Title

var svar — Structural vector autoregressive models

Description

`svar` fits a vector autoregressive (VAR) model subject to short- or long-run constraints you place on the resulting impulse–response functions (IRFs). Economic theory typically motivates the constraints, allowing a causal interpretation of the IRFs to be made. See [TS] **var intro** for a list of commands that are used in conjunction with `svar`.

Quick start

Structural VAR model for `y1`, `y2`, and `y3` using `tsset` data with short-run constraints on impulse responses given by predefined matrices `A` and `B`

```
svar y1 y2 y3, aeq(A) beq(B)
```

Structural VAR model for `y1`, `y2`, and `y3` with long-run constraint on impulse responses given by the predefined matrix `C`

```
svar y1 y2 y3, lreq(C)
```

Add exogenous variables `x1` and `x2`

```
svar y1 y2 y3, lreq(C) exog(x1 x2)
```

Same as above, but include third and fourth lags of the dependent variables instead of first and second

```
svar y1 y2 y3, lreq(C) exog(x1 x2) lags(3 4)
```

Menu

Statistics > Multivariate time series > Structural vector autoregression (SVAR)

Syntax

Short-run constraints

`svar` *depvarlist* [*if*] [*in*] , { `aconstraints(`*constraints*$_a$`)` `aeq(`*matrix*$_{\text{aeq}}$`)`

`acns(`*matrix*$_{\text{acns}}$`)` `bconstraints(`*constraints*$_b$`)` `beq(`*matrix*$_{\text{beq}}$`)` `bcns(`*matrix*$_{\text{bcns}}$`)` }

[*short_run_options*]

Long-run constraints

`svar` *depvarlist* [*if*] [*in*] , { `lrconstraints(`*constraints*$_{\text{lr}}$`)` `lreq(`*matrix*$_{\text{lreq}}$`)`

`lrcns(`*matrix*$_{\text{lrcns}}$`)` } [*long_run_options*]

short_run_options	Description
Model	
noconstant	suppress constant term
* aconstraints(*constraints*$_a$)	apply previously defined *constraints*$_a$ to $\mathbf{A}$
* aeq(*matrix*$_{\rm aeq}$)	define and apply to $\mathbf{A}$ equality constraint matrix *matrix*$_{\rm aeq}$
* acns(*matrix*$_{\rm acns}$)	define and apply to $\mathbf{A}$ cross-parameter constraint matrix *matrix*$_{\rm acns}$
* bconstraints(*constraints*$_b$)	apply previously defined *constraints*$_b$ to $\mathbf{B}$
* beq(*matrix*$_{\rm beq}$)	define and apply to $\mathbf{B}$ equality constraint matrix *matrix*$_{\rm beq}$
* bcns(*matrix*$_{\rm bcns}$)	define and apply to $\mathbf{B}$ cross-parameter constraint *matrix*$_{\rm bcns}$
lags(*numlist*)	use lags *numlist* in the underlying VAR model
Model 2	
exog(*varlist*$_{\rm exog}$)	use exogenous variables *varlist*
varconstraints(*constraints*$_v$)	apply *constraints*$_v$ to underlying VAR model
noislog	suppress SURE iteration log
isiterate(#)	set maximum number of iterations for SURE; default is isiterate(1600)
istolerance(#)	set convergence tolerance of SURE
noisure	use one-step SURE
dfk	make small-sample degrees-of-freedom adjustment
small	report small-sample t and F statistics
noidencheck	do not check for local identification
nobigf	do not compute parameter vector for coefficients implicitly set to zero
Reporting	
level(#)	set confidence level; default is level(95)
full	show constrained parameters in table
var	display underlying var output
lutstats	report Lütkepohl lag-order selection statistics
nocnsreport	do not display constraints
display_options	control columns and column formats
Maximization	
maximize_options	control the maximization process; seldom used
coeflegend	display legend instead of statistics

* aconstraints(*constraints*$_a$), aeq(*matrix*$_{\rm aeq}$), acns(*matrix*$_{\rm acns}$), bconstraints(*constraints*$_b$), beq(*matrix*$_{\rm beq}$), bcns(*matrix*$_{\rm bcns}$): at least one of these options must be specified.

coeflegend does not appear in the dialog box.

long_run_options	Description
Model	
noconstant	suppress constant term
* lrconstraints(*constraints*$_{\mathrm{lr}}$)	apply previously defined *constraints*$_{\mathrm{lr}}$ to **C**
* lreq(*matrix*$_{\mathrm{lreq}}$)	define and apply to **C** equality constraint matrix *matrix*$_{\mathrm{lreq}}$
* lrcns(*matrix*$_{\mathrm{lrcns}}$)	define and apply to **C** cross-parameter constraint matrix *matrix*$_{\mathrm{lrcns}}$
lags(*numlist*)	use lags *numlist* in the underlying VAR model
Model 2	
exog(*varlist*$_{\mathrm{exog}}$)	use exogenous variables *varlist*
varconstraints(*constraints*$_v$)	apply *constraints*$_v$ to underlying VAR model
noislog	suppress SURE iteration log
isiterate(#)	set maximum number of iterations for SURE; default is isiterate(1600)
istolerance(#)	set convergence tolerance of SURE
noisure	use one-step SURE
dfk	make small-sample degrees-of-freedom adjustment
small	report small-sample t and F statistics
noidencheck	do not check for local identification
nobigf	do not compute parameter vector for coefficients implicitly set to zero
Reporting	
level(#)	set confidence level; default is level(95)
full	show constrained parameters in table
var	display underlying var output
lutstats	report Lütkepohl lag-order selection statistics
nocnsreport	do not display constraints
display_options	control columns and column formats
Maximization	
maximize_options	control the maximization process; seldom used
coeflegend	display legend instead of statistics

* lrconstraints(*constraints*$_{\mathrm{lr}}$), lreq(*matrix*$_{\mathrm{lreq}}$), lrcns(*matrix*$_{\mathrm{lrcns}}$): at least one of these options must be specified.

coeflegend does not appear in the dialog box.

You must tsset your data before using svar; see [TS] **tsset**.

depvarlist and *varlist*$_{\mathrm{exog}}$ may contain time-series operators; see [U] **11.4.4 Time-series varlists**.

by, collect, fp, rolling, statsby, and xi are allowed; see [U] **11.1.10 Prefix commands**.

