Statistics/Data Analysis

Title

svarih bacchiocchi —— Heteroskedasticity-identified SVAR models, Bacchiocchi (2011) methodology

Syntax

svarih bacchiocchi *depvarlist* [*if*] [*in*] **, rgmvar(***rgmvarname***) rgmmat(***rgmmatname***)** [*optional_options*]

options	Description
Model: IH part	
rgmvar(*rgmvarname***)**	identify regimes by contents of variable *rgmvarname*
rgmmat(*rgmmatname***)**	indicate through matrix *rgmmatname* which variables are volatile during each regime
aconstraints(*constraints_a***)**	apply previously defined *constraints_a* to **A**
aeq(*matrix_aeq***)**	define and apply to **A** equality constraint matrix *matrix_aeq*
acns(*matrix_acns***)**	define and apply to **A** cross-parameter constraint matrix *matrix_acns*
bconstraints(*constraints_b***)**	apply previously defined *constraints_b* to **B**
beq(*matrix_beq***)**	define and apply to **B** equality constraint matrix *matrix_beq*
bcns(*matrix_bcns***)**	define and apply to **B** cross-parameter constraint matrix *matrix_bcns*
econstraints(*constraints_e***)**	apply previously defined *constraints_e* to **E**
eeq(*matrix_eeq***)**	define and apply to **E** equality constraint matrix *matrix_eeq*
ecns(*matrix_ecns***)**	define and apply to **E** cross-parameter constraint matrix *matrix_ecns*
noidencheck	do not check for local identification
idendetail	display details of identification check if it fails
Model: VAR part	
lags(*numlist***)**	use lags *numlist* in the underlying VAR
exog(*varlist_exog***)**	use exogenous variables *varlist_exog*
noconstant	suppress constant term
dfk	make small-sample degrees-of-freedom adjustment
small	report small-sample t and F statistics
Reporting and screen output	
level(*#***)**	set confidence level
var	display underlying **var** output
nocnsreport	do not display constraints
notable	do not display estimation results table
display_options	control column formats
Maximization	
evalmode(*modenum***)**	choose between d0, d1, and d2 evaluators; rarely used
glsiter(*glsiternum***)**	perform a maximum number of *glsiternum* GLS iterations; default: 0
stolerance(*#***)**	convergence criterion for GLS iteration on regime-specific residual covariance matrices
btolerance(*#***)**	convergence criterion for GLS iteration on ML coefficient vector
glstrace	display output of GLS iterations
fixedfrom	always use the same starting values for ML optimizations within the GLS iteration
maximize_options	control the maximization process

coeflegend display legend instead of statistics

You must **tsset** your data before using **svarih bacchiocchi**; see **[TS] tsset**.
varlist_exog may contain time-series operators; see tsvarlist.
depvarlist may NOT contain time-series operators.
See svarih postestimation for features available after estimation.

Description

svarih bacchiocchi provides an alternative identification scheme for structural VARs than those implemented in svar. It implements a slightly generalized version of the "Identification through Heteroskedasticity" (IH) method as put forth in Bacchiocchi (2011). For details of the model setup, see the Remarks section. For general information on IH methods and for other IH methods that are available, see svarih.

Abbreviations, definitions, notation

This help entry uses abbreviations and definitions from svarih.

Options

Model: IH part

rgmvar(*rgmvarname***)** determines the variable whose observations identify the volatility regimes. It must be a numeric variable holding non-negative integer values where each integer value identifies a particular volatility regime.

It is not a requirement that a particular regime be contiguous.

A practical encoding is to define a baseline period to occur when *rgmvarname* is equal to one, and to encode each regime of different volatility by other non-negative integers.

rgmmat(*rgmmatname***)** specifies an existing (*sbar x n+1*) matrix that contains information about the set of shocks that are volatile during a particular volatility regime. The first column must contain non-negative integers. No value may occur more than once. Sorting these values ascendingly may be useful but is not a requirement. Each element of the first column links its matrix row to observations of the estimation sample during which the level of *rgmvarname* is equal to the value of the element. Columns 2 through *n*+1 correspond to the *n* endogenous variables of the system. These columns may only have elements equal to zero or one. A one indicates that the shock that corresponds to a particular column is volatile.

If this sounds confusing, have a look at the examples section which illustrates the concepts involved with a simple example.

