

---

# MULCOM 2.00

## Econometric Toolkit for Multiple Comparisons

PETER REINHARD HANSEN

*Department of Economics, Stanford University,  
Landau Economics Building, 579 Serra Mall,  
Stanford, CA 94305-6072, USA  
& CREATES*

ASGER LUNDE

*School of Economics and Management,  
University of Aarhus  
Bartholins Allé 10, Aarhus, Denmark  
& CREATES*

Email: `alunde@econ.au.dk`

---

# Contents

---

<b>Preface</b>	<b>vi</b>
<b>Chapter 1 Introduction and Installation</b>	<b>1</b>
<b>1.1</b> Disclaimer . . . . .	1
<b>1.2</b> Availability and Citation . . . . .	1
<b>1.3</b> Installation . . . . .	2
1.3.1 The files <code>mulcom.zip</code> . . . . .	2
<b>1.4</b> Using MULCOM . . . . .	3
1.4.1 Console Application . . . . .	3
1.4.2 OxEdit Application . . . . .	3
1.4.3 OxPack Application though OxMetrics . . . . .	3
<b>1.5</b> Sample session using MULCOM in OxPack . . . . .	3
<b>Chapter 2 Tests for Superior Predictive Ability</b>	<b>8</b>
<b>2.1</b> What is the Test for SPA? . . . . .	8
2.1.1 SPA $p$ -values . . . . .	9
2.1.2 Critical values for 10%, 5%, and 1% . . . . .	9
2.1.3 Relation to the Reality Check . . . . .	9
<b>2.2</b> Interpreting SPA Results . . . . .	10
<b>2.3</b> OxEdit Application . . . . .	11
2.3.1 User specified loss function . . . . .	12
<b>2.4</b> OxPack Application . . . . .	13

---

<b>Chapter 3</b>	<b>Model Confidence Set</b>	<b>15</b>
3.1	What is a MCS? . . . . .	15
3.2	Theory for General Confidence Set . . . . .	16
3.2.1	MCS $p$ -Values . . . . .	16
3.2.2	Equivalence Tests and Elimination Rules . . . . .	17
3.3	Interpreting MCS Results . . . . .	18
3.4	OxEdit Application . . . . .	19
3.5	OxPack Application . . . . .	21
<b>Chapter 4</b>	<b>MulCom Reference</b>	<b>24</b>
4.1	MULCOM summary . . . . .	24
4.2	MULCOM member functions . . . . .	24
	<b>Bibliography</b>	<b>28</b>
	<b>Index</b>	<b>29</b>

# List of Figures

---

1.1	OxPack: Selection of multiple comparison module . . . . .	4
1.2	OxPack: Data Selection window . . . . .	4
1.3	OxPack: Data Selection window . . . . .	5
1.4	OxPack: Selection of SPA and/or MCS; Loss function and test statistics. . . . .	6
1.5	OxPack: Selection of Sample, Bootstrap, and output settings. . . . .	7

# List of Programs

---

2.1	Testing for SPA using OxEdit . . . . .	11
2.2	Testing for SPA with an user specified loss function . . . . .	12
3.1	Estimating a MCS using OxEdit . . . . .	20
3.2	Estimating a MCS with an user specified loss function . . . . .	21

# List of Outputs

---

2.1	SPA test results	10
3.1	MCS results	19

# Preface

---

This program grew from a series of papers written by authors during the period 2000-2003. Financial support from the Danish Research Agency, grant no. 24-00-0363, is gratefully acknowledged.

Peter R. Hansen and Asger Lunde  
September 2007

Version 2.00 follows the publication of Hansen, Lunde and Nason (2010). To this version there has been a few updates to the MCS part of the code. Most importantly, the  $T_{\max}$  statistic is now the default option in the MCS procedure. Moreover, some new methods has been add for accessing the estimated MCS and for controlling the amount of output that is printed to screen.

Peter R. Hansen and Asger Lunde  
September 2010

# Chapter 1

---

## INTRODUCTION AND INSTALLATION

This documentation describes the MULCOM (Multiple Comparison) package version 2.00 for Ox 5 or later, see Doornik (2006).

MULCOM is a package that provides tools for simultaneous comparison of any number of models. MULCOM is designed for forecasting evaluation to analyze whether any among a set of competing models are significantly better than the others, in terms of predictive accuracy. It is however, applicable to a much wider range of problems. It can be applied in any setting where one what to compare the means of two or more populations.

The current implementation of MULCOM can do two things. Test for Superior Predictive Ability (SPA) and estimate a Model Confidence Set (MCS). The test for SPA is that of Hansen (2005) which is similar to the *reality check* by White (2000). The key difference is that the SPA-test is more powerful in many situations. Both the test for SPA and the reality check are implemented in MULCOM. The MCS methodology is due to Hansen et al. (2010).

MULCOM is a class written in Ox (see Doornik, 2006), and it is used by writing small Ox programs which create and use an object of the MULCOM class. Some knowledge of Ox will be required when using MULCOM in this way.

MULCOM can also be used interactively in conjunction with OxPack for OxMetrics (see Doornik and Hendry, 2006). This is probably the easiest way to use the program, but it requires access to this software. Below we will give examples of applications, both using the console- and the interactive version.

### 1.1. Disclaimer

This package is functional, but no warranty is given whatsoever. The most appropriate forum to discuss problems and issues related to the MULCOM package is the ox-users discussion group.<sup>1</sup> Any bugs, typos in the manual, or suggestions for improvements should be reported to Asger Lunde (email: [alunde@asb.dk](mailto:alunde@asb.dk)) and will be greatly appreciated.

### 1.2. Availability and Citation

The MULCOM package is available at [mit.econ.au.dk/vip/htm/alunde/mulcom/mulcom.htm](http://mit.econ.au.dk/vip/htm/alunde/mulcom/mulcom.htm). The use of MULCOM is free provided that it is cited in any application.

---

<sup>1</sup>See [www.mailbase.ac.uk/lists/ox-users](http://www.mailbase.ac.uk/lists/ox-users).



*The package must be cited whenever it is used. When testing for SPA cite also Hansen (2005), and/or when estimating a MCS please cite Hansen et al. (2010).*

To use the package you must have one of the following configurations on your computer:

- Ox version 4.00 or later. For academic use, a free version of Ox is available from the web site.  
<http://www.doornik.com/ox/>
- Ox Professional and OxMetrics. Which can be purchased from the following web site.  
<http://www.timberlake.co.uk/>

### 1.3. Installation

1. Make sure you have properly installed Ox version 4.00 or later. The MULCOM package does not work fully with earlier versions of Ox. Type `oxl` at the command prompt to check.
2. Unzip `mulcom.zip` to the `ox\packages` folder.
3. Read the `read.me` file for info on the last updates.
4. If Ox has been installed properly, this will allow using the MULCOM package from any directory. To use the package in your code, add the command  
`#import <packages/MulCom/MulCom>`  
at the top of all files which require it.

#### 1.3.1. The files `mulcom.zip`

Program files

- `Mulcom.oxo` - the complied source code.

Example programs

- `Inflation-SW-tab2-SPA-v01.ox` - use MULCOM to test for SPA in the model space in Table 2 in Stock and Watson (1999).
- `Inflation-SW-tab2-SPA-v02.ox` - produce the same output as the former program, but shows how the user can specify an alternative loss function.
- `Inflation-SW-tab2-MCS-v01.ox` - use MULCOM to estimate a MCS in the model space in Table 2 in Stock and Watson (1999).

Included sample data sets

- `tab2-data-1.xls`, `tab2-data-2.xls`, `tab2-data-3.xls`, `tab2-data-4.xls` - The forecasts used to produce Table 2 in Stock and Watson (1999).

## 1.4. Using MULCOM

MULCOM can be used in one of three ways.

1. Applying Ox to your program file (the .ox) in a MS-DOS console.
2. Running the program from OxEdit.
3. Applying OxPack, together with Ox Professional and OxMetrics

In the following sections we give the basics of these methods.

### 1.4.1. Console Application

To run a program at hand (e.g. `Inflation-SW-tab2-SPA-v01.ox` in chapter 2), from the command prompt in a console window simply type

```
oxl Inflation-SW-tab2-SPA-v01.ox
```

### 1.4.2. OxEdit Application

Open `Inflation-SW-tab2-SPA-v01.ox` (see Program 2.1 on page 11) in OxEdit and press [Ctrl]+r. For how to use Ox in conjunction with OxEdit consult Doornik and Ooms (2005).

### 1.4.3. OxPack Application through OxMetrics

Installation of the interactive version of MULCOM :

1. Install MULCOM into `ox/packages/mulcom` as described above.
2. OxPack requires a properly installed OxMetrics version 4.00 or later. Check the version number in the OxMetrics Help menu.
3. Start OxMetrics, and then OxPack from the OxMetrics Modules menu. From the OxPack Package menu choose Add/Remove Package. Locate `MulCom.oxo` (in the MULCOM folder) using the Browse button, and press Add.

## 1.5. Sample session using MULCOM in OxPack

If a data set is open in OxMetrics, then you are now ready to use MULCOM. From the OxPack Package menu choose MULCOM. The title bar of the OxPack window shows MULCOM is loaded and the message `MulCom package version 1.00, object created on ...` is displayed in the OxMetrics Results window

When you start OxPack it looks as shown in Figure 1.1. To start comparing models you simply select Formulate in the Model drop down window. If you do so then the Formulate dialog will appear, and if you opened the data file, `tab2-data-1.xls`, then this window will look as shown in Figure 1.2.

