Time Series
for GAUSS™
Version 4.0

Aptech Systems, Inc.
# Contents

1 Installation .......................................................... 1

1.1 UNIX ........................................................................ 1
   1.1.1 Download ......................................................... 1
   1.1.2 Floppy ............................................................. 1
   1.1.3 Solaris 2.x Volume Management ............................ 2

1.2 Windows/NT/2000 .................................................... 3
   1.2.1 Download ......................................................... 3
   1.2.2 Floppy ............................................................. 3

1.3 Differences Between the UNIX and Windows/NT/2000 Versions .. 3

2 Getting Started .......................................................... 5

2.0.1 README Files ..................................................... 5

3 VARMA ................................................................. 7

3.1 Introduction .......................................................... 7
   3.1.1 VARMA Global Variables ..................................... 9
   3.1.2 Printing Output .................................................. 10

3.2 VARMA Models ...................................................... 11
Chapter 1

Installation

1.1 UNIX

If you are unfamiliar with UNIX, see your system administrator or system
documentation for information on the system commands referred to below. The device
names given are probably correct for your system.

1.1.1 Download

1. Copy the .tar.gz file to /tmp.
2. Unzip the file.
   `gunzip appxxx.tar.gz`
3. cd to the GAUSS or GAUSS Engine installation directory. We are assuming
   /usr/local/ gauss in this case.
   `cd /usr/local/ gauss`
4. Untar the file.
   `tar xvf /tmp/appxxx.tar`

1.1.2 Floppy

1. Make a temporary directory.
   `mkdir /tmp/workdir`
1. INSTALLATION

2. cd to the temporary directory.
   
   cd /tmp/workdir

3. Use tar to extract the files.

   tar xvf device_name

   If this software came on diskettes, repeat the tar command for each diskette.

4. Read the README file.

   more README

5. Run the install.sh script in the work directory.

   ./install.sh

   The directory the files are install to should be the same as the install directory of GAUSS or the GAUSS Engine.

6. Remove the temporary directory (optional).

The following device names are suggestions. See your system administrator. If you are using Solaris 2.x, see Section 1.1.3.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>3.5-inch diskette</th>
<th>1/4-inch tape</th>
<th>DAT tape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solaris 1.x SPARC</td>
<td>/dev/rfd0</td>
<td></td>
<td>/dev/rst8</td>
</tr>
<tr>
<td>Solaris 2.x SPARC</td>
<td>/dev/rfd0a (vol. mgt. off)</td>
<td>/dev/rst12</td>
<td>/dev/rmt/11</td>
</tr>
<tr>
<td>Solaris 2.x SPARC</td>
<td>/vol/dev/aliases/floppy0</td>
<td>/dev/rst12</td>
<td>/dev/rmt/11</td>
</tr>
<tr>
<td>Solaris 2.x x86</td>
<td>/dev/rfd0c (vol. mgt. off)</td>
<td>/dev/rst12</td>
<td>/dev/rmt/11</td>
</tr>
<tr>
<td>Solaris 2.x x86</td>
<td>/vol/dev/aliases/floppy0</td>
<td>/dev/rst12</td>
<td>/dev/rmt/11</td>
</tr>
<tr>
<td>HP-UX</td>
<td>/dev/rfd0</td>
<td></td>
<td>/dev/rmt/0m</td>
</tr>
<tr>
<td>IBM AIX</td>
<td>/dev/rfd0</td>
<td></td>
<td>/dev/rmt.0</td>
</tr>
<tr>
<td>SGI IRIX</td>
<td>/dev/rdsk/fds0d2.3.5hi</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1.1.3 Solaris 2.x Volume Management

If Solaris 2.x volume management is running, insert the floppy disk and type

   volcheck

to signal the system to mount the floppy.

