

Model Confidence Set for Stata

Martin Fankhauser* Jakub Przewoski[†] Gabriele Nespoli[‡]

September 2026

Abstract

This article introduces `modelconfset`, a Stata command for implementing the model confidence set procedure of Hansen, Lunde, and Nason (2011). The procedure starts from a user-specified set of competing models and returns a subset that contains the best model, or models, with a chosen level of confidence, where best is defined by a user-supplied loss function. Unlike procedures that select a single winning model, the model confidence set explicitly recognizes that the data may not be informative enough to separate closely performing alternatives. The command works directly with loss variables stored in a Stata dataset, supports different test statistics, implements bootstrap inference, reports the sequential elimination path, stores the resulting superior set, and generates forecast combinations if model forecasts are supplied. We illustrate the syntax and output using reproducible examples and validate the implementation against existing results and simulations.

Keywords: model confidence set, forecast comparison, model selection, bootstrap, loss functions, multiple comparisons, Stata

*Bocconi University, Milan, Italy

[†]Bocconi University, Milan, Italy

[‡]Bocconi University, Milan, Italy

1 Introduction

Applied researchers often compare several plausible models, forecasts, or decision rules, but the data often do not clearly identify a single dominant alternative. Reporting the model with the smallest average loss is simple, but it ignores uncertainty about the ranking. Pairwise comparisons are also unsatisfactory when many alternatives are evaluated, because unadjusted tests do not provide a coherent statement about the set of models that remain statistically competitive.

Hansen, Lunde, and Nason (2011) proposed the model confidence set (MCS) procedure for this problem. Starting from an initial set of models $\mathcal{M}^0$ and a user-specified loss criterion, the procedure sequentially tests equal predictive ability among the remaining models. If the null is rejected, the model with the weakest relative performance is eliminated. The procedure stops when equal performance can no longer be rejected. The surviving models form a confidence set for the best model, or models, at the chosen confidence level.

This article introduces `modelconfset`, a Stata command that implements the MCS procedure for user-supplied loss variables. Each variable represents the loss of one competing model, forecast, rule, or method. The command supports the $T_{\max}$ and T_R statistics, uses bootstrap inference, reports the sequential elimination path, and returns model-specific MCS p -values and the final superior set in `r()`. The command is loss-function agnostic: users may supply squared-error loss, absolute-error loss, quantile loss, negative log likelihood, coverage loss, or any other criterion appropriate to the application.

For forecasting applications, `modelconfset` also provides an optional post-selection forecast-combination feature. When forecast variables are supplied, the command can construct a combined forecast from the models that survive the MCS procedure, with equal weights as the default. This option is a convenience feature for forecast-comparison workflows; the MCS determines the set of statistically competitive forecasts, while the forecast combination summarizes the predictions of the surviving models.

The contribution of this article is practical rather than theoretical. The asymptotic

theory, sequential testing principle, and bootstrap justification are due to Hansen, Lunde, and Nason (2011). Our contribution is a Stata-native implementation that integrates loss construction, block-bootstrap inference, model-specific MCS p -values, stored post-estimation results, and forecast combination in a single command. We illustrate its use by revisiting the inflation-forecasting application of Hansen, Lunde, and Nason (2011). We then validate the implementation in three complementary ways. First, we compare its output with the existing R implementation of Bernardi and Catania (2018). Second, we reproduce one simulation study in Hansen, Lunde, and Nason (2011). Third, we conduct a forecasting experiment in which the MCS removes most models that contain no predictive signal while retaining the correctly specified model with high probability. In that experiment, combining only the forecasts retained by the MCS substantially improves out-of-sample RMSE relative to averaging across the full candidate set.

The remainder of the article is organized as follows. Section 2 summarizes the MCS procedure and briefly describes how to construct forecast combinations based on the resulting set. Section 3 describes the syntax, options, output, stored results, bootstrap implementation, and forecast-combination feature of `modelconfset`. Section 4 presents an illustrative example. Section 5 validates the command through comparison with an existing implementation and Monte Carlo evidence. Section 6 concludes.

2 The Model Confidence Set Procedure

The model confidence set (MCS) procedure of Hansen, Lunde, and Nason (2011) compares a finite set of competing models, forecasts, or more generally objects. Let

$$\mathcal{M}^0 = \{1, \dots, m_0\}$$

denote the initial set of alternatives. Each alternative is evaluated by a loss function chosen by the researcher. Let $L_{i,t}$ denote the loss of model i at observation $t = 1, \dots, n$, where lower

loss is preferred. The MCS procedure is not tied to a particular loss function. Depending on the application, $L_{i,t}$ may be squared-error loss, absolute-error loss, quantile loss, negative log likelihood, or another user-specified criterion.