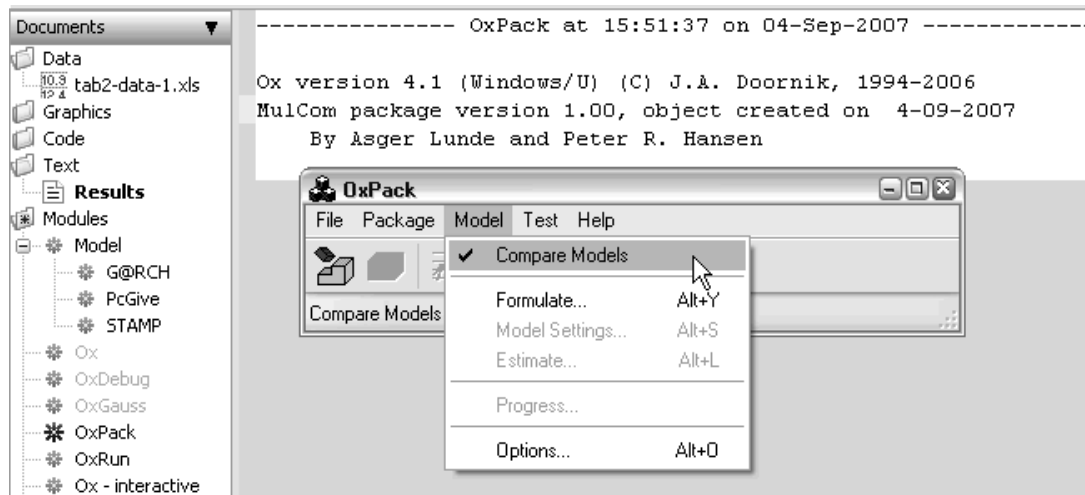


Figure 1.1: OxPack: Selection of multiple comparison module

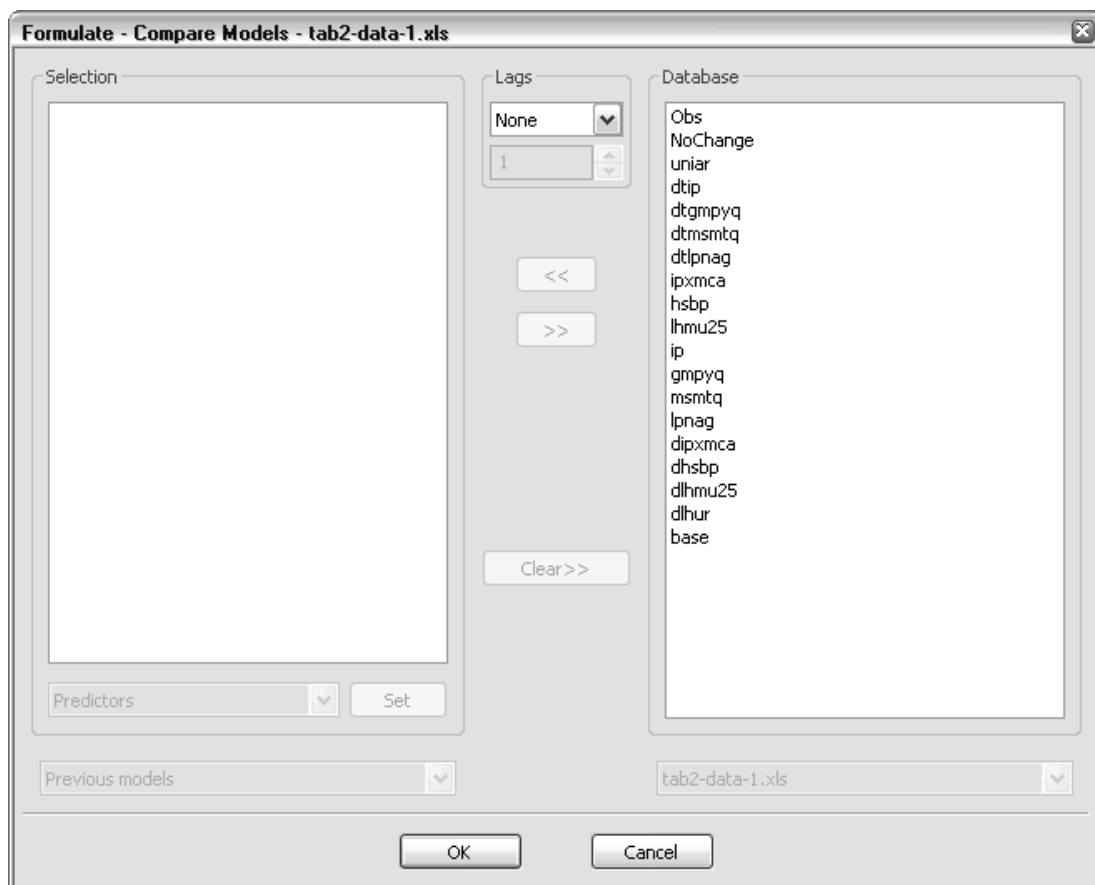


Figure 1.2: OxPack: Data Selection window

In the Database area you see all the variables that are available in your data set. For use with MULCOM

these variables will typically be forecasts, forecasts errors, or some kind of performance score. You move the variables that you want to compare into the Selection area using the controls between the areas. From the dropdown menu just below the Selection area you can label your selection of variables as follows.

- Ex Post Obs: The variable marked with an O is the variable containing the ex post observations of the entity to be forecasted.
- Benchmark: The variable marked with a B is the variable containing the forecasts of the benchmark model.
- Predictor: The variables marked with a P contain the forecasts of the models which are to be compared with the benchmark or each other.

To illustrate the use of the Data selection window we have made some selections that are shown in Figure 1.3

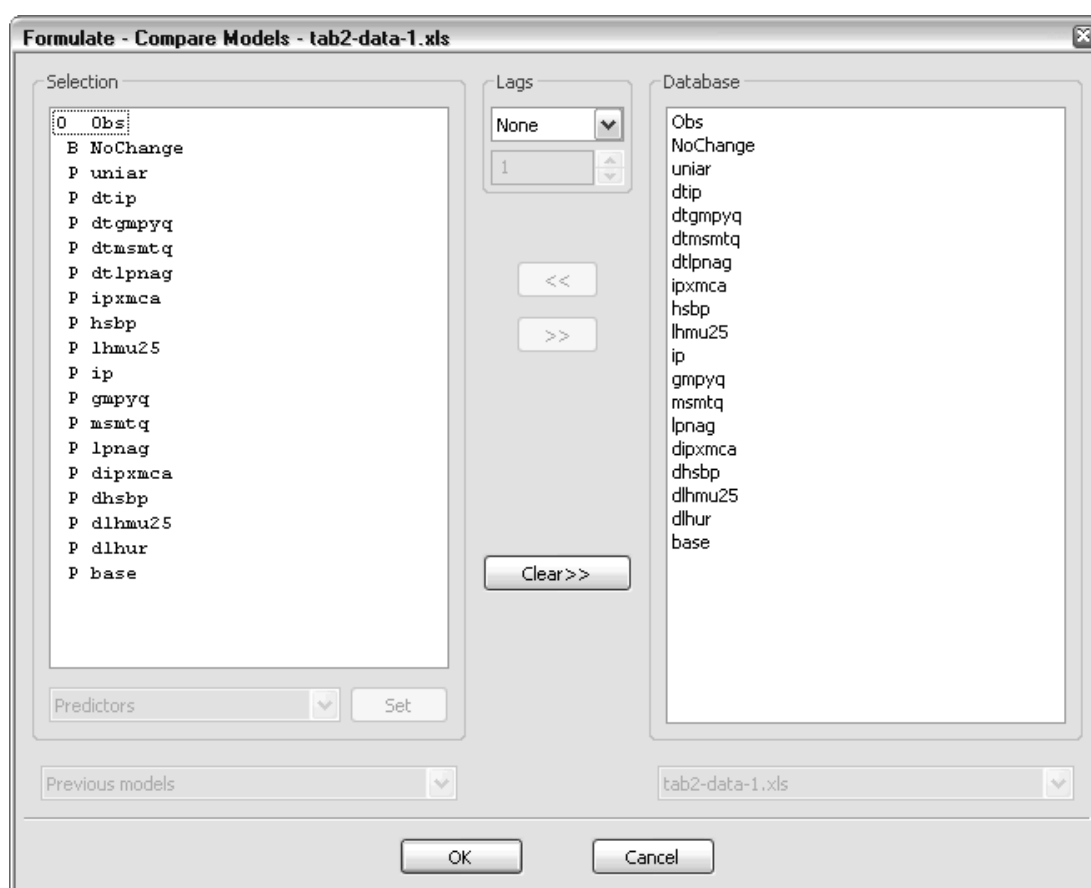


Figure 1.3: OxPack: Data Selection window

If you click OK then the Model Settings menu will appear as in Figure 1.4. Here you must choose if you want to test for SPA or estimate a MCS.

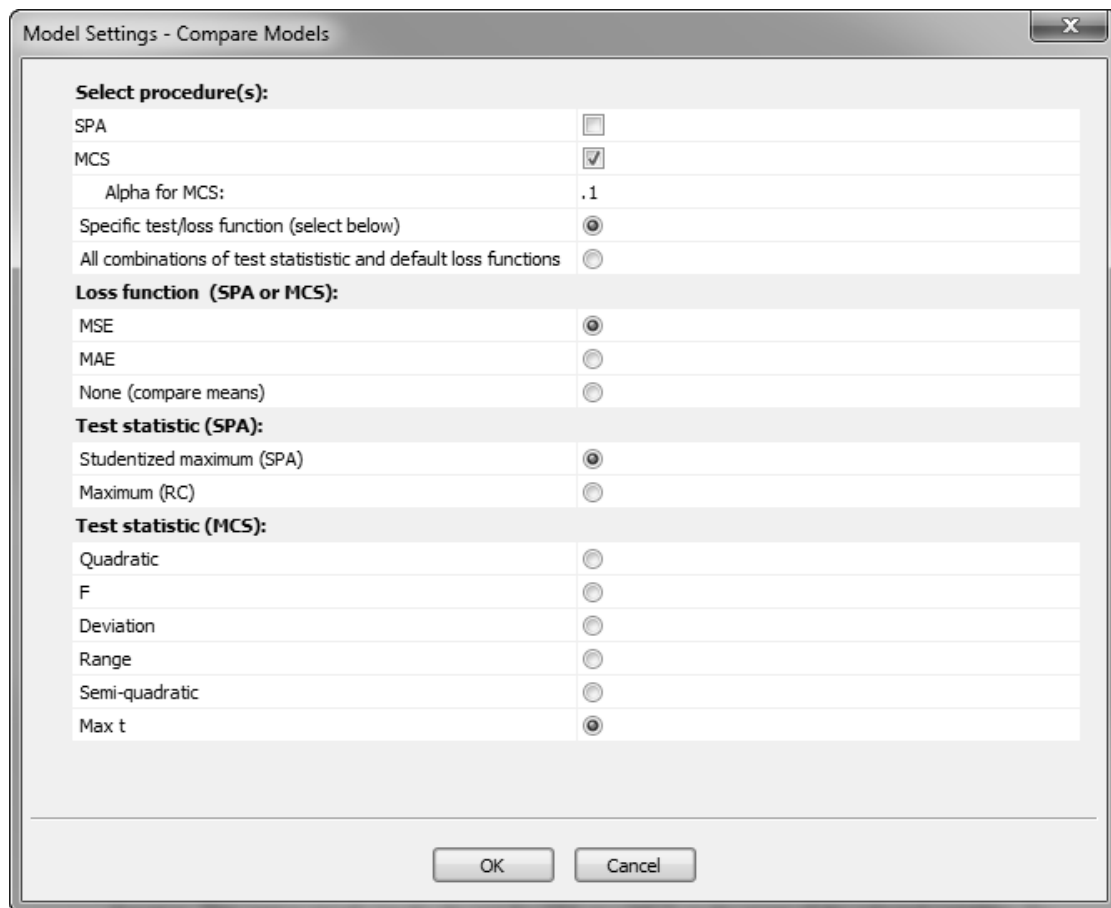


Figure 1.4: OxPack: Selection of SPA and/or MCS; Loss function and test statistics.