See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

`noconstant`; see [R] **Estimation options**.

`aconstraints(`*constraints*$_a$`)`, `aeq(`*matrix*$_{\text{aeq}}$`)`, `acns(`*matrix*$_{\text{acns}}$`)`

`bconstraints(`*constraints*$_b$`)`, `beq(`*matrix*$_{\text{beq}}$`)`, `bcns(`*matrix*$_{\text{bcns}}$`)`

These options specify the short-run constraints in an SVAR model. To specify a short-run SVAR model, you must specify at least one of these options. The first list of options specifies constraints on the parameters of the $\mathbf{A}$ matrix; the second list specifies constraints on the parameters of the $\mathbf{B}$ matrix (see *Short-run SVAR models*). If at least one option is selected from the first list and none are selected from the second list, `svar` sets $\mathbf{B}$ to the identity matrix. Similarly, if at least one option is selected from the second list and none are selected from the first list, `svar` sets $\mathbf{A}$ to the identity matrix.

None of these options may be specified with any of the options that define long-run constraints.

`aconstraints(`*constraints*$_a$`)` specifies a *numlist* of previously defined Stata constraints to be applied to $\mathbf{A}$ during estimation.

`aeq(`*matrix*$_{\text{aeq}}$`)` specifies a matrix that defines a set of equality constraints. This matrix must be square with dimension equal to the number of equations in the underlying VAR model. The elements of this matrix must be *missing* or real numbers. A missing value in the (i,j) element of this matrix specifies that the (i,j) element of $\mathbf{A}$ is a free parameter. A real number in the (i,j) element of this matrix constrains the (i,j) element of $\mathbf{A}$ to this real number. For example,

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ . & 1.5 \end{bmatrix}$$

specifies that $\mathbf{A}[1,1] = 1$, $\mathbf{A}[1,2] = 0$, $\mathbf{A}[2,2] = 1.5$, and $\mathbf{A}[2,1]$ is a free parameter.

`acns(`*matrix*$_{\text{acns}}$`)` specifies a matrix that defines a set of exclusion or cross-parameter equality constraints on $\mathbf{A}$. This matrix must be square with dimension equal to the number of equations in the underlying VAR model. Each element of this matrix must be *missing*, 0, or a positive integer. A missing value in the (i,j) element of this matrix specifies that no constraint be placed on this element of $\mathbf{A}$. A zero in the (i,j) element of this matrix constrains the (i,j) element of $\mathbf{A}$ to be zero. Any strictly positive integers must be in two or more elements of this matrix. A strictly positive integer in the (i,j) element of this matrix constrains the (i,j) element of $\mathbf{A}$ to be equal to all the other elements of $\mathbf{A}$ that correspond to elements in this matrix that contain the same integer. For example, consider the matrix

$$\mathbf{A} = \begin{bmatrix} . & 1 \\ 1 & 0 \end{bmatrix}$$

Specifying `acns(A)` in a two-equation SVAR model constrains $\mathbf{A}[2,1] = \mathbf{A}[1,2]$ and $\mathbf{A}[2,2] = 0$ while leaving $\mathbf{A}[1,1]$ free.

`bconstraints(`*constraints*$_b$`)` specifies a *numlist* of previously defined Stata constraints to be applied to $\mathbf{B}$ during estimation.

`beq(`*matrix*$_{\text{beq}}$`)` specifies a matrix that defines a set of equality constraints. This matrix must be square with dimension equal to the number of equations in the underlying VAR model. The elements of this matrix must be either *missing* or real numbers. The syntax of implied constraints is analogous to the one described in `aeq()`, except that it applies to $\mathbf{B}$ rather than to $\mathbf{A}$.

`bcns(`*matrix*$_{\text{bcns}}$`)` specifies a matrix that defines a set of exclusion or cross-parameter equality constraints on **B**. This matrix must be square with dimension equal to the number of equations in the underlying VAR model. Each element of this matrix must be *missing*, 0, or a positive integer. The format of the implied constraints is the same as the one described in the `acns()` option above.

`lrconstraints(`*constraints*$_{\text{lr}}$`)`, `lreq(`*matrix*$_{\text{lreq}}$`)`, `lrcns(`*matrix*$_{\text{lrcns}}$`)`

These options specify the long-run constraints in an SVAR model. To specify a long-run SVAR model, you must specify at least one of these options. The list of options specifies constraints on the parameters of the long-run **C** matrix (see *Long-run SVAR models* for the definition of **C**). None of these options may be specified with any of the options that define short-run constraints.

`lrconstraints(`*constraints*$_{\text{lr}}$`)` specifies a *numlist* of previously defined Stata constraints to be applied to **C** during estimation.

`lreq(`*matrix*$_{\text{lreq}}$`)` specifies a matrix that defines a set of equality constraints on the elements of **C**. This matrix must be square with dimension equal to the number of equations in the underlying VAR model. The elements of this matrix must be either *missing* or real numbers. The syntax of implied constraints is analogous to the one described in option `aeq()`, except that it applies to **C**.

`lrcns(`*matrix*$_{\text{lrcns}}$`)` specifies a matrix that defines a set of exclusion or cross-parameter equality constraints on **C**. This matrix must be square with dimension equal to the number of equations in the underlying VAR model. Each element of this matrix must be *missing*, 0, or a positive integer. The syntax of the implied constraints is the same as the one described for the `acns()` option above.

`lags(`*numlist*`)` specifies the lags to be included in the underlying VAR model. The default is `lags(1 2)`. This option takes a *numlist* and not simply an integer for the maximum lag. For instance, `lags(2)` would include only the second lag in the model, whereas `lags(1/2)` would include both the first and second lags in the model. See **[U] 11.1.8 numlist** and **[U] 11.4.4 Time-series varlists** for further discussion of *numlist*s and lags.

Model 2

`exog(`*varlist*$_{\text{exog}}$`)` specifies a list of exogenous variables to be included in the underlying VAR model.

`varconstraints(`*constraints*$_v$`)` specifies a list of constraints to be applied to the coefficients in the underlying VAR model. Because `svar` estimates multiple equations, the constraints must specify the equation name for all but the first equation.

`noislog` prevents `svar` from displaying the iteration log from the iterated seemingly unrelated regression algorithm. When the `varconstraints()` option is not specified, the VAR model coefficients are estimated via OLS, a noniterative procedure. As a result, `noislog` may be specified only with `varconstraints()`. Similarly, `noislog` may not be combined with `noisure`.

