The Stata Journal (*yyyy*) *vv*, Number *ii*, pp. 1–15

Conditional Evaluation of Predictive Models: The cspa Command

Jia Li
Singapore Management University
jiali@smu.edu.sg

Zhipeng Liao
UCLA
zhipeng.liao@econ.ucla.edu

Rogier Quaedvlieg
Erasmus University Rotterdam
quaedvlieg@ese.eur.nl

Wenyu Zhou
Zhejiang University
wenyuzhou@intl.zju.edu.cn

Abstract. In this article, we introduce a command, `cspa`, that implements the conditional superior predictive ability (CSPA) test developed in Li et al. (2021). With the conditional performance of predictive methods measured nonparametrically by the conditional expectation functions of their predictive losses, we test the null hypothesis that a benchmark model weakly outperforms a collection of competitors uniformly across the conditioning space. The proposed command can be used to implement this test for both independent cross-sectional data and serially dependent time-series data. Confidence sets for the most superior model can be obtained by inverting the test, for which the `cspa` command also offers a convenient implementation.

Keywords: st????, `cspa`, conditional moment inequality, forecast evalution, functional inference, series estimation.

1 Introduction

Making accurate predictions is a central task of data analysis. Economists in central banks routinely update their forecasts on a broad range of macroeconomic indicators to inform policy making. Investors in financial markets devote a tremendous amount of effort to predict asset prices' movements. Data analytics for various socioeconomic activities has also simulated new generations of predictive models on the "micro" level for firms, communities, households, and individuals.

In view of the evergrowing abundance of potentially good predictive methods, evaluating their relative performance is evidently of great practical importance. Arguably the most popular evaluation method in economic applications is the pseudo out-of-sample Diebold–Mariano test (Diebold and Mariano (1995)) for the null hypothesis that the ex post losses (e.g., the squared prediction error) of two predictive methods are equal in expectation. Diebold and Mariano's proposal is effectively a t-test, which may be implemented by regressing the observed loss differential between the competing methods on a constant term, and check whether the estimated intercept is significantly different from zero. This can be easily done in Stata via [R] **regress** for independent data or, more generally, [TS] **newey** for data with serial dependence.

Since the Diebold–Mariano test is based on the *unconditional* average performance of the competing predictions, it invariably "integrates out" the heterogeneity across subsample periods (e.g., expansion or recession episodes) and/or sub-populations (e.g., income or age groups), and hence, may be too coarse for certain practical applications. Unraveling the state- or characteristic-dependent relative performance among predictive models naturally calls for a *conditional* evaluation approach.

In this article, we propose a new Stata command, `cspa`, which offers a convenient implementation for the *conditional superior predictive ability* (CSPA) test recently proposed by Li et al. (2021). The null hypothesis of interest states that the conditional expected loss of a benchmark predictive model is no larger than those of the competing models *uniformly* across all conditioning states (specified by a conditioning variable chosen a priori by the user). In this sense, the CSPA null hypothesis asserts that the benchmark is a uniformly weakly dominating method among all predictive models under consideration. "Passing" the test indicates that the benchmark method is likely to perform well not only on average, but also across all sub-populations "sliced" by the conditioning variable, which may be a macroeconomic cyclical indicator, a financial risk measure, or an individual characteristic, chosen by the user depending on the empirical context.

The main functionality of the proposed `cspa` command is to implement the CSPA test for a given benchmark against a collection of competitors. A rejection indicates that some competitor outperforms the benchmark over some conditioning states. A non-rejection, on the other hand, suggests that the benchmark is weakly dominating. Moreover, by rotating the benchmark role across all models, we may form the *confidence set for the most superior* (CSMS) as the collection of all non-rejected benchmarks. This operation can also be accomplished easily by calling the `csms` option. We shall demonstrate how to carry out these inferential tasks in an empirical example on asset volatility forecasting using a dataset from Li et al. (2021).

The remainder of this article is organized as follows. Section 2 provides a brief review on the CSPA testing procedure and the underlying intuition. Section 3 documents the syntax of the `cspa` command and available options. Section 4 demonstrates the key functionalities of the proposed command in an empirical example.

2 The conditional evaluation method

This section provides a brief review of the CSPA test developed in Li et al. (2021). Section 2.1 describes the setting. Section 2.2 details the CSPA testing procedure. To simplify the discussion, we mainly focus on the time-series setting, with the understanding that random samples with independent observations may be considered as a special "time series" with no serial dependence.

