ECONSTOR
Make Your Publications Visible.

A Service of

ZBW Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Barde, Sylvain

Working Paper

A fast algorithm for finding the confidence set of large collections of models

School of Economics Discussion Papers, No. 1519

Provided in Cooperation with:
University of Kent, School of Economics

Suggested Citation: Barde, Sylvain (2015) : A fast algorithm for finding the confidence set of large collections of models, School of Economics Discussion Papers, No. 1519, University of Kent, School of Economics, Canterbury

This Version is available at:
https://hdl.handle.net/10419/130001

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.

WWW.ECONSTOR.EU

Mitglied der Leibniz-Gemeinschaft

University of Kent

School of Economics Discussion Papers

A fast algorithm for finding the confidence set of large collections of models

Sylvain Barde

September 2015

KDPE 1519

A fast algorithm for finding the confidence set of large collections of models*

Sylvain Barde†‡

September 2015

Abstract

The paper proposes a new algorithm for finding the confidence set of a collection of forecasts or prediction models. Existing numerical implementations for finding the confidence set use an elimination approach where one starts with the full collection of models and successively eliminates the worst performing until the null of equal predictive ability is no longer rejected at a given confidence level. The intuition behind the proposed implementation lies in reversing the process: one starts with a collection of two models and as models are successively added to the collection both the model rankings and p-values are updated. The first benefit of this updating approach is a reduction of one polynomial order in both the time complexity and memory cost of finding the confidence set of a collection of M models, falling respectively from $\mathcal{O}\left(M^3\right)$ to $\mathcal{O}\left(M^2\right)$ and from $\mathcal{O}\left(M^2\right)$ to $\mathcal{O}\left(M\right)$. This theoretical prediction is confirmed by a Monte Carlo benchmarking analysis of the algorithms. The second key benefit of the updating approach is that it intuitively allows for further models to be added at a later point in time, thus enabling collaborative efforts using the model confidence set procedure.

JEL Classification: C12, C18, C52, C55.

Keywords: Model selection, model confidence set, bootstrapped statistics

*The author is extremely grateful to Stefano Grassi and Guy Tchuente for their helpful advice on the MCS methodology and to Jagjit Chadha for his comments on an earlier version of this paper. Special thanks goes to James Holdsworth for his invaluable work in maintaining the computer cluster on which the Monte Carlo analysis was run. Any errors in the manuscript remain of course the author's.

†School of Economics, Keynes College, University of Kent, Canterbury, CT2 7NP, UK
tel : +44 (0)1 227 824 092, email: s.barde@kent.ac.uk

‡Affiliate, Observatoire Français des Conjonctures Economiques

Non-Technical Summary

Social scientists often do not have the benefit of being able to run or replicate experiments in order to generate new data and end up re-using the same datasets when evaluating the explanatory power of new models compared to older models. As pointed out by White (2000), such sequential testing of models on a fixed amount of data leads to the problem known as 'data snooping', in which the initially small probability of a poor model appearing good by random chance gets amplified by repeated testing. Ignoring this effect or naively selecting the best-fitting model without further testing can often lead to incorrectly identifying as a 'best' a model that in fact has no real predictive power on the data. In order to help avoid this problem, White proposes a 'Reality Check' procedure which tests the null hypothesis that no model in a given collection outperforms a given benchmark model.

Hansen et al. (2011) offer a generalisation of the reality check, in the form of the Model Confidence Set (MCS) approach, which identifies the subset of models which have equal predictive power on some data and has the benefit of not requiring an *a priori* benchmark model. Starting with the full collection of models, the procedure successively eliminates the worst performing model until the null of equal predictive ability is no longer rejected at a given confidence level. The surviving subset of models makes up the MCS at that confidence level. The flexibility of the MCS procedure has made it very popular as a way of evaluating the forecasting ability of multivariate GARCH models of volatility, leading to numerous comparison exercises with relatively large collections of models (such as 600 models in Liu et al. (2015)) compared on a small range of data sets such as the S&P 500 index (for example in Neumann and Skiadopoulos (2013) or Wilhelmsson (2013)).

In spite of its flexibility and popularity, this highlights two drawbacks to the MCS approach. The first is that because the elimination process starts with the full collection of models and gradually shrinks it down to the subset of models that forms the MCS, it is not possible to add extra models to a collection *ex post* without having to rerun the MCS procedure on the entire, larger, collection. It is therefore not possible to simply update the confidence set with a few more candidate models, which is something that was possible with White's Reality Check, and which one might argue is desirable in view of the many parallel exercises carried out using similar specifications on the same volatility data. A second, related, drawback of the elimination process is that its time complexity and memory requirements increase rapidly with the size of the model collection, thus making it cumbersome to analyse large model collections.

This paper proposes an alternative approach to obtaining the MCS of a collection of models which preserves the attractiveness and flexibility of the methodology while addressing the two potential drawbacks mentioned above. The intuition is that the iterative process used to find the MCS can be reversed: rather than starting with the full collection of models and shrinking it down the the subset of models that form the MCS, a collection is initially made up of two models and MCS is gradually updated as models are added to the collection. The MCS is obtained once all the models in the collection are processed. Growing the collection of models rather than shrinking it intuitively allows for further models to be added to the collection at a later point in time. Furthermore, because only deviations of the new model with respect to the existing collection need to be calculated and stored for each iteration, this reducing the time complexity of the MCS procedure from $\mathcal{O}\left(M^3\right)$ to $\mathcal{O}\left(M^2\right)$ and the memory requirement from $\mathcal{O}\left(M^2\right)$ to $\mathcal{O}\left(M\right)$. This is confirmed by a Monte Carlo analysis carried out in order to validate this updating approach.

1 Introduction

Data snooping is a well-known problem which occurs when attempting to use a fixed amount of data to identify the best predictor from a collection of models: as the number of prediction models in the collection increases, the probability of accidentally selecting a model with no real predictive power increases. As pointed out by Corradi and Swanson (2013), the starting point of the literature on this issue is Diebold and Mariano (1995), who provide a test for the null hypothesis that two prediction models have equal accuracy. This test forms the basis of the "reality check" (RC) test for data snooping, proposed by White (2000), which sequentially tests a collection of forecasts against a benchmark model in order to test the null that no model in the collection outperforms the benchmark. The data snooping bias incurred when sequentially testing different models on the same data is avoided by the use of a bootstrap implementation. The procedure is improved by Hansen (2005) with the test for superior predictive accuracy (SPA), which offers better protection against the inclusion of irrelevant models into the collection.

A further generalisation is proposed by Hansen et al. (2011) in the form of the model confidence set (MCS) procedure, which unlike the RC and SPA tests does not require an a priori benchmark, therefore providing greater flexibility when no obvious benchmark is available. Given a collection of model losses on a dataset, the MCS procedure sequentially tests the null hypothesis that all models in the collection have equal predictive power on the data and eliminates worst performing the model from the set if the hypothesis is rejected. The next worst model is then tested and the process continues until the null hypothesis can no longer be rejected. The surviving subset of models then forms the MCS.[1]

Despite the relatively recent nature of the MCS procedure, it has rapidly become a popular method of assessing the accuracy of volatility forecasts, leading to a large volume of research. A few examples of this literature, which focus heavily on multivariate GARCH forecasting methods, are Patton et al. (2009), Laurent et al. (2012), Amado and Teräsvirta

[1]Numerical implementations of the MCS procedure are available in the MULCOM package for Ox, available at `http://mit.econ.au.dk/vip_htm/alunde/MULCOM/MULCOM.HTM`, in the MFE toolbox for Matlab, available at `https://www.kevinsheppard.com/MFE_Toolbox`, and in a recent R package provided by Bernardi and Catania (2014). This paper uses the MFE toolbox implementation as the reference, due to its wide use in the literature.