The relative performance of models i and j is measured by the loss differential

$$d_{ij,t} = L_{i,t} - L_{j,t}.$$

The corresponding sample average is

$$\bar{d}_{ij} = n^{-1} \sum_{t=1}^n d_{ij,t}.$$

Let $\mu_{ij} = E(d_{ij,t})$. Model i is preferred to model j if $\mu_{ij} < 0$. The population set of superior models is

$$\mathcal{M}^* = \{i \in \mathcal{M}^0 : \mu_{ij} \leq 0 \text{ for all } j \in \mathcal{M}^0\}.$$

Thus, $\mathcal{M}^*$ contains the model, or models, with the lowest expected loss among the initial candidates.

For any current set of models $\mathcal{M} \subseteq \mathcal{M}^0$, the MCS procedure tests the null hypothesis of equal predictive ability,

$$H_{0,\mathcal{M}} : E(d_{ij,t}) = 0 \quad \text{for all } i, j \in \mathcal{M}.$$

Equivalently, the null states that all models that remain in $\mathcal{M}$ have the same expected loss. If this null is not rejected, the procedure stops and reports the current set as the model confidence set. If the null is rejected, the procedure eliminates one model with poor relative sample performance and repeats the test on the reduced set.

The resulting set, $\widehat{\mathcal{M}}_{1-\alpha}^*$, is constructed to contain the population superior set $\mathcal{M}^*$ with

asymptotic probability at least $1 - \alpha$. The procedure therefore does not force the data to select a single model. If the data are highly informative, the final set may contain only a few models. If the data are weakly informative, several models may remain.

The command implements the two t -statistic-based tests emphasized by Hansen, Lunde, and Nason (2011). For the current model set $\mathcal{M}$, define the average loss of model i relative to the other models in the set as

$$\bar{d}_{i\cdot} = |\mathcal{M}|^{-1} \sum_{j \in \mathcal{M}} \bar{d}_{ij}.$$

The corresponding t -statistics are

$$t_{ij} = \frac{\bar{d}_{ij}}{\widehat{\text{var}}(\bar{d}_{ij})^{1/2}}, \quad t_{i\cdot} = \frac{\bar{d}_{i\cdot}}{\widehat{\text{var}}(\bar{d}_{i\cdot})^{1/2}}.$$

The first statistic compares two models directly, while the second compares one model with the average performance of the models in the current set.

The maximum statistic is

$$T_{\max, \mathcal{M}} = \max_{i \in \mathcal{M}} t_{i\cdot},$$

and the range statistic is

$$T_{R, \mathcal{M}} = \max_{i, j \in \mathcal{M}} |t_{ij}|.$$

The statistic $T_{\max, \mathcal{M}}$ is based on the largest standardized loss of one model relative to the average loss in the current set. The statistic $T_{R, \mathcal{M}}$ is based on the largest standardized pairwise loss differential.

Because the limiting distributions of these statistics are nonstandard and depend on nuisance parameters, inference is obtained by bootstrap methods. The bootstrap is used to

estimate the variances entering the t -statistics and to approximate the null distribution of the test statistic. In time-series forecasting applications, block-bootstrap methods are useful because loss differentials may be serially dependent.

The elimination rule is chosen to be coherent with the test statistic. When $T_{\max, \mathcal{M}}$ is used, the eliminated model is

$$e_{\max, \mathcal{M}} = \arg \max_{i \in \mathcal{M}} t_{i.}.$$

When $T_{R, \mathcal{M}}$ is used, the eliminated model is

$$e_{R, \mathcal{M}} = \arg \max_{i \in \mathcal{M}} \left\{ \sup_{j \in \mathcal{M}} t_{ij} \right\}.$$

Thus, in each case the model removed from the set is the one that contributes most strongly to the rejection of equal predictive ability.

The MCS algorithm starts from the full model set $\mathcal{M} = \mathcal{M}^0$. At each step, the null hypothesis $H_{0, \mathcal{M}}$ is tested at significance level α . The first time that the null is not rejected, the current set is reported as $\widehat{\mathcal{M}}_{1-\alpha}^*$. If the null is rejected, the elimination rule removes the worst-performing model from $\mathcal{M}$, and the equal predictive ability test is repeated on the reduced set. The procedure continues until equal predictive ability can no longer be rejected.

This sequential construction is central to the interpretation of the MCS. The procedure eliminates models only when there is statistically significant evidence that the remaining models do not have equal expected loss. Therefore, the final set should be interpreted as the set of models that remain statistically competitive at the chosen confidence level.