You must also specify a loss function. `MULCOM` currently offers the choice of two predefined loss functions, and a third option, None (compare means), that can be used to implement any other loss function. This option simply invoke the test for SPA or a MCS on the mean of the selected variables. So by appropriately transforming the variables in advance the desired loss function will be used. Finally, a test statistics must be specified in order to test for SPA or to compute a MCS. We return to the available statistics in Chapters 2 and 3.

Note that you can have `MULCOM` do all combinations of the available Loss functions and test statistics.

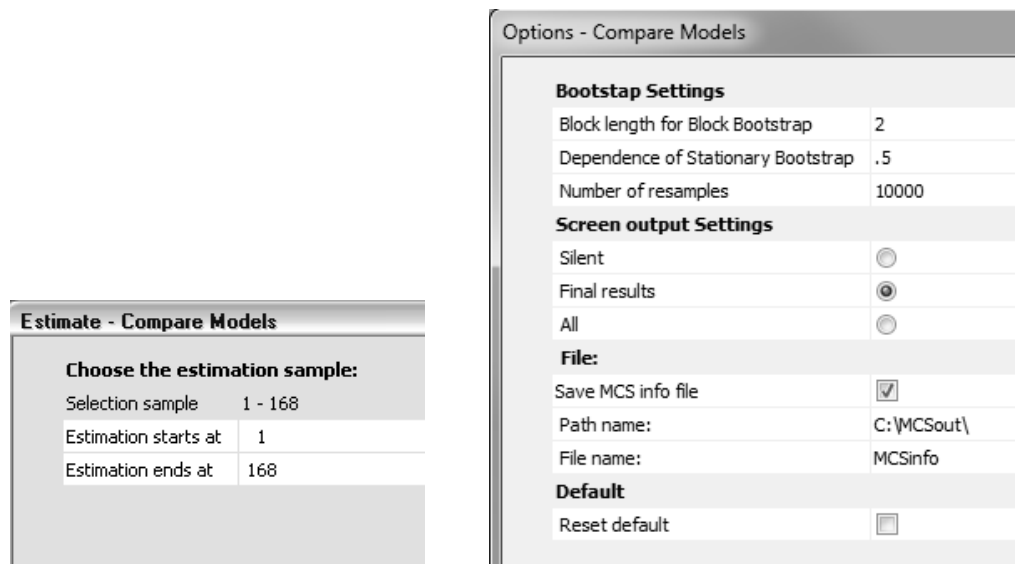


Figure 1.5: OxPack: Selection of Sample, Bootstrap, and output settings.

Upon clicking OK you can in the Estimate Model menu (see Figure 1.5) select the desired sample, and through the Options menu (see Figure 1.1) the settings for the bootstrap, and output to screen and file may be altered.

We will return to the output of both a SPA and a MCS sample session, when we have explained what these concepts cover.

# Chapter 2

## TESTS FOR SUPERIOR PREDICTIVE ABILITY

### 2.1. What is the Test for SPA?

The Superior Predictive Ability test is a test that can be used for comparing the performances of two or more forecasting models. The forecasts are evaluated using a prespecified loss function, and the “best” forecast model is the model that produces the smallest expected loss. Two loss functions are predefined in the SPA code the two are: the Mean Squared Error (mse) and Mean Absolute Deviation (mad).

Let  $L(Y_t, \hat{Y}_t)$  denote the loss if one had made the prediction,  $\hat{Y}_t$ , when the realized value turned out to be  $Y_t$ . The performance of model  $k$ , relative to the benchmark model (at time  $t$ ), can be defined as:

$$d_{k,t} = L(Y_t, \hat{Y}_{0,t}) - L(Y_t, \hat{Y}_{k,t}), \quad k = 1, \dots, m, \quad t = 1, \dots, n.$$

The question of interest is whether any of the models  $k = 1, \dots, m$  are better than the benchmark model. To analyze this question we formulate the testable hypothesis that the benchmark model is the best forecasting model. This hypothesis can be expressed parametrically as

$$H_0 : \mu_k = E[d_{k,t}] \leq 0, \quad \text{for all } k = 1, \dots, m. \quad (2.1.1)$$

For notational convenience, we define an  $m$ -dimensional vector  $\boldsymbol{\mu}$  by

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_m \end{pmatrix} = E \begin{pmatrix} d_{1,t} \\ \vdots \\ d_{m,t} \end{pmatrix}.$$

Since a positive value of  $\mu_k$  corresponds to model  $k$  being better than the benchmark, we want to test the null hypothesis stated in (2.1.1). The equivalent vector formulation is

$$H_0 : \boldsymbol{\mu} \leq \mathbf{0}.$$

One way to test this hypothesis is to consider the test statistic

$$T_n^{SPA} = \max_k \frac{n^{1/2} \bar{d}_k}{\hat{\omega}_k},$$

where

$$\bar{d}_k = \frac{1}{n} \sum_{t=1}^n d_{k,t},$$

and

$$\hat{\omega}_k^2 = \widehat{\text{var}}(n^{1/2}\bar{d}_k),$$

is a consistent estimator of the asymptotic variance,  $\omega_k^2 = \lim_{n \rightarrow \infty} \text{var}(n^{1/2}\bar{d}_k)$ .

Under the regularity condition it holds that

$$n^{-1/2}T_n^{SPA} = \max_k \frac{\bar{d}_k}{\hat{\omega}_k} \xrightarrow{p} \max_k \frac{\mu_k}{\omega_k},$$

which is greater than zero if and only if  $\mu_k > 0$  for some  $k$ . So we can test  $H_0$  using the test statistic  $T_n^{SPA}$ . The only remaining problem is to derive the distribution of  $T_n^{SPA}$ , under the assumption of a true null hypothesis. Testing multiple inequalities is more complicated than testing equalities (or a single inequality) because the distribution is not unique under the null hypothesis. Nevertheless, a consistent estimate of the  $p$ -value can be obtained, as well as an upper and a lower bound, and those are the  $p$ -values produced by the SPA program. The test for SPA have been applied by among other Hansen and Lunde (2005) and Koopman, Jungbacker and Hol (2005).

### 2.1.1. SPA $p$ -values

The upper bound is the  $p$ -value of a conservative test which tacitly assumes that all the competing models ( $k = 1, \dots, m$ ) are as precisely as good as the benchmark in terms of expected loss ( $\mu_1 = \dots = \mu_m$ ). The lower bound is the  $p$ -value of a liberal test whose null hypothesis assumes that the models with worse performance than the benchmark are poor models in the limit. Therefore, these can be viewed as asymptotic upper and lower bounds for the actual  $p$ -value, respectively. The consistent  $p$ -value is produced by the test for SPA of Hansen (2005). This test will asymptotically determine which models are worse than the benchmark and asymptotically prevent them from influencing the estimated distribution of the test statistic, as should be the case. While the conservative test is sensitive to including poor and irrelevant models in the comparison, the consistent and liberal tests are not affected, at least not asymptotically.

### 2.1.2. Critical values for 10%, 5%, and 1%

Since  $p$ -values are designed to control for the size of a test, they are not informative of the power of a test, which is the probability of rejecting  $H_0$  when  $H_1$  is true. When you fail to reject the null hypothesis, the critical value can be informative about the power of a test. By definition, the critical values show how well a model would need to perform (in standardized performance  $\bar{d}_k/\hat{\omega}_k$ ) in order to reject the null hypothesis and conclude that a competing model is better than the benchmark.

If the critical values are large, it suggests that the data are not very informative about the hypothesis of interest. Hypotheses testing using relatively uninformative data results in tests with low power. Lack of power in the present context makes it difficult to discover a superior model, even if one existed. It is therefore useful to pay attention to critical values when the SPA test yields insignificant  $p$ -values.

### 2.1.3. Relation to the Reality Check

The SPA test differs from White's reality check in two ways. The SPA test employs a different test statistic and is based on the consistent null distribution.



The reality check is based on the statistic

$$T_n^{RC} = \max_k n^{1/2} \bar{d}_k.$$

As is the case for  $T_n^{SPA}$ , the asymptotic distribution of  $T_n^{RC}$  depends on the number of inequalities in (2.1.1) that are binding. For this statistics one can also construct lower and upper bounds for  $p$ -values. The reality check is based on the conservative upper bound  $p$ -values.