There are some more requirements that *rgmmatname* must fulfill: The matrix column names must be equal to the string "rgmcode" followed by the variable names of the *n* endogenous variables, where the endogenous variables must be in the same order as in *depvarlist*.

As mentioned, the first column of the regime matrix contains entries that correspond to the regime encoding used by the regime variable. Each level that the regime variable takes on in the estimation sample must be matched by a corresponding entry in the first column of the regime matrix. Otherwise the command would not know which shocks are volatile during these periods.

The reverse is not true: The first column of the regime matrix may contain entries that are not matched by levels of the regime variable within the estimation sample. That way you are able to generate general regime matrices and use them in different contexts. For example, suppose you have a 5-equation system, i.e. 5 shocks (referred to through the dependent variable of their equation). Then the maximum number of regimes is 2^5=32. You can create a regime matrix with 32 rows that covers each possible state. Columns 2 through *n*+1 of each row of this matrix identify the volatile shocks in that state. You can now use this particular encoding with different data sets, estimation samples, regime variables, etc. and always use this regime encoding, regardless of which regimes actually occur in the regime variable.

The actual regime matrix that is used by **svarih bacchiocchi** will be returned as e(rgmmat_used). It is equal to *rgmmatname*, with rows irrelevant for estimation removed. For example, if the regime variable has numerical encodings 1 and 2 in the estimation sample but the regime matrix lists three rows corresponding to encodings 1, 2, 5, the row corresponding to encoding 5 will not be contained in e(rgmmat_used).

Columns in e(rgmmat_used) that consist of zeroes only imply that the corresponding shock is never modeled to be volatile which implies zero constraints on the E-matrix for corresponding columns. If these constraints are not specified in options **econstraints**, **eeq**, or **ecns**, the command will automatically generate the appropriate constraints on E for estimation.

aconstraints(*constraints_a***)**, **aeq(***matrix_aeq***)**, **acns(***matrix_acns***)** have the same meaning and can be specified in a similar manner as **aconstraints**, **aeq**, **acns** in **svar**. See the exposition there. They define linear constraints on the contemporaneous coefficients. A difference to **svar** is that **aconstraints()** does accept restrictions across model matrices A, B, E.

aeq(*matrix_aeq***)** defines equality constraints. *matrix_aeq* usually is an existing matrix but it may also be an expression as in matrix input or a simple matrix function. For example, **aeq((.,0\.,.))** and **aeq(I(2))** are allowed

acns(*matrix_acns***)** defines cross-equation constraints. Argument *matrix_acns* can be supplied in the same way as *matrix_aeq* from option **aeq()**.

aconstraints(*constraints_a***)** can define either one. They are defined using the constraint command.

bconstraints(*constraints_b***)**, **beq(***matrix_beq***)**, **bcns(***matrix_bcns***)** work in the same way as **aconstraints**, **aeq** and **acns**. They define linear constraints on the regular shock transmission matrix B.

econstraints(*constraints_e***)**, **eeq(***matrix_eeq***)**, **ecns(***matrix_ecns***)** work in the same way as **aconstraints**, **aeq** and **acns**. They define linear constraints on the shock impact modification matrix E.

If your regime matrix contains columns that consist of ones only, the elements of the corresponding column in E are automatically constrained to equal zero.

noidencheck skips the check for local identification of parameters. The check consists of examining the rank of a matrix at the parameter estimates. Full rank of this matrix is needed for identification. For details, see the associated PDF document on methods and formulas to **svarih**.

idendetail displays additional information on the identification check if it fails. The identification check computes the rank of a matrix at the parameter estimates. Full rank is required for local identification. **idendetail** displays rank information on submatrices also. This may aide in finding new estimation specifications that pass the identification check. For details, see the associated PDF document on methods and formulas to **svarih**.

If option **noidencheck** is used, option **idendetail** is ignored.

Model: VAR part

lags(*numlist***)** see var / svar.

exog(*varlist_exog***)** see var / svar.

noconstant; see **[R] estimation options**.

dfk see var / svar.

small see var / svar.

Reporting and screen output

level(*#***)**; see **[R] estimation options**.

var specifies that the output from **var** also be displayed. By default, the underlying VAR is fit **quietly**. This option is not available when you replay estimates.

nocnsreport; see **[R] estimation options**.

notable does not display the estimation output table. This is useful for example if you just want to look at the constraints that have been applied when obtaining **svarih** results. Replaying results automatically reports these constraints. You suppress most other output by

. svarih, notable

display_options: **cformat(***%fmt***)**, **pformat(***%fmt***)**, and **sformat(***%fmt***)**; see **[R] estimation options**.