The MCS procedure also reports model-specific MCS p -values. These p -values account for the sequential nature of the elimination algorithm. A model belongs to the MCS at confidence level $1 - \alpha$ if and only if its MCS p -value is at least α :

$$i \in \widehat{\mathcal{M}}_{1-\alpha}^* \iff \widehat{p}_i \geq \alpha.$$

The MCS p -value is therefore a convenient summary of the confidence levels at which each model remains in the superior set. It should not be interpreted as the probability that a particular model is the best model.

2.1 Forecast Combinations

In forecasting applications, the model confidence set can also be used as a trimming device before forecast combination. The idea is not to average over all available forecasts, but first to eliminate forecasts that are statistically inferior according to the MCS procedure. The final forecast is then constructed from the models that remain in $\widehat{\mathcal{M}}_{1-\alpha}^*$.

Let $\widehat{y}_{i,t+h}$ denote the h -step-ahead forecast from model i . Given a set of surviving models $\widehat{\mathcal{M}}_{1-\alpha}^*$, a combined forecast can be written as

$$\widehat{y}_{t+h}^{\text{MCS}} = \sum_{i \in \widehat{\mathcal{M}}_{1-\alpha}^*} w_{i,t} \widehat{y}_{i,t+h}, \quad \sum_{i \in \widehat{\mathcal{M}}_{1-\alpha}^*} w_{i,t} = 1.$$

The simplest choice, and the default implemented in `modelconfset`, is the equal-weighted combination

$$w_{i,t} = \frac{1}{|\widehat{\mathcal{M}}_{1-\alpha}^*|}, \quad i \in \widehat{\mathcal{M}}_{1-\alpha}^*,$$

which gives

$$\widehat{y}_{t+h}^{\text{MCS}} = \frac{1}{|\widehat{\mathcal{M}}_{1-\alpha}^*|} \sum_{i \in \widehat{\mathcal{M}}_{1-\alpha}^*} \widehat{y}_{i,t+h}.$$

This approach follows the forecast-combination interpretation of the MCS used by Samuels and Sekkel (2013), who propose trimming the set of candidate forecasts using MCS before averaging, and by Shang and Haberman (2018), who combine forecasts by assigning

equal weights¹ to the selected superior models. The motivation is that simple forecast averages are often difficult to improve upon, but averaging over many weak forecasts can increase noise. MCS-based trimming assigns zero weight to models that are excluded from the superior set and distributes the remaining weight over statistically competitive models.

The MCS confidence level controls the amount of trimming. A small value of α eliminates only models with strong evidence against equal predictive ability and therefore tends to retain a larger set of forecasts. A larger value of α is more aggressive and tends to produce a smaller superior set. Thus, the user chooses not a fixed number of models to combine, but a statistical threshold for excluding forecasts from the combination.

The forecast-combination option should be interpreted as a post-MCS convenience feature. The MCS procedure provides inference for the selected set of superior models. The combined forecast itself does not inherit a separate coverage guarantee from the MCS procedure; it is a forecast constructed from the models that survive the sequential elimination algorithm.

3 The modelconfset command

3.1 Syntax

```
modelconfset varlist [if] [in], alpha(#) reps(#) statistic(tmax|tr)
min_k(#) p_max(#) seed(#) force_k(#) verbose loss_type(SE1|SE2|QL)
realized(varname) tau(#) generate(newvar)
```

3.2 Options

alpha

specifies the significance level of the sequential tests. The resulting model confidence set has nominal coverage $1 - \alpha$. The default is `alpha(.10)`.

¹alternatively one can also use a weighted average here. In the forecasting simulation we compare the simple average with a forecast combination based on the MCS p-values.

reps

specifies the number of bootstrap replications. The default is **reps(5000)**.

statistic(Tmax|TR)

specifies the test statistic used in the MCS procedure. **statistic(Tmax)** uses $T_{\max, \mathcal{M}}$, while **statistic(TR)** uses $T_{R, \mathcal{M}}$. The default is **statistic(Tmax)**.

seed

sets the random-number seed used for the bootstrap. The default is **seed(1)**.

min_k(#)

Select the minimum block length for the bootstrap. The minimum k bounds the block length from the bottom. It determines the minimum level of autocorrelation allowed in the procedure. The default is **min_k(3)**.

force_k(#)

Force a preferred block length k , skipping the data-driven selection. This parameter should be set if one has already established a suitable autocorrelation structure via other statistical tests or institutional constraints. A forced value must be an integer in $[1, T - 1]$, where T is the number of observations. By default (**force_k(0)**) the block length is selected automatically, by the order of the $AR(p)$ fitted to the average loss difference of model i against all other models. This is done at the beginning of the procedure using the full set $\mathcal{M}_0$.

p_max(#)

The maximum AR order searched when the block length k is selected automatically. It therefore bounds the block length k from above. The default is the smaller value between $T - 1$ (length of the dataset) and $10 \cdot \log_{10}(T)$.

verbose

Display intermediate messages during the procedure.