`isiterate(`*#*`)` sets the maximum number of iterations for the iterated seemingly unrelated regression algorithm. The default limit is 1,600. When the `varconstraints()` option is not specified, the VAR model coefficients are estimated via OLS, a noniterative procedure. As a result, `isiterate()` may be specified only with `varconstraints()`. Similarly, `isiterate()` may not be combined with `noisure`.

`istolerance(`*#*`)` specifies the convergence tolerance of the iterated seemingly unrelated regression algorithm. The default tolerance is `1e-6`. When the `varconstraints()` option is not specified, the VAR model coefficients are estimated via OLS, a noniterative procedure. As a result, `istolerance()` may be specified only with `varconstraints()`. Similarly, `istolerance()` may not be combined with `noisure`.

noisure specifies that the VAR model coefficients be estimated via one-step seemingly unrelated regression when varconstraints() is specified. By default, svar estimates the coefficients in the VAR model via iterated seemingly unrelated regression when varconstraints() is specified. When the varconstraints() option is not specified, the VAR model coefficient estimates are obtained via OLS, a noniterative procedure. As a result, noisure may be specified only with varconstraints().

dfk specifies that a small-sample degrees-of-freedom adjustment be used when estimating $\mathbf{\Sigma}$, the covariance matrix of the VAR disturbances. Specifically, $1/(T - \overline{m})$ is used instead of the large-sample divisor $1/T$, where $\overline{m}$ is the average number of parameters in the functional form for $\mathbf{y}_t$ over the K equations.

small causes svar to calculate and report small-sample t and F statistics instead of the large-sample normal and χ^2 statistics.

noidencheck requests that the Amisano and Giannini (1997) check for local identification not be performed. This check is local to the starting values used. Because of this dependence on the starting values, you may wish to suppress this check by specifying the noidencheck option. However, be careful in specifying this option. Models that are not structurally identified can still converge, thereby producing meaningless results that only appear to have meaning.

nobigf requests that svar not compute the estimated parameter vector that incorporates coefficients that have been implicitly constrained to be zero, such as when some lags have been omitted from a model. e(bf) is used for computing asymptotic standard errors in the postestimation commands irf create and fcast compute. Therefore, specifying nobigf implies that the asymptotic standard errors will not be available from irf create and fcast compute. See *Fitting models with some lags excluded* in [TS] **var**.

Reporting

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

full shows constrained parameters in table.

var specifies that the output from var also be displayed. By default, the underlying VAR model is fit quietly.

lutstats specifies that the Lütkepohl versions of the lag-order selection statistics be computed. See *Methods and formulas* in [TS] **varsoc** for a discussion of these statistics.

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

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

Maximization

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**. These options are seldom used.

The following option is available with svar but is not shown in the dialog box:

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

Remarks and examples

stata.com

Remarks are presented under the following headings:

Introduction

This entry assumes that you have already read [TS] **var intro** and [TS] **var**; if not, please do. Here we illustrate how to fit SVAR models in Stata subject to short-run and long-run restrictions. For more detailed information on SVAR models, see Amisano and Giannini (1997) and Hamilton (1994). For good introductions to VAR models, see Lütkepohl (2005), Hamilton (1994), Stock and Watson (2001), and Becketti (2020).

Short-run SVAR models

A short-run SVAR model without exogenous variables can be written as

$$\mathbf{A}(\mathbf{I}_K - \mathbf{A}_1 L - \mathbf{A}_2 L^2 - \cdots - \mathbf{A}_p L^p)\mathbf{y}_t = \mathbf{A}\boldsymbol{\epsilon}_t = \mathbf{B}\mathbf{e}_t$$

where L is the lag operator, $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{A}_1, \ldots, \mathbf{A}_p$ are $K \times K$ matrices of parameters, $\boldsymbol{\epsilon}_t$ is a $K \times 1$ vector of innovations with $\boldsymbol{\epsilon}_t \sim N(\mathbf{0}, \boldsymbol{\Sigma})$ and $E[\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_s'] = \mathbf{0}_K$ for all $s \neq t$, and $\mathbf{e}_t$ is a $K \times 1$ vector of orthogonalized disturbances; that is, $\mathbf{e}_t \sim N(\mathbf{0}, \mathbf{I}_K)$ and $E[\mathbf{e}_t \mathbf{e}_s'] = \mathbf{0}_K$ for all $s \neq t$. These transformations of the innovations allow us to analyze the dynamics of the system in terms of a change to an element of $\mathbf{e}_t$. In a short-run SVAR model, we obtain identification by placing restrictions on $\mathbf{A}$ and $\mathbf{B}$, which are assumed to be nonsingular.

▷ Example 1: Short-run just-identified SVAR model

Following Sims (1980), the Cholesky decomposition is one method of identifying the impulse–response functions in a VAR model; thus, this method corresponds to an SVAR model. There are several sets of constraints on $\mathbf{A}$ and $\mathbf{B}$ that are easily manipulated back to the Cholesky decomposition, and the following example illustrates this point.

One way to impose the Cholesky restrictions is to assume an SVAR model of the form

$$\widetilde{\mathbf{A}}(\mathbf{I}_K - \mathbf{A}_1 - \mathbf{A}_2 L^2 - \cdots \mathbf{A}_p L^p)\mathbf{y}_t = \widetilde{\mathbf{B}}\mathbf{e}_t$$

where $\widetilde{\mathbf{A}}$ is a lower triangular matrix with ones on the diagonal and $\widetilde{\mathbf{B}}$ is a diagonal matrix. Because the $\mathbf{P}$ matrix for this model is $\mathbf{P}_{\rm sr} = \widetilde{\mathbf{A}}^{-1}\widetilde{\mathbf{B}}$, its estimate, $\widehat{\mathbf{P}}_{\rm sr}$, obtained by plugging in estimates of $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$, should equal the Cholesky decomposition of $\widehat{\boldsymbol{\Sigma}}$.

To illustrate, we use the German macroeconomic data discussed in Lütkepohl (2005) and used in [TS] **var**. In this example, $\mathbf{y}_t = $ (`dln_inv`, `dln_inc`, `dln_consump`), where `dln_inv` is the first difference of the log of investment, `dln_inc` is the first difference of the log of income, and `dln_consump` is the first difference of the log of consumption. Because the first difference of the natural log of a variable can be treated as an approximation of the percentage change in that variable, we will refer to these variables as percentage changes in `inv`, `inc`, and `consump`, respectively.