Maximization

evalmode(*modenum***)** will choose between d0, d1, and d2 evaluators. Rarely used. The default is 2. Can be used to check the numerical robustness of large models.

glsiter(*glsiternum***)** determines the maximum number of GLS iterations. Note that the default is 0, i.e. no GLS iterations at all.

The following options are relevant only if **glsiter(***glsiternum***)**, *glsiternum*>0 is specified:

stolerance(*#***)** and **btolerance(***#***)** are two criteria that must be both fulfilled for GLS convergence to be declared.

stolerance(*#***)** calculates a matrix relative difference of the regime-specific reduced-form covariance matrices of the current iteration to the ones from the previous iteration. This criterion is fulfilled if the matrix relative difference is less than #. The default for # is 1e-4.

btolerance(*#***)** calculates a matrix relative difference of the estimated ML coefficient vector of the current iteration to the one from the previous iteration. This criterion is fulfilled if the matrix relative difference is less than #. The default for # is 1e-4.

glstrace outputs the following during each GLS iteration: the estimated coefficient vector, the regime-specific reduced-form covariance matrices, and the ML optimization of the current GLS iteration.

fixedfrom specifies that the same starting values are used for the ML optimization in each GLS iteration. By default, the ML estimates from the previous iteration are used as starting values for the current ML optimization.

maximize_options: **difficult**, **technique(***algorithm_spec***)**, **iterate(***#***)**, [**no**]**log**, **trace**, **gradient**, **showstep**, **hessian**, **showtolerance**, **tolerance(***#***)**, **ltolerance(***#***)**, **nrtolerance(***#***)**, **nonrtolerance(***#***)**, and **from(***init_specs***)**; see **[R] maximize**.

coeflegend; see **[R] estimation options**.

Remarks

Remarks are presented under the following headings:

Model equations
GLS iteration to achive ML estimates

Model equations

The IH-BAC method consists of an extended maximum likelihood structural VAR framework. The extension consists of positing a priori knowledge about different regimes of volatility, i.e. time periods during which the structural shocks have different variances and possibly nonzero covariances. Consequently, a prerequisite for estimation is the specification of a variable that identifies these regimes. In addition, to complete the information necessary for running the command, a matrix must be supplied that specifies for each volatility state the variables that are volatile. In the following, these two components are referred to as the "regime variable" and the "regime matrix".

In terms of a typical structural VAR equation that relates VAR residuals u_t to structural shocks e_t, the model in Baccchiocchi (2011) reads

(1) A*u_t = (I + B*D_s)*e_t

The matrix A models the contemporaneous effects among the endogenous variables, I is the identity matrix, B models the contemporaneous impact of shocks, and D_s is a diagonal regime-specific selection matrix that switches columns of B on and off. The generalized version of this equation implemented in **svarih bacchiocchi** reads

(2) A*u_t = (B + E*D_s)*e_t

where the identity matrix I has now been replaced by a potentially unrestricted matrix of coefficients B and the E-matrix in (2) corresponds to the B-matrix in (1). If D_s is the zero matrix, the model (2) collapses into the standard SVAR model implemented in svar.

GLS iteration to achieve ML estimates

The GLS iteration is done in an analogous fashion to the one in **svarih llutkepohl**. See the discussion there.

Examples

The following example illustrates the mechanics of **svarih bac**. It does not discuss the important issues of the interpretation of the shocks. It focuses on the mechanics. It builds on the example given in **svar** so you can compare the IH-BAC setup to the one of **svar**.

Throughout this example section, we store estimated results in Stata's estimation results catalogue for later access. The utility svarih examples allows you to easily re-generate these estimates at any point.