## 2.2. Interpreting SPA Results

When running the standard configuration of the test SPA using MULCOM the result will appear as displayed in Output 2.1.

```

Ox version 3.30 (Windows) (C) J.A. Doornik, 1994-2003
MulCom package version 0.1, object created on 13-10-2003
  By Asger Lunde and Peter R. Hansen

----- TEST FOR SUPERIOR PREDICTIVE ABILITY -----
SPA version 1.12, March 2003. (C) 2001, 2002, 2003 Peter R. Hansen

Number of models:      l=18
Sample size:           n=168
Loss function:         mse_spa
Test Statistic:        TestStatScaledMax()
Bootstrap parameters:  B=1000 (resamples), q=0.5 (dependence)

```

	Model number	Performance	t-stat	"p-value"
Benchmark	0.00000	-0.00108	-	-
Most Significant	11.00000	-0.00047	3.48586	0.00300
Best	11.00000	-0.00047	3.48586	0.00300
Model_75%	16.00000	-0.00054	3.07528	0.00300
Median	6.00000	-0.00056	2.89055	0.00699
Model_25%	5.00000	-0.00062	2.58918	0.00899
Worst	14.00000	-0.00073	2.13435	0.02597
	Lower	Consistent	Upper	
SPA p-values:	0.00300	0.00300	0.00300	
Critical values: 10%	1.67521	1.67521	1.67521	
5%	2.11804	2.11804	2.11804	
1%	2.80970	2.80970	2.80970	

Output 2.1: SPA test results. For code see Program 2.1

This output can be divided into three parts. The first part contains descriptive statistics, such as the number of competing models, sample size, bootstrap parameters, etc. The second part informs about model performance, and six pair-wise comparisons. The six models being compared to the benchmark model are: (1) The "Most Significant" model is the model that had the most "significant" performance relative to the benchmark model in the sample being analyzed, and (2-6) those models with a performance that corresponded to the 75%, 50% (median), 25%, and 0% (worst) quantile of model performances. This provides some information about the population of model performances in the sample being analyzed. The Performance is measured in terms of the loss function that has been specified by the user, in this case the mean absolute deviation (mad) has been used, another predefined loss-function is the mean squared error (mse). The t-stat indicates the significance of a model's performance relative to the benchmark

model, although it is important to note that this statistic is not  $t$ -distributed, and the “ $p$ -value” reported next to it cannot be interpreted as a  $p$ -value, rather it is a number that is calculated like a  $p$ -value, but it ignores the search over models that preceded the selection of the model being compared to the benchmark.

The third part contains the relevant information for testing the hypothesis that the benchmark model is the best forecasting model. The SPA  $p$ -value takes the space of models into account and the SPA-Consistent  $p$ -value instructs you if there is evidence against the hypothesis. A low  $p$ -value (less than .05–.10 say) informs you that the benchmark model is inferior to one or more of the competing models. A high  $p$ -value tells you that the sample being analyzed does not yield strong evidence that the benchmark is outperformed. The number the left (right) of the SPA-Consistent  $p$ -value, is a lower (upper) bound for the true  $p$ -value. Critical values at the 10%, 5%, and 1% significance levels are given below the SPA  $p$ -values.

### 2.3. OxEdit Application

In this section we show how to use MULCOM through in OxEdit. In Program 2.1 we present the sample code that produced Output 2.1.

```

/*-----
 * Code (C) Lunde & Hansen, 2007
 *
 * FILE: Inflation-SW-tab2-SPA.ox
 * Example: Use MulCom to Test for SPA
 *
 *-----*/

#include <oxstd.h>
#import <packages/MulCom/MulCom>

main()
{
    // Declare MulCom object
    decl MCobj = new MulCom();

    // Get dataset
    MCobj.Load("data/tab2-data-1.xls");

    // Select realization, benchmark, and forecasts
    MCobj.Select(O_VAR, {"Obs",0,0} );
    MCobj.Select(B_VAR, {"NoChange",0,0} );
    MCobj.SelectByIndex(P_VAR, <2:23>, 0, 0 );

    // Select sample period
    MCobj.SetSelSample(-1, 1, -1, 1);

    // Select test specification
    MCobj.Do_SPA(0.5,10000); // Prob. for stationary bootstrap, and
                           // # bootstrap resamples.
    MCobj.Set_Loss("mse"); // Loss func.: "mse", "mad", "ident"
    MCobj.Set_TestStatSPA("TestStatScaledMax"); // Test stat.: "TestStatScaledMax",
                                                // "TestStatMax"

    MCobj.DoEstimation();

    delete MCobj;
}

```

Program 2.1: Sample code Inflation-SW-tab2-SPA-v01.ox to testing for SPA using OxEdit

The lines of codes does the follwing:

```

decl MCobj = new MulCom(); declare an object of the MULCOM class.
MCobj.Load("data/tab2-data-1.xls"); reads in the relevant data set.
MCobj.Select(O_VAR, "Obs", 0, 0); specify the target of the forecast.
MCobj.Select(B_VAR, "NoChange", 0, 0); specify the benchmark.
MCobj.SelectByIndex(P_VAR, <2:23>, 0, 0); specify the forecasts to be included in the
comparison.
MCobj.SetSelSample(-1, 1, -1, 1); select the sample size.
MCobj.Do_SPA(0.5, 10000); settings for bootstrapping. The first number is the dependence param-
eter for the stationary bootstrap, and the second is the number of bootstrap resamples.
MCobj.Set_Loss("mse"); select the loss function.
MCobj.Set_TestStat("TestStatScaledMax"); select the test statistics.
MCobj.DoEstimation(); run the test.
delete MCobj; delete the MULCOM object.

```

To run the program in OxEdit simply press control-r. For more details regarding Ox in conjunction with OxEdit consult Doornik and Ooms (2005).

### 2.3.1. User specified loss function

Alternatively one may desire some other loss function. In Program 2.2 we present some code that also will produce Output 2.1, but with the loss function designed before calling `MCobj.Do_SPA`.

```

.
.
.
    // Get dataset
    MCobj.Load("data/tab2-data-1.xls");

    decl alldata = MCobj.GetAll();
    alldata = (alldata[][0]-alldata[][1:]).^2;

    // replace the forecast with forecast errors
    MCobj.Renew(alldata, MCobj.GetAllNames()[1:]);

.
.
.

    MCobj.Set_Loss("ident");    // Loss func.: "mse", "mad", "ident"

.
.
.

```

Program 2.2: Sample code Inflation-SW-tab2-SPA-v02.ox to testing for SPA with an user specified loss function.

The interesting lines of codes are:

```

decl alldata = MCobj.GetAll(); extract the data matrix from the MCobj object.
alldata = (alldata[][0]-alldata[][1:]).^2; for each a every date and model, compute
the squared forecast error.

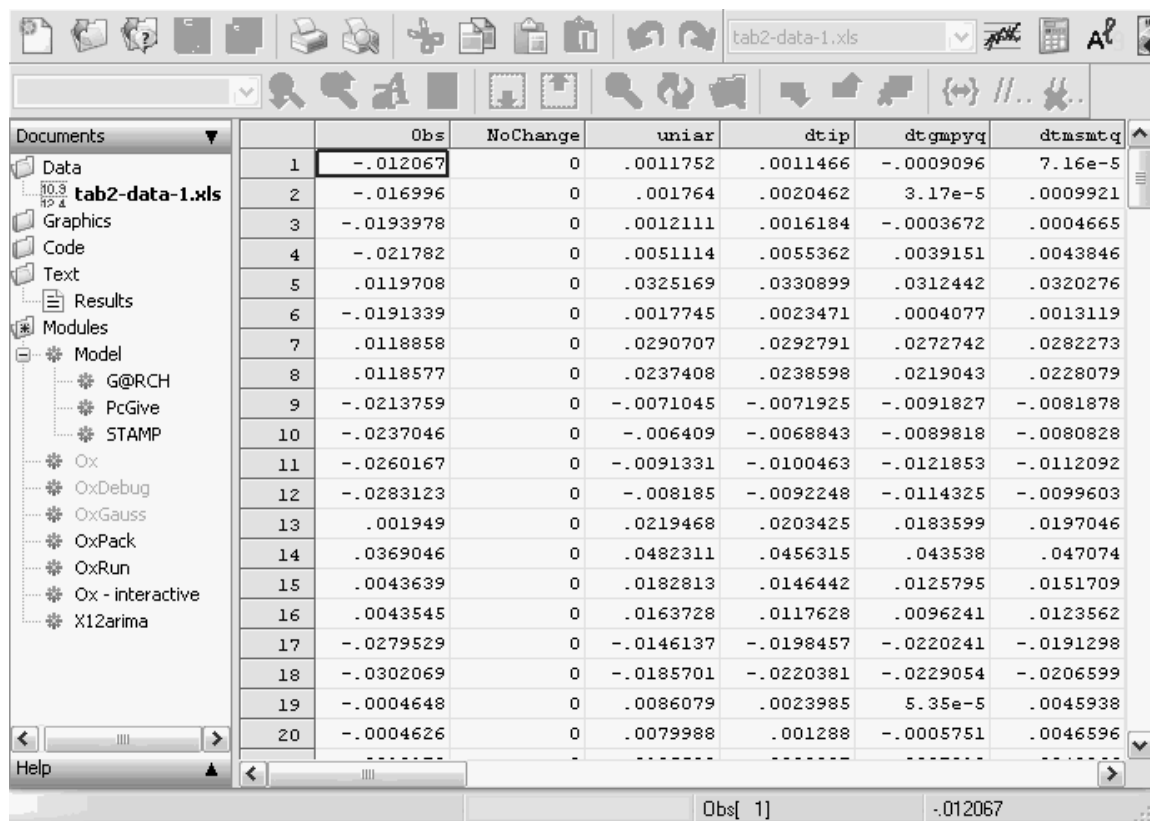
```

`MCobj.Renew(alldata,MCobj.GetAllNames()[1:]);` replace all forecasts with the corresponding squared forecast error.

`MCobj.SetLoss("ident");...` the loss function must be changed to "ident" as the actual losses have already been specified.