We will impose the Cholesky restrictions on this system by applying equality constraints with the constraint matrices

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ . & 1 & 0 \\ . & . & 1 \end{bmatrix} \qquad \text{and} \qquad \mathbf{B} = \begin{bmatrix} . & 0 & 0 \\ 0 & . & 0 \\ 0 & 0 & . \end{bmatrix}$$

With these structural restrictions, we assume that the percentage change in `inv` is not contemporaneously affected by the percentage changes in either `inc` or `consump`. We also assume that the percentage change of `inc` is affected by contemporaneous changes in `inv` but not `consump`. Finally, we assume that percentage changes in `consump` are affected by contemporaneous changes in both `inv` and `inc`.

The following commands fit an SVAR model with these constraints.

```
. use https://www.stata-press.com/data/r18/lutkepohl2
(Quarterly SA West German macro data, Bil DM, from Lutkepohl 1993 Table E.1)
. matrix A = (1,0,0\.,1,0\.,.,1)
. matrix B = (.,0,0\0,.,0\0,0,.)
. svar dln_inv dln_inc dln_consump if qtr<=tq(1978q4), aeq(A) beq(B)
Estimating short-run parameters
```

(output omitted)

```
Structural vector autoregression
 ( 1)  [/A]1_1 = 1
 ( 2)  [/A]1_2 = 0
 ( 3)  [/A]1_3 = 0
 ( 4)  [/A]2_2 = 1
 ( 5)  [/A]2_3 = 0
 ( 6)  [/A]3_3 = 1
 ( 7)  [/B]1_2 = 0
 ( 8)  [/B]1_3 = 0
 ( 9)  [/B]2_1 = 0
 (10)  [/B]2_3 = 0
 (11)  [/B]3_1 = 0
 (12)  [/B]3_2 = 0
Sample: 1960q4 thru 1978q4                      Number of obs     =         73
Exactly identified model                        Log likelihood    =    606.307
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/A						
1_1	1	(constrained)				
2_1	-.0336288	.0294605	-1.14	0.254	-.0913702	.0241126
3_1	-.0435846	.0194408	-2.24	0.025	-.0816879	-.0054812
1_2	0	(constrained)				
2_2	1	(constrained)				
3_2	-.424774	.0765548	-5.55	0.000	-.5748187	-.2747293
1_3	0	(constrained)				
2_3	0	(constrained)				
3_3	1	(constrained)				
/B						
1_1	.0438796	.0036315	12.08	0.000	.036762	.0509972
2_1	0	(constrained)				
3_1	0	(constrained)				
1_2	0	(constrained)				
2_2	.0110449	.0009141	12.08	0.000	.0092534	.0128365
3_2	0	(constrained)				
1_3	0	(constrained)				
2_3	0	(constrained)				
3_3	.0072243	.0005979	12.08	0.000	.0060525	.0083962

The SVAR output has four parts: an iteration log, a display of the constraints imposed, a header with sample and SVAR log-likelihood information, and a table displaying the estimates of the parameters from the $\mathbf{A}$ and $\mathbf{B}$ matrices. From the output above, we can see that the equality constraint matrices supplied to svar imposed the intended constraints and that the SVAR header informs us that the model we fit is just identified. The estimates of /A:2_1, /A:3_1, and /A:3_2 are all negative. Because the off-diagonal elements of the $\mathbf{A}$ matrix contain the negative of the actual contemporaneous effects, the estimated effects are positive, as expected.

The estimates $\widehat{\mathbf{A}}$ and $\widehat{\mathbf{B}}$ are stored in e(A) and e(B), respectively, allowing us to compute the estimated Cholesky decomposition.

```
. matrix Aest = e(A)

. matrix Best = e(B)

. matrix chol_est = inv(Aest)*Best

. matrix list chol_est

chol_est[3,3]
                dln_inv      dln_inc  dln_consump
    dln_inv    .04387957           0            0
    dln_inc    .00147562    .01104494           0
dln_consump    .00253928     .0046916    .00722432
```

svar stores the estimated $\mathbf{\Sigma}$ from the underlying var in e(Sigma). The output below illustrates the computation of the Cholesky decomposition of e(Sigma). It is the same as the output computed from the SVAR estimates.

```
. matrix sig_var = e(Sigma)

. matrix chol_var = cholesky(sig_var)

. matrix list chol_var

chol_var[3,3]
                dln_inv      dln_inc  dln_consump
    dln_inv    .04387957           0            0
    dln_inc    .00147562    .01104494           0
dln_consump    .00253928     .0046916    .00722432
```

◁

We might now wonder why we bother obtaining parameter estimates via nonlinear estimation if we can obtain them simply by a transform of the estimates produced by var. When the model is just identified, as in the previous example, the SVAR parameter estimates can be computed via a transform of the VAR estimates. However, when the model is overidentified, such is not the case.

▷ Example 2: Short-run overidentified SVAR model

The Cholesky decomposition example above fit a just-identified model. This example considers an overidentified model. In example 1, the /A:2_1 parameter was not significant, which is consistent with a theory in which changes in our measure of investment affect only changes in income with a lag. We can impose the restriction that /A:2_1 is zero and then test this overidentifying restriction. Our $\mathbf{A}$ and $\mathbf{B}$ matrices are now

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ . & . & 1 \end{bmatrix} \qquad \text{and} \qquad \mathbf{B} = \begin{bmatrix} . & 0 & 0 \\ 0 & . & 0 \\ 0 & 0 & . \end{bmatrix}$$

The output below contains the commands and results we obtained by fitting this model on the Lütkepohl data.

```
. matrix B = (.,0,0\0,.,0\0,0,.)