The floppy device names for Solaris 2.x change when the volume manager is turned off and on. To turn off volume management, become the superuser and type

   /etc/init.d/volmgt off

To turn on volume management, become the superuser and type

   /etc/init.d/volmgt on
1. INSTALLATION

1.2 Windows/NT/2000

1.2.1 Download

Unzip the .zip file into the GAUSS or GAUSS Engine installation directory.

1.2.2 Floppy

1. Place the diskette in a floppy drive.
2. Call up a DOS window
3. In the DOS window log onto the root directory of the diskette drive. For example:
   
   A:<enter>
   cd\<enter>

4. Type: `ginstall source_drive target_path`

   `source_drive`  Drive containing files to install with colon included
                    
                    For example: A:

   `target_path`  Main drive and subdirectory to install to without a final \
                   
                   For example: C:\GAUSS

   A directory structure will be created if it does not already exist and the files will be copied over.

   `target_path\src`  source code files
   `target_path\lib`  library files
   `target_path\examples`  example files

1.3 Differences Between the UNIX and Windows/NT/2000 Versions

- If the functions can be controlled during execution by entering keystrokes from the keyboard, it may be necessary to press `Enter` after the keystroke in the UNIX version.
On the Intel math coprocessors used by the Windows/NT/2000 machines, intermediate calculations have 80-bit precision, while on the current UNIX machines, all calculations are in 64-bit precision. For this reason, GAUSS programs executed under UNIX may produce slightly different results, due to differences in roundoff, from those executed under Windows/NT/2000.
Chapter 2

Getting Started

GAUSS version 3.6.16 or greater and Run–Time Library version 3.6.3 or greater are required to use these routines. See \_rtl\_ver in src/gauss.dec.

The Time Series version number is stored in one of the global variables:

\_ts\_ver  3×1 matrix, the first element contains the major version number, the second element the minor version number, and the third element the revision number.

If you call for technical support, you may be asked for the version of your copy of Time Series.

2.0.1 README Files

The file README.ts contains any last minute information on the Time Series procedures. Please read it before using them.
2. GETTING STARTED
Chapter 3

VARMA

3.1 Introduction

The VARMA library in the TIME SERIES module contains procedures for estimating and analyzing VARMA, VARMAX, ARMA, ARMAX and ECM models.

varmax is the main procedure for estimating VARMA, VARMAX, ARMA, and ARMAX models. Linear and nonlinear equality and inequality constraints may be placed on the parameter estimates, calling the sqpsolve procedure. varmax calls a number of subordinate procedures that enable identification, estimation, diagnostic checking, and forecasting. These are described in the sections below. varmax returns parameter estimates, residuals, and various summary statistics.

ecm is the main procedure for dealing with ECM models. It calls a number of subordinate procedures that enable the recovery and analysis of long-run and short-run parameters and cointegrating vectors. ecm returns parameter estimates (including cointegration coefficients), eigenvalues and eigenvectors (computed using full information maximum likelihood), residuals, and summary statistics.

The varmax and ecm procedures use a full information maximum likelihood (FIML, exact, unconditional) estimation procedure adapted from code developed by Jose Alberto Mauricio of the Universidad Complutense de Madrid. The code was published as Algorithm AS311 in Applied Statistics. It is also described in “Exact maximum likelihood estimation of stationary vector ARMA models”, JASA, 90:282-264. The estimation algorithm assumes that a covariance stationary process is passed to it. Sample means are removed from all data prior to estimation. Further discussion of the estimation method and requirements is contained in Section 3.5.
The `sqpsolve` procedure in the **GAUSS Run-Time Library** links Mauricio’s FIML (exact, unconditional) estimation to constraints. **sqpsolve** uses Newton’s method to minimize the negative of a log-likelihood function subject to different types of constraints.

The following procedures are in the **VARMA** library. In order to use these procedures the **VARMA** library must be active. This is done by including `varma` in the **library** statement at the top of your program, as given below. The first **library** command makes the **varma** and **pgraph** libraries active. The second **library** command makes all the libraries in the **TIME SERIES** module active.

```gauss
library varma, pgraph;
library arima, autoreg, tscs, varma;
```

This enables **GAUSS** to find the procedures and global variables catalogued in these libraries.