(2014), Hamid (2014), Caporin and McAleer (2014), Hansen et al. (2014). Boudt et al. (2013) uses the MCS to assess the improvement in forecasting performance brought by incorporating jumps into the standard GARCH forecasts, while Iltuzer and Tas (2013) uses the full set of RC, SPA and MCS tests to evaluate the measurement of forecast accuracy. In a different type of application, Neumann and Skiadopoulos (2013) and Wilhelmsson (2013) who both use the MCS to evaluate density forecasts that integrate higher moments beyond simple volatility, and both use the S&P 500 index to carry out their analysis.

An attractive aspect of the RC bootstrap implementation which is unfortunately lost with the MCS procedure is the ability to perform incremental testing. As explained by White (2000, p. 1110), a collaborative effort can be carried out by posting the bootstrap indices, the values of the RC test statistic and bootstrapped test statistics, allowing "researchers at different locations or at different times to further understanding of the phenomenon modeled without needing to know the specifications tested by their collaborators or competitors". Ferrari and Yang (2015, p.3) point out this is not possible with the MCS, which "is meant to handle only a fixed and small number of models to begin with".

Given the large literature mentioned above, as well as the tight focus on multivariate GARCH modeling of volatility and use of very similar stock or foreign exchange market data, one could argue that it would be desirable to have a methodology that allows researchers confront their findings. The related issue, pointed out by Ferrari and Yang (2015) is the relative inability of the MCS implementation to deal with the large model collections that would result from such a comparison exercise. Liu et al. (2015), for instance, examine a collection of 600 different realised measures of asset price volatility in an attempt to assess if these can outperform 5-minute realised variance, which is to the author's knowledge the largest collection used in a single MCS implementation. In a collaborative effort, this number would increase rapidly: Laurent et al. (2012) alone compare 125 models, and while other papers use much smaller collections, the cumulative amount of specifications to test would be substantial.

This paper outlines and benchmarks an alternative algorithmic approach for determining the MCS of a collection of models which provides both a dramatic improvement in performance on large collections and allows the possibility of updating the MCS *ex post*

with further models. The intuition underpinning the approach is that rather than starting with the full collection of models and shrinking it down the the subset of models that form the MCS, the collection is initially made up of two models and MCS is gradually updated as models are added to the collection. In this manner, only deviations of the new model with respect to the existing collection need to be calculated and stored, rather than the full set of pairwise deviations, thus reducing the computational complexity of the MCS procedure by one polynomial order. Similarly, growing the collection of models rather than shrinking it intuitively allows for further models to be added to the collection at a later point in time.

The remainder of the paper is organised as follows: Section 2 describes the MCS approach and the existing elimination algorithm. The updating algorithm for obtaining the MCS is then presented in section 3 and the results of the benchmarking exercise are detailed in section 4 before section 5 concludes.

2 The Model Confidence Set elimination implementation

Before presenting the proposed MCS updating implementation it is important to briefly present the MCS approach and its existing implementation. The notation used below broadly follows Hansen et al. (2011), with a few modifications: n is used to index the N observations available in the data. Similarly, b will be used to index the B bootstrap resamples. The indexing notation for the M models to be compared is slightly more complex: i, j are used where necessary to index the models used in the calculation of the test statistics, while m is used specifically to identify the model examined in the current iteration of an algorithm.

The MCS procedure rests on the combination of an equivalence test statistic and an elimination rule, which are used to test the null hypothesis that all models currently in the set have equivalent loss, and to identify which model needs to be excluded should the null be rejected. The MCS is very a general approach in that many combinations of tests and rules can be chosen, however the focus here will be on the range rule (R), which (a) is typically used in existing MCS implementations and (b) possesses those interesting properties, discussed in section 3, which allow for fast updating to be performed.

Under the R rule the MCS procedure requires an $N \times M$ set of losses L, in order to calculate the mean loss $d_{i,j}$ of model i relative to model j:[2]

$$d_{i,j} = \frac{1}{N} \sum\nolimits_n \left(L_{n,i} - L_{n,j} \right) \tag{1}$$

These are used to calculate the following set of t-statistics under the null hypothesis that all pairwise deviations are zero-valued, i.e.$H_0 : d_{i,j} = 0$. The variance $\widehat{\text{var}}\left(d_{i,j}\right)$ is estimated using a bootstrapping procedure explained further below, using B resamples of the loss data L.

$$t_{i,j} = \frac{d_{i,j}}{\sqrt{\widehat{\text{var}}\left(d_{i,j}\right)}} \tag{2}$$

The R rule uses the following equivalence test statistic and elimination rule, which identifies the worst-performing model as the one having the largest pairwise t-statistic $t_{i,j}$, i.e. the one that has the highest average loss relative to any other model still in the set.

$$T_m = \max_{i \in \mathcal{M}} \max_{j \in \mathcal{M}} |t_{i,j}| \qquad\qquad e_m = \arg\max_{i \in \mathcal{M}} \sup_{j \in \mathcal{M}} t_{i,j} \tag{3}$$

As stated above, a bootstrapping procedure is used to test null hypothesis that all M models are in the MCS. Using a set of $N \times B$ bootstrap indexes $\mathcal{B}$ allows us to generate B re-sampled loss matrices $\mathcal{L}_{n,m,b}$ and pairwise deviation matrices $\delta_{i,j,b} = \text{mean}_n \left(\mathcal{L}_{n,i,b} - \mathcal{L}_{n,j,b} \right)$, which in turn provide the following bootstrapped t-statistics:[3]

$$\tau_{i,j,b} = \frac{\delta_{i,j,b} - d_{i,j}}{\sqrt{\text{var}_b\left(\delta_{i,j,b}\right)}} \tag{4}$$

These can be used to generate the bootstrapped distribution of t-statistics and p-values as follows, where $I(\dots)$ is the boolean indicator function:

[2]The choice of loss function to use is important for model selection, and the effect of that choice on the MCS procedure is discussed in Laurent et al. (2013). As is the case in Kevin Sheppard's MFE toolbox, however, we simply assume that a set of losses is available for the model forecasts.

[3]The implementation uses the bootstrap scheme proposed by Hansen et al. (2011) for reasons of comparability, however, alternative mechanisms such as the Politis and Romano (1994) stationary bootstrap can also be used.

$$\mathcal{T}_{m,b} = \max_{i \in \mathcal{M}} \max_{j \in \mathcal{M}} |\tau_{i,j,b}| \qquad P_m = \frac{1}{B} \sum\nolimits_b I\left(\mathcal{T}_{m,b} \geq T_m\right) \tag{5}$$

If $P_m > \alpha$ then all remaining models are deemed to be part of the MCS at confidence level $1 - \alpha$. If this is not the case, the null is rejected and e_m can be eliminated. It is important to point out that the MFE toolbox implementation, which is used as the benchmark, uses the following elimination rule where $d_i = \sum_{j \in \mathcal{M}} d_{i,j}$ and $\delta_{i,b} = \text{mean}_{j \in \mathcal{M}} \text{mean}_n \left(\mathcal{L}_{n,i,b} - \mathcal{L}_{n,j,b}\right)$ are average of the relative mean losses (1) over all the other models j still in the collection:

$$e_m = \arg\max_{i \in \mathcal{M}} \frac{d_i}{\sqrt{\text{var}_b\left(\delta_{i,b}\right)}} \tag{6}$$

As pointed out in Hansen et al. (2011, p. 466), (6) is actually the elimination rule used for the 'max' test statistic. However, the MFE toolbox implementation still uses test statistic (3) and critical values (5), and section 4 shows that this change in elimination rule makes little difference in the power of the procedure in practice. For the sake of clarity, rule (6) we will be referred to as the R_{max} rule in section 4.

Equations (1)-(5) form the basis of the elimination algorithm 1 for determining the MCS of a set of candidate models, presented in appendix A. Starting with the full set of candidate models M, each iteration m identifies the worst model e_m and corresponding test statistic T_m using the elimination rule (3) and calculates the bootstrapped distribution and p-value (4). If the null hypothesis is rejected at the $1 - \alpha$ level, the model e_m is removed from the set and the algorithm proceeds to the following iteration. This process continues until the null hypothesis ceases to be rejected. The elimination sequence can be stored in a vector e, which lists the models in the order in which they are are eliminated. This provides the ranking of the candidate models from worst to best in terms of their test statistic T_i. The reverse ranking of models (best to worst), which will be required later, is defined as r.