`loss_type(SE1|SE2|QL)`

The `modelconfset` procedure relies on losses for the construction of the final set. The function accepts data in the form of pre-computed losses, but also in the form of forecasted values. This argument specifies the user's preferred loss function in case they provide the forecasted values as an input. Writing f for a forecast and y for the realized value, the available types are $SE1 = (f - y)^2$, $SE2 = (f^2 - y^2)^2$, $QL = (\tau - \mathbf{1}\{f \geq y\})(y - f)$ (quantile loss). When specified, `loss_type()` requires `realized(varname)`.

`realized(varname)`

Specifies the column name holding realized values of the outcome variable.

`tau(#)`

Quantile level in $(0, 1)$ used by the QL loss; ignored for the other loss types. The default is `tau(0.5)`.

`generate(newvar)`

If the user supplies forecasted values and a realized column, the function can automatically produce a combined forecast from the survivor models. `generate(newvar)` specifies the column name (`newvar`) for this forecast.

`weights_type(string)`

When generating a combined forecast, the forecasts in the model confidence set can be weighted in a number of ways. The function allows for equal weighting (`weights_type(equal)`, default) or MCS p-value weighting (`weights_type(pvalue)`), where every model gets weighted by its p-value over the sum of all p-values in the model confidence set.

3.3 Output

The `modelconfset` command has two printing modes. By default, it prints output that consists of three parts. First, a small table containing the parameters of the procedure.

These consist of the chosen statistic, alpha, number of bootstrap repetitions, number of observations, number of models, block length, and the first non-significant MCS p-value. The second part of the output is table with three columns. The first column contains the raw p-values generated in the sequential testing procedure. The second column contains the MCS p -values as defined by Hansen, Lunde, and Nason (2011). The third and final column contains the average loss for each model. This table is sorted according to the MCS p-values, therefore showing the models in the order of elimination. The last part of the default output contains two vectors. One, displaying the estimated model confidence set and a second one holding the eliminated set. An example of this output is shown in Section 4. If `verbose` is selected when calling the procedure, the function additionally displays progress messages while carrying out the testing.

3.4 Stored results

The command stores the following results:

Scalars

<code>r(pvalue)</code>	is the first p -value for which the procedure fails to reject
<code>r(k_selected)</code>	reports the block length k , selected at the beginning of the procedure
<code>r(alpha)</code>	is the significance level of the sequential equivalence tests
<code>r(reps)</code>	is the number of bootstrap replications
<code>r(min_k)</code>	is the minimum block length k
<code>r(p_max)</code>	is the maximum AR order used to select k
<code>r(tau)</code>	is the quantile loss level selected by the user; it is reported only if quantile loss is selected

Macros

<code>r(statistic)</code>	type of the chosen statistic
<code>r(mcs_models)</code>	string of survivor model names (the reported model confidence set)
<code>r(elim_set)</code>	string of eliminated model names, in elimination order

Matrices

<code>r(pvalues_path)</code>	vector of raw p -values computed at every hypothesis test
<code>r(mcs_pvalues)</code>	vector of MCS p -values
<code>r(weights)</code>	row vector of weights over all models; eliminated models receive weight 0, and selected models are weighted according to <code>weights_type()</code>
<code>r(table)</code>	contains the same three columns printed in the output: raw p -values, MCS p -values, and average loss. Models are ordered from the first eliminated to the final surviving model.

3.5 Remarks on missing values and samples

The function does not accept missing values for the construction of the set. When provided with a matrix of estimated forecasts and a column of realized values, the function can only estimate the model confidence set using values common to both columns. Therefore, the user is required to subset the data to a conformable dataset using *if* and *in* qualifiers. The function only accepts rows with missing values during the forecasting procedure, when the values of realized y are missing, but the forecasts are present.

3.6 Remarks on bootstrap settings

Our implementation of the bootstrap is in line with the circular moving block bootstrap proposed by Hansen, Lunde, and Nason (2011) for the MCS procedure. The function re-samples the periods in the following way. For a user-specified block length k , it defines T

possible starting periods, corresponding to the observations in the sample. Then, $\lfloor \frac{T}{k} \rfloor$ starting periods are drawn with replacement. For each starting period i , the function generates indexes $(i, i+1, \dots, i+k-1)$, creating blocks of observations. For indices equal to $T+j$, with $j \geq 1$, the function sets $T+j = j$. This convention specifies that the implemented bootstrap procedure is circular. The corresponding blocks are concatenated and truncated to length T , to create a single index vector that is applied to the loss series of all models, to preserve the serial correlation within the blocks. Then, the procedure is repeated B times and the resulting $B \times T$ index matrix is stored, so that the same resamples are reused throughout the elimination sequence.