2.1 The setting and hypotheses of interest

Let $(F_t^\dagger)_{1\leq t\leq n}$ denote the time series to be predicted. Correspondingly, we consider a series of benchmark predictions $\{F_{0,t}\}_{1\leq t\leq n}$, and a collection of J competing alternatives, $\{F_{j,t}\}_{1\leq t\leq n}$, $1\leq j\leq J$. The performance of these predictive methods is gauged by a user-specified loss function $L(\cdot,\cdot)$, so that the loss for model j at period t is $L(F_t^\dagger, F_{j,t})$. Commonly used loss functions include the quadratic loss, absolute deviation loss, and Stein's loss. The performance of the benchmark method relative to the jth competing alternative is thus measured by the loss differential series

$$Y_{j,t} = L(F_t^\dagger, F_{j,t}) - L(F_t^\dagger, F_{0,t}).$$

Its conditional mean function given a user-specified conditioning variable X_t is further defined as

$$h_j(x) \equiv \mathbb{E}[Y_{j,t}|X_t = x]. \tag{1}$$

Note that $h_j(x)\geq 0$ indicates that the benchmark method is expected to (weakly) outperform the jth competitor conditional on $X_t = x$.

The null hypothesis of the CSPA test asserts that

$$H_0:\ h_j(x)\geq 0,\quad \text{for all } x\in\mathcal{X},\quad 1\leq j\leq J, \tag{2}$$

where $\mathcal{X}$ is the support of X_t. Under the null hypothesis, the benchmark method outperforms all competing alternatives uniformly across all conditioning states. Evidently, by the law of iterated expectations, this also implies that the unconditional expected loss of the benchmark is lower than those of the competing methods (i.e., $\mathbb{E}[Y_{j,t}]\geq 0$). But the CSPA null hypothesis is generally much more stringent than its unconditional counterpart. As such, the (uniform) conditional dominance criterion may help the researcher discriminate competing predictive methods that appear "unconditionally similar," and so, complements conventional evaluation methods such as the Diebold–Mariano test.

2.2 The testing procedure

We now detail how the `cspa` command implements the CSPA test in the background. The first step is to nonparametrically estimate the conditional expectation functions of the loss differentials (i.e., $\{h_j(\cdot): 1\leq j\leq J\}$) by running series regressions. Specifically, let $P(x) = (p_1(x),\ldots,p_m(x))^\top$ be an m-dimensional vector of approximating basis functions. For each j, we regress $Y_{j,t}$ on $P(X_t)$ with the resulting regression coefficient given by

$$\hat{b}_j = \widehat{Q}^{-1}\Big(n^{-1}\sum_{t=1}^n P(X_t)Y_{j,t}\Big)\quad \text{where}\quad \widehat{Q} = n^{-1}\sum_{t=1}^n P(X_t)P(X_t)^\top.$$

The nonparametric series estimator for $h_j(\cdot)$ is then constructed as

$$\hat{h}_j(\cdot) = P(\cdot)^\top \hat{b}_j. \tag{3}$$

The underlying nonparametric inference theory requires the number of series terms $m \to \infty$ in large samples, so that the unknown conditional expectation functions can be well approximated.

The current version of `cspa` employs Legendre polynomials to form the approximating functions $P(X_t)$. An important property of the Legendre polynomials is that they are orthogonal on the $[-1, 1]$ interval with respect to the uniform distribution. This orthogonality property helps mitigate the multicollinearity among series terms, and hence, improves the numerical stability of the estimation procedure. Other types of orthogonal series basis may be adopted to serve the same purpose as well, which is left for future development. To better exploit the orthogonality of Legendre polynomials, it is advisable (though not required in theory) to perform a preliminary transformation on the conditioning variable X_t so as to make it approximately uniformly distributed on the $[-1, 1]$ interval; see the `method` option for available choices provided in `cspa`.

To carry out the test, it is crucial to account for the sampling variability of the functional estimates $\hat{h}_j(\cdot)$. Let $\hat{u}_{j,t} = Y_{j,t} - \hat{h}_j(X_t)$ be the residual from the jth regression and collect them in a J-dimensional column vector $\hat{u}_t$. We estimate the joint variance-covariance matrix for the series regression coefficients $\hat{b}_j$, $1 \le j \le J$, using

$$\widehat{\Omega} \equiv \big(I_J \otimes \widehat{Q}\big)^{-1} \widehat{A} \big(I_J \otimes \widehat{Q}\big)^{-1},$$

where I_J denotes the $J \times J$ identity matrix, $\otimes$ is the Kronecker product, and $\widehat{A}$ is a Newey–West estimator for the "long-run" covariance matrix of the Jm-dimensional vector $\hat{u}_t \otimes P(X_t)$. The lag parameter for the Newey–West estimator may be set via the `lag(#)` option in `cspa`; also see [TS] **newey**. Note that, if it is known a priori that there is no serial correlation in the data (e.g., when the observations are assumed to be independent), one may set `lag(0)`, which is also the default option in `cspa`. We further partition $\widehat{\Omega}$ into $J \times J$ blocks of $m \times m$ sub-matrices. The standard error function of $\hat{h}_j(\cdot)$ is then estimated as

$$\hat{\sigma}_j(x) \equiv (P(x)^\top \widehat{\Omega}(j,j) P(x))^{1/2},$$

where $\widehat{\Omega}(j,j)$ is the (j,j) block extracted from the aforementioned partition of $\widehat{\Omega}$.

At a significance level α, the rejection decision of the CSPA test is determined as follows.