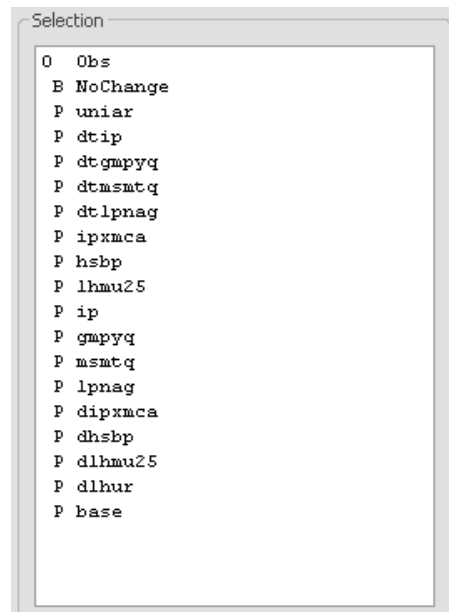
## 2.4. OxPack Application

Below we show the sequence of screen shots that will produce Output 2.1. `tab2-data-1.xls` must be open in OxMetrics:

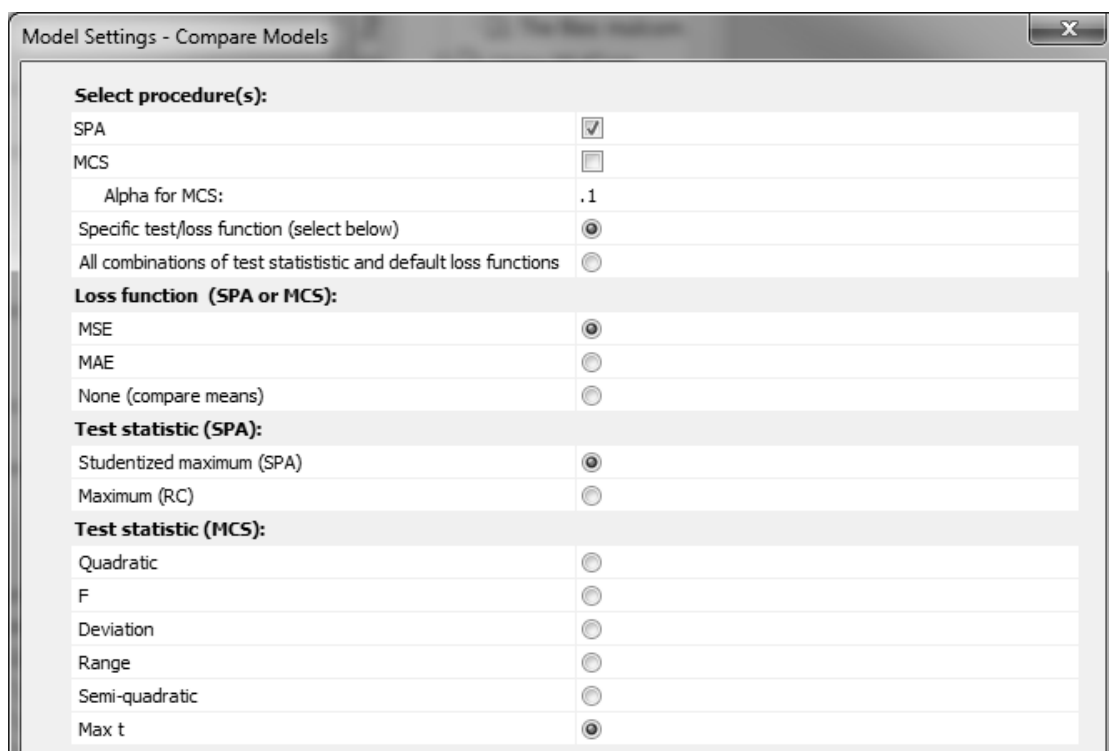


	Obs	NoChange	uniar	dtip	dtgmpyq	dtmsmtq
1	-.012067	0	.0011752	.0011466	-.0009096	7.16e-5
2	-.016996	0	.001764	.0020462	3.17e-5	.0009921
3	-.0193978	0	.0012111	.0016184	-.0003672	.0004665
4	-.021782	0	.0051114	.0055362	.0039151	.0043846
5	.0119708	0	.0325169	.0330899	.0312442	.0320276
6	-.0191339	0	.0017745	.0023471	.0004077	.0013119
7	.0118858	0	.0290707	.0292791	.0272742	.0282273
8	.0118577	0	.0237408	.0238598	.0219043	.0228079
9	-.0213759	0	-.0071045	-.0071925	-.0091827	-.0081878
10	-.0237046	0	-.006409	-.0068843	-.0089818	-.0080828
11	-.0260167	0	-.0091331	-.0100463	-.0121853	-.0112092
12	-.0283123	0	-.008185	-.0092248	-.0114325	-.0099603
13	.001949	0	.0219468	.0203425	.0183599	.0197046
14	.0369046	0	.0482311	.0456315	.043538	.047074
15	.0043639	0	.0182813	.0146442	.0125795	.0151709
16	.0043545	0	.0163728	.0117628	.0096241	.0123562
17	-.0279529	0	-.0146137	-.0198457	-.0220241	-.0191298
18	-.0302069	0	-.0185701	-.0220381	-.0229054	-.0206599
19	-.0004648	0	.0086079	.0023985	5.35e-5	.0045938
20	-.0004626	0	.0079988	.001288	-.0005751	.0046596

Then start OxPack and invoke `MULCOM`. Select `Formulate` in the `Model` drop down window. Now make selections such that the `Selection` area appears as below.



When clicking OK the Model Settings menu will appear. The setting should be as follows:



The selection in the Test statistics (MCS) part has no effect when testing for SPA.

# Chapter 3

## MODEL CONFIDENCE SET

### 3.1. What is a MCS?

The objective of the model confidence set (MCS) procedure is to determine the set,  $\mathcal{M}^*$ , that consists of the ‘best’ model(s) from a collection of models,  $\mathcal{M}_0$ , where ‘best’ is defined in terms of a criterion that is user-specified. The MCS procedure yields a model confidence set,  $\widehat{\mathcal{M}}^*$ , that is a set of models constructed to contain the best models with a given level of confidence. The models in  $\widehat{\mathcal{M}}^*$  are evaluated using sample information about the relative performances of the models in  $\mathcal{M}_0$ . Thus, the MCS is a *random* data-dependent set of models that includes the best forecasting model(s), as a standard confidence interval covers the population parameter.

An attractive feature of the MCS approach is that it acknowledges the limitations of the data. Informative data will result in a MCS that contains only the best model. Less informative data makes it difficult to distinguish between models and may result in a MCS that contains several (or possibly all the) models. Thus, the MCS differs from extant model selection criteria that choose a single model without regard to the information content of the data. Another advantage is that the MCS procedure makes it possible to make statements about significance that are valid in the traditional sense. A property that is not satisfied by the commonly used approach of reporting  $p$ -values from multiple pairwise comparisons. Another attractive feature of the MCS procedure is that it allows for the possibility that more than one model can be the ‘best’, i.e.,  $\mathcal{M}^*$  may contain more than a single model.

A MCS is constructed from a collection of competing objects,  $\mathcal{M}_0$ , and a criterion for evaluating these objects empirically. The MCS procedure is based on an *equivalence test*,  $\delta_{\mathcal{M}}$ ; and an *elimination rule*,  $e_{\mathcal{M}}$ . The equivalence test is applied to the set of objects  $\mathcal{M} = \mathcal{M}_0$ . If  $\delta_{\mathcal{M}}$  is rejected, there is evidence that the models in  $\mathcal{M}$  are not equally ‘good’ and  $e_{\mathcal{M}}$  is used to eliminate an object with poor sample performance from  $\mathcal{M}$ . This procedure is repeated until  $\delta_{\mathcal{M}}$  is ‘accepted’, and the MCS is now defined by the set of ‘surviving’ models. The same significance level,  $\alpha$ , is employed in all tests, which asymptotically guarantees that  $P(\mathcal{M}^* \subset \widehat{\mathcal{M}}_{1-\alpha}^*) \geq 1 - \alpha$ , and in the case where  $\mathcal{M}^*$  consists of one object we have the stronger results that  $\lim_{n \rightarrow \infty} P(\mathcal{M}^* = \widehat{\mathcal{M}}_{1-\alpha}^*) = 1$ . The MCS procedure also yields  $p$ -values for each of the models. For a given model  $i \in \mathcal{M}_0$ , the MCS  $p$ -value,  $\hat{p}_i$ , is the threshold at which  $i \in \widehat{\mathcal{M}}_{1-\alpha}^*$ , if and only if  $\hat{p}_i \geq \alpha$ . Thus, a model with a small MCS  $p$ -value makes it unlikely that model  $i$  is one of the ‘best’ models (is a member of  $\mathcal{M}^*$ ).

### 3.2. Theory for General Confidence Set

In this section, we briefly review the theory of model confidence sets for general objects, using the comparison of forecasting model as our leading example. For more details the reader should consult Hansen et al. (2010).

Consider a set,  $\mathcal{M}_0$ , that contains a finite number of models, indexed by  $i = 1, \dots, m_0$ . The objects are evaluated over the sample  $t = 1, \dots, n$ , in terms of a loss function and we denote the loss that is associated with object  $i$  in period  $t$  as  $L_{i,t}$ .<sup>1</sup>

Define the relative performance variables  $d_{ij,t} \equiv L_{i,t} - L_{j,t}$ , for all  $i, j \in \mathcal{M}_0$ . Then the set of superior objects is defined by

$$\mathcal{M}^* \equiv \{i \in \mathcal{M}_0 : E(d_{ij,t}) \leq 0 \text{ for all } j \in \mathcal{M}_0\}.$$

In the following, we denote by  $\mathcal{M}^\dagger$  the complement to  $\mathcal{M}^*$ , and use  $i^*$  and  $i^\dagger$  to represent typical elements of  $\mathcal{M}^*$  and  $\mathcal{M}^\dagger$ , respectively.

The objective of the MCS procedure is to determine  $\mathcal{M}^*$ . This is done through a sequence of significance tests, where objects that are found to be significantly inferior to other elements of  $\mathcal{M}_0$  are eliminated. The hypotheses that are being tested take the form:

$$H_{0,\mathcal{M}} : E(d_{ij,t}) = 0 \quad \text{for all } i, j \in \mathcal{M}, \quad (3.2.1)$$

where  $\mathcal{M} \subset \mathcal{M}_0$ . We denote the alternative hypothesis ( $E(d_{ij,t}) \neq 0$  for some  $i, j \in \mathcal{M}$ ) by  $H_{A,\mathcal{M}}$ .

The MCS procedure is based on an *equivalence test*,  $\delta_{\mathcal{M}}$ , and an *elimination rule*,  $e_{\mathcal{M}}$ . The equivalence test,  $\delta_{\mathcal{M}}$ , is used to test the hypothesis  $H_{0,\mathcal{M}}$  for any  $\mathcal{M} \subset \mathcal{M}_0$ ,<sup>2</sup> and  $e_{\mathcal{M}}$  identifies the object of  $\mathcal{M}$  that is to be removed from  $\mathcal{M}$ , in the event that  $H_{0,\mathcal{M}}$  is rejected. Hence, the MCS Algorithm is based on the following three steps:

Step 0: Initially set  $\mathcal{M} = \mathcal{M}_0$ .

Step 1: Test  $H_{0,\mathcal{M}}$  using  $\delta_{\mathcal{M}}$  at level  $\alpha$ .

