Statistics/Data Analysis

Title

dsimih create —— Obtain SIRFs and SFEVDs after svarih

Syntax

After **svarih bacciocchi**, **svarih bfanelli** and **svarih llutkepohl**:

dsimih create [, *options*]

options	Description
Main	
regimes(*rgmlist***)**	generate results for all regime encodings in *rgmlist*; default: all regimes
step(*#***)**	set forecast horizon to *#*; default is **step(8)**
nose	do not calculate standard errors
saving(*filename*[**, replace**]**)**	save results to file *filename*
Bootstrap	
bs	obtain standard errors from residual bootstrap
bsp	obtain standard errors from parametric bootstrap
reps(*#***)**	use *#* bootstrap replications; there is no default value
bsaving(*filename*[**, replace**]**)**	save bootstrap results in *filename*
bsbmat(*matname***)**	store coefficient estimates from bootstrap replications in matrix *matname*
nodots	do not display "." for each bootstrap replication or for each *step* of asymptotic standard error calculations
nomaxopts	use default *maximize* options in bootstrap, even if estimates were obtained using different settings
fromb	use estimates as starting values for bootstrap optimizations
verbose	display optimization output for all bootstrap replications
notable	do not show coefficient tables of bootstrap replications when option **verbose** is used

The default is to use asymptotic standard errors if no options are specified.

Description

dsimih create calculates structural impulse-response functions and structural forecast error variance decompositions after svarih. Results are stored in **e()** by default, among the results from **svarih** estimation.

Once you have created a set of **dsimih** results, use **dsimih** subcommands other than **create** to analyze them.

Abbreviations, definitions, notation, syntax elements

This help entry uses terminology defined in svarih and dsimih.

Options

Main

regimes(*rgmlist***)** enumerates the regimes in their numerical encoding for which SIRFs and SFEVDs shall be created. The default is to create results for all regimes in the **svarih** estimation.

step(*#***)** specifies the step (forecast horizon); the default is eight periods.

nose specifies that no standard errors be calculated. Asymptotic standard errors are the default. See options **bs** and **bsp** for bootstrap standard errors. Note that options **nose**, **bs** and **bsp** are mutually exclusive alternatives.

saving(*filename*[**, replace**]**)** saves the DS results as a dsimih file. You can choose to preserve results this way in addition to the results saved by **dsimih create** in **e()**. If your maximum matsize setting is too small for DS results, you must specify the **saving()** option.

If *filename* is specified without an extension, **.dta** is assumed.

Bootstrap

bs specifies that standard errors be calculated by a residual bootstrap. Note that options **nose**, **bs** and **bsp** are mutually exclusive alternatives.

bsp specifies that standard errors be calculated via a multivariate-normal parametric bootstrap. Note that options **nose**, **bs** and **bsp** are mutually exclusive alternatives.

The following options are relevant only if **bs** or **bsp** is specified.

reps(*#***)** specifies the number of bootstrap replications to be performed. Since there is no default value, you must specify this options when you are bootstrapping standard errors. The minimum of # is 3 which is obviously too small a number to produce sensible results. Nevertheless, using a small number may be useful when learning the usage of **dsimih create**.

bsaving(*filename*[**, replace**]**)** specifies that the bootstrap replications be saved in file *filename*. *filename* is a regular Stata data file that can be loaded later using **use**. If *filename* is specified without an extension, **.dta** is assumed.

bsbmat(*matname***)** saves the coefficient estimates from successful bootstrap replications in matrix *matname*. If matrix *matname* already exists, it will get overwritten. If the number of requested replications from options **reps()** is larger than your current matsize setting, the option is ignored in the sense that after **dsimih create** finishes, matrix *matname* will not exist.

nodots specifies that dots not be displayed each time **dsimih create** performs a bootstrap replication. It also suppresses the output of dots for each forecast step when calculating asymptotic standard errors.

In the case of bootstrap standard errors, usage of this option is discouraged since **dsimih create** displays characters other than dots for replications that are not counted as being successful. It displays a "**f**" for a replication that failed. This might be caused, for example, by infeasible starting values. Furthermore, a "**n**" is displayed if estimates were obtained for a particular replication, but the last ML optimization and/or the GLS iteration did not terminate with the declaration of convergence, and a "**i**" is displayed if estimation went through but parameter estimates did not pass the local identification check. The local identification check is only performed for **svarih bacchiocchi** and **svarih bfanelli**.