It is important to properly specify the bootstrap settings to arrive at sound results. The user can specify the length of the block themselves (based on prior knowledge), or let the function estimate the length of the block by fitting an AR order to the data. The fixed block length can be set by using the `force_k()` parameter. For example, if a user knows that the forecasts are independent every period, `force_k(1)` sets the length of the block to one observation only. If k is estimated by the function, we allow the user to restrict the set of possible block lengths, by setting `min_k()` (the minimum block length) and `p_max()` (the maximum order of the AR process used to estimate k). Furthermore, the function requires the `reps()` parameter, which determines the number of bootstrap repetitions. Generally, the specified value of bootstrap repetitions creates a trade-off between the computational cost (as the number of repetitions increases) and simulation precision.

4 Examples

In this section we show how to implement the `modelconfset` procedure using the inflation-forecasting application of Stock and Watson (1999), as revisited by Hansen, Lunde, and Nason (2011).² In this example, the model confidence set is used to reduce the number

²A corrigendum to Hansen, Lunde, and Nason (2011) corrected an error in the original code for this application. The corrected tables can be obtained from the original authors.

of models available for out-of-sample inflation forecasting. We extract the forecasts from the replication package and include them in our package as `hansen.dta`. We replicate the first part of Table IV using the following commands and output:

```
. * ---- import data and create forecast set
. use "input/hansen"
. ds col0 obs, not
nochangea nochangeb uniar dtip dtgmpyq dtmsmtq dtlpnag
ipxmca hsbp lhmu25 ip gmpyq msmtq lpnag
dipxmca dhsbp dlhmu25 dlhur lhur

. local models `r(varlist)'
```

```
. * ---- run the procedure
. modelconfset `models', loss_type(SE1) realized(obs) statistic("TR") ///
>     alpha(0.10) force_k(12) reps(25000) seed(1) ///
>     generate(combined_forecast) weights_type(equal)
```

```
-----
Model Confidence Set              Number of obs      =          168
Statistic: TR                    Number of models   =           19
Alpha: 0.10                     Block length (k)  =           12
Bootstrap reps: 25000           MCS p-value      =          0.1167
-----
```

Model	P-value	MCS P-value	Avg. Loss
nochangea	0.045	0.045	0.001082
dtip	0.063	0.063	0.000674
dhsbp	0.117	0.117	0.000812
uniar	0.131	0.131	0.000785
ipxmca	0.160	0.160	0.000681
nochangeb	0.170	0.170	0.000783
lhur	0.200	0.200	0.000696
dtgmpyq	0.249	0.249	0.000757
dtlpnag	0.253	0.253	0.000672
lhmu25	0.226	0.253	0.000881
dipxmca	0.420	0.420	0.000566
ip	0.481	0.481	0.000549
dlhmu25	0.465	0.481	0.000568
dlhur	0.751	0.751	0.000527
lpnag	0.735	0.751	0.000580
gmpyq	0.809	0.809	0.000532
dtmsmtq	0.885	0.885	0.000485
msmtq	0.789	0.885	0.000466

```

hsbp          |    1.000          1.000          0.000447
-----+-----

90% Model Confidence Set (17 models):
nochangeb uniar dtgmpyq dtmsmtq dtlpnag ipxmca hsbp lhmu25 ip gmpyq
msmtq lpnag dipxmca dhsbp dlhmu25 dlhur lhur

Eliminated (2 models):
nochangea dtip

. * ---- extract outputs from the function
. local elim 'r(elim_set)'
. local surv 'r(mcs_models)'
. local nsurv : word count 'surv'
. local nelim : word count 'elim'
. scalar mcs_p = r(pvalue)
. matrix mdlist = r(table)

```

The top panel of the output reports the chosen settings and the first MCS p -value at which the null can no longer be rejected. The middle panel lists the models in elimination order together with the raw test p -value, the model-specific MCS p -value, and average loss. The final panel reports the estimated model confidence set and the eliminated models.

All elements of the printed output can also be accessed after estimation. For example:

```

. * ---- print the extracted outputs
. di as result "Number of survivors: 'nsurv'"
Number of survivors: 17

. di as result "Number of eliminated models: 'nelim'"
Number of eliminated models: 2

. di as result "The smallest insignificant MCS p-value: " mcs_p
The smallest insignificant MCS p-value: .11668

. di as text "The extracted table of results: "
The extracted table of results:

. matrix list mdlist

mdlist[19,3]