. matrix A = (1,0,0\0,1,0\.,.,1)
```

```
. svar dln_inv dln_inc dln_consump if qtr<=tq(1978q4), aeq(A) beq(B)
Estimating short-run parameters
```

(*output omitted*)

```
Structural vector autoregression
 ( 1)  [/A]1_1 = 1
 ( 2)  [/A]1_2 = 0
 ( 3)  [/A]1_3 = 0
 ( 4)  [/A]2_1 = 0
 ( 5)  [/A]2_2 = 1
 ( 6)  [/A]2_3 = 0
 ( 7)  [/A]3_3 = 1
 ( 8)  [/B]1_2 = 0
 ( 9)  [/B]1_3 = 0
 (10)  [/B]2_1 = 0
 (11)  [/B]2_3 = 0
 (12)  [/B]3_1 = 0
 (13)  [/B]3_2 = 0

Sample: 1960q4 thru 1978q4                      Number of obs     =         73
Overidentified model                            Log likelihood    =   605.6613
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/A						
1_1	1	(constrained)				
2_1	0	(constrained)				
3_1	-.0435911	.0192696	-2.26	0.024	-.0813589	-.0058233
1_2	0	(constrained)				
2_2	1	(constrained)				
3_2	-.4247741	.0758806	-5.60	0.000	-.5734973	-.2760508
1_3	0	(constrained)				
2_3	0	(constrained)				
3_3	1	(constrained)				
/B						
1_1	.0438796	.0036315	12.08	0.000	.036762	.0509972
2_1	0	(constrained)				
3_1	0	(constrained)				
1_2	0	(constrained)				
2_2	.0111431	.0009222	12.08	0.000	.0093356	.0129506
3_2	0	(constrained)				
1_3	0	(constrained)				
2_3	0	(constrained)				
3_3	.0072243	.0005979	12.08	0.000	.0060525	.0083962

```
LR test of identifying restrictions: chi2(1) = 1.292       Prob > chi2 = 0.256
```

The footer in this example reports a test of the overidentifying restriction. The null hypothesis of this test is that any overidentifying restrictions are valid. In the case at hand, we cannot reject this null hypothesis at any of the conventional levels.

◁

▷ Example 3: Short-run SVAR model with constraints

`svar` also allows us to place constraints on the parameters of the underlying VAR model. We begin by looking at the underlying VAR model for the SVAR models that we have used in the previous examples.

```
. var dln_inv dln_inc dln_consump if qtr<=tq(1978q4)
Vector autoregression
Sample: 1960q4 thru 1978q4                       Number of obs     =         73
Log likelihood =   606.307                       AIC               =  -16.03581
FPE            =  2.18e-11                       HQIC              =  -15.77323
Det(Sigma_ml)  =  1.23e-11                       SBIC              =  -15.37691
Equation           Parms      RMSE     R-sq      chi2     P>chi2

dln_inv               7     .046148   0.1286   10.76961   0.0958
dln_inc               7     .011719   0.1142   9.410683   0.1518
dln_consump           7     .009445   0.2513   24.50031   0.0004


             Coefficient  Std. err.      z    P>|z|     [95% conf. interval]

dln_inv
     dln_inv
         L1.   -.3196318   .1192898    -2.68   0.007    -.5534355   -.0858282
         L2.   -.1605508    .118767    -1.35   0.176      -.39333    .0722283

     dln_inc
         L1.    .1459851   .5188451     0.28   0.778    -.8709326    1.162903
         L2.    .1146009    .508295     0.23   0.822     -.881639    1.110841

 dln_consump
         L1.    .9612288   .6316557     1.52   0.128    -.2767936    2.199251
         L2.    .9344001   .6324034     1.48   0.140    -.3050877    2.173888

       _cons   -.0167221   .0163796    -1.02   0.307    -.0488257    .0153814

dln_inc
     dln_inv
         L1.    .0439309   .0302933     1.45   0.147    -.0154427    .1033046
         L2.    .0500302   .0301605     1.66   0.097    -.0090833    .1091437

     dln_inc
         L1.   -.1527311    .131759    -1.16   0.246    -.4109741    .1055118
         L2.    .0191634   .1290799     0.15   0.882    -.2338285    .2721552

 dln_consump
         L1.    .2884992   .1604069     1.80   0.072    -.0258926    .6028909
         L2.      -.0102   .1605968    -0.06   0.949    -.3249639    .3045639

       _cons    .0157672   .0041596     3.79   0.000     .0076146    .0239198

dln_consump
     dln_inv
         L1.    -.002423   .0244142    -0.10   0.921     -.050274     .045428
         L2.    .0338806   .0243072     1.39   0.163    -.0137607    .0815219

     dln_inc
         L1.    .2248134   .1061884     2.12   0.034     .0166879    .4329389
         L2.    .3549135   .1040292     3.41   0.001     .1510199     .558807

 dln_consump
         L1.   -.2639695   .1292766    -2.04   0.041     -.517347    -.010592
         L2.   -.0222264   .1294296    -0.17   0.864    -.2759039     .231451

       _cons    .0129258   .0033523     3.86   0.000     .0063554    .0194962
```

The equation-level model tests reported in the header indicate that we cannot reject the null hypotheses that all the coefficients in the first equation are zero, nor can we reject the null that all the coefficients in the second equation are zero at the 5% significance level. We use a combination of theory and the p-values from the output above to place some exclusion restrictions on the underlying VAR(2) model. Specifically, in the equation for the percentage change of `inv`, we constrain the coefficients on `L2.dln_inv`, `L.dln_inc`, `L2.dln_inc`, and `L2.dln_consump` to be zero. In the equation for `dln_inc`, we constrain the coefficients on `L2.dln_inv`, `L2.dln_inc`, and `L2.dln_consump` to be zero. Finally, in the equation for `dln_consump`, we constrain `L.dln_inv` and `L2.dln_consump` to be zero. We then refit the SVAR model from the previous example.

```
. constraint 1 [dln_inv]L2.dln_inv = 0

. constraint 2 [dln_inv ]L.dln_inc = 0

. constraint 3 [dln_inv]L2.dln_inc = 0

. constraint 4 [dln_inv]L2.dln_consump = 0

. constraint 5 [dln_inc]L2.dln_inv = 0

. constraint 6 [dln_inc]L2.dln_inc = 0

. constraint 7 [dln_inc]L2.dln_consump = 0

. constraint 8 [dln_consump]L.dln_inv = 0

. constraint 9 [dln_consump]L2.dln_consump = 0

. svar dln_inv dln_inc dln_consump if qtr<=tq(1978q4), aeq(A) beq(B)
> varconst(1/9) noislog

Estimating short-run parameters
  (output omitted )

Structural vector autoregression

 ( 1)  [/A]1_1 = 1
 ( 2)  [/A]1_2 = 0
 ( 3)  [/A]1_3 = 0
 ( 4)  [/A]2_1 = 0
 ( 5)  [/A]2_2 = 1
 ( 6)  [/A]2_3 = 0
 ( 7)  [/A]3_3 = 1
 ( 8)  [/B]1_2 = 0
 ( 9)  [/B]1_3 = 0
 (10)  [/B]2_1 = 0
 (11)  [/B]2_3 = 0
 (12)  [/B]3_1 = 0
 (13)  [/B]3_2 = 0
```

```
Sample: 1960q4 thru 1978q4                      Number of obs     =         73
Overidentified model                            Log likelihood    =   601.8591
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/A						
1_1	1	(constrained)				
2_1	0	(constrained)				
3_1	-.0418708	.0187579	-2.23	0.026	-.0786356	-.0051061
1_2	0	(constrained)				
2_2	1	(constrained)				
3_2	-.4255808	.0745298	-5.71	0.000	-.5716565	-.2795051
1_3	0	(constrained)				
2_3	0	(constrained)				
3_3	1	(constrained)				
/B						
1_1	.0451851	.0037395	12.08	0.000	.0378557	.0525145
2_1	0	(constrained)				
3_1	0	(constrained)				
1_2	0	(constrained)				
2_2	.0113723	.0009412	12.08	0.000	.0095276	.013217
3_2	0	(constrained)				
1_3	0	(constrained)				
2_3	0	(constrained)				
3_3	.0072417	.0005993	12.08	0.000	.006067	.0084164