Step 1. Simulate a Jm-dimensional normal random vector $(\xi_1^{*\top}, \ldots, \xi_J^{*\top}) \sim \mathcal{N}(0, \widehat{\Omega})$, where each ξ_j^* is m-dimensional. Set $\hat{t}_j^*(x) \equiv P(x)^\top \xi_j^* / \hat{\sigma}_j(x)$.

Step 2. Repeat step 1 many times. For some constant $z > 0$, define $\hat{K}$ as the $1 - z/\log(n)$-quantile of $\max_{1 \le j \le J} \sup_{x \in \mathcal{X}} \hat{t}_j^*(x)$ in the simulated sample and then set

$$\widehat{V} \equiv \Big\{ (j,x) : \hat{h}_j(x) \le \min_{1 \le j \le J} \inf_{x \in \mathcal{X}} \big(\hat{h}_j(x) + n^{-1/2} \hat{K} \hat{\sigma}_j(x) \big) + 2 n^{-1/2} \hat{K} \hat{\sigma}_j(x) \Big\}. \quad (4)$$

The default value of z is 0.1, which may be modified by calling the `ais(#)` option.

Step 3. Set $\hat{k}_{1-\alpha}$ as the $(1-\alpha)$-quantile of $\sup_{(j,x)\in\widehat{V}} \hat{t}_j^*(x)$. Reject the null hypothesis (2) if and only if

$$\hat{\eta}_{1-\alpha} \equiv \min_{1\leq j\leq J} \inf_{x\in\mathcal{X}} \left[\hat{h}_j(x) + n^{-1/2}\hat{k}_{1-\alpha}\hat{\sigma}_j(x)\right] < 0. \tag{5}$$

It is instructive to clarify some intuition underlying this testing procedure. The set $\widehat{V}$ described in step 2 implements an adaptive inequality selection (corresponding to the `ais(#)` option). It can be shown that, with probability approaching 1, $\widehat{V}$ contains all (j,x)'s that minimize $h_j(x)$. From (2), it is easy to see that whether the null hypothesis holds or not is solely determined by the functions' values at these extreme points. The selection step focuses the test on the relevant conditioning region, and so, improves its power. We also note that the rejection decision described in (5) naturally admits a graphical representation. Indeed, the functional estimates $\hat{h}_j(\cdot) + n^{-1/2}\hat{k}_{1-\alpha}\hat{\sigma}_j(\cdot)$ are $1-\alpha$ upper confidence bounds for $h_j(\cdot)$ uniformly over the selected set $\widehat{V}$. If some of these upper bounds dip below zero over some part of the conditioning space, we interpret it as significant statistical evidence against the null hypothesis (i.e., $h_j(x) \geq 0$ for all (j,x)), which leads to a formal rejection.

For some applications, it may be more natural to treat all competing predictive methods "symmetrically," rather than picking a particular one as the benchmark. In this situation, one may naturally rotate the benchmark role across all competing methods, and then collect all non-rejected benchmarks in a set

$$\begin{aligned}\widehat{M}_{1-\alpha} = \{0 \leq j \leq J :\ &\text{the } \alpha\text{-level CSPA test} \\ &\text{with method } j \text{ as the benchmark does not reject}\}.\end{aligned}$$

This operation is formally an inversion of the CSPA test and, by the duality between tests and confidence sets, $\widehat{M}_{1-\alpha}$ is a $1-\alpha$ level confidence set for the most superior model (which has the lowest conditional expected loss uniformly across all conditioning states), namely, the CSMS. The implementation can be easily carried out via `cspa` by calling the `csms` option.

3 The `cspa` command

This section documents the syntax and functionalities of the `cspa` command. The command requires the `moremata` package, which may be installed in command line via `ssc install moremata`.

3.1 Syntax

The Stata syntax of the `cspa` command is as follows:

`cspa` *condvar benchmark competitors* [*if*] [*in*] [`, ais(#) lag(#) m(#)`
`method(`*transtype*`) siglevel(#) ngrid(#) mc(#) triml(#) trimr(#) csms`

`plot plotu detail excel`]

where *condvar* is the conditioning variable, *benchmark* is the loss of the benchmark, and *competitors* is a list of loss variables associated with the competitors.

3.2 Options

`ais(#)` specifies the degree of adaptive inequality selection. The default is `ais(0.1)`. The selection may be disabled by setting `ais(0)`.

`lag(#)` specifies the number of lags for computing the Newey–West estimator $\hat{A}_n$. The default is `lag(0)`.

`m(#)` specifies the number of Legendre polynomial terms used in series estimation. The default is `m(5)`.

`method(`*transtype*`)` specifies the transformation implemented on the conditioning variable. The main purpose of doing so is to make the regressors approximately orthogonal, which generally improves the numerical stability of the series regression, especially when a large number of series terms are included. The approximating functions are Legendre polynomials of the transformed variable. The current version supports the following transformation methods, with `method(`*rank*`)` being the default.