Replications are only classified as being successful if a dot is displayed. In all other cases the replication is discarded and does not enter the calculation of bootstrap standard errors.

nomaxopts applies default maximization options for bootstrap maximum likelihood optimizations as described in **[R] maximize**. If you do not use this option, **dsimih create** applies the same maximization options that were used to obtain the **svarih** estimates. For example, if you obtained **svarih** estimates by switching between the NR and DFP algorithms, the omission of **nomaxopts** will apply the same procedure when performing estimation based on the bootstrap samples. If you use **nomaxopts**, the optimization will only be based on the maximize default, which is the NR algorithm.

Note that options **nomaxopts** is only concerns the ML optimization. If the **svarih** subcommand in question has a nested GLS procedure, GLS optimization settings in the bootstrap are always identical to the ones used in estimation.

fromb will feed the estimated parameter values from **svarih** as starting values into the bootstrap estimations. The default is to use the starting values that were used in obtaining **svarih** estimates.

verbose will display the optimization iteration output for each bootstrap replication.

notable does not display the coefficient estimates for each bootstrap replication. The default is to display these tables if option **verbose** is used. If option **verbose** is not used, option **notable** has no effect.

Remarks

Remarks are presented under the following headings:

Calculated statistics, standard errors and error bands

Statistics calculated are SIRFs and SFEVDs. They are calculated for each requested regime separately. Within each regime, calculations are performed in an analogous fashion to **[TS] irf create** but adjusted for the IH setting. The basic principle is to obtain *conditional* SIRFs and SFEVDs assuming that one is and stays within one regime.

There are three different ways of obtaining error bands. The default one is to use bands based on asymptotic standard errors. Calculations differ slightly across **svarih** methods.

The second choice is option **bsp** which applies a parametric bootstrap or option **bs** which applies a residual-based bootstrap. For each bootstrap estimate of the coefficients, **dsimih create** calculates an SIRF/SFEVD profile. From these profiles, it calculates the standard error for each step. The distribution at each step is taken to be normal.

When you use the above methods, you can make full use of the capabilities of **dsimih table** and **dsimih graph**. They use the calculated standard errors to generate error bands.

The third method of obtaining error bands is only useful if you are an advanced Stata user. In consists in using option **bs** or **bsp** in combination with option **bsaving()**. The latter saves the results for each replication in an output file. That is, for each replication you can recover the SIRF/SFEVD profile from the saved file. From this you can generate, for each step in the SIRF/SFEVD horizon, say, the 3rd and 97th percentiles. This will generate asymmetric bands. With this method, you cannot make use of the graphing capabilities of **dsimih**. You will have to draw all graphs by yourself. The data for the error bands to be used in these graphs is based on the output file of option **bsaving()**.

Treatment of gaps

Applying a bootstrap in order to calculate SIRF/SFEVD standard errors involves the simulation of the dependent variables. To make the simulation of dependent variables in a recursive (dynamic) system possible when there are gaps in the data, **dsimih create** conditions on initial values of the dependent variables for each gap that occurs in the data. You must make your own judgement as to whether this is tenable within the particular setting that you are working in.

Replacing existing results

In general, new results generated by **dsimih create** are merged with existing DS results. For example, if you are creating results for regimes 1 and 4 and results for regime 2 already exist, results for regimes 1, 2, 4 will be available after you run **dsimih create**.

It may happen that you want to modify existing DS results for a specific *rgmlist*. This makes sense in the following situations:

- You want to increase the forecast horizon (option **step()**).
- You want to replace bootstrapped standard errors by ones that are based on a higher number of replications.

Depending on what you are doing, you may have to erase parts of the DS results before you can replace them with new values. You can do so with dsimih drop. **dsimih create** will prompt you to use **dsimih drop** and not take any action if your **dsimih create** statement does not imply a longer forecast horizon for any regimes in *rgmlist*. If only part of the regimes in *rgmlist* will receive a longer forecast horizon, DS results for those regimes only are updated.

Matsize issues

Before creating DS results, **dsimih create** calculates how many rows of a matrix or of a data set are necessary to store results. If this number is above your current matsize setting, **dsimih create** aborts with an informative error message. In these cases, you can increase your **matsize** setting accordingly and re-run the **dsimih create** statement. Should your Stata flavor not allow to increase the **matsize** sufficiently, you must re-run the **dsimih create** statement using the **saving()** option. In this case results will only be available in the saved file but not in **e()**.