```

	P-values	MCS P-values	Avg Loss
--	----------	--------------	----------

nochangea	.04484	.04484	.00108227
dtip	.06252	.06252	.00067443
dhsbp	.11668	.11668	.00081237
uniar	.13144	.13144	.00078536
ipxmca	.16012	.16012	.00068079
nochangeb	.17020	.17020	.00078293
lhur	.20044	.20044	.00069560
dtgmpyq	.24868	.24868	.00075687
dtlpnag	.25340	.25340	.00067157
lhmu25	.22592	.25340	.00088081
dipxmca	.42032	.42032	.00056614
ip	.48056	.48056	.00054928
dlhmu25	.46472	.48056	.00056767
dlhur	.75056	.75056	.00052702
lpnag	.73508	.75056	.00057976
gmpyq	.80908	.80908	.00053158
dtmsmtq	.88512	.88512	.00048476
msmtq	.78944	.88512	.00046587
hsbp	1.00000	1.00000	.00044706

The table reproduces the first column of the corrected version of Table IV, yielding the same model confidence set. In particular, models **nochangea** and **dtip** are excluded. Small differences in the reported p -values can arise from differences in pseudorandom-number generation between Stata and the original code.

Because the replication data contain both forecasts and realized values, we can also generate a combined forecast with the **generate** option. The last ten values are:

```
. list combined_forecast in -10/1, clean noobs

combined_forecast
    -.01595684
    -.07241292
    -.02289755
    -.00187827
    -.01513015
    -.00001773
    .00683565
    .00190559
    .01228055
    .0122289
```

5 Simulation and Validation

5.1 Monte Carlo Simulation

We evaluate the finite-sample behavior of the Stata implementation using a simplified version of Design I.A in Hansen, Lunde, and Nason (2011). The loss vector is generated as

$$L_t = \theta + \frac{a_t}{\{E(a_t^2)\}^{1/2}} X_t, \quad X_t \sim N_m(0, \Sigma),$$

where $\Sigma_{ii} = 1$ and $\Sigma_{ij} = \rho$ for $i \neq j$.³ The relative performance vector is governed by

$$\theta = \left(0, \frac{1}{m-1}, \dots, \frac{m-2}{m-1}, 1\right)' \frac{\lambda}{\sqrt{n}}.$$

Thus, when $\lambda = 0$ all models are equally good, whereas for $\lambda > 0$ there is a unique best model. We fix $m = 40$, $n = 250$, $\rho = 0.5$, $\phi = 0.5$, use $B = 1000$ bootstrap replications with block length $\ell = 2$, and report results from $R = 1000$ Monte Carlo replications. The MCS is computed at confidence level $1 - \alpha = 0.90$ using the $T_{\max}$ statistic.

Table 1 reports the results of the Monte Carlo experiment for Design I.A and compares them with the corresponding values in the corrected benchmark table. The comparison is restricted to the design used in the simulation, namely $m = 40$, $\rho = 0.5$, $\phi = 0.5$, and $\lambda \in \{0, 5, 10, 20\}$. The table reports two quantities: the empirical frequency with which the estimated 90% model confidence set contains the true superior set, and the average number of models retained in the estimated set. Overall, the simulation reproduces the benchmark results closely. Under the null design, $\lambda = 0$, the estimated coverage is 0.930, compared with the benchmark value of 0.908. The estimated average number of retained models is 39.929,

³The process a_t is a positive, time-varying common scale factor defined by

$$a_t = \exp(y_t), \quad y_t = -\frac{\phi}{2(1+\phi)} + \phi y_{t-1} + \sqrt{\phi} \varepsilon_t, \quad \varepsilon_t \sim \text{i.i.d. } N(0, 1).$$

It introduces persistent heteroskedasticity common to all model losses. The normalization by $\{E(a_t^2)\}^{1/2}$ keeps the unconditional variance of each loss component equal to one. When $\phi = 0$, $a_t = 1$ and the losses are homoskedastic.

essentially identical to the benchmark value of 39.90, which confirms that the procedure retains almost all models when all models are equally good. For positive values of λ , the empirical coverage is equal to one in all three cases, matching the benchmark. The average MCS size also tracks the corrected benchmark closely: the simulation gives 27.909, 13.303, and 6.014 for $\lambda = 5, 10$, and 20 , respectively, compared with benchmark values of 27.91, 13.27, and 6.050. The main qualitative pattern is therefore the same as in the benchmark table. When $\lambda = 0$, the experiment assesses the size of the procedure and produces coverage close to the nominal 90% level. As λ increases, the inferior models become easier to detect, and the average MCS size decreases sharply. This decline in the number of retained models indicates increasing power while maintaining coverage of the true superior model set. The small discrepancies relative to the benchmark are minor and consistent with Monte Carlo variation and implementation differences in the bootstrap simulation.