- *none*: no transformation;
- *affine*: affine transformation $x \mapsto \frac{2(x-\min(x))}{\max(x)-\min(x)} - 1$;
- *normal*: normal transformation $x \mapsto 2\Phi[(x-\overline{x})/\sigma] - 1$, where $\overline{x}$ and σ are the sample mean and standard deviation of x, and Φ is the cumulative distribution function of the standard normal distribution;
- *lognormal*: log-normal transformation $x \mapsto 2\Phi[(\log x - \overline{\log x})/\Sigma] - 1$, where $\overline{\log x}$ and Σ are the sample mean and standard deviation of $\log x$, and Φ is the cumulative distribution function of the standard normal distribution;
- *rank*: $x \mapsto 2q(x) - 1$, where $q(x)$ is the empirical quantile of x.

`siglevel(#)` specifies the significance level (in percentage) for the CSPA test. The default is `siglevel(5)`.

`ngrid(#)` specifies the number of grid points used for discretizing the support of the conditioning state variable. The default is `ngrid(1000)`.

`triml(#)` sets the left limit of the conditioning region $\mathcal{X}$ to be the # empirical quantile of *condvar*. The default is `triml(0)`.

`trimr(#)` sets the right limit of the conditioning region $\mathcal{X}$ to be the $1 - \#$ empirical quantile of *condvar*. The default is `trimr(0)`.

`mc(#)` specifies the number of Monte Carlo simulations used to construct the distribution of the critical values. The default is `mc(5000)`.

`csms` calculates the confidence set of the most superior (CSMS) by rotating the benchmark role across all competing methods.

`plot` produces plots of the lower envelope of estimated conditional mean functions of loss differentials and its upper confidence bound, with the transformed conditioning variable plotted on the horizontal axis.

`plotu` produces plots of the lower envelope of estimated conditional mean functions of loss differentials and its upper confidence bound, with the original conditioning variable plotted on the horizontal axis.

`detail` adds the estimated conditional mean functions of loss differentials onto the plot generated by `plot` or `plotu`.

`excel` creates an Excel file that contains the requisite estimates for producing the plots generated by `plot` or `plotu`.

3.3 Stored results

The `cspa` command stores the following results in `e()`:

Scalars

`e(N)`	number of observations
`e(ts)`	CSPA test statistic
`e(pvalue)`	p-value of the CSPA test

Macros

`e(condvar)`	name of the conditioning variable
`e(loss)`	name of the loss variables
`e(method)`	transformation method
`e(cmd)`	`cspa`

Matrices

`e(xgrid)`	grid points of the conditioning variable
`e(h_hat)`	estimates of the conditional expected loss differentials
`e(lowerenvelope)`	estimate of the lower envelope of the conditional expected loss differentials
`e(cb)`	estimate of the confidence bound

4 An empirical example

4.1 Data description

Our real-data demonstration is based a dataset from the empirical study of Li et al. (2021). The `rv.dta` dataset contains a times series of daily realized volatilities of the Boeing Company (BA), computed as the sum of squared 5-minute returns within regular trading hours, and six time series of one-day-ahead volatility forecasts that are generated

by AR(1), AR(22), AR(22) with Lasso selection, the HAR model of Corsi (2009), the HARQ model developed in Bollerslev et al. (2016), and an ARFIMA model, respectively. These forecasts are formed under a rolling-window scheme with $1,000$ daily observations, and the evaluation sample ranges from May 2001 to December 2013, resulting in $3,180$ daily forecasts. We refer the reader to Li et al. (2021) for details on their constructions. The dataset also contains a time series of lagged CBOE volatility index (VIX), which shall be used as the conditioning variable X_t for conducting the CSPA test.

Following Li et al. (2021), we use Stein's loss function to measure the performance of the forecasts, that is,

$$L(F_t^\dagger, F_{j,t}) = \frac{F_{j,t}}{F_t^\dagger} - \log\left(\frac{F_{j,t}}{F_t^\dagger}\right) - 1.$$

This is a natural choice in view of the fact that volatility is a scale parameter, and Stein's loss is commonly used in the study of scale problems. The losses of the six volatility forecasts are then generated as follows.

```
. *** Load Data ***
. use "rv.dta", clear
.
. *** Calculate Loss ***
. gen ar1         =       (rv_ar1/true)       - log((rv_ar1/true) ) - 1
. gen ar22        =       (rv_ar22/true)      - log((rv_ar22/true) ) - 1
. gen ar22lasso   =       (rv_ar22lasso/true) - log((rv_ar22lasso/true) ) - 1
. gen har         =       (rv_har/true)       - log((rv_har/true) ) - 1
. gen harq        =       (rv_harq/true)      - log((rv_harq/true) ) - 1
. gen arfima      =       (rv_arfima/true)    - log((rv_arfima/true) ) - 1
```

4.2 One-versus-one CSPA test

We illustrate the most basic usage of the `cspa` command by conducting a one-versus-one conditional evaluation for two competing models: AR(1) and HAR. Specifically, we perform the CSPA test with one of them as the benchmark and the other as the competitor (so $J = 1$). The tests are implemented as follows.

```
. cspa vix ar1 har, lag(11) plot

Transformation on vix: Rank.
```