Each iteration of the R rule elimination algorithm requires finding the maximum value of the matrices of true and bootstrapped t-statistics $t_{i,j}$ and $\tau_{i,j,b}$. Given that the number of models in the confidence set starts at M and shrinks by one in each iteration, the total

number of operations required is proportional to the square pyramidal number $\sum_{i=1}^{M} i^2$, which implies a time complexity of $\mathcal{O}(M^3)$. Similarly, the implementation requires storing the bootstrapped t-statistics $\tau_{i,j,b}$ in a $M \times M \times B$ array, leading to a $\mathcal{O}(M^2)$ memory cost.

3 A fast updating implementation for the R elimination rule

The proposed updating implementation takes advantage of two important properties offered by the R rule (3) to improve both the time complexity and memory requirement of finding the MCS. The first is that the values of the pairwise t-statistics $t_{i,j}$ and $\tau_{i,j,b}$ underpinning T_m, e_m and $\mathcal{T}_{m,b}$ are independent of the order $e_1, e_2, ..., e_M$ in which models are eliminated, which is visible from the fact that in algorithm (1) both can are pre-computed before the start of the iterative elimination procedure. As models are eliminated from the MCS, T_m and $\mathcal{T}_{m,b}$ are drawn from successively smaller sub-matrices of the initial set of $t_{i,j}$ and $\tau_{i,j,b}$ statistics. Conceptually, this means that these initial sets of pairwise t-statistics for M models can themselves be considered as the submatrices of a larger set of $M+1$ models. This opens the door to the possibility of growing and updating the MCS as models are added rather than shrinking the MCS as models are eliminated.

The second property is that by construction the matrix of t-statistics (2) and (4) are skew symmetric, with $t_{i,j} = -t_{j,i}$ and $\tau_{i,j,b} = -\tau_{j,i,b}$. This implies that when an additional model m is added to the comparison set, only the vector of t-statistics pertaining to the deviations from m need be stored and processed rather than the full matrix of pairwise statistics. As will be shown below, this is what enables the reduction of both the time complexity and memory requirements by one polynomial degree.

3.1 Updating the equivalence test and elimination rule

The first task that needs to be carried out is to update the test statistic and elimination rule (3) as the set of candidate models increases from 2 to M, in order to produce the entries of the elimination vector e_i and corresponding test statistics T_i. This is done by initialising the vector of test statistics with $T_i = 0 \ \forall \ i$, then for $m = 2 \rightarrow M$, calculating the m-length vector of t-statistics t_i from the deviations $d_i = \text{mean}_n(L_{n,i} - L_{n,m})$ of all

past models i from the the current model m. This vector is used to update the test statistics T_i as follows:

$$\begin{cases} T_m = \max_i (-t_i) \quad \text{s.t.} \quad -t_{i*} > T_i* \\ T_i = \max (T_i, t_i) \quad \forall \ i < m : \ \ t_i > T_m \end{cases} \tag{7}$$

The first part of the rule uses the fact that $t_{j,i} = -t_{i,j}$ to identify the maximum t-statistic of current model m against the models that have already been processed. The constraint that $-t_{i*} > T_i*$ ensures that model $i*$, against which the maximum statistic for m is obtained, is itself ranked better than m and would therefore still be included in the set of models under the elimination rule (3). The second rule updates the t-statistics of the $m-1$ previously processed models. Again, the condition $t_i > T_m$ ensures that only those models ranked worse then the current model m can be updated with pairwise t-statistics calculated with respect to it, as under the elimination rule (3) these statistics would no longer be included in the set once m is eliminated.

Once all M models have been processed using the updating rules (7), the entries in the resulting vector T contain the largest deviation of a model i with respect to all the other j models ranked batter than itself, i.e. $T_i = \max_{j \in \mathcal{M}} t_{i,j}$. Sorting the test statistics T_i from largest to smallest value provides the elimination order e for the M models, or equivalently sorting from smallest to largest provides the model rankings r, which will play an important role in the updating of the the bootstrapped distribution below.

$$e, r \leftarrow \text{sortindex} (T_i) \tag{8}$$

For a given set of $N \times M$ losses L and boostrap indices $\mathcal{B}$, this updating procedure will provide the same rankings r_i and test statistics T_i as those obtained using the R rule (3) in the elimination algorithm 1. A similar updating scheme can be used to obtain the second element required for the MCS procedure, the bootstrapped distribution of t-statistics (5).

3.2 Updating the bootstrapped distribution

Updating the bootstrapped distribution (5) is more complicated than updating the rankings, as in certain circumstances the value of $\mathcal{T}_{m,b} = \max_{i \in \mathcal{M}} \max_{j \in \mathcal{M}} |\tau_{i,j,b}|$ can fall as the

rankings are updated using (7). In order to help illustrate this issue and the updating rules required to get around it, a simple example of the updating process for the bootstrapped distribution $\mathcal{T}_{m,b}$ for a given bootstrap replication b is provided in figure 1, which shows a situation where the vector of t-statistics $\tau_{i,m}$ from model $m = 6$ (in light grey) is added to an existing collection of five models.

Supposing for the moment that the sixth model enters the rankings without disturbing them, as shown in figure 1, then only two simple sets of updating rules are required. The first initialises a bootstrapped distribution $\mathcal{T}_{m,b}$ for the new model m, and the second updates the existing bootstrapped distributions of those past models ranked worse than m (models 1,3 and 5 here)[4]

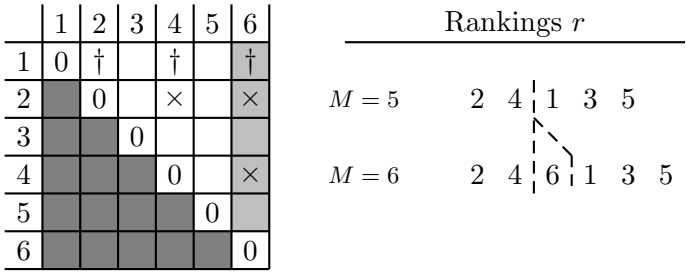


Figure 1: Updating rule for bootstrapped t-statistics

The first set of rules (9) creates a distribution $\mathcal{T}_{m,b}$ for the new model m. Let k be the ranking of the current model m, such that $r_k = m$. If the if the current model is ranked best ($k = 1$), one sets $\mathcal{T}_{m,b} = \ 0 \ \forall \ b$ and if $k > 1$ the bootstrapped distribution (5) is initialised as such:

$$\begin{cases} \tau_b^{max} = \max_{i \in \{r_1, \ldots, r_k\}} |\tau_{i,b}| \\ \mathcal{T}_{m,b} = \max\left(\tau_b^{max}, \mathcal{T}_{r_{k-1},b}\right) \end{cases} \qquad \forall \ b \tag{9}$$

In figure 1, the possible locations of $\mathcal{T}_{m,b} = \max_{i \in \mathcal{M}} \max_{j \in \mathcal{M}} |\tau_{i,j,b}|$ for $m = 6$ and $\mathcal{M} = \{2, 4, 6\}$ are marked with a '×' symbol.[5] The first part of the rule defines τ_b^{max} as the vector of running maxima of $\tau_{i,m}$ over the B bootstrap resamples, and in figure 1 this identifies which of the two '×' locations in the sixth column is largest. The second part of

[4]It should be obvious from the elimination rule (3) that the bootstrapped distributions $\mathcal{T}_{i,b}$ of those models ranked better than m do not require updating, as once m is eliminated from the confidence set, the information in $\tau_{i,m}$ is discarded.

[5]Because the B $|\tau_{i,j,b}|$ matrices are symmetric, the information in the dark grey area can be discarded.

the rule then selects $\mathcal{T}_{6,b}$ as the largest of either τ_b^{max} itself, or of $\mathcal{T}_{4,b}$, which is available in the white '$\times$' location.