Step 2: If  $H_{0,\mathcal{M}}$  is ‘accepted’ we define the  $\widehat{\mathcal{M}}_{1-\alpha}^* = \mathcal{M}$ , otherwise we use  $e_{\mathcal{M}}$  to eliminate an object from  $\mathcal{M}$  and repeat the procedure beginning with Step 1.

The set,  $\widehat{\mathcal{M}}_{1-\alpha}^*$ , which consists of the set of ‘surviving’ objects (those that survived all tests without being eliminated) is referred to as the *model confidence set*.

#### 3.2.1. MCS $p$ -Values

To define the MCS  $p$ -values, let  $m_0$  denote the number of elements in  $\mathcal{M}_0$ . Moreover, suppose for simplicity that the elements of  $\mathcal{M}_0 = \{1, \dots, m_0\}$  are ordered such that  $k = e_{\mathcal{M}_{(k)}}$  where  $\mathcal{M}_{(k)} = \{k, k+1, \dots, m_0\}$ ,  $k = 1, \dots, m_0$ . Hence  $e_{\mathcal{M}_0} = e_{\mathcal{M}_{(1)}} = 1$  is the first model to be eliminated in the event that  $H_{0,\mathcal{M}_{(1)}}$  is rejected,  $e_{\mathcal{M}_{(2)}} = 2$  is the next model, etc.

**Definition 3.1 (MCS  $p$ -values)** Let  $p(k)$  be the  $p$ -value of the hypothesis  $H_{0,\mathcal{M}_{(k)}}$ , with the convention:  $p(m_0) \equiv 1$ . The MCS  $p$ -value for model  $i \in \mathcal{M}_0$  is defined by  $\hat{p}_i \equiv \max_{k \leq i} p(k)$ .

<sup>1</sup>In the situation where a point forecast,  $\hat{Y}_{i,t}$ , of  $Y_t$  is evaluated in terms of a loss function,  $L$ , we define  $L_{i,t} = L(Y_t, \hat{Y}_{i,t})$ .

<sup>2</sup>We let  $\delta_{\mathcal{M}} = 0$  and  $\delta_{\mathcal{M}} = 1$  correspond to the cases where  $H_{0,\mathcal{M}}$  are ‘accepted’ and ‘rejected’ respectively.

The following table describes how the MCS  $p$ -values are defined and how they relate to the  $p$ -values of the individual tests,  $p(k)$ ,  $k = 1, \dots, m_0$ .

$e_i$	$p$ -value of $H_{0,\mathcal{M}_{(k)}}$	MCS $p$ -value
1	$p(1) = 0.01$	$\hat{p}_1 = 0.01$
2	$p(2) = 0.04$	$\hat{p}_2 = 0.04$
3	$p(3) = 0.02$	$\hat{p}_3 = 0.04$
4	$p(4) = 0.03$	$\hat{p}_4 = 0.04$
5	$p(5) = 0.07$	$\hat{p}_5 = 0.07$
6	$p(6) = 0.04$	$\hat{p}_6 = 0.07$
7	$p(7) = 0.11$	$\hat{p}_7 = 0.11$
8	$p(8) = 0.25$	$\hat{p}_8 = 0.25$
$\vdots$	$\vdots$	$\vdots$
$m_0$	$p(m_0) \equiv 1.00$	$\hat{p}_{m_0} = 1.00$

The MCS  $p$ -values are convenient because they make it easy to determine whether a particular object is in  $\widehat{\mathcal{M}}_{1-\alpha}^*$ . Thus, the MCS  $p$ -values are an effective way of conveying the information in the data.

The interpretation of a MCS  $p$ -value is analogous to that of a classical  $p$ -value. So the MCS  $p$ -value cannot be interpreted as the probability that a particular model is the best model, analogous to the fact that a classical  $p$ -value does not give the probability that a particular hypothesis is true. The MCS is a *random* subset of models, that contains  $\mathcal{M}^*$  with a certain probability, and the probability interpretation of a MCS  $p$ -value is tied to the random nature of the MCS. The analogy to the classical setting is that of a  $(1 - \alpha)$  confidence interval that contains the ‘true’ parameter with a probability no less than  $1 - \alpha$ .

### 3.2.2. Equivalence Tests and Elimination Rules

Now we consider the specific equivalence tests and an elimination rule of Hansen et al. (2010). Let  $\mathcal{M}$  be some subset of  $\mathcal{M}_0$  and let  $m$  be the number of models in  $\mathcal{M} = \{i_1, \dots, i_m\}$ . We define the vector of loss-variables,  $\mathbf{L}_t \equiv (L_{i_1,t}, \dots, L_{i_m,t})'$ ,  $t = 1, \dots, n$ , and its sample average,  $\bar{\mathbf{L}} \equiv n^{-1} \sum_{t=1}^n \mathbf{L}_t$ , and we let  $\mathbf{1} \equiv (1, \dots, 1)'$  be the column vector where all  $m$  entries equal one. The orthogonal complement to  $\mathbf{1}$ , is an  $m \times (m - 1)$  matrix,  $\mathbf{1}_\perp$ , that has full column rank and satisfies  $\mathbf{1}_\perp' \mathbf{1} = 0$  (an vector of zeros). The  $m - 1$  dimensional vector  $\mathbf{X}_t \equiv \mathbf{1}_\perp' \mathbf{L}_t$  can be viewed as  $m - 1$  contrasts. Define  $\boldsymbol{\mu} \equiv E(\mathbf{X}_t)$ , then the null hypothesis  $H_{0,\mathcal{M}}$  is equivalent to  $\boldsymbol{\mu} = \mathbf{0}$  and it holds that  $n^{1/2}(\bar{\mathbf{X}} - \boldsymbol{\mu}) \xrightarrow{d} N(\mathbf{0}, \boldsymbol{\Sigma})$ , where  $\bar{\mathbf{X}} \equiv n^{-1} \sum_{t=1}^n \mathbf{X}_t$  and  $\boldsymbol{\Sigma} \equiv \lim_{n \rightarrow \infty} \text{var}(n^{1/2}\bar{\mathbf{X}})$ . So  $H_{0,\mathcal{M}}$  can be tested using traditional quadratic-form tests, such as those that are based on the test statistics

$$T_Q \equiv n \bar{\mathbf{X}}' \hat{\boldsymbol{\Sigma}}^\# \bar{\mathbf{X}} \xrightarrow{d} \chi_{(q)}^2 \quad \text{and} \quad T_F \equiv \frac{n - q}{q(n - 1)} T_Q \xrightarrow{d} F_{(q, n - q)},$$

where  $\hat{\boldsymbol{\Sigma}}$  is some consistent estimator of  $\boldsymbol{\Sigma}$ ,  $q \equiv \text{rank}(\hat{\boldsymbol{\Sigma}})$ , and  $\hat{\boldsymbol{\Sigma}}^\#$  denotes the Moore-Penrose inverse of  $\hat{\boldsymbol{\Sigma}}$ . Here  $q$  denotes the effective number of *contrasts* (the number of linearly independent comparisons) under  $H_{0,\mathcal{M}}$ , and  $\chi_{(q)}^2$  denotes the  $\chi^2$ -distribution with  $q$  degrees of freedom and  $F_{(q, n - q)}$  is the  $F$ -distribution with  $(q, n - q)$  degrees of freedom

An empirical problem arises when the number of elements,  $m$ , become large relative to the sample size,  $n$ . In this case, it is useful to consider alternative tests that do not require an estimate of the  $(m -$



1)  $\times (m - 1)$  covariance matrix,  $\Sigma$ . Such tests can be constructed from the  $t$ -statistics

$$t_{ij} = \frac{\bar{d}_{ij}}{\sqrt{\widehat{\text{var}}(\bar{d}_{ij})}} \quad \text{and} \quad t_{i\cdot} = \frac{\bar{d}_{i\cdot}}{\sqrt{\widehat{\text{var}}(\bar{d}_{i\cdot})}}, \quad \text{for } i, j \in \mathcal{M},$$

where we have defined  $\bar{d}_{ij} \equiv n^{-1} \sum_{t=1}^n d_{ij,t}$  and  $\bar{d}_{i\cdot} \equiv m^{-1} \sum_{j \in \mathcal{M}} \bar{d}_{ij}$ . The variable  $\bar{d}_{ij}$  measures the sample loss differential between model  $i$  and  $j$ , whereas  $\bar{d}_{i\cdot}$  is a contrast of model  $i$ 's sample loss to that of the average across all models.

The null hypothesis,  $H_{0,\mathcal{M}}$ , is equivalent to  $E(\bar{d}_{i\cdot}) = 0$  for all  $i \in \mathcal{M}$ , (and equivalent to  $E(\bar{d}_{ij}) = 0$  for all  $i, j \in \mathcal{M}$  by definition). Test statistics, such as  $T_{\max} = \max_{i \in \mathcal{M}} t_{i\cdot}$ ,  $T_D \equiv \sum_{i \in \mathcal{M}} t_{i\cdot}^2$ ,  $T_R \equiv \max_{i,j \in \mathcal{M}} |t_{ij}|$ , and  $T_{SQ} \equiv \sum_{i,j \in \mathcal{M}} t_{ij}^2$ , can be used to test the hypothesis  $H_{0,\mathcal{M}}$ . The subscripts refer to *maximum deviation*, *deviation* (from common average), *range*, and *semi-quadratic*, respectively.

The asymptotic distributions of the test statistics,  $T_{\max}$ ,  $T_D$ ,  $T_R$ , and  $T_{SQ}$ , are non-standard because they depend on nuisance parameters (under both the null and the alternative). However, this poses no obstacle as their distributions are easily estimated using bootstrap methods that implicitly solve the nuisance parameter problem. For details about the implementation of the bootstrap we refer to Hansen et al. (2010)

For the test statistic  $T_{\max}$  the natural elimination rule is  $e_{\max,\mathcal{M}} \equiv \arg \max_{i \in \mathcal{M}} t_{i\cdot}$  because a rejection of the null hypothesis identifies the hypothesis  $\mu_{j\cdot} = 0$  as false, for  $j = e_{\max,\mathcal{M}}$ . In this case the elimination rule removes the model that contributes most to the test statistic. This model has the largest standardized excess loss relative to the average across all models in  $\mathcal{M}$ . The current implementation uses  $\arg \max_i t_{i\cdot}$  as the elimination rule together with the  $T_Q$ ,  $T_F$ ,  $T_D$ ,  $T_R$ ,  $T_{SQ}$  or  $T_{\max}$  equivalence test in the MCS algorithm.