Table 1: Design I.A simulation results compared with benchmark values

λ	Frequency at which $\mathcal{M}^* \subseteq \widehat{\mathcal{M}}_{90\%}^*$			Average number of elements in $\widehat{\mathcal{M}}_{90\%}^*$		
	Benchmark	Simulation	Difference	Benchmark	Simulation	Difference
0	0.908	0.930	0.022	39.90	39.929	0.029
5	1.000	1.000	0.000	27.91	27.909	-0.001
10	1.000	1.000	0.000	13.27	13.303	0.033
20	1.000	0.998	-0.002	6.050	6.014	-0.036

Notes: The benchmark values are taken from the corrected Design I.A table for $m = 40$, $\rho = 0.5$, and $\phi = 0.5$. The simulation uses 1,000 Monte Carlo replications. The first three numerical columns report the empirical frequency with which the estimated 90% model confidence set contains the true superior set. The last three numerical columns report the average number of models retained in the estimated model confidence set. “Difference” is computed as Simulation minus Benchmark.

5.2 Forecast Simulation

We next evaluate whether the MCS can improve forecast performance in a linear prediction environment where the candidate set contains both correctly specified and misspec-

ified models. The data-generating process is $y_t = x_{1t} + 0.75x_{2t} + u_t$, so that only x_1 and x_2 are relevant predictors. The predictor vector is generated independently over time as $X_t = (x_{1t}, \dots, x_{6t})' \sim N_6(0, \Sigma)$, where Σ has unit diagonal elements and common off-diagonal correlation $\rho = 0.30$. The disturbance is generated independently as $u_t \sim N(0, 1)$.

The candidate set contains $m = 20$ linear forecasting models constructed from six predictors. It includes the true model (x_1, x_2) , four one-variable models containing only a noise predictor, eight models containing one relevant predictor and one noise predictor, six two-variable models containing only noise predictors, and the overfitted model containing all six predictors.

For each Monte Carlo replication ($R = 1000$), the observations are divided into three non-overlapping samples of equal size. The models are estimated using $n_{\text{train}} = 100$ training observations. Their losses over a separate validation sample of $n_{\text{val}} = 100$ observations are used to construct the MCS and to identify the model with the largest MCS p -value. Forecast performance is then measured on an independent test sample of $n_{\text{test}} = 100$ observations. The MCS is constructed at the 90 percent confidence level using the T_{max} statistic, a block bootstrap with block length 2, and $B = 1000$ bootstrap draws.

We evaluate four forecast rules: the equal-weighted combination of all candidate models, the equal-weighted combination of the models retained in the MCS, a p -value-weighted combination of the retained models, and the individual model with the largest MCS p -value in the validation sample. For each rule, test-sample RMSE is normalized by the test-sample RMSE of the all-model combination. The p -value-weighted combination is included as a predictive heuristic; it does not interpret the MCS p -values as probabilities that the corresponding models are optimal.

Table 2 reports test-sample forecast performance and summarizes the types of models eliminated by the MCS. The equal-weighted combination of all candidate forecasts is used as the normalization, so values below one indicate an improvement relative to combining all forecasts. The equal-weight MCS combination has a relative RMSE of 0.821, corresponding

to an approximately 18 percent reduction in RMSE relative to the all-model combination. The p -value-weighted MCS combination has a relative RMSE of 0.800, slightly improving over the simple forecast average.⁴ Selecting the model with the largest validation-sample MCS p -value yields a relative RMSE of 0.786, an improvement of approximately 21 percent.

Table 2: Forecast simulation results

Forecast rule	Rel. RMSE	Avg. MCS size	True-model cov.	Exclusion rate			
				True only	Mixed	Wrong only	Overfitted
All-model combination	1.000	–	–	–	–	–	–
Equal-weight MCS combination	0.821	6.70	0.998	0.002	0.428	0.981	0.068
p -value-weighted MCS combination	0.800	–	–	–	–	–	–
Highest MCS- p model	0.786	–	–	–	–	–	–

Notes: Relative RMSE is computed on the independent test sample and normalized by the RMSE of the equal-weighted combination of all 20 candidate models. True-model coverage is the fraction of replications in which the model containing x_1 and x_2 belongs to the 90% MCS. Exclusion rates are average class-specific exclusion frequencies. Mixed models contain one relevant predictor, wrong-only models contain no relevant predictors, and the overfitted model contains all six predictors.

The MCS contains an average of 6.7 of the 20 candidate models and includes the correctly specified model in 99.8 percent of the replications. The elimination pattern depends strongly on how much predictive signal a model retains. The MCS excludes approximately 98.1 percent of the models containing no relevant predictors. In contrast, it excludes only 42.8 percent of the mixed models, which contain one relevant predictor and one noise predictor. Models that retain part of the predictive signal are therefore much harder to distinguish from the best model in the finite validation sample. The overfitted model containing all six predictors is excluded in only 6.8 percent of the replications, although it contains four irrelevant predictors.