Estimation data set requirements

If options **bs** and **bsp** are not used, **dsimih create** does not require the data set used in estimation, nor does it require that **e(sample)** be set appropriately. By contrast, if you use either option **bs** or option **bsp**, **dsimih create** needs to construct bootstrap data samples, and for that both the data set used in estimation as well as **e(sample)** are necessary. In addition, the time variable must be set correctly (see tsset).

Computational issues

While bootstrap procedures are often recommended in the literature for creating SIRF/SFEVD error bands, they do have downsides. One example are numerical difficulties that may occur. A prominent example for SVARs is that the signs of individual columns of model matrices may switch. While this just changes the interpretation of the sign of the shock if this occurs for your coefficient estimates, it does invalidate the calculation of bootstrap standard errors when some of the bootstrap samples show this kind of reversal of signs. **svarih** may be more susceptible to this than **svar** since its models contain a larger number of parameters, so numerical difficulties are more likely to occur. To deal with this, **dsimih create** provides the option **fromb**. It causes the ML maximization to start from the coefficient estimates instead from standard starting values. In many cases it is an effective means to prevent sign reversals. To check whether sign reversals occur, you can utilize option **bsbmat**.

Another difficulty consists in deciding whether estimates based on one particular bootstrap sample should be counted as valid. In **dsimih create**, bootstrap coefficient estimates are only counted as successful if the optimization did not fail, parameters pass the local identification check (methods IH-BAC and IH-BFA only) and ML convergence was declared. If the **svarih** subcommand in question uses a nested GLS iteration, declaration of GLS convergence is also a prerequisite for a replication to be included in the calculation of standard errors. In all other cases the bootstrap coefficient estimates are discarded. This means that the effective number of replications that go into the calculation of bootstrap standard errors may below the number supplied in option **reps()**. If you have a highly and difficult-to-maximize model, the number of effective (successful) replications may be substantially below the requested number of replications. Therefore, usage of option **nodots** when calculating bootstrap standard errors is discouraged since its output gives an impression of how well the bootstrap works. When you save **dsimih** results in **e()**, matrices e(dsimih_bs) and e(dsimih_bsp) provide a record of successful, failed, non-convergent, and not-identified replications. Usage of option **bootstrap** of **dsimih describe** will display this information.

If you are in the situation that many bootstrap replications fail, you can re-run the bootstrap using option **verbose** and a low number of **reps()**. **verbose** causes **dsimih create** to display the output of any ML and GLS iterations performed. By default, it also outputs the estimated coefficients for each replication. You can suppress the latter output using option **notable**. The output so produced may give you hints as to why bootstrap replications fail.

Yet another downside of bootstrapped standard errors is that calculating them is time-consuming for larger models and for a large number of replications. For a large model, Stata may well be busy for an hour until a sufficient number of replications have been calculated. Therefore, the suggested workflow for using **dsimih create** is to use asymptotic standard errors for interactive work sessions and to cross-check them from time to time against bootstrap error bands.

Asymptotic standard errors are typically calculated much quicker than bootstrap ones. They are produced almost instantly for smaller values of *step*. Unfortunately, the order of the algorithm that calculates asymptotic standard errors is quadratic so the calculations eventually can become very slow. To give you an idea what you can roughly expect, many models will have asymptotic standard error calculation times of a few seconds for *step*=20-50 and may take half a minute for *step*=70-80. They may become less suitable for interactive work beyond *step*=100. These values obviously depend on your computer, the Stata flavor, and other things. **dsimih create** by default displays dots for each *step* calculated so you know how quickly calculations proceed.

Option **modelstats** of **dsimih describe** displays information on the **svarih** model that is most relevant for the calculation time of DS standard errors.

Setting the seed

When creating bootstrapped standard errors, the exact outcome depends on the state of the seed for the random number generator at the time **dsimih create** was invoked. Results may also depend on the version of the random number generator used. See set seed. While results should not differ qualitatively for the same statement run from different states of the seed when using a sufficiently large number for **reps()**, it is imperative to set the seed systematically if exact replicability of results is of importance to you.

If you want to make your bootstrap results exactly reproducible, it is your responsibility to augment your do-files with appropriate **set seed** statements. In the case of **dsimih create**, you need to be a little careful, since you can create, drop, replace and add calculations to **dsimih** results. Suppose that you have estimated a model with regime encodings 1 and 2 and you generate DS results with standard errors from a residual bootstrap by executing

```
. version 11.2: set seed 123456
. dsimih create, bs reps(500) step(10)
```

dsimih create generates 500 bootstrap samples and then calculates the DS statistics for both regimes, on the same bootstrap samples. Let's assume you decide to increase the forecast horizon for regime 2.

```
. version 11.2: set seed 123456
. dsimih create, bs reps(500) step(20) regime(2)
```

The results for forecast steps 0-10 of regime 2 will be identical to what you had obtained previously since **dsimih create** generated 500 identical bootstrap samples. A problem would have occurred if you had not preceeded the second **dsimih create** statement with a statement that set the seed to the appropriate state. Then you would have ended up with results for regimes 1 and 2 that are not strictly comparable because they would not be based on identical bootstrap samples.