Benchmark	CSPA Test (5%)	P>|t|
ar1	-0.1029 (reject)	0.0002

```
. cspa vix har ar1, lag(11) plot

Transformation on vix: Rank.
```

Benchmark	CSPA Test (5%)	P>|t|

`har`	`0.0402 (non-reject)`	`0.9998`

The output table of `cspa` reports the test statistic $\hat{\eta}_{1-\alpha}$ at the user-specified significance level α, which by default is 5%. The null hypothesis that the benchmark uniformly weakly dominates the competitor is rejected when $\hat{\eta}_{1-\alpha} < 0$. The table also reports the p-value of the test. Looking at the tables above, we see that the CSPA null hypothesis with AR(1) being the benchmark is strongly rejected, but the test does not reject the CSPA null for HAR at any conventional significance level, suggesting that HAR is the superior predictive method.

With the `plot` option turned on, `cspa` also plots the estimated conditional expected loss differential function $\hat{h}_1(\cdot)$ along with the upper confidence bound $\hat{h}_1(\cdot) + n^{-1/2}\hat{k}_{1-\alpha}\hat{\sigma}_1(\cdot)$ as shown in Figure 1. Recall that a negative loss differential indicates that the benchmark underperforms the competitor, and vice versa. Moreover, the CSPA test rejects the null hypothesis if the upper confidence bound dips below zero over *some* part of the conditioning space. From the left panel of Figure 1, we see that AR(1) actually underperforms HAR over the entire conditioning space with high statistical significance. Meanwhile, the upper confidence bound plotted on the right panel is always above zero, which is consistent with the fact that the CSPA test does not reject the null hypothesis that HAR uniformly weakly dominates AR(1).

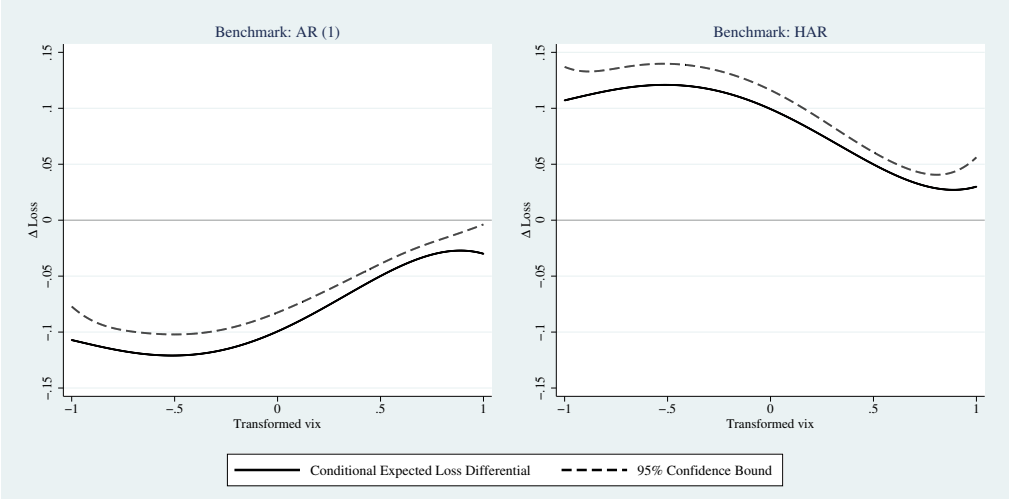

Figure 1: Diagnostic plots for one-versus-one CSPA tests.

The results presented above are generated under the default setting of `cspa`, except that we set the Newey–West lag parameter at `lag(11)` to account for serial dependence following Li et al. (2021). The significance level is $\alpha = 5\%$, which may be modified via the `siglevel(#)` option. By default, the series basis consists of $m = 5$ series terms (i.e., `m(5)`), corresponding to a fourth-order Legendre polynomial. In addition, the

conditioning variable `vix` is transformed onto the $[-1, 1]$ interval via the rank transformation (i.e., `method(`*rank*`)`). Robustness checks with respect to these choices can be conveniently implemented by modifying the corresponding options, as illustrated in Section 4.5 below.

4.3 One-versus-all CSPA test

We next demonstrate how to implement the CSPA test with multiple competitors. Recall that the `rv.dta` dataset contains six competing volatility forecasts. We may thus perform one-versus-all CSPA tests with one model as the benchmark and all the other five models as competitors, corresponding to $J = 5$. Under the null hypothesis, the benchmark model weakly dominates all the others uniformly across the conditioning space spanned by the VIX. The following command implements such a test with HAR being the benchmark.

```
. cspa vix har ar1 ar22 ar22lasso harq arfima, lag(11) plot

Transformation on vix: Rank.
```

Benchmark	CSPA Test (5%)	P>|t|
har	-0.0094 (reject)	0.0002

From the table, we see that the CSPA null hypothesis for the HAR benchmark is strongly rejected. Although the one-versus-one test presented in Section 4.2 suggests that HAR outperforms AR(1), the more stringent one-versus-all test reveals that HAR is no longer uniformly superior once additional competitors join the competition.