The second set of rules is used to update the distributions $\mathcal{T}_{i,b}$ of the set of $m-k$ models that are ranked worse than the current model, i.e. $i \in \{r_{k+1}, ..., r_m\}$. For the case of model 1 in figure 1, this involves incorporating the $\tau_{i,j}$ values indicated with the '†' symbol. The first part of the rule updates the running maxima over $\tau_{i,m}$ by taking the maximum of either τ_b^{max} or the grey '†' and the second part of the rule then compares the value of the updated τ_b^{max} the the existing value of $\mathcal{T}_{1,b}$, which is already the maximum over the white '$\times$' and '†' locations.

$$\begin{cases} \tau_b^{max} = \max\left(\tau_b^{max}, |\tau_{r_{k+i},b}|\right) \\ \mathcal{T}_{r_{k+i},b} = \max\left(\tau_b^{max}, \mathcal{T}_{r_{k+i},b}\right) \end{cases} \qquad \forall\ b \tag{10}$$

Unfortunately, in general when the m^{th} model enters the rankings in k^{th} place, relative rankings in the set of worse models $\{r_{k+1}, ...r_m\}$ can change. As an example, suppose now that the introduction of model 6 in figure (1) switches the ranking of the three worse models as follows: $1, 3, 5 \rightarrow 5, 3, 1$. In the following discussion this will be refereed to as a 'ranking swap'. When this happens the second set of updating rules (10) will not produce the correct outcome, as the existing bootstrapped distributions $\mathcal{T}_{r_{k+i},b}$ no longer correctly reflect the model rankings prior to being updated with the current model. In our example for instance, $\mathcal{T}_{1,b}$ will tend to be undervalued: in the 5^{th} iteration it was calculated on the basis of $\mathcal{M} = \{2, 4, 1\}$, however because model 1 is now ranked last, $\mathcal{T}_{1,b}$ should reflect information from the full set of model $\mathcal{M} = \{2, 4, 5, 3\}$. Conversely, $\mathcal{T}_{5,b}$ will be overvalued, having used taken the maximum of $\tau_{i,j}$ over the full set of 5 models instead of the more restricted set $\mathcal{M} = \{2, 4, 5\}$.

Two strategies are proposed to get around ranking swaps without having to incur the time cost of recalculating the full pairwise bootstrapped distributions every time one occurs. The first is based on the observation that ranking swaps only affect worse ranked models, and are therefore not an issue if the current model always enter rankings in last place. In such a case, only the updating rule for the current model (9) needs to be used, as there are no worse models to update. This forms the basis of the fast updating algorithm 2, which calculates the model rankings (7) over the full collection of models in a first pass

before updating the bootstrapped distributions in a second pass, relying on the rankings obtained in the first pass to process the models from best to worst.

The fast updating algorithm 2 is designed to produce exactly the same result as the elimination algorithm 1 for a given set of losses L and bootstrap indices $\mathcal{B}$.[6] Importantly, however, each of the M iterations now only requires finding the maximum of the true and bootstrapped vectors of t-statistics t_i and $\tau_{i,b}$ with respect to the current model m rather than the full matrices of pairwise statistics $t_{i,j}$, $\tau_{i,j,b}$ for all models i and j still in the confidence set. The total number of operations required is therefore proportional to the triangular number $\sum_{i=1}^{M} i$, which leads to a smaller theoretical time complexity of $\mathcal{O}(M^2)$. Because there is now no need to store the full set of pairwise bootstrapped t-statistics $\tau_{i,j,b}$, the memory requirement is essentially dominated by the $B \times M$ bootstrapped distribution for each model $\mathcal{T}_{i,b}$, leading to a linear $\mathcal{O}(M)$ memory requirement.

3.3 Enabling incremental testing

While algorithm 2 allows for faster comparison and larger model sets than the elimination version 1, the fact it requires two passes on the Loss data to get around the problem of ranking swaps means it will also suffer from the same problem as the elimination algorithm 1, in that it cannot easily update a previously calculated MCS. Should one wish to add a handful of models to a previously processed collection of models, one would have no choice but to re-run the entire procedure on the larger collection.

The second strategy proposed for dealing with ranking swaps when updating the bootstrapped distribution uses a heuristic which specifically allows for such incremental updating of model collections. This heuristic, which forms the basis of the incremental algorithm 3, is based on the fact that while the updating rules (9) and (10) cannot identify the true value of $\mathcal{T}_{i,b}$ when a ranking swap occurs they can be modified to generate a reliable interval for the value, thus providing a 'best guess' for $\mathcal{T}_{i,b}$.

In practice, once the model rankings have been updated using (7) the algorithm identifies the location of any ranking swaps amongst models that are ranked worse than the current model $m = r_k$, i.e. $\{r_{k+1}, ...r_m\}$. If none have occurred, then rule (10) can safely

[6] This is illustrated in a demonstration file (`demo_update.m`) provided in the supplementary material, which compares the output of the elimination and updating algorithms for identical losses and bootstrap indices.

be applied on all models. If some ranking swaps have occurred, the swap range is identified as the location of the first affected model up to the location of the last affected model. Rule (10) is applied on those models outside the affected range and the heuristic rule (11) below is applied to those within the range.

$$\begin{cases} \tau_b^{max} = \max\left(\tau_b^{max}, |\tau_{r_{k+i},b}|\right) \\ \underline{\mathcal{T}}_{r_{k+i},b} = \max\left(\tau_b^{max}, \mathcal{T}_{r_{k+i-1},b}\right) \quad \forall\ b \\ \overline{\mathcal{T}}_{r_{k+i},b} = \max\left(\underline{\mathcal{T}}_{r_{k+i},b}, \mathcal{T}_{r_{k+i},b}\right) \end{cases} \tag{11}$$

The first part of the rule simply updates the running maximum τ_b^{max} over the $\tau_{i,b}$ bootstrapped t-statistics, as was the case for updating rules (9) and (10). The second part corresponds to (9) and treats model r_{k+i} as if it were a new addition to the collection, by disregarding the existing (and incorrect) value of $\mathcal{T}_{r_{k+i},b}$. Because this information is not included, the procedure either produces the correct value (if it happens that the maximum is in fact located in τ_b^{max}) or an incorrect but lower value that can be used as a lower bound for the heuristic. The third part corresponds to (10) combined with a constraint that under rule (5) the values of the bootstrapped distribution $\mathcal{T}_{i,b}$ cannot decrease as the models get worse. This can be written as $\overline{\mathcal{T}}_{r_{k+i},b} = \max\left(\mathcal{T}_{r_{k+i-1},b}, \tau_b^{max}, \mathcal{T}_{r_{k+i},b}\right)$, which can be rearranged into the specification in (11) to generate the upper value for the interval. The updated value for the bootstrapped distribution of each model is then simply set to the midpoint of the interval, i.e. $\mathcal{T}_{r_{k+i},b} = \frac{1}{2}\left(\overline{\mathcal{T}}_{r_{k+i},b} + \underline{\mathcal{T}}_{r_{k+i},b}\right)$.

This heuristic is unable to systematically find the true value of $\mathcal{T}_{r_{k+i},b}$ for those models r_{k+i}, affected by a ranking swap, nevertheless it should provide good performance. The main reason is that the updating of the bootstrapped distributions is an iterative process and ranking swaps will not systematically affect all models on any given iteration, therefore all that is required is that the heuristic track the true value of $\mathcal{T}_{r_{k+i},b}$ for models affected by ranking swaps well enough to 'patch the gap' so that they can be correctly updated by (10) in those later iterations where they are unaffected. The heuristic (11) will occasionally provide the correct value[7] and is also anchored to the true $\mathcal{T}_{r_{k+i},b}$ values either side of the range of ranking swaps by the use of the correct rule (10). Combined with the fact that (11) provides values of $\mathcal{T}_{r_{k+i},b}$ that are non-decreasing in model rankings r_{k+i}, as required by

[7]This will happen in (11) when the true value is located in τ_b^{max} and $\underline{\mathcal{T}}_{r_{k+i},b} > \mathcal{T}_{r_{k+i},b}$ in the third line.

