Bayesian Dynamic Model Averaging." *Technical report*, Center for Quantitative Economics (CQE), University of Muenster. URL `https://www.wiwi.uni-muenster.de/cqe/sites/cqe/files/CQE_Paper/CQE_WP_29_2014.pdf`.

Byrne JP, Korobilis D, Ribeiro PJ (2018). "On the Sources of Uncertainty in Exchange Rate Predictability." *International Economic Review*, **59**(1), 329–357. `doi:10.1111/iere.12271`.

Catania L, Nonejad N (2018). ***eDMA**: Dynamic Model Averaging with Grid Search.* R package version 1.5-0, URL `https://CRAN.R-project.org/package=eDMA`.

Chambers JM (1998). *Programming with Data: A Guide to the S Language.* Springer-Verlag, New York.

Chang W, Luraschi J (2017). ***profvis**: Interactive Visualizations for Profiling R Code.* R package version 0.3.3, URL `https://CRAN.R-project.org/package=profvis`.

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

Croissant Y, Millo G (2008). "Panel Data Econometrics in R: The **plm** Package." *Journal of Statistical Software*, **27**(2), 1–43. `doi:10.18637/jss.v027.i02`.

Dangl T, Halling M (2012). "Predictive Regressions with Time-Varying Coefficients." *Journal of Financial Economics*, **106**(1), 157–181.

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 JJ, 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 JJJ, 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. `doi:10.1016/j.jeconom.2005.07.020`.

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`.

McCormick TH, Raftery AE, Madigan D, Burd RS (2012). "Dynamic Logistic Regression and Dynamic Model Averaging for Binary Classification." *Biometrics*, **68**(1), 23–30. `doi:10.1111/j.1541-0420.2011.01645.x`.

Mersmann O (2015). ***microbenchmark**: Accurate Timing Functions.* R package version 1.4-2.1, URL `https://CRAN.R-project.org/package=microbenchmark`.

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`.

Paye BS (2012). "'Déjà Vol': Predictive Regressions for Aggregate Stock Market Volatility Using Macroeconomic Variables." *Journal of Financial Economics*, **106**(3), 527–546. `doi:10.1016/j.jfineco.2012.06.005`.

Prado R, West M (2010). *Time Series: Modeling, Computation, and Inference.* CRC Press, Boca Raton.

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 (1995). "Bayesian Model Selection in Social Research." *Sociological Methodology*, **25**, 111–163. doi:10.2307/271063.

Raftery AE, Kárný 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.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Riskmetrics (1996). "Riskmetrics – Technical Document." *Technical report*, J.P. Morgan/Reuters, New York. URL https://www.msci.com/www/research-paper/1996-riskmetrics-technical/018482266.

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. NBER Working Paper No. 14322, URL http://www.nber.org/papers/w14322.

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

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.

A. The mathematics of dynamic linear models

Below, we briefly outline the Kalman recursions for the ith DLM model in the model averaging. We refer the reader to Prado and West (2010) for more details on the Kalman recursions. Based on the information up to time $t-1$, the prior if the state vector, $\boldsymbol{\theta}_t^{(i)}$, at time t follows $\mathcal{N}\left(\mathbf{a}_t^{(i)}, \mathbf{R}_t^{(i)}\right)$, where

$$\begin{aligned}
\mathbf{a}_t^{(i)} &= \mathbf{m}_{t-1}^{(i)}, \\
\mathbf{R}_t^{(i)} &= \mathbf{C}_{t-1}^{(i)} + \mathbf{W}_t^{(i)}.
\end{aligned} \tag{17}$$

Conditional on $V_t^{(i)}$, the one-step-ahead predictive mean and variance of $y_t^{(i)}$ follows a Normal distribution with mean, $f_t^{(i)}$, and variance, $Q_t^{(i)}$, where

$$\begin{aligned}
f_t^{(i)} &= \mathbf{F}_t^{(i)\top} \mathbf{a}_t^{(i)}, \\
Q_t^{(i)} &= \mathbf{F}_t^{(i)\top} \mathbf{R}_t^{(i)} \mathbf{F}_t^{(i)} + V_t^{(i)}.
\end{aligned} \tag{18}$$