For comparison, we further implement the test for the HARQ benchmark as follows.

```
. cspa vix harq ar1 ar22 ar22lasso har arfima, lag(11) plot

Transformation on vix: Rank.
```

Benchmark	CSPA Test (5%)	P>|t|
harq	0.0053 (non-reject)	0.6712

Here, we see that the CSPA null hypothesis cannot be rejected at any conventional significance level, offering formal statistical evidence for HARQ's superiority relative to the other competitors across conditioning states.

In the present setting with multiple competitors, the `plot` option draws the estimated lower envelope function $\min_{1\leq j\leq J} \hat{h}_j(\cdot)$, which depicts the worst-case relative performance of the benchmark (in comparison to the "toughest" competitor) across the different conditioning states. The associated upper confidence bound $\min_{1\leq j\leq J}[\hat{h}_j(x) + n^{-1/2}\hat{k}_{1-\alpha}\hat{\sigma}_j(\cdot)]$ is also plotted; if this bound ever falls below zero, the null hypothesis is rejected.

The left (resp. right) panel of Figure 2 shows the plots associated with the HAR (resp. HARQ) benchmark. From the left panel, we see that the lower envelope function is always below zero, suggesting that at every conditioning state the HAR benchmark is outperformed by some competitor. The upper confidence bound dips below zero over the low-VIX region, which explains the CSPA test's rejection decision. On the other hand, we see from the right panel that the lower envelope function for the HARQ benchmark is generally "more positive," indicating its better relative performance. The associated upper confidence bound is always above zero, which is why the null hypothesis is not rejected for HARQ.

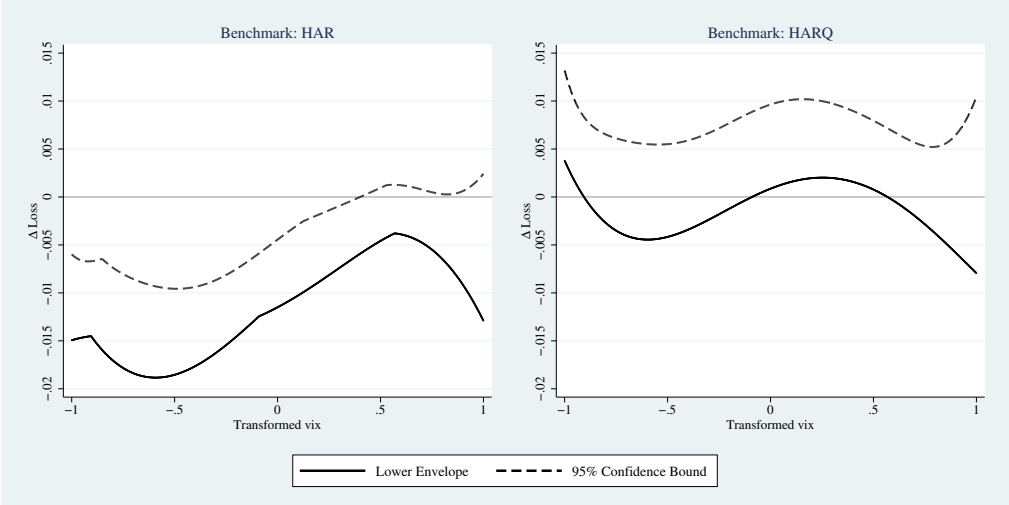

Figure 2: Diagnostic plots for one-versus-all CSPA tests.

Note that for the one-versus-all test, the `plot` option only draws the lower envelope function and its upper confidence bound, which concisely summarize the information from all J functional estimates. That noted, it may be useful for diagnostic purpose to also examine the individual $\hat{h}_j(\cdot)$ estimates of conditional expected loss differentials. Their plots can be added by further calling the `detail` option, as illustrated in Figure 3. In the present example, the `detail` option draws five curves for $\hat{h}_j(\cdot)$, $1 \leq j \leq 5$, though parts of these curves are overlaid by the lower envelope function. The left panel of Figure 3 reveals that, although HAR does not pass the CSPA test, its estimated conditional expected loss is lower than three competitors as evidenced by the three $\hat{h}_j(\cdot)$ curves above zero; but it is "beaten" by the other two competitors across all conditioning states, which in turn form the lower envelope. Meanwhile, it is interesting to note from the right panel that the lower envelope for the HARQ benchmark coincides with one particular $\hat{h}_j(\cdot)$ curve (overlaid by the lower envelope), which is associated with the ARFIMA model.