If you decided to compare results from the residual bootstrap (option **bs**) to results from a parametric bootstrap (option **bsp**) by issuing

```
. dsimih create, bsp reps(500) step(20)
```

it would not matter that the random number generator was at another or unknown state. You could still compare your **bs** results to your **bsp** results, since the method for drawing from the residual distribution is both conceptionally and computationally different. Still, in order to keep things orderly and managable, it is recommended that you set the seed each time before you issue a **dsimih create** statement that resorts to bootstrap methods.

Examples

Excecuting the following statements will change current **e()**-results.

Bootstrap replication numbers are set to values that are inappropriate for analysis but appropriate for quick execution of example statements.

We will make frequent use of **dsimih describe**, which describes both **svarih** estimation results as well as **dsimih** results. See dsimih describe

Generate example estimates (see svarih examples):

```
. webuse lutkepohl2
. svarih examples bac first , ereplace
. dsimih describe , modelstats
```

Create SIRFs and SFEVDs with asymptotic standard errors (the default):

```
. dsimih create , step(12)
. dsimih describe
```

Add residual bootstrap standard errors. We set the seed!

```
. version 11.2: set seed 123456
. dsimih create , bs fromb step(12) reps(10)
. dsimih describe , bootstrap
```

Trying to recreate numbers that already exist will not execute.

```
. dsimih create , step(12)
. dsimih create , step(18) regime(2)
. dsimih describe
```

It is important to store or save estimates before replacing results in **e()**. Otherwise they are lost.

```
. estimates store bac first
```

Moving on to different estimates:

```
. svarih examples llu first , ereplace
. dsimih describe , modelstats
```

Horizons of *stats* can differ across regimes. Horizons of standard errors can differ across *setypes* and regimes. In particular, they can be lower than the horizons for the *stats*.

```
. dsimih create , step(12)
. dsimih create , step(24) nose
. dsimih describe
. estimates store llu_first
```

We are now in a postition to display and analyze the DS statistics.

```
. estimates restore bac_first
. dsimih describe , modelstats
. dsimih table sfevd, impulse(*inc) regime(1) step(1/3 6 12)

. estimates restore llu_first
. dsimih describe , modelstats
. dsimih graph sirf, impulse(*inv) ustep(10) byopts(rows(2) yrescale)
    level(68) name(dsimih_gr1, replace)
```

The following command requires a matsize setting of 882, which Stata/IC users do not have. We therefore save the results in a dsimih file.

```
. dsimih create , step(48) nose saving(dsimih_examplefile.dta, replace)
```

If your current matsize setting is below 882, results in **e()** remained the same. Otherwise the results from the last **dsimih create** run have been added to **e()**. The saved file only contains the results generated by the statement containing the **saving()** option.

```
. dsimih describe
. dsimih describe using dsimih_examplefile.dta
```

The file saved by the **saving()** option is a dsimih file. All **dsimih** subcommands for results analysis have a **using** modifier that lets you access these files.

```
. dsimih graph sirf using dsimih_examplefile, imp(*inc) reg(1) noci
    name(dsimih_gr2, replace)

. erase dsimih_examplefile.dta
```

Saved results

If allowed by your matsize setting, **dsimih create** adds or modifies the following in **e()**:

Matrices

e(dsimih)	matrix of dynamic simulation results

according to usage of option **bs** or **bsp**:

e(dsimih_bs)	detail for bootstrap replications for the residual bootstrap
e(dsimih_bsp)	detail for bootstrap replications for the parametric bootstrap

Author

Daniel C. Schneider, Goethe University Frankfurt, dan_schneider@outlook.com

Acknowledgements

The code of official Stata's **irf create** has served as a point of reference throughout the development of **dsimih create**. Any remaining errors in **dsimih create** are mine.

Also see

Help: **[TS] irf**, **dsimih**, **dsimih table**, **dsimih graph**, **dsimih describe**, **dsimih use**, **dsimih drop**, **dsimih etodta**