```
LR test of identifying restrictions: chi2(1) = .8448       Prob > chi2 = 0.358
```

If we displayed the underlying VAR(2) results by using the `var` option, we would see that most of the unconstrained coefficients are now significant at the 10% level and that none of the equation-level model statistics fail to reject the null hypothesis at the 10% level. The `svar` output reveals that the p-value of the overidentification test rose and that the coefficient on `/A:3_1` is still insignificant at the 1% level but not at the 5% level.

◁

Before moving on to models with long-run constraints, consider these limitations. We cannot place constraints on the elements of $\mathbf{A}$ in terms of the elements of $\mathbf{B}$, or vice versa. This limitation is imposed by the form of the check for identification derived by Amisano and Giannini (1997). As noted in *Methods and formulas*, this test requires separate constraint matrices for the parameters in $\mathbf{A}$ and $\mathbf{B}$. Another limitation is that we cannot mix short-run and long-run constraints.

Long-run SVAR models

As discussed in [TS] **var intro**, a long-run SVAR model has the form

$$\mathbf{y}_t = \mathbf{C}\mathbf{e}_t$$

In long-run models, the constraints are placed on the elements of $\mathbf{C}$, and the free parameters are estimated. These constraints are often exclusion restrictions. For instance, constraining $\mathbf{C}[1,2]$ to be zero can be interpreted as setting the long-run response of variable 1 to the structural shocks driving variable 2 to be zero.

Similar to the short-run model, the $\mathbf{P}_{\rm lr}$ matrix such that $\mathbf{P}_{\rm lr}\mathbf{P}_{\rm lr}' = \mathbf{\Sigma}$ identifies the structural impulse–response functions. $\mathbf{P}_{\rm lr} = \mathbf{C}$ is identified by the restrictions placed on the parameters in $\mathbf{C}$. There are K^2 parameters in $\mathbf{C}$, and the order condition for identification requires that there be at least $K^2 - K(K+1)/2$ restrictions placed on those parameters. As in the short-run model, this order condition is necessary but not sufficient, so the Amisano and Giannini (1997) check for local identification is performed by default.

▷ Example 4: Long-run SVAR model

Suppose that we have a theory in which unexpected changes to the money supply have no long-run effects on changes in output and, similarly, that unexpected changes in output have no long-run effects on changes in the money supply. The $\mathbf{C}$ matrix implied by this theory is

$$\mathbf{C} = \begin{bmatrix} . & 0 \\ 0 & . \end{bmatrix}$$

```
. use https://www.stata-press.com/data/r18/m1gdp
. matrix lr = (.,0\0,.)
. svar d.ln_m1 d.ln_gdp, lreq(lr)
Estimating long-run parameters
  (output omitted)
Structural vector autoregression
 ( 1)  [/C]1_2 = 0
 ( 2)  [/C]2_1 = 0
Sample: 1959q4 thru 2002q2                        Number of obs     =        171
Overidentified model                              Log likelihood    =   1151.614