We first load the data set and define constraints. Then we estimate an **svar** model that is similar to the one of the examples section of **[TS] svar**, except that we choose to not restrict the estimation sample here.

```
. webuse lutkepohl2
. matrix aeq = (1,0,0 \ .,1,0 \ .,.,1)
. matrix beq = (.,0,0 \ 0,.,0 \ 0,0,.)
. svar dln_inv dln_inc dln_consump, aeq(aeq) beq(beq)
. est store bac_svar
```

In order to move to IH-BAC, we first must define a regime variable and a corresponding regime matrix. In order to illustrate the mechanics of **svarih bac**, we make the following assumptions: We have prior knowledge that the volatilities of the shocks in our model have changed in the 1970s. We fix a date of change of volatilities of 1974q1. The following generates a regime variable that contains values of 1 and 2. While we could have coded regimes with any integers of our choosing, this particular encoding will be useful when comparing estimation methods to **svarih bfa** and **svarih llu**, for which the regime encoding is hardcoded.

```
. gen byte rgmvar = (qtr>=tq(1974q1)) + 1
. matrix rgmmat = (1 , 0 , 0 , 0 \ 2 , 1 , 1 , 1 \ 3 , 1 , 0 , 1 )
. matrix colnames rgmmat = rgmcode dln_inv dln_inc dln_consump
. matrix list rgmmat
```

It is worth reiterating that the occurence of regimes can be modeled to be much more complicated than this. Any sequence and any multiplicity of regimes, with any occurence of gaps in the data, are allowed. If there are enough observations, **svarih bac** will produce estimates.

The regime matrix contains a regime encoding "3" which does not occur in the sample. This is no problem, and this line is merely included in the matrix to make this point. **svarih bac** would not allow this the other way round: All levels of the regime variable that occur in the sample must have an entry in the regime matrix. Lastly, we have to decide which constraints we impose on the shock modification matrix E. In the current setup, it makes sense to define analogous constraints to those of B.

```
. matrix eeq = (.,0,0 \ 0,.,0 \ 0,0,.)
. svarih bac dln_inv dln_inc dln_consump , rgmvar(rgmvar) rgmmat(rgmmat)
    aeq(aeq) beq(beq) eeq(eeq)
. est store bac_first
```

Estimation replay can be done in the standard way:

```
. svarih
```

but there exists a convenient postestimation option **cmat** that changes replay behavior. See svarih cmat. Option **star** encodes significance levels 0.01, 0.05, 0.1 by .a, .b, .c.

```
. svarih , cmat format(%12.3f) star
```

According to our model, the volatility of the shock of the first equation is significantly lower after 1974.

A major advantage of IH methods is that they can be used to relax required constraints of standard SVAR models. We further relax constraints on A, which is not possible in a standard SVAR model. As a rule of thumb, the more variables are modeled to have changes in volatility, the less constraints you have to impose on A, B, and E. In the following, we constrain the insignificant elements of E to zero. As an alternative to that, we could have redefined the regime matrix so that only the shock of the first equation is volatile in regime 2.

In order to obtain proper ML estimates, we explicitly specify the option **glsiter()**.

```
. matrix aeq = (1,.,. \ .,1,0 \ .,.,1)
. matrix eeq = (.,0,0 \ 0,0,0 \ 0,0,0)
. svarih bac dln_inv dln_inc dln_consump , rgmvar(rgmvar) rgmmat(rgmmat)
    aeq(aeq) beq(beq) eeq(eeq) glsiter(100)
. est store bac_unconstr_gls
. svarih , cmat format(%12.3f) star
```

The estimation output informs us that parameters are locally identified. If we relax the exclusion retriction of element [2,3] of A, this is no longer the case. Below we suppress most of the output.

```
. matrix aeq = (1,.,. \ .,1,. \ .,.,1)
. svarih bac dln_inv dln_inc dln_con, rgmv(rgmvar) rgmm(rgmmat) aeq(aeq)
    beq(beq) eeq(eeq) glsi(100) not nocnsr
. est store bac_notident
```

Going back to the identified model, we can perform LR-tests on model adequacy. We do this for the model with A-constraints as in **svar** and the one that relaxes the constraints on the first equation. We first need to estimate the constrained model.

```
. matrix aeq = (1,0,0 \ .,1,0 \ .,.,1)
. svarih bac dln_inv dln_inc dln_con, rgmv(rgmvar) rgmm(rgmmat) aeq(aeq)
    beq(beq) eeq(eeq) glsi(100) not nocnsr
. est store bac_constr_gls
. lrtest bac_constr_gls bac_unconstr_gls , stat
```

The LR-test rejects the constraints at the 1% level.

predict after **svarih bacchiocchi** generates prediced values, residuals, shocks, and historical decompositions. **dsimih** generates dynamic simulation statistics. For all features available after estimation, see svarih postestimation.