Hansen, Lunde, and Nason (2011) emphasize that the MCS may contain several models

⁴The p -value-weighted MCS combination assigns model $i \in \widehat{\mathcal{M}}_{90\%}^*$ the weight

$$w_i = \frac{\widehat{p}_i}{\sum_{j \in \widehat{\mathcal{M}}_{90\%}^*} \widehat{p}_j},$$

so that models with larger MCS p -values receive greater weight and the weights sum to one.

when the available data do not provide enough information to separate all alternatives and that a model remains in the set until it is found to be significantly inferior. In the present simulation, the validation sample is highly informative about models that discard the entire predictive signal, but less informative about models that retain one or all relevant predictors.

5.3 Comparison with the R Implementation

Finally, we benchmark our Stata implementation against the existing R package of Bernardi and Catania (2018). The benchmark data consist of the out-of-sample losses of $m_0 = 40$ competing volatility models over $n = 2000$ observations distributed with the R implementation. We set $\alpha = 0.15$, use $B = 1000$ bootstrap replications, and report results for both the $T_{\max}$ and T_R statistics.

The deterministic components agree to numerical precision. Across the 40 models, the average losses $n^{-1} \sum_t L_{i,t}$ reported by the two programs differ by at most 5.4×10^{-19} . Both implementations also select the same block length, $k = 33$. For $T_{\max}$, the equal-predictive-ability null is not rejected at the first step by either implementation. Consequently, no model is eliminated and both programs return the full set of 40 models as the estimated 85% MCS.

The bootstrap MCS p -values need not coincide exactly because the two programs use different random-number streams and slightly different resampling implementations. As shown in Table 3, however, the model-specific p -values are strongly correlated across implementations. For both $T_{\max}$ and T_R , the median absolute difference between matched model-specific p -values is only modestly larger in the Stata–R comparison than between two runs of the R implementation using different random seeds.

Table 3: Numerical comparison of `modelconfset` and the R implementation

Quantity	Stata vs. R		R vs. R (seed)	
	median $ \Delta $	corr.	median $ \Delta $	corr.
MCS p -value ($T_{\max}$)	0.017	0.9929	0.014	0.9965
MCS p -value (T_R)	0.041	0.9750	0.019	0.9957
Average loss	5.4×10^{-19}	–	–	–

Notes: “median $|\Delta|$ ” is the median absolute difference across matched model-specific quantities and “corr.” is the cross-program correlation. The right panel repeats the exercise for two runs of the R implementation that differ only in the random seed, as a benchmark for sensitivity to simulation randomness.

6 Conclusions

This article introduces `modelconfset`, a Stata command for implementing the model confidence set procedure of Hansen, Lunde, and Nason (2011). The command compares an arbitrary set of models using user-supplied losses, implements the $T_{\max}$ and T_R statistics with moving-block bootstrap inference, and reports the sequential elimination path, model-specific MCS p -values, and the final set of statistically competitive models. For forecasting applications, it can additionally construct an equal-weighted or p -value-weighted combination of the forecasts retained in the model confidence set if the individual forecasts are supplied.

We assess the implementation through comparison with Bernardi and Catania (2018), replication of a simulation design and an application from Hansen, Lunde, and Nason (2011), and a simulated forecasting example. The forecasting experiment further illustrates how the MCS can remove poorly performing forecasts while retaining models that cannot be reliably distinguished from the best alternative, thereby improving forecast combinations when the initial model set contains weak specifications.

References

- Bernardi, M. and L. Catania (2018). “The model confidence set package for R”. In: *International Journal of Computational Economics and Econometrics* 8, pp. 144–158.
- Hansen, Peter R., Asger Lunde, and James M. Nason (2011). “The Model Confidence Set”. In: *Econometrica* 79.2, pp. 453–497. DOI: <https://doi.org/10.3982/ECTA5771>.
- Samuels, Jon D. and Rodrigo M. Sekkel (Apr. 2013). *Forecasting with Many Models: Model Confidence Sets and Forecast Combination*. Working Paper 2013-11. Bank of Canada Working Paper 2013-11. Bank of Canada.
- Shang, Han Lin and Steven Haberman (2018). “Model Confidence Sets and Forecast Combination: An Application to Age-Specific Mortality”. In: *Genus* 74.19, pp. 1–23. DOI: [10.1186/s41118-018-0043-9](https://doi.org/10.1186/s41118-018-0043-9).
- Stock, James H and Mark W Watson (1999). “Forecasting inflation”. In: *Journal of Monetary Economics* 44.2, pp. 293–335. DOI: [https://doi.org/10.1016/S0304-3932\(99\)00027-6](https://doi.org/10.1016/S0304-3932(99)00027-6).