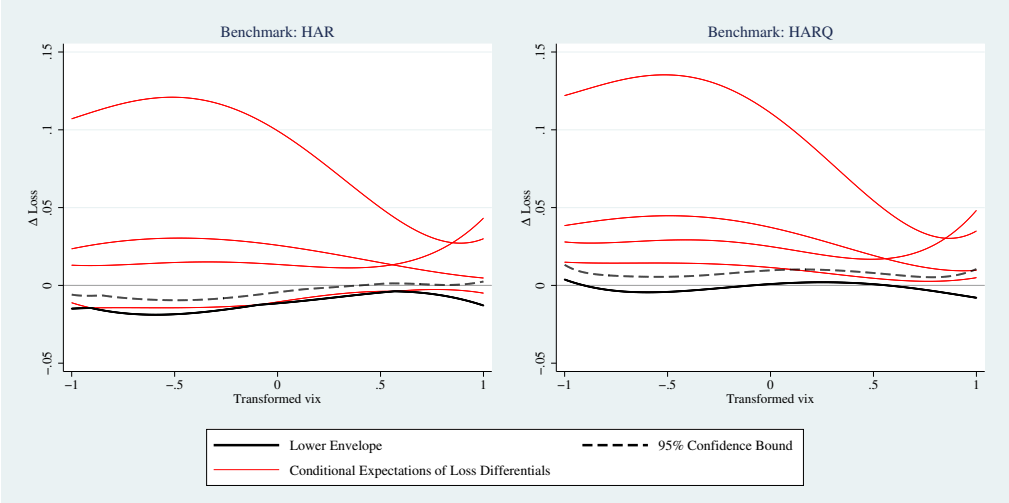

Figure 3: Detailed diagnostic plots for one-versus-all CSPA tests.

4.4 Confidence set for the most superior

As discussed in Section 2.2, the CSMS may be constructed by collecting all benchmarks that are not rejected by the CSPA test. In particular, the one-versus-all tests performed in Section 4.3 suggest that at the 95% confidence level, HARQ belongs to the CSMS, but HAR does not. We may further implement the test for the other benchmarks in order to decide whether they should be included in the CSMS or not.

The `cspa` command offers a more compact way to accomplish this task. With the `csms` option turned on, the command will automatically perform the requisite tests by rotating the benchmark role across all competing models, as shown below.

```
. cspa vix ar1 ar22 ar22lasso har harq arfima, lag(11) csms

Transformation on vix: Rank.
```

Benchmark	CSPA Test (5%)	P>|t|
ar1	-0.1160 (reject)	0.0002
ar22	-0.0173 (reject)	0.0002
ar22lasso	-0.0344 (reject)	0.0002
har	-0.0095 (reject)	0.0002
harq	0.0054 (non-reject)	0.6636
arfima	0.0058 (non-reject)	0.7906

```
The 95% CSMS = {harq, arfima}.
```

Here, the output table reports the CSPA test statistics and the associated p-values for all six one-versus-all tests with different benchmarks. It also reports the CSMS, which in the current example consists of the HARQ and ARFIMA models. Note that when `csms` in active, it is unnecessary to distinguish *benchmark* from *competitors*, because all models under consideration are treated in a symmetric manner.

4.5 Options

We highlight a few options that are more likely to be useful in typical empirical applications. For brevity, the illustration below focuses on the one-versus-all CSPA test for the HAR benchmark. Recall from Section 4.3 that the null hypothesis is rejected when the testing procedure is implemented under `cspa`'s default setting. Our goal here is to perform some robustness checks with respect to this choice.

We first consider the number of series terms m, which is 5 by default. We may change it to 4, 6, 8, or 10 by modifying the `m(#)` option as shown below. The null hypothesis is still strongly rejected in all these settings.

```
. cspa vix har ar1 ar22 ar22lasso harq arfima, lag(11) m(4)

Transformation on vix: Rank.
```

Benchmark	CSPA Test (5%)	P>|t|
har	-0.0101 (reject)	0.0002

```
. cspa vix har ar1 ar22 ar22lasso harq arfima, lag(11) m(6)

Transformation on vix: Rank.
```

Benchmark	CSPA Test (5%)	P>|t|
har	-0.0092 (reject)	0.0002

```
. cspa vix har ar1 ar22 ar22lasso harq arfima, lag(11) m(8)

Transformation on vix: Rank.
```

Benchmark	CSPA Test (5%)	P>|t|
har	-0.0088 (reject)	0.0002

```
. cspa vix har ar1 ar22 ar22lasso harq arfima, lag(11) m(10)

Transformation on vix: Rank.
```

Benchmark	CSPA Test (5%)	P>|t|
har	-0.0086 (reject)	0.0002

Next, we recall that under the default setting, the approximating functions are formed as Legendre polynomials of the rank-transformed `vix`. The purpose of using the

rank transformation is to make the conditioning variable approximately uniformly distributed on $[-1, 1]$, which helps reduce the multicolinearity among the series regressors. We may also employ the other types of transformations by modifying the `method` option. Since the distribution of VIX is close to be log-normal, `method(`*lognormal*`)` is a natural choice for this task. From the implementation below, we see that this modification does not affect the rejection decision of the CSPA test.

```
. cspa vix har ar1 ar22 ar22lasso harq arfima, lag(11) method(lognormal)

Transformation on vix: Lognormal.
```