As an aside, when constraining the E-matrix of the model to a null matrix, IH-BAC collapses to a standard SVAR model, and estimates coincide with those of **svar**.

```
. matrix eeq = (0,0,0 \ 0,0,0 \ 0,0,0)
. qui svarih bac dln_inv dln_inc dln_consump , rgmvar(rgmvar) rgmmat(rgmmat)
    aeq(aeq) beq(beq) eeq(eeq)
. est store bac_svarrepl
. estimates table bac_svar bac_svarrepl , p
```

Saved results

svarih bacchiocchi saves the following in **e()**:

Scalars

e(N)	number of observations
e(N_cns)	number of independent constraints
e(k_eq)	number of equations in **e(b)**
e(k_dv)	number of dependent variables
e(k_aux)	number of auxiliary parameters
e(ll)	log likelihood
e(N_gaps)	number of gaps in the sample
e(k_var)	number of coefficients in VAR
e(k_eq_var)	number of equations in underlying VAR
e(k_dv_var)	number of dependent variables in underlying VAR
e(df_eq_var)	average number of parameters in an equation
e(df_r_var)	if **small**, VAR residual degrees of freedom
e(mlag)	highest lag in VAR
e(tmin)	first time period in the sample
e(tmax)	maximum time
e(rank)	rank of **e(V)**
e(ic_ml)	number of iterations in the final ML optimization
e(rc_ml)	return code from **ml**
e(converged_ml)	**1** if **ml** declared convergence, **0** otherwise
e(converged_gls)	**1** if the GLS iteration declared convergence, **0** otherwise
e(ic_gls)	number of GLS iterations performed
e(glsiter)	maximium # of GLS iterations allowed in command execution
e(numregimes)	number of regimes marked by *regimevar*

Macros

e(cmd)	**svarih**
e(method)	**Bacchiocchi**
e(version)	version number of command
e(cmdline)	command as typed
e(lags)	lags in model
e(depvar)	names of dependent variables
e(rgmvar)	name of variable that identifies regimes
e(rgmmatname)	name of regime matrix
e(exog)	names of exogenous variables, if specified
e(nocons)	noconstant, if **noconstant** specified
e(cns_a)	comprehensive list of constraints on A
e(cns_b)	comprehensive list of constraints on B
e(cns_e)	comprehensive list of constraints on E
e(dfk_var)	alternate divisor (**dfk**), if specified
e(small)	**small**, if specified
e(tsfmt)	format of timevar
e(timevar)	name of timevar
e(title)	title in estimation output
e(predict)	program used to implement **predict**
e(from)	contents of maximization option **from**, if specified as a string
e(mlopts)	maximization options used
e(idencheck)	result of identification check; one of **passed**, **failed**, or **skipped**
e(regimes)	regime encodings that occur in sample
e(regimes_Ns)	number of observations in each regime
e(glsopts)	GLS-related options supplied to command by user

Matrices

e(b)	coefficient vector
e(Cns)	constraints matrix
e(Sigma)	Residual covariance matrix of underlying VAR
e(Sigma_rgm#)	Residual covariance matrix for obs of regime #, based on the underlying GLS-VAR (if **glsiter(***#***)**, #>0) or of the underlying VAR
e(V)	variance-covariance matrix of the estimators
e(b_var)	coefficient vector of underlying VAR model
e(V_var)	VCE of underlying VAR model
	if GLS iteration was performed
e(b_vargls)	coefficient vector of underlying GLS-VAR model
e(V_vargls)	VCE of underlying GLS-VAR model
e(aeq)	**aeq(***matrix***)**, if specified
e(acns)	**acns(***matrix***)**, if specified
e(beq)	**beq(***matrix***)**, if specified
e(bcns)	**bcns(***matrix***)**, if specified
e(eeq)	**eeq(***matrix***)**, if specified
e(ecns)	**ecns(***matrix***)**, if specified
e(A)	estimated A matrix
e(B)	estimated B matrix
e(E)	estimated E matrix
e(rgmmat)	regime matrix as passed to the command
e(rgmmat_used)	equal to *rgmmatname*, with rows irrelevant for estimation removed
e(from)	matrix of maximization option **from**, if specified as a matrix

Functions

e(sample)	marks estimation sample

Author

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

Acknowledgements

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

References

Bacchiocchi, E. (2011): Identification in Structural VAR Models with Different Volatility Regimes. Universita Degli Studi di Milano, Working Paper No.2011-39.

Also see

Help: **[TS] svar**, **svarih**, **svarih bfa**, **svarih llu**, **svarih postestimation**, **svarih cmat**, **dsimih**