### 3.3. Interpreting MCS Results

When running the standard configuration of the a MCS estimation using MULCOM the result will appear as displayed in Output 3.1. The first part contains some information about estimation configuration, that is the number of competing models, sample size, loss function, equivalence test statistic and bootstrap parameters. Next follows the list of all models in the model space together with their loss and their MCS  $p$ -value. Finally,  $\widehat{\mathcal{M}}_{1-\alpha}^*$  is presented for the requested  $\alpha$ -level.

```

----- Ox at 22:34:48 on 13-Dec-2010 -----

Ox Professional version 6.00 (Windows/U/MT) (C) J.A. Doornik, 1994-2009
MulCom package version 2.00, object created on 13-12-2010
By Asger Lunde and Peter R. Hansen

----- MODEL CONFIDENCE SET ESTIMATION -----
Number of models:      l=18
Sample size:           n=168
Loss function:         mse
Test Statistic:        MaxT
Resample by:           BlockBootResamp
Bootstrap parameters:  B=10000 (resamples), d=2 (block length)

Model Name              mse(*10^3)      MCS p-val.
NoChange                1.08227      0.0000
uniar                   0.71582      0.0003
dtip                    0.63440      0.0012
dtgmpyq                 0.69928      0.0002
dtmsmtq                 0.54816      0.0255
dtlpnag                 0.61621      0.0016
ipxmca                  0.56309      0.0070
hsbp                    0.48615      0.6838 *
lhmu25                  0.59208      0.0023
ip                      0.56827      0.0055
gmpyq                   0.49850      0.4079 *
msmtq                   0.47035      1.0000 *
lpnag                   0.53272      0.0462
dipxmca                 0.55470      0.0097
dhsbp                   0.72934      0.0006
dlhmu25                 0.55303      0.0132
dlhur                   0.53882      0.0629
base                    0.56957      0.0037

Level 0.1 Model Confidence Set

Model Name              mse(*10^3)      MCS p-val.
hsbp                    0.48615      0.6838 *
gmpyq                   0.49850      0.4079 *
msmtq                   0.47035      1.0000 *

```

Output 3.1: MCS results. For code see Program 3.1

### 3.4. OxEdit Application

In this section we show how to use MULCOM through in OxEdit. In Program 3.1 we present the sample code that produced Output 3.1.

```

/*-----
 * Code (C) Lunde & Hansen, 2010
 *
 * FILE: Inflation-SW-tab2-MCS-v01.ox
 * Example: Use MulCom to Estimate Model Confidence Set (MCS)
 *
 *-----*/

#include <oxstd.h>
#import <packages/MulCom/MulCom>

main()
{
    decl MCobj = new MulCom();          // Declare MulCom object

    MCobj.Load("data/tab2-data-1.xls"); // Get dataset

    // Select realization, benchmark, and forecasts
    MCobj.Select(O_VAR, {"Obs",0,0} );
    MCobj.SelectByIndex(P_VAR, <1:23>, 0, 0 );

    // Select sample period
    MCobj.SetSelSample(-1, 1, -1, 1);

    // Select test specification
    MCobj.Do_MCS(0.1,2,10000);          // alpha, block-length, #resamples
    MCobj.Set_Loss("mse");              // Loss func.: "mse", "mad", "ident"
    MCobj.Set_TestStatMCS("maxT");      // Test stat.: "maxT", "Chi", "F",
                                         //              "ComAve", "Range", "SemiQ"

    MCobj.Set_MCS_Save_Name("output\\", "tab-1");
    MCobj.DoEstimation();

    delete MCobj;
}

```

Program 3.1: Sample code Inflation-SW-tab2-MCS-v01.ox to estimate a MCS using OxEdit

`decl MCobj = new MulCom();` declare an object of the MULCOM class.

`MCobj.Load("data/tab2-data-1.xls");` reads in the relevant data set.

`MCobj.Select(O_VAR, "Obs", 0, 0 );` specify the target of the forecast.

`MCobj.SelectByIndex(P_VAR, <1:23>, 0, 0 );` specify the forecasts to be included in the comparison.

`MCobj.SetSelSample(-1, 1, -1, 1);` select the sample size.

`MCobj.Do_MCS(0.1, 2, 10000);` the first number is the  $\alpha$ -level to be used for  $\widehat{\mathcal{M}}_{1-\alpha}^*$ . The second number is the block-length for the block bootstrap, and third is the number of bootstrap resamples that are used to assess the equivalence tests.

`MCobj.Set_Loss("mse");` select the loss function.

`MCobj.Set_TestStat("MaxT");` select the equivalence test .

`MCobj.Set_MCS_Save_Name("output\\", "tab-1");` specify the path and filename where the MCS automatic output should go.

`MCobj.DoEstimation();` run the test.

`delete MCobj;` delete the MULCOM object.

To run the program in OxEdit simply press control-r. For more details regarding Ox in conjunction with OxEdit consult Doornik and Ooms (2005).

Alternatively one may desire some other loss function. In Program 3.2 we present some code that also will produce Output 3.1, but with the loss function designed before calling `MCobj.Do_MCS`.

```

/*-----
 * Code (C) Lunde & Hansen, 2010
 *
 * FILE: Inflation-SW-tab2-MCS-v02.ox
 * Example: Use MulCom to Estimate Model Confidence Set (MCS)
 *          using a user specified loss
 *-----*/

#include <oxstd.h>
#import <packages/MulCom/MulCom>

main()
{
    decl MCobj = new MulCom();          // Declare MulCom object

    MCobj.Load("data/tab2-data-1.xls"); // Get dataset

    decl alldata = MCobj.GetAll();
    alldata = (alldata[][0]-alldata[][1:]).^2;

    // replace the forecast with forecast errors
    MCobj.Renew(alldata,MCobj.GetAllNames()[1:]);

    // Select forecasts errors to compare
    MCobj.SelectByIndex(P_VAR, <1:23>, 0, 0 );

    // Select sample period
    MCobj.SetSelSample(-1, 1, -1, 1);

    // Select test specification
    MCobj.Do_MCS(0.1,2,10000);          // alpha, block-length, #resamples
    MCobj.Set_Loss("mse");               // Loss func.: "mse", "mad", "ident"
    MCobj.Set_TestStatMCS("maxT");       // Test stat.: "maxT", "Chi", "F",
                                         //          "ComAve", "Range", "SemiQ"

    MCobj.Set_MCS_Save_Name("output\\", "tab-1");
    MCobj.DoEstimation();

    delete MCobj;
}

```

Program 3.2: Sample code `Inflation-SW-tab2-MCS-v02.ox` to estimate a MCS with an user specified loss function

The interesting lines of codes are:

`decl alldata = MCobj.GetAll();` extract the data matrix from the `MCobj` object.

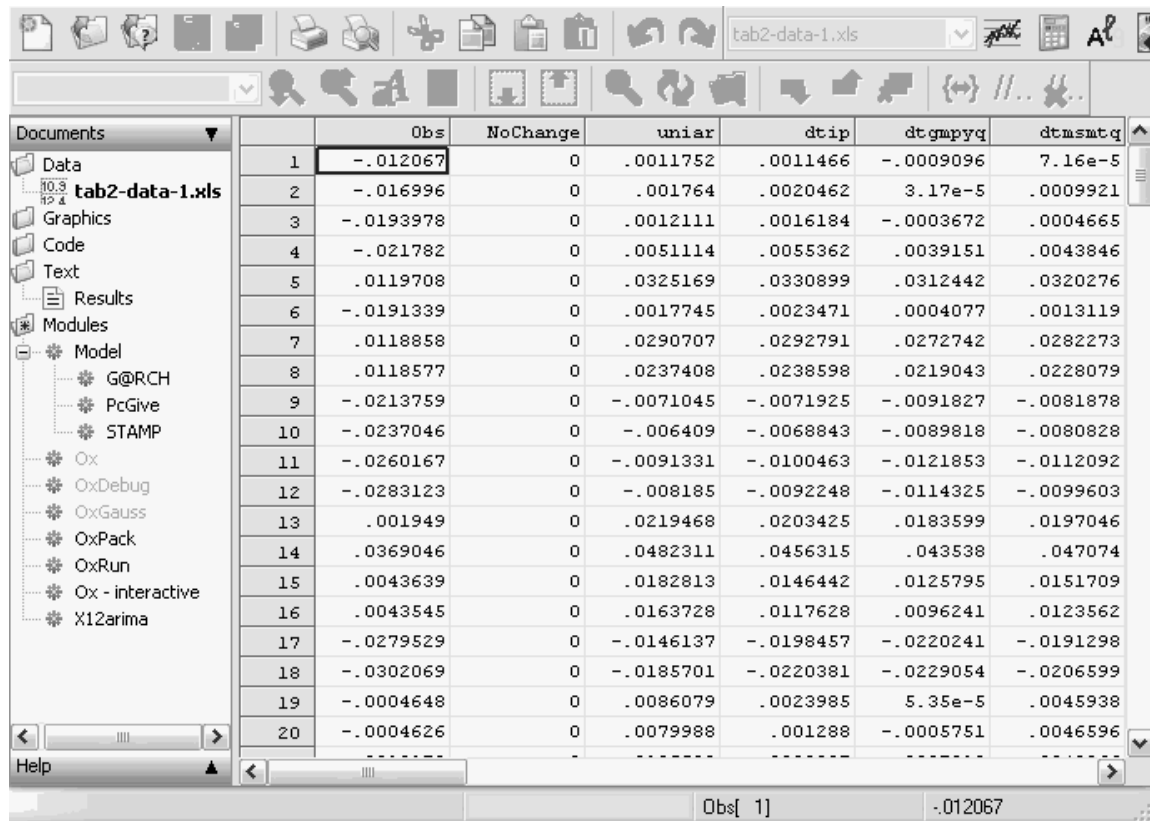
`alldata = -(alldata[][0]-alldata[][1:]).^2;` for each a every date and model, compute the squared forecast error.

`MCobj.Renew(alldata,MCobj.GetAllNames()[1:]);` replace all forecasts with the corresponding squared forecast error.