------------------------------------------------------------------------------
             | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
/C           |
         1_1 |   .0301007   .0016277    18.49   0.000     .0269106    .0332909
         2_1 |          0  (constrained)
         1_2 |          0  (constrained)
         2_2 |   .0129691   .0007013    18.49   0.000     .0115946    .0143436
------------------------------------------------------------------------------
LR test of identifying restrictions: chi2(1) = .1368        Prob > chi2 = 0.712
```

We have assumed that the underlying VAR model has 2 lags; four of the five selection-order criteria computed by `varsoc` (see [TS] **varsoc**) recommended this choice. The test of the overidentifying restrictions provides no indication that it is not valid. ◁

Stored results

svar stores the following in e():

Scalars

e(N)	number of observations
e(N_cns)	number of constraints
e(k_eq)	number of equations in e(b)
e(k_dv)	number of dependent variables
e(ll)	log likelihood from svar
e(N_gaps_var)	number of gaps in the sample
e(k_var)	number of coefficients in a VAR model
e(k_eq_var)	number of equations in an underlying VAR model
e(k_dv_var)	number of dependent variables in an underlying VAR model
e(df_eq_var)	average number of parameters in an equation
e(df_r_var)	if small, residual degrees of freedom
e(obs_#_var)	number of observations on equation #
e(k_#_var)	number of coefficients in equation #
e(df_m#_var)	model degrees of freedom for equation #
e(df_r#_var)	residual degrees of freedom for equation # (small only)
e(r2_#_var)	R^2 for equation #
e(ll_#_var)	log likelihood for equation # VAR model
e(chi2_#_var)	χ^2 statistic for equation #
e(F_#_var)	F statistic for equation # (small only)
e(rmse_#_var)	root mean squared error for equation #
e(mlag_var)	highest lag in VAR model
e(tparms_var)	number of parameters in all equations
e(aic_var)	Akaike information criterion
e(hqic_var)	Hannan–Quinn information criterion
e(sbic_var)	Schwarz's Bayesian information criterion
e(fpe_var)	final prediction error
e(ll_var)	log likelihood from var
e(detsig_var)	determinant of e(Sigma)
e(detsig_ml_var)	determinant of $\widehat{\Sigma}_{\text{ml}}$
e(tmin)	first time period in the sample
e(tmax)	maximum time
e(chi2_oid)	overidentification test
e(oid_df)	number of overidentifying restrictions
e(rank)	rank of e(V)
e(ic_ml)	number of iterations
e(rc_ml)	return code from ml

Macros

e(cmd)	svar
e(cmdline)	command as typed
e(lrmodel)	long-run model, if specified
e(lags_var)	lags in model
e(depvar_var)	names of dependent variables
e(endog_var)	names of endogenous variables
e(exog_var)	names of exogenous variables, if specified
e(nocons_var)	nocons, if noconstant specified
e(cns_lr)	long-run constraints
e(cns_a)	cross-parameter equality constraints on **A**
e(cns_b)	cross-parameter equality constraints on **B**
e(dfk_var)	alternate divisor (dfk), if specified
e(eqnames_var)	names of equations
e(lutstats_var)	lutstats, if specified
e(constraints_var)	constraints_var, if there are constraints on VAR model
e(small)	small, if specified
e(tsfmt)	format of timevar
e(timevar)	name of timevar
e(title)	title in estimation output
e(properties)	b V
e(predict)	program used to implement predict

Matrices

e(b)	coefficient vector
e(Cns)	constraints matrix
e(Sigma)	$\widehat{\boldsymbol{\Sigma}}$ matrix
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
e(bf_var)	full coefficient vector with zeros in dropped lags
e(G_var)	Gamma matrix stored by var; see *Methods and formulas* in [TS] **var**
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(lreq)	lreq(*matrix*), if specified
e(lrcns)	lrcns(*matrix*), if specified
e(Cns_var)	constraint matrix from var, if varconstraints() is specified
e(A)	estimated A matrix, if a short-run model
e(B)	estimated B matrix
e(C)	estimated C matrix, if a long-run model
e(A1)	estimated $\overline{\mathrm{A}}$ matrix, if a long-run model

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

The log-likelihood function for models with short-run constraints is

$$L(\mathbf{A},\mathbf{B}) = -\frac{NK}{2}\ln(2\pi) + \frac{N}{2}\ln(|\mathbf{W}|^2) - \frac{N}{2}\mathrm{tr}(\mathbf{W}'\mathbf{W}\widehat{\boldsymbol{\Sigma}})$$

where $\mathbf{W} = \mathbf{B}^{-1}\mathbf{A}$.

When there are long-run constraints, because $\mathbf{C} = \overline{\mathbf{A}}^{-1}\mathbf{B}$ and $\mathbf{A} = \mathbf{I}_K$, $\mathbf{W} = \mathbf{B}^{-1} = \mathbf{C}^{-1}\overline{\mathbf{A}}^{-1} = (\overline{\mathbf{A}}\mathbf{C})^{-1}$. Substituting the last term for $\mathbf{W}$ in the short-run log likelihood produces the long-run log likelihood

$$L(\mathbf{C}) = -\frac{NK}{2}\ln(2\pi) + \frac{N}{2}\ln(|\widetilde{\mathbf{W}}|^2) - \frac{N}{2}\mathrm{tr}(\widetilde{\mathbf{W}}'\widetilde{\mathbf{W}}\widehat{\boldsymbol{\Sigma}})$$

where $\widetilde{\mathbf{W}} = (\overline{\mathbf{A}}\mathbf{C})^{-1}$.

For both the short-run and the long-run models, the maximization is performed by the scoring method. See Harvey (1990) for a discussion of this method.

Based on results from Amisano and Giannini (1997), the score vector for the short-run model is

$$\frac{\partial L(\mathbf{A},\mathbf{B})}{\partial[\text{vec}(\mathbf{A}),\text{vec}(\mathbf{B})]} = N\left[\{\text{vec}(\mathbf{W}'^{-1})\}' - \{\text{vec}(\mathbf{W})\}'(\widehat{\boldsymbol{\Sigma}}\otimes\mathbf{I}_K)\right]\times$$
$$\left[(\mathbf{I}_K\otimes\mathbf{B}^{-1}), -(\mathbf{A}'\mathbf{B}'^{-1}\otimes\mathbf{B}^{-1})\right]$$

and the expected information matrix is

$$I\left[\text{vec}(\mathbf{A}),\text{vec}(\mathbf{B})\right] = N\begin{bmatrix}(\mathbf{W}^{-1}\otimes\mathbf{B}'^{-1})\\ -(\mathbf{I}_K\otimes\mathbf{B}'^{-1})\end{bmatrix}(\mathbf{I}_{K^2}+\oplus)\left[(\mathbf{W}'^{-1}\otimes\mathbf{B}^{-1}), -(\mathbf{I}_K\otimes\mathbf{B}^{-1})\right]$$

where $\oplus$ is the commutation matrix defined in Magnus and Neudecker (2019, 54–55).

Using results from Amisano and Giannini (1997), we can derive the score vector and the expected information matrix for the case with long-run restrictions. The score vector is

$$\frac{\partial L(\mathbf{C})}{\partial \text{vec}(\mathbf{C})} = N\left[\{\text{vec}(\mathbf{W}'^{-1})\}' - \{\text{vec}(\mathbf{W})\}'(\widehat{\boldsymbol{\Sigma}}\otimes\mathbf{I}_K)\right]\left[-(\overline{\mathbf{A}}'^{-1}\mathbf{C}'^{-1}\otimes\mathbf{C}^{-1})\right]$$

and the expected information matrix is