(5), the heuristic is expected to provide this good tracking performance. Finally, because the heuristic (11) produces a value equal to, smaller, or larger than the correct value of $\mathcal{T}_{r_{k+i},b}$, the bootstrapped nature of the MCS procedure also ensures that when averaged over B resamples, the p-values (5) are close to the correct ones even when models affected by ranking swaps are not subsequently updated.

The use of the heuristic (11) enables each model to be processed in a single iteration, resulting in the incremental algorithm 3, in appendix. The main implication of this single-pass updating is that the MCS procedure need not be carried out in one go: a subset of the full collection can be processed at a given point in time, with the rest of the models added later on.[8] As is the case for the White (2000) RC test, this theoretically allows for collaborative research and incremental testing of models. As for the RC test, some information needs to be made available for this to be possible. Specifically, researchers attempting to compare their models to a pre-existing collection would require the corresponding model losses L, bootstrap indices $\mathcal{B}$ (or equivalently the seed of a random number generator if the bootstrap method is known), the test statistics T and the bootstrap distributions $\mathcal{T}$.

4 Monte Carlo benchmarking

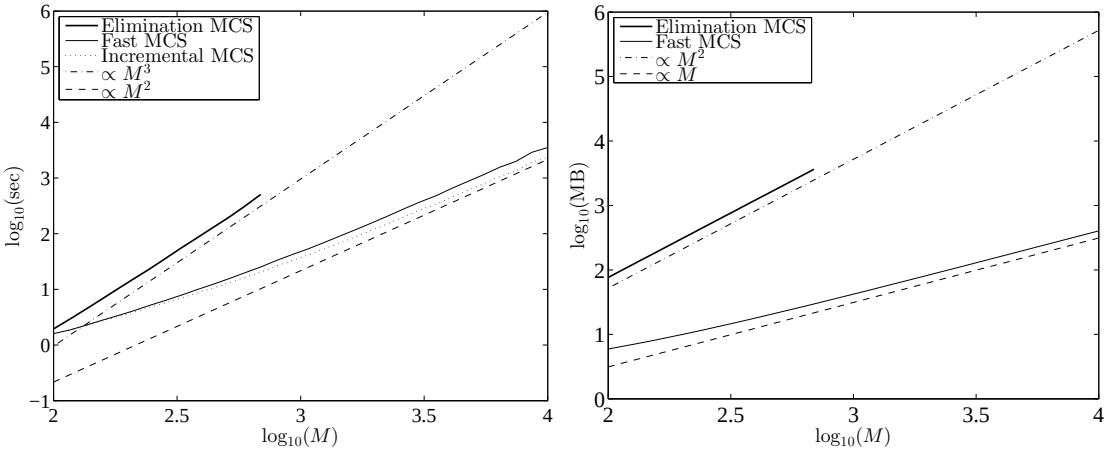
A series of benchmarking exercises are carried out on the algorithms 1 - 3 in order to confirm that their time complexity and memory costs are in line with what is expected, as well as to check that their power corresponds to the findings of Hansen et al. (2011).[9] In order to ensure comparability, the Monte Carlo framework follows the one used in Hansen et al. (2011). A synthetic $N \times M$ matrix of losses L is generated as follows:

$$L_{n,i} = \theta_i + \frac{a_n}{\sqrt{E\left(a_n^2\right)}} X_{n,i} \tag{12}$$

Where the individual components are given by:

[8] An illustration is provided in the demonstration file `demo_increment.m` in which a collection of models is processed in two separate runs, with the MCS outcome being compared to that of algorithm 2.

[9] All the benchmarking work was carried out on a 32-worker cluster made up of 8 machines with the following specifications: An ASUS B85M-G motherboard with a Core i5 4670 (3.40GHz) processor and 16 GB (DDR3 - 1600 MHz) of memory. The MATLAB code used is available as supplementary material from the author.



(a) Time complexity (b) Memory requirement

Figure 2: MCS implementation benchmarking

$$\begin{cases} \theta_i = \frac{\lambda}{\sqrt{N}} \left(0, \frac{1}{M-1}, ..., \frac{M-2}{M-1}, 1\right) \\ a_n = \exp\left(y_n\right) \\ y_n = -\frac{\varphi}{2(1+\varphi)} + \varphi y_{n-1} + \sqrt{\varphi}\varepsilon_n \end{cases} \tag{13}$$

The free parameters are λ, which controls the dispersion of the mean loss θ_i of a given model, ρ which controls the degree of correlation of losses across models and φ which allows for the possibility of conditional heteroskedasticity in the losses. The two stochastic elements are $X_{n,i} \underset{iid}{\sim} \mathrm{N}_M\left(0, \Sigma\right)$, where the diagonal elements of the variance covariance matrix Σ are equal to one, with all the off-diagonal elements set to $\rho \in [0, 1]$ and $\varepsilon_n \underset{iid}{\sim} \mathrm{N}\left(0, 1\right)$. Finally, each time a loss matrix is generated using (12) and (13) the M columns are randomly shuffled in order to ensure that models are not processed in order of worsening performance. This is important for testing the heuristic (11), as the ranking swaps which can affect the bootstrapped distribution do not occur if models are processed from best to worse.

The first Monte Carlo exercise is a simple replication of the Hansen et al. (2011, p. 478) symmetric loss setting, which aims to establish that the R_{max} elimination algorithms from the MFE toolbox and the R elimination algorithm 1 have the expected power and size, and have therefore been implemented correctly. The choice of parameters is therefore similar to Hansen et al. (2011), in that we set $M = 100$, $N = 250$, $\lambda \in \{5, 10, 20, 40\}$,

$\rho \in \{0, 0.5, 0.75, 0.95\}$, $\varphi \in \{0, 0.5, 0.8\}$ with a bootstrap block of $l = 2$ with $B = 1000$ resamples. Table 1 in appendix B shows that the MFE toolbox and algorithm 1 implementations of the MCS replicate the size and power characteristics in table (II) of Hansen et al. (2011). While the MFE toolbox R_{max} rule (6) has slightly more power than the R rule (3) on models with low dispersion λ and low correlation ρ, their general performance is comparable.

The main benchmarking exercise is carried out by measuring the running time and memory requirements of algorithms 1 - 3 for 32 values of M such that $\log_{10}(M)$ is evenly spaced over $[2, 4]$ with N, l and B unchanged. The values of λ, ρ and ϕ do not matter here as for the purpose of comparing the worst case runtime performance all three algorithms examine all M models regardless of their accuracy. Figure 2 shows the average running time and memory consumption over 32 separate runs. As expected, the time complexity of the R rule elimination algorithm 1 is $\mathcal{O}\left(M^3\right)$, with a memory requirement of $\mathcal{O}\left(M^2\right)$. In fact, as is visible in figure 2 this exercise had to be stopped at $M = 690$ models for algorithm 1, as by this point the 3.6 GB memory requirement was reaching the maximum manageable level of 4 GB per cluster worker.[10] It also confirms that the updating algorithms 2 and 3 both have $\mathcal{O}\left(M^2\right)$ time complexity and $\mathcal{O}\left(M\right)$ memory requirement. Finally, as expected, because the incremental algorithm 3 only requires a single pass on the loss data $L_{n,m}$, it is slightly faster than the updating algorithm 2 which requires one pass to establish the rankings r and another to obtain the bootstrapped distribution $\mathcal{T}$.

The final exercise evaluates the power of the updating algorithms on large scale model comparisons, with $M \in \{1000, 2500, 5000\}$, in order to ensure that the effectiveness of the R rule remains when the size of model collections become increasingly large. Because of the increased time required for running 1000 Monte Carlo replications of these large model collections, the parameter space is reduced to $\lambda \in \{10, 20, 40\}$, $\rho \in \{0.5, 0.75, 0.95\}$ with $\varphi = 0.5$. The bootstrap settings remain the same with $l = 2$ and $B = 1000$, as does the number of observations $N = 250$. The results of this analysis, shown in table 2, confirm that the size test is unaffected and that although the power of the R rule falls slightly as collection size increases, as expected, the values of the test remain comparable to the

[10] Larger values of M could have been run by using virtual memory to supplement the RAM but the resulting paging would have biased the measurement of the time complexity.

corresponding entries in table 1. A second result of interest is that the power statistics of the updating algorithm 2 and the incremental algorithm 3 are very similar. Furthermore, the mean difference in p-values across the algorithms is around 10^{-3} and is smaller in magnitude than the standard deviation of the difference in p-values, suggesting that the heuristic updating rule (11) is reliable, even when models are processed in a completely random order.