### 3.5. OxPack Application

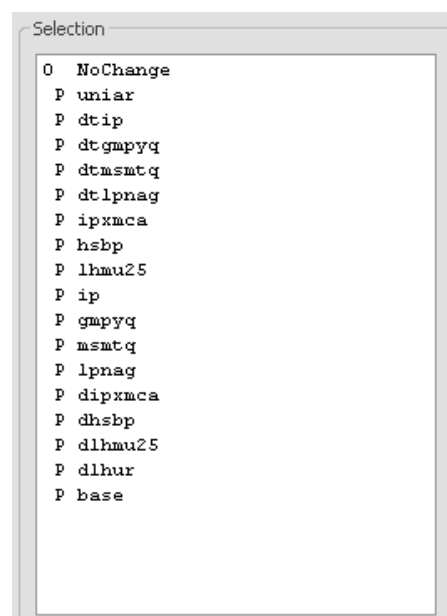
In this section we show the sequence of screen shots that will produce Output 3.1.

The data set, `tab2-data-1.xls`, should be open in OxMetrics:



	Obs	NoChange	uniar	dtip	dtgmpyq	dtmsmtq
1	-.012067	0	.0011752	.0011466	-.0009096	7.16e-5
2	-.016996	0	.001764	.0020462	3.17e-5	.0009921
3	-.0193978	0	.0012111	.0016184	-.0003672	.0004665
4	-.021782	0	.0051114	.0055362	.0039151	.0043846
5	.0119708	0	.0325169	.0330899	.0312442	.0320276
6	-.0191339	0	.0017745	.0023471	.0004077	.0013119
7	.0118858	0	.0290707	.0292791	.0272742	.0282273
8	.0118577	0	.0237408	.0238598	.0219043	.0228079
9	-.0213759	0	-.0071045	-.0071925	-.0091827	-.0081878
10	-.0237046	0	-.006409	-.0068843	-.0089818	-.0080828
11	-.0260167	0	-.0091331	-.0100463	-.0121853	-.0112092
12	-.0283123	0	-.008185	-.0092248	-.0114325	-.0099603
13	.001949	0	.0219468	.0203425	.0183599	.0197046
14	.0369046	0	.0482311	.0456315	.043538	.047074
15	.0043639	0	.0182813	.0146442	.0125795	.0151709
16	.0043545	0	.0163728	.0117628	.0096241	.0123562
17	-.0279529	0	-.0146137	-.0198457	-.0220241	-.0191298
18	-.0302069	0	-.0185701	-.0220381	-.0229054	-.0206599
19	-.0004648	0	.0086079	.0023985	5.35e-5	.0045938
20	-.0004626	0	.0079988	.001288	-.0005751	.0046596

Then start OxPack and invoke `MULCOM`. Select `Formulate` in the Model drop down window. Now make selections such that the Selection area appears as below.



```

Selection
0 NoChange
P uniar
P dtip
P dtgmpyq
P dtmsmtq
P dtlpnag
P ipxmca
P hsbp
P lhm25
P ip
P gmpyq
P msmtq
P lpnag
P dipxmca
P dhsbp
P dlhm25
P dlhur
P base
  
```

When clicking OK the Model Settings menu will appear. The setting should be as follows:

**Model Settings - Compare Models**

**Select procedure(s):**

SPA	<input type="checkbox"/>
MCS	<input checked="" type="checkbox"/>

Alpha for MCS: .1

Specific test/loss function (select below) ☒

All combinations of test statistic and default loss functions ☐

**Loss function (SPA or MCS):**

MSE	<input checked="" type="radio"/>
MAE	<input type="radio"/>
None (compare means)	<input type="radio"/>

**Test statistic (SPA):**

Studentized maximum (SPA)	<input checked="" type="radio"/>
Maximum (RC)	<input type="radio"/>

**Test statistic (MCS):**

Quadratic	<input type="radio"/>
F	<input type="radio"/>
Deviation	<input type="radio"/>
Range	<input type="radio"/>
Semi-quadratic	<input type="radio"/>
Max t	<input checked="" type="radio"/>

The selection in the Test statistics (SPA) part has no effect when estimating a MCS. EndExpansion

# Chapter 4

---

## MULCOM REFERENCE

The `MULCOM` class derives from `MODELBASE`, which in turn derives from `DATABASE`. In this chapter the most common `MULCOM` functions are summarized. Consult the header file `mulcom.h` for definitions of member variables, and undocumented functions, such as those for communication with `OxPack`.

### 4.1. `MULCOM` summary

#### Constructor

<code>MulCom</code>	Constructor
---------------------	-------------

#### Test formulation

<code>Do_MCS</code>	Do MCS and specify options
<code>Do_SPA</code>	Do SPA and specify options
<code>Set_Loss</code>	Specify the loss function
<code>Set_MCS_Save_Name</code>	Where to save MCS information
<code>Set_PrintLevel</code>	How much output to print to screen
<code>Set_TestStatMCS</code>	Specify the test statistic
<code>Set_TestStatSPA</code>	Specify the test statistic
<code>GetMCSpval</code>	Returns the MCS p-values

### 4.2. `MULCOM` member functions

This section documents the main member functions of `MULCOM` in alphabetical order.

#### `MulCom::MulCom`

`MulCom( ) ;`

*No return value.*

*Description*

Constructor function.

#### `MulCom::Do_SPA`

`Do_SPA(const statboot_q, const cresamples);`

`statboot_q`    in: double, dependence parameter for the stationary bootstrap  
`cresamples`   in: int, number of bootstrap resamples

*No return value.*

*Description*

Specify to test for SPA. The first number is the dependence parameter for the stationary bootstrap, and the second is the number of bootstrap resamples.

### **MulCom::Do\_MCS**

`Do_MCS(const TheAlpha, const cblocklength, const cresamples);`

`TheAlpha`        in: double,  $\alpha$ -level to be used for  $\widehat{\mathcal{M}}_{1-\alpha}^*$   
`cblocklength`   in: int, block-length for the block bootstrap  
`cresamples`     in: int, number of bootstrap resamples

*No return value.*

*Description*

Specify that a MCS should be estimated. Specify the  $\alpha$ -level to be used for  $\widehat{\mathcal{M}}_{1-\alpha}^*$ . The second number is the block-length for the block bootstrap, and third is the number of bootstrap resamples that are used to assess the equivalence tests.

### **MulCom::Get\_MCSpval**

`Get_MCSpval();`

*Return value.* A matrix with the loss and the MCS  $p$ -values for each model.

*Description*

Get the MCS  $p$ -values.

### **MulCom::Set\_Loss**

`Set_Loss(const NameLoss);`

`NameLoss`    in: string, name of loss function

*No return value.*

*Description*

Select the loss function, choices are:

`mse`        : mean squared error  
`mad`        : mean absolute deviation  
`ident`     : identity

### **MulCom::Set\_MCS\_Save\_Name**

`Set_MCS_Save_Name(const Path, const ForecFilename);`

`Path`                in: string, the path to where you want to place the MCS information  
`ForecFilename`    in: string, name of file to hold output with MCS information

*No return value.*

*Description*



Name of file to hold output with MCS information

**MulCom::Set\_PrintLevel**

```
Set_PrintLevel(const Prt);
```

Prt: int, 0: Print nothing, 1: Print final MCS results, 2: Print also step by step progress

*No return value.*

*Description*

Name of file to hold output with MCS information

**MulCom::Set\_TestStatMCS**

```
Set_TestStat(const NameTest);
```

    NameTest  in:  string, name of test statistic

*No return value.*

*Description*

Select the test statistic:

maxT	:	$T_{\max}$ , the max deviation to the average loss
chi	:	$T_Q$ , traditional quadratic-form test
F	:	$T_F$ , as chi but which sample size correction
ComAve	:	$T_D$ , deviation from common average
Range	:	$T_R$ , range based statistic
SemiQ	:	$T_{SQ}$ , semi-quadratic

**MulCom::Set\_TestStatSPA**

```
Set_TestStat(const NameTest);
```

    NameTest  in:  string, name of test statistic

*No return value.*

*Description*

Select the test statistic:

TestStatScaledMax	:	studentized maximum (SPA)
TestStatMax	:	maximum (RC)

# Bibliography

---

- Doornik, J. A. (2006), *Ox: An Object-Orientated Matrix Programming Language*, 5 edn, Timberlake Consultants Ltd., London. [1](#)
- Doornik, J. A. and Hendry, D. F. (2006), *OxMetrics: An Interface to Empirical Modelling*, 5 edn, Timberlake Consultants Ltd., London. [1](#)
- Doornik, J. A. and Ooms, M. (2005), *Introduction to Ox*, 5 edn, Timberlake Consultants Ltd., London. [3](#), [12](#), [20](#)
- Hansen, P. R. (2005), ‘A test for superior predictive ability’, *Journal of Business and Economic Statistics* **23**, 365–380. [1](#), [2](#), [9](#)
- Hansen, P. R. and Lunde, A. (2005), ‘A forecast comparison of volatility models: Does anything beat a GARCH(1,1)?’, *Journal of Applied Econometrics* **20**, 873–889. [9](#)
- Hansen, P. R., Lunde, A. and Nason, J. M. (2010), ‘Model confidence sets for forecasting models’. Forthcoming in *Econometrica*. [1](#), [2](#), [16](#), [17](#), [18](#)
- Koopman, S. J., Jungbacker, B. and Hol, E. (2005), ‘Forecasting daily variability of the S&P 100 stock index using historical, realised and implied volatility measurements’, *Journal of Empirical Finance* **12**, 445–475. [9](#)
- Stock, J. H. and Watson, M. W. (1999), ‘Forecasting inflation’, *Journal of Monetary Economics* **44**, 293–335. [2](#)
- White, H. (2000), ‘A reality check for data snooping’, *Econometrica* **68**, 1097–1126. [1](#)

# Index

---

MULCOM Summary, [25](#)

MULCOM member functions, [25](#)

Comparing Multiple Populations, [25](#)

Example files, [2](#)

Installation, [2](#)

Model Confidence Set, [15](#)

    OxEdit application, [19](#)

    OxPack application, [21](#)

Ox console version, [2](#)

Ox Professional, [2](#)

Superior Predictive Ability, [8](#)

$p$ -values, [9](#)

    critical values, [9](#)

    OxEdit application, [11](#)

    OxPack application, [13](#)

    Relation to the reality check, [9](#)

    Test, [8](#)

    User specified loss function, [12](#)

Using MULCOM , [3](#)

    console application, [3](#)

    OxEdit application, [3](#)

    OxPack application though OxMetrics, [3](#)