Once we observe y_t, we can compute the forecast error as $e_t^{(i)} = y_t - f_t^{(i)}$. The posterior distribution for $\boldsymbol{\theta}_t^{(i)}$ given the current information set, $\mathcal{F}_t$, is then updated as

$$\begin{aligned}
\mathbf{m}_t^{(i)} &= \mathbf{a}_t^{(i)} + \mathbf{A}_t^{(i)} e_t^{(i)}, \\
\mathbf{C}_t^{(i)} &= \mathbf{R}_t^{(i)} - \mathbf{A}_t^{(i)} \mathbf{A}_t^{(i)\top} Q_t^{(i)},
\end{aligned} \tag{19}$$

where $\mathbf{A}_t^{(i)}$ is the adaptive coefficient vector $\mathbf{A}_t^{(i)} = \mathbf{R}_t^{(i)} \mathbf{F}_t^{(i)} / Q_t^{(i)}$.

B. True out-of-sample forecast

There might be cases where the practitioner desires to predict $T+1$ conditional on observations until time T in a true out-of-sample fashion (i.e., without having the possibility of backtesting the forecast since y_{T+1} cannot be observed). In such circumstances, the user can substitute the future value of the dependent variable with an `NA`. This way, the code treats the last observation as missing and does not perform backtesting or updating of the coefficients. However, the estimation procedure provides us with the necessary quantities to perform prediction. The predicted value $\widehat{y}_{T+1} = \mathsf{E}[y_{T+1}|\mathcal{F}_T]$ as well as the predicted variance decomposition defined in Equation 12 can then be extracted using the `getLastForecast` method available in the **eDMA** package. The other quantities that can be extracted, for example via the `as.data.frame` method, will ignore the presence of the last `NA` and report results only for the first T observations.

For example, consider the simulated data, `SimData`, detailed in Section 4.1

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

Recall that this is a 500×6 dataframe simulated from the model defined in Equations 15–16. The first column represents the dependent variable, y_t, while the last five columns the predictors $x_{i,t}$ for $i = 2, \ldots, 6$ and $t = 1, \ldots, T = 500$. Assume that we observe (or that we have previously forecasted) the values for the predictors at time $T+1$, i.e., $x_{i,T+1} \in \mathcal{F}_T$ for $i = 2, \ldots, 6$, and these are $x_{2,T+1} = -0.07$, $x_{3,T+1} = 1.61$, $x_{4,T+1} = -2.07$, $x_{5,T+1} = 0.17$, $x_{6,T+1} = -0.80$. What we need to do it is simply add a new row to the `SimData` dataframe

```
R> newData <- c(NA, -0.07, 1.61, -2.07, 0.17, -0.80)
R> SimData <- rbind(SimData, newData)
```

and run DMA

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

In order to extract the predicted value $\widehat{y}_{T+1} = \mathsf{E}[y_{T+1}|\mathcal{F}_T]$ and the predicted variance decomposition, we simply run

```
R> getLastForecast(Fit)

$PointForecast
[1] 11.5293

$VarianceDecomposition
      vtotal         vobs       vcoeff         vmod         vtvp
4.290887e-01 2.805227e-01 1.478767e-01 6.682507e-04 2.108273e-05
```

Affiliation:

Leopoldo Catania
Department of Economics and Business Economics
Aarhus University and CREATES
Fuglesangs Allé 4
8210 Aarhus V, Denmark
E-mail: leopoldo.catania@econ.au.dk

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

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
April 2018, Volume 84, Issue 11 *Submitted:* 2016-06-17
doi:10.18637/jss.v084.i11 *Accepted:* 2017-05-02