5 Conclusion

The main result of this paper is that one can obtain the confidence set of a collection of models using the R rule with an updating approach that has a smaller time complexity and memory requirement than the existing elimination approach, allowing for faster comparisons and larger collections. The first proposed version of the methodology is a two-pass algorithm that replicates exactly the elimination implementation of the R rule for a given set of bootstrap indices, but offers one polynomial order less time and memory cost. As an illustration of the resulting performance gain, finding the MCS of a collection of models similar to Liu et al. (2015) (600 models) using their bootstrap settings takes the elimination implementation 5.5 minutes and requires 2.75 GB of memory. The fast updating algorithm can carry out the same task in 28 seconds with only 53.5 MB of memory. Pushing the comparison to the extreme, finding the MCS of a collection of 5000 models would take the fast algorithm around 15 minutes using 200 MB of memory, while extrapolating from the $\mathcal{O}\left(M^3\right)$ time complexity and $\mathcal{O}\left(M^2\right)$ memory requirement suggests the elimination algorithm would require 53 hours and an untractable 186 GB to provide the same result.

Because large model collections would probably result from a gradual accretion over time, as newly developed models or forecasts are compared to past methods, a second version is proposed which allows for incremental and collaborative testing. This is achieved by using a single-pass algorithm in which model rankings and p-values are updated simultaneously when a model is added to the collection. While this single pass incremental version allows *ex post* updating and is faster than the two-pass updating algorithm, it does come at the cost of ability to replicate *exactly* the p-values of the elimination algorithm

for a given set of bootstrap indices. This is because the ranking of the existing models in the collection can change when new models are added, requiring the use of a heuristic in order to be able to update the bootstrapped distribution of t-statistics. The Monte-Carlo analysis confirms that this heuristic performs well, providing very similar p-values to the updating algorithm, suggesting that this tradeoff between exact replication of the elimination algorithm and the ability to add models *ex post* is acceptable.

Finally, given the order of magnitude gap in the collection sizes used in this paper compared to those in the existing literature, and given that the time and memory costs of the elimination algorithm are perfectly acceptable for small to medium collections, it is important to comment on the practical usefulness of these updating algorithms. The first and immediate benefit is of course the gain in speed illustrated above. The wider aspiration, similar to what was suggested by White (2000) for the reality check, is to facilitate future collaborative efforts over very large scale model collections. While clearly there are other obstacles to such large scale collaborative efforts, providing a methodology that enables these efforts in the first place is an important step.

References

Amado, Cristina and Timo Teräsvirta (2014) "Conditional Correlation Models of Autoregressive Conditional Heteroscedasticity With Nonstationary GARCH Equations," *Journal of Business & Economic Statistics*, Vol. 32, pp. 69–87.

Bernardi, Mauro and Leopoldo Catania (2014) "The Model Confidence Set package for R," *arXiv:1410.8504*.

Boudt, Kris, Jon Danielsson, and Sébastien Laurent (2013) "Robust forecasting of dynamic conditional correlation GARCH models," *International Journal of Forecasting*, Vol. 29, pp. 244–257.

Caporin, Massimiliano and Michael McAleer (2014) "Robust ranking of multivariate GARCH models by problem dimension," *Computational Statistics & Data Analysis*, Vol. 76, pp. 172–185.

Corradi, Valentina and Norman R Swanson (2013) "A survey of recent advances in forecast

accuracy comparison testing, with an extension to stochastic dominance," in *Recent Advances and Future Directions in Causality, Prediction, and Specification Analysis*, pp. 121–143.

Diebold, Francis X and Roberto S Mariano (1995) "Comparing predictive accuracy," *Journal of Business & Economic Statistics*, Vol. 13, pp. 134–144.

Ferrari, Davide and Yuhong Yang (2015) "Confidence sets for model selection by F-testing," *Statistica Sinica*, Vol. Forthcoming.

Hamid, Alain (2014) "Prediction power of high-frequency based volatility measures: a model based approach," *Review of Managerial Science*, Vol. 9.

Hansen, Peter R., Asger Lunde, and James M. Nason (2011) "The Model Confidence Set," *Econometrica*, Vol. 79, pp. 453–497.

Hansen, Peter Reinhard (2005) "A test for superior predictive ability," *Journal of Business & Economic Statistics*, Vol. 23, pp. 365–380.

Hansen, Peter Reinhard, Asger Lunde, and Valeri Voev (2014) "Realized beta GARCH: a multivariate GARCH model with realized measures of volatility," *Journal of Applied Econometrics*, Vol. 29, pp. 774–799.

Iltuzer, Zeynep and Oktay Tas (2013) "The Forecasting Performances of Volatility Models in Emerging Stock Markets: Is a Generalization Really Possible?" *Journal of Applied Finance and Banking*, Vol. 3, pp. 49–73.

Laurent, Sébastien, Jeroen VK Rombouts, and Francesco Violante (2012) "On the forecasting accuracy of multivariate GARCH models," *Journal of Applied Econometrics*, Vol. 27, pp. 934–955.

——— (2013) "On loss functions and ranking forecasting performances of multivariate volatility models," *Journal of Econometrics*, Vol. 173, pp. 1–10.

Liu, Lily, Andrew J. Patton, and Kevin Sheppard (2015) "Does Anything Beat 5-Minute RV? A Comparison of Realized Measures Across Multiple Asset Classes," *Journal of Econometrics*, Vol. 187, pp. 293–311.

Neumann, Michael and George Skiadopoulos (2013) "Predictable dynamics in higher-order risk-neutral moments: Evidence from the S&P 500 options," *Journal of Financial and Quantitative Analysis*, Vol. 48, pp. 947–977.

Patton, Andrew J., , and Kevin Sheppard (2009) *Handbook of Financial Time Series*, Chap. Evaluating Volatility and Correlation Forecasts: Oxford University Press.

Politis, Dimitris N. and Joseph P. Romano (1994) "The Stationary Bootstrap," *Journal of the American Statistical Association*, Vol. 89, pp. 1303–1313.

White, Halbert (2000) "A Reality Check For Data Snooping," *Econometrica*, Vol. 68, pp. 1097–1126.

Wilhelmsson, Anders (2013) "Density Forecasting with Time-Varying Higher Moments: A Model Confidence Set Approach," *Journal of Forecasting*, Vol. 32, pp. 19–31.

A Algorithms

Algorithm 1 MCS elimination algorithm with R rule

Require: L: $N \times M$ matrix of losses
Require: $\mathcal{B}$: $N \times B$ matrix of bootstrap indexes
Require: α: cutoff criterion of the MCS

Part 1: preprocess data
$\mathcal{L} \leftarrow L(\mathcal{B})$ ▷ Bootstrap resamples
$d_{i,j} \leftarrow \text{mean}_n(L_{n,i} - L_{n,j}) \quad \forall\ i,j$
$\delta_{i,j,b} \leftarrow \text{mean}_n(\mathcal{L}_{n,i,b} - \mathcal{L}_{n,j,b}) \quad \forall\ i,j,b$
$t_{i,j} \leftarrow d_{i,j} / \sqrt{\text{var}_b(\delta_{i,j,b})}$
$\tau_{i,j,b} \leftarrow (\delta_{i,j,b} - d_{i,j}) / \sqrt{\text{var}_b(\delta_{i,j,b})}$ ▷ δ can be deleted