$$I\left[\text{vec}(\mathbf{C})\right] = N(\mathbf{I}_K\otimes\mathbf{C}'^{-1})(\mathbf{I}_{K^2}+\oplus)(\mathbf{I}_K\otimes\mathbf{C}'^{-1})$$

Checking for identification

This section describes the methods used to check for identification of models with short-run or long-run constraints. Both methods depend on the starting values. By default, `svar` uses starting values constructed by taking a vector of appropriate dimension and applying the constraints. If there are m parameters in the model, the jth element of the $1\times m$ vector is $1+m/100$. `svar` also allows the user to provide starting values.

For the short-run case, the model is identified if the matrix

$$\mathbf{V}^*_{\text{sr}} = \begin{bmatrix}\mathbf{N}_K & \mathbf{N}_K\\ \mathbf{N}_K & \mathbf{N}_K\\ \mathbf{R}_a(\mathbf{W}'\otimes\mathbf{B}) & \mathbf{0}_{K^2}\\ \mathbf{0}_{K^2} & \mathbf{R}_a(\mathbf{I}_K\otimes\mathbf{B})\end{bmatrix}$$

has full column rank of $2K^2$, where $\mathbf{N}_K = (1/2)(\mathbf{I}_{K^2}+\oplus)$, $\mathbf{R}_a$ is the constraint matrix for the parameters in $\mathbf{A}$ (that is, $\mathbf{R}_a\text{vec}(\mathbf{A}) = \mathbf{r}_a$), and $\mathbf{R}_b$ is the constraint matrix for the parameters in $\mathbf{B}$ (that is, $\mathbf{R}_b\text{vec}(\mathbf{B}) = \mathbf{r}_b$).

For the long-run case, based on results from the $\mathbf{C}$ model in Amisano and Giannini (1997), the model is identified if the matrix

$$\mathbf{V}^*_{\text{lr}} = \begin{bmatrix}(\mathbf{I}\otimes\mathbf{C}'^{-1})(2\mathbf{N}_K)(\mathbf{I}\otimes\mathbf{C}^{-1})\\ \mathbf{R}_c\end{bmatrix}$$

has full column rank of K^2, where $\mathbf{R}_c$ is the constraint matrix for the parameters in $\mathbf{C}$; that is, $\mathbf{R}_c\text{vec}(\mathbf{C}) = \mathbf{r}_c$.

The test of the overidentifying restrictions is computed as

$$\text{LR} = 2(\text{LL}_{\text{var}} - \text{LL}_{\text{svar}})$$

where LR is the value of the test statistic against the null hypothesis that the overidentifying restrictions are valid, LL_{var} is the log likelihood from the underlying VAR(p) model, and LL_{svar} is the log likelihood from the SVAR model. The test statistic is asymptotically distributed as $\chi^2(q)$, where q is the number of overidentifying restrictions. Amisano and Giannini (1997, 38–39) emphasize that, because this test of the validity of the overidentifying restrictions is an omnibus test, it can be interpreted as a test of the null hypothesis that all the restrictions are valid.

Because constraints might not be independent either by construction or because of the data, the number of restrictions is not necessarily equal to the number of constraints. The rank of `e(V)` gives the number of parameters that were independently estimated after applying the constraints. The maximum number of parameters that can be estimated in an identified short-run or long-run SVAR model is $K(K+1)/2$. This implies that the number of overidentifying restrictions, q, is equal to $K(K+1)/2$ minus the rank of `e(V)`.

The number of overidentifying restrictions is also linked to the order condition for each model. In a short-run SVAR model, there are $2K^2$ parameters. Because no more than $K(K+1)/2$ parameters may be estimated, the order condition for a short-run SVAR model is that at least $2K^2 - K(K+1)/2$ restrictions be placed on the model. Similarly, there are K^2 parameters in long-run SVAR model. Because no more than $K(K+1)/2$ parameters may be estimated, the order condition for a long-run SVAR model is that at least $K^2 - K(K+1)/2$ restrictions be placed on the model.

Acknowledgment

We thank Gianni Amisano of the Board of Governors of the Federal Reserve System for his helpful comments.

References

Amisano, G., and C. Giannini. 1997. *Topics in Structural VAR Econometrics.* 2nd ed, revised and enlarged. Heidelberg: Springer.

Baum, C. F., and S. Hurn. 2021. *Environmental Econometrics Using Stata.* College Station, TX: Stata Press.

Becketti, S. 2020. *Introduction to Time Series Using Stata.* Rev. ed. College Station, TX: Stata Press.

Christiano, L. J., M. Eichenbaum, and C. L. Evans. 1999. Monetary policy shocks: What have we learned and to what end? In *Handbook of Macroeconomics: Volume 1A*, ed. J. B. Taylor and M. Woodford. New York: Elsevier. https://doi.org/10.1016/S1574-0048(99)01005-8.

Hamilton, J. D. 1994. *Time Series Analysis.* Princeton, NJ: Princeton University Press.

Harvey, A. C. 1990. *The Econometric Analysis of Time Series.* 2nd ed. Cambridge, MA: MIT Press.

Lütkepohl, H. 1993. *Introduction to Multiple Time Series Analysis.* 2nd ed. New York: Springer.

——. 2005. *New Introduction to Multiple Time Series Analysis.* New York: Springer.

Magnus, J. R., and H. Neudecker. 2019. *Matrix Differential Calculus with Applications in Statistics and Econometrics.* 3rd ed. Hoboken, NJ: Wiley.

Rothenberg, T. J. 1971. Identification in parametric models. *Econometrica* 39: 577–591. https://doi.org/10.2307/1913267.

Schenck, D. 2016a. Long-run restrictions in a structural vector autoregression. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2016/10/27/long-run-restrictions-in-a-structural-vector-autoregression/.

——. 2016b. Structural vector autoregression models. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2016/09/20/structural-vector-autoregression-models/.

Sims, C. A. 1980. Macroeconomics and reality. *Econometrica* 48: 1–48. https://doi.org/10.2307/1912017.

Stock, J. H., and M. W. Watson. 2001. Vector autoregressions. *Journal of Economic Perspectives* 15: 101–115. https://doi.org/10.1257/jep.15.4.101.

Watson, M. W. 1994. Vector autoregressions and cointegration. In Vol. 4 of *Handbook of Econometrics*, ed. R. F. Engle and D. L. McFadden. Amsterdam: Elsevier. https://doi.org/10.1016/S1573-4412(05)80016-9.

Also see

[TS] **var svar postestimation** — Postestimation tools for svar

[TS] **tsset** — Declare data to be time-series data

[TS] **var** — Vector autoregressive models+

[TS] **var intro** — Introduction to vector autoregressive models

[TS] **var ivsvar** — Instrumental-variables structural vector autoregressive models+

[TS] **varbasic** — Fit a simple VAR and graph IRFs or FEVDs

[TS] **vec** — Vector error-correction models

[U] **20 Estimation and postestimation commands**

Stata Dynamic Stochastic General Equilibrium Models Reference Manual

For suggested citations, see the FAQ on citing Stata documentation.