Benchmark	CSPA Test (5%)	P>|t|
har	-0.0089 (reject)	0.0002

Finally, we note that the diagnostic figures generated by the `plot` option are plotted on the transformed scale over the $[-1, 1]$ interval. Alternatively, we may also set the horizontal axis to the original untransformed scale by using the `plotu` option. To clarify, the only effect of switching from `plot` to `plotu` is to compress/stretch the plotted curves along the horizontal axis, resulting in an alternative graphical presentation. The underlying testing results are completely unchanged. A concrete demonstration is given below, with the new plots displayed on Figure 4. From the figure, we see that the statistical evidence against the null hypothesis mainly stems from the subregion with the VIX below 25%. We also note that the gap between the confidence bound and nonparametric estimate is fairly wide when the VIX is, say, above 50%. This is because the effective sample size for the local estimation on this region is relatively small, resulting in imprecise nonparametric estimates. This is also why the corresponding segment is "compressed" under the rank-transformed scale as previously shown on the left panel of Figure 2.

```
. cspa vix har ar1 ar22 ar22lasso harq arfima, lag(11) method(lognormal)

Transformation on vix: Lognormal.
```

Benchmark	CSPA Test (5%)	P>|t|
har	-0.0089 (reject)	0.0002

5 References

Bollerslev, T., A. J. Patton, and R. Quaedvlieg. 2016. Exploiting the Errors: A Simple Approach for Improved Volatility Forecasting. *Journal of Econometrics* 192(1): 1–18.

Corsi, F. 2009. A Simple Approximate Long-memory Model of Realized Volatility. *Journal of Financial Econometrics* 7(2): 174–196.

Diebold, F. X., and R. S. Mariano. 1995. Comparing Predictive Accuracy. *Journal of Business & economic statistics* 13(3): 253–263.

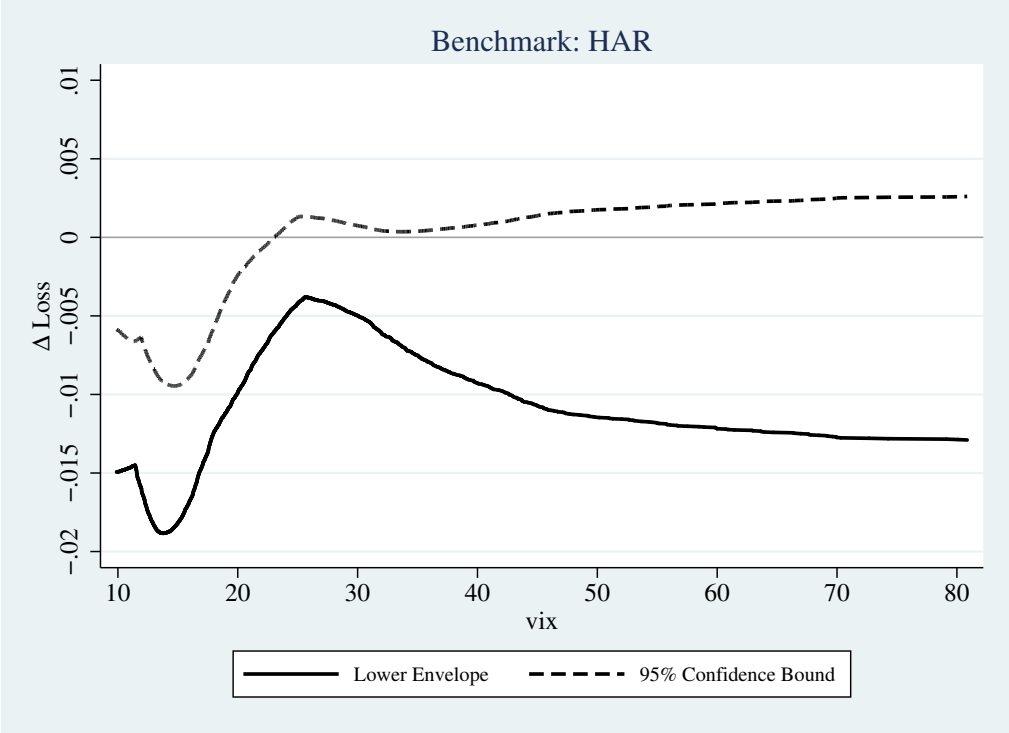

Figure 4: Diagnostic plot for the one-versus-all CSPA test generated by `plotu`.

Li, J., Z. Liao, and R. Quaedvlieg. 2021. Conditional Superior Predictive Ability. *The Review of Economic Studies, Forthcoming* .

About the authors

Jia Li is the Lee Kong Chian Professor of Economics at the School of Economics, Singapore Management University.

Zhipeng Liao is an associate professor of economics at the Department of Economics, UCLA.

Rogier Quaedvlieg is an associate professor of finance at the Erasmus School of Economics, Erasmus University Rotterdam.

Wenyu Zhou is an assistant professor at the International Business School, Zhejiang University.