Part 2: iterative elimination
for $m = 1 \rightarrow M - 1$ **do**
  $T \leftarrow \max_{i \in \mathcal{M}} \max_{j \in \mathcal{M}} |t_{i,j}|$ ▷ Get test statistic
  $e_m \leftarrow \arg\max_{i \in \mathcal{M}} \sup_{j \in \mathcal{M}} t_{i,j}$ ▷ Identify worst model
  $\mathcal{T}_b \leftarrow \max_{i \in \mathcal{M}} \max_{j \in \mathcal{M}} |\tau_{i,j,b}| \quad \forall\ b$ ▷ Bootstrapped statistic
  $P_m \leftarrow \max\left(\sum_b (\mathcal{T}_b > T)/B, P_{m-1}\right)$ ▷ Non-decreasing p-values
  $\mathcal{M} \leftarrow \mathcal{M} \setminus e_m$ ▷ Remove worst model
end for
$m* \leftarrow \arg\min_m (P_m > \alpha)$
$\mathcal{M}_{excl} \leftarrow \{e_1, e_2, ..., e_{m*}\}$ ▷ Excluded models
$\mathcal{M}_{1-\alpha} \leftarrow \{e_{m*+1}, e_{m*+2}, ..., e_M\}$ ▷ Confidence set
return P, $\mathcal{M}_{excl}$, $\mathcal{M}_{1-\alpha}$

Algorithm 2 MCS fast updating algorithm with R rule

Require: L: $N \times M$ matrix of losses
Require: $\mathcal{B}$: $N \times B$ matrix of bootstrap indexes
Require: α: cutoff criterion of the MCS

Pass 1: update rankings
for $m = 2 \rightarrow M$ **do**
  $d_i \leftarrow \text{mean}_n \left(L_{n,i} - L_{n,m} \right) \quad \forall \ i \leq m$
  $\mathcal{L}_{i,b} \leftarrow L_i \left(\mathcal{B} \right)$ ▷ Bootstrap resamples
  $\delta_{i,b} \leftarrow \text{mean}_n \left(\mathcal{L}_{i,b} - \mathcal{L}_{m,b} \right) \quad \forall \ b, i \leq m$
  $t_i \leftarrow d_i / \sqrt{\text{var}_b \left(\delta_{i,b} \right)}$
  $T_m \leftarrow \max_i \left(-t_i \right) \quad \text{s.t.} \ -t_{i*} > T_i*$
  $T_i \leftarrow \max \left(T_i, t_i \right) \quad \forall \ i < m: \ t_i > T_m$
end for
$r \leftarrow \text{sort} \left(T \right)$ ▷ Sort best to worst
$e \leftarrow \text{sort} \left(T \right)$ ▷ Sort worst to best

Pass 2: update distributions
for $m = 2 \rightarrow M$ **do**
  $\delta_{i,b} \leftarrow \text{mean}_n \left(\mathcal{L}_{i,b} - \mathcal{L}_{m,b} \right) \quad \forall \ b, i \leq m$
  $\tau_{i,b} \leftarrow \left(\delta_{i,b} - d_i \right) / \sqrt{\text{var}_b \left(\delta_{i,b} \right)} \quad \forall \ b, i \leq m$
  $\tau_b^{max} \leftarrow \max_{i \leq m} \left| \tau_{i,b} \right| \quad \forall \ b$ ▷ Current maximum
  $\mathcal{T}_{m,b} \leftarrow \max \left(\mathcal{T}_{r_{m-1},b}, \tau_b^{max} \right) \quad \forall \ b$ ▷ Update statistic
end for
$P_i \leftarrow \sum_b \left(\mathcal{T}_{i,b} > T_i \right) / B \quad \forall \ i$ ▷ Get p-values
$m* \leftarrow \arg\min_m \left(P_m > \alpha \right)$
$\mathcal{M}_{excl} \leftarrow \{ e_1, e_2, ..., e_{m*} \}$ ▷ Excluded models
$\mathcal{M}_{1-\alpha} \leftarrow \{ e_{m*+1}, e_{m*+2}, ..., e_M \}$ ▷ Confidence set
return P, $\mathcal{M}_{excl}$, $\mathcal{M}_{1-\alpha}$, T (optional), $\mathcal{T}$ (optional)

Algorithm 3 MCS incremental updating algorithm with R rule

Require: L: $N \times M$ matrix of losses
Require: $\mathcal{B}$: $N \times B$ matrix of bootstrap indexes
Require: α: cutoff criterion of the MCS
Require: T_0: M_0 vector of previous scores (optional)
Require: $\mathcal{T}_0$: $B \times M_0$ matrix of previous bootstrapped statistics (optional)

if $M_0 = 0$ **then**
  $\rho \leftarrow 1$ ▷ Initialise past ranking
else
  $\rho \leftarrow \text{sort}\,(T)$ ▷ Sort past models best to worst
end if
for $m = M_0 + 1 \rightarrow M$ **do**
  $d_i \leftarrow \text{mean}_n\,(L_{n,i} - L_{n,m}) \quad \forall\ i \leq m$
  $\mathcal{L}_{i,b} \leftarrow L_i\,(\mathcal{B}) \quad \forall\ b, i \leq m$ ▷ Bootstrap resamples
  $\delta_{i,b} \leftarrow \text{mean}\,(\mathcal{L}_{i,b} - \mathcal{L}_{m,b}) \quad \forall\ b, i \leq m$
  $t_i \leftarrow d_i / \sqrt{\text{var}_b\,(\delta_{i,b})}$
  $T_m \leftarrow \max_i\,(-t_i) \quad \text{s.t.} \quad -t_{i*} > T_i*$
  $T_i \leftarrow \max\,(T_i, t_i) \quad \forall\ i < m: \ \ t_i > T_m$
  $r \leftarrow \text{sort}\,(T)$ ▷ Sort best to worst
  $k, s \leftarrow \text{compare}(r_i, \rho_i)$ ▷ Get rank of m & swap range
  $\tau_{i,b} \leftarrow (\delta_{i,b} - d_i) / \sqrt{\text{var}_b\,(\delta_{i,b})} \quad \forall\ b, i \leq m$
  $\tau_b^{max} \leftarrow \max_{i \in \{r_1, \ldots, r_k\}} |\tau_{i,b}| \quad \forall\ b$ ▷ Current maximum
  $\mathcal{T}_{m,b} \leftarrow \max\left(\mathcal{T}_{r_{k-1},b}, \tau_b^{max}\right) \quad \forall\ b$ ▷ Statistics for current model
  for $j = k + 1 \rightarrow m$ **do**
    $\tau_b^{max} \leftarrow \max\left(\tau_b^{max}, |\tau_{r_j,b}|\right) \quad \forall\ b$ ▷ Update running maximum
    if $s_j = 1$ **then** ▷ In the swap range
      $\underline{\mathcal{T}}_{r_j,b} \leftarrow \max\left(\tau_b^{max}, \mathcal{T}_{r_{j-1},b}\right) \quad \forall\ b$ ▷ use heuristic
      $\overline{\mathcal{T}}_{r_j,b} \leftarrow \max\left(\underline{\mathcal{T}}_{r_j,b}, \mathcal{T}_{r_j,b}\right) \quad \forall\ b$
      $\mathcal{T}_{r_{k+j},b} \leftarrow 0.5 \times \left(\underline{\mathcal{T}}_{r_j,b} + \overline{\mathcal{T}}_{r_j,b}\right) \quad \forall\ b$
    else
      $\mathcal{T}_{r_j,b} \leftarrow \max\left(\mathcal{T}_{r_j,b}, \tau_b^{max}\right) \quad \forall\ b$ ▷ Normal updating
    end if
  end for
  $\rho \leftarrow r$ ▷ Save rankings
end for
$e \leftarrow \text{sort}\,(T)$ ▷ Sort worst to best
$P_i \leftarrow \sum_b\,(\mathcal{T}_{i,b} > T_i)\,/B \quad \forall\ i$ ▷ Get p-values
$m* \leftarrow \arg\min_m\,(P_m > \alpha)$
$\mathcal{M}_{excl} \leftarrow \{e_1, e_2, ..., e_{m*}\}$ ▷ Excluded models
$\mathcal{M}_{1-\alpha} \leftarrow \{e_{m*+1}, e_{m*+2}, ..., e_M\}$ ▷ Confidence set
return P, $\mathcal{M}_{excl}$, $\mathcal{M}_{1-\alpha}$, T (optional), $\mathcal{T}$ (optional)

B Monte Carlo benchmarking

Table 1: Monte Carlo analysis of MCS elimination algorithms

	ρ, R_{max} elimination rule				ρ, R elimination rule			
λ	0	0.5	0.75	0.95	0	0.5	0.75	0.95
$\varphi = 0$								
Size test: frequency at which $\mathcal{M}^* \in \hat{\mathcal{M}}_{90\%}$								
5	1.000	0.997	0.999	0.994	1.000	0.998	1.000	0.994
10	0.996	0.997	0.993	0.994	0.997	0.997	0.993	0.994
20	0.997	0.998	0.996	0.996	0.997	0.998	0.996	0.996
40	0.992	0.992	0.994	1.000	0.992	0.992	0.994	1.000
Power test: Average number of models in $\hat{\mathcal{M}}_{90\%}$								
5	74.875	53.517	37.852	16.412	77.717	55.651	39.042	16.593
10	37.154	26.214	18.390	8.041	38.389	26.758	18.618	8.057
20	18.556	12.880	9.068	3.928	18.807	12.994	9.099	3.929
40	8.937	6.066	4.283	1.979	8.974	6.071	4.283	1.979
$\varphi = 0.5$								
Size test: frequency at which $\mathcal{M}^* \in \hat{\mathcal{M}}_{90\%}$								
5	0.997	0.996	0.992	0.997	1.000	0.998	0.997	0.998
10	0.998	0.991	0.990	0.993	1.000	0.995	0.993	0.993
20	0.998	0.996	0.995	0.995	1.000	0.997	0.995	0.995
40	0.996	0.992	0.995	0.998	0.998	0.992	0.995	0.998
Power test: Average number of models in $\hat{\mathcal{M}}_{90\%}$								
5	64.522	47.100	33.215	15.257	74.313	53.053	36.539	15.931
10	33.597	23.832	16.741	7.634	37.192	25.673	17.651	7.790
20	16.805	12.094	8.489	3.810	17.830	12.475	8.668	3.819
40	8.505	5.880	4.243	1.974	8.708	5.934	4.256	1.976
$\varphi = 0.8$								
Size test: frequency at which $\mathcal{M}^* \in \hat{\mathcal{M}}_{90\%}$								
5	0.975	0.982	0.992	0.993	1.000	1.000	1.000	1.000
10	0.986	0.984	0.992	0.995	1.000	1.000	0.999	0.997
20	0.994	0.993	0.991	0.994	0.999	0.999	0.995	0.994
40	0.998	0.995	0.997	1.000	0.999	0.995	0.997	1.000
Power test: Average number of models in $\hat{\mathcal{M}}_{90\%}$								
5	45.283	33.626	25.934	13.065	64.170	45.117	32.441	14.723
10	25.025	18.837	14.060	6.762	31.926	22.639	16.038	7.149
20	13.651	10.217	7.265	3.447	15.997	11.310	7.770	3.505
40	7.461	5.168	3.925	1.907	7.936	5.416	4.025	1.913

The size and power averages are calculated over 1000 Monte Carlo replications.
The R_{max} elimination rule (6) is the method provided in the MFE toolbox, while the R rule corresponds to (3).

Table 2: Monte Carlo analysis of fast algorithms on large scale collections

	ρ, $M = 1000$			ρ, $M = 2500$			ρ, $M = 5000$		
λ	0.5	0.75	0.95	0.5	0.75	0.95	0.5	0.75	0.95
Updating algorithm									
Size test: frequency at which $\mathcal{M}^* \in \hat{\mathcal{M}}_{90\%}$									
10	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
20	0.999	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
40	1.000	0.998	0.998	1.000	1.000	1.000	1.000	1.000	0.999
Power test: Average number of models in $\hat{\mathcal{M}}_{90\%}$ as a percentage of M									
10	27.952	19.482	8.634	28.570	19.984	8.719	28.830	20.207	8.933
20	13.721	9.574	4.199	14.061	9.885	4.332	14.302	10.015	4.412
40	6.654	4.726	2.056	6.872	4.812	2.101	7.039	4.958	2.160
Incremental algorithm									
Size test: frequency at which $\mathcal{M}^* \in \hat{\mathcal{M}}_{90\%}$									
10	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
20	0.999	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
40	1.000	0.998	0.998	1.000	1.000	1.000	1.000	1.000	0.999
Power test: Average number of models in $\hat{\mathcal{M}}_{90\%}$ as a percentage of M									
10	28.003	19.505	8.618	28.651	20.043	8.735	28.913	20.268	8.958
20	13.720	9.562	4.175	14.097	9.902	4.328	14.341	10.041	4.420
40	6.637	4.701	2.029	6.880	4.810	2.090	7.057	4.967	2.157
Heuristic diagnostics									
Mean difference in bootstrapped P-values ($\times 10^{-3}$)									
10	1.592	0.993	0.088	1.671	1.266	0.505	1.495	1.144	0.537
20	0.466	0.174	−0.182	0.817	0.570	0.102	0.825	0.566	0.234
40	−0.014	−0.184	−0.197	0.351	0.140	−0.061	0.425	0.265	0.057
Standard deviation of differences in bootstrapped P-values ($\times 10^{-3}$)									
10	2.582	1.910	0.797	2.000	1.811	0.914	1.626	1.331	0.743
20	1.293	0.880	0.367	1.315	1.005	0.397	1.146	0.795	0.408
40	0.545	0.337	0.200	0.783	0.450	0.149	0.636	0.473	0.207

The size and power averages are calculated over 1000 Monte Carlo replications.
For the power tests, the number of models in $\hat{\mathcal{M}}_{90\%}$ is normalised by M to make the numbers comparable to table 1.
The third section provides the mean and standard deviations of the difference between the bootstrapped P-values of the fast and incremental algorithms 2 and 3.

Recent Kent Discussion Papers in Economics

15/18: 'Trend Dominance in Macroeconomic Fluctuations', Katsuyuki Shibayama

15/17: 'Efficient estimation with many weak instruments using regularization techniques', Marine Carrasco and Guy Tchuente

15/16: 'High school human capital portfolio and college outcomes', Guy Tchuente

15/15: 'Regularized LIML for many instruments, Marine Carrasco and Guy Tchuente

15/14: 'Agglomeration Economies and Productivity Growth: U.S. Cities, 1880-1930', Alexander Klein and Nicholas Crafts

15/13: 'Microcredit with Voluntary Contributions and Zero Interest Rate - Evidence from Pakistan', Mahreen Mahmud

15/12: 'Act Now: The Effects of the 2008 Spanish Disability Reform', Matthew J. Hill, Jose Silva and Judit Vall

15/11: 'Testing for Level Shifts in Fractionally Integrated Processes: a State Space Approach', Davide Delle Monache, Stefano Grassi and Paolo Santucci de Magistris

15/10: 'German Wage Moderation and European Imbalances: Feeding the Global VAR with Theory', Timo Bettendorf and Miguel A. León-Ledesma

15/09: 'Repaying Microcredit Loans: A Natural Experiment on Liability Structure', Mahreen Mahmud

15/08: 'Fundamental shock selection in DSGE models', Filippo Ferroni, Stefano Grassi and Miguel A. León-Ledesma

15/07: 'Direct calibration and comparison of agent-based herding models of financial markets', Sylvain Barde

15/06: 'Efficiency in a forced contribution threshold public good game', Edward Cartwright and Anna Stepanova

15/05: 'Military Aid, Direct Intervention and Counterterrorism', María D.C. García-Alonso, Paul Levine and Ron Smith

15/04: 'A Practical, Universal, Information Criterion over Nth Order Markov Processes', Sylvain Barde

15/03: 'Public Good Provision in Indian Rural Areas: the Returns to Collective Action by Microfinance Groups', Paolo Casini, Lore Vandewalle and Zaki Wahhaj

15/02: 'Fiscal multipliers in a two-sector search and matching model', Konstantinos Angelopoulos, Wei Jiang and James Malley