*Note that library statements completely replace previous ones. It is therefore highly recommended to have a single library statement in a program.*

**VARMA Library Procedures**

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>armanames</td>
<td>Generates AR and MA names</td>
</tr>
<tr>
<td>coeffprt</td>
<td>Prints coefficient estimates and standard errors</td>
</tr>
<tr>
<td>corm</td>
<td>Prints the correlation matrix of parameters</td>
</tr>
<tr>
<td>covm</td>
<td>Prints the covariance matrix of parameters</td>
</tr>
<tr>
<td>ecm</td>
<td>Estimates an Error Correction Model</td>
</tr>
<tr>
<td>identify</td>
<td>Returns ACF, PACF (univariate), and portmanteau statistics</td>
</tr>
<tr>
<td>macf</td>
<td>Returns an ACF matrix for multivariate models</td>
</tr>
<tr>
<td>nw</td>
<td>Returns the Newey-West Covariance matrix</td>
</tr>
<tr>
<td>paramconfig</td>
<td>Returns coefficient and standard error matrices</td>
</tr>
<tr>
<td>readdata</td>
<td>Reads GAUSS data sets and matrices</td>
</tr>
<tr>
<td>rewwmatrix</td>
<td>Reverses rows of a matrix, in block order</td>
</tr>
<tr>
<td>sumstat</td>
<td>Returns summary statistics from the <code>ecm</code> and <code>varmax</code> procedures</td>
</tr>
<tr>
<td>unitroots</td>
<td>Prints unit root and cointegration test results</td>
</tr>
<tr>
<td>varmaset</td>
<td>Resets the varma global variables</td>
</tr>
<tr>
<td>varmax</td>
<td>Estimates a <strong>VARMAX</strong> model</td>
</tr>
<tr>
<td>vmadf</td>
<td>Computes the Augmented Dickey Fuller statistic, allowing for deterministic</td>
</tr>
<tr>
<td></td>
<td>polynomial time trends of an arbitrary order</td>
</tr>
<tr>
<td>vmcadf</td>
<td>Computes the Augmented Dickey Fuller statistic applied to the residuals of</td>
</tr>
<tr>
<td></td>
<td>a cointegrating regression, allowing for deterministic polynomial time</td>
</tr>
<tr>
<td></td>
<td>trends of an arbitrary order</td>
</tr>
<tr>
<td>vmc__sja</td>
<td>Returns critical values for Johansen’s Maximum Eigenvalue statistic.</td>
</tr>
<tr>
<td>vmc__sjt</td>
<td>Returns critical values for Johansen’s Trace statistic.</td>
</tr>
</tbody>
</table>
3. VARMA

vmdetrend
Returns residuals from regressing on a time trend polynomial

vmdiff
Differences a time series matrix

vmforecast
Forecasts VARMAX models

vmpp
Performs Phillips-Perron unit root tests

vmptrend
Creates a polynomial matrix of time trends of order p.

vmyroots
Finds characteristic roots of AR and MA equations

vmrztcrit
Returns τ critical values for the Augmented Dickey-Fuller statistic, derived from the residuals of a cointegrating regression. Depends on p, the AR order in the fitted regression, the number of observations, and the number of explanatory variables.

vmsj
Calculates Johansen’s Trace and Maximum Eigenvalue test statistics.

vmztcrit
Returns τ critical values for the Augmented Dickey-Fuller test statistic, depending on the number of observations and p, the AR order in the fitted regression.

3.1.1 VARMA Global Variables

The table below contains a list of the numerous VARMA library global variables. They give the user considerable control over the model’s specification and estimation. A more complete description of their use is in the following sections.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_vm_A</td>
<td>matrix, linear equality constraint coefficients</td>
</tr>
<tr>
<td>_vm_adforder</td>
<td>scalar, number of AR lags in the ADF test statistic</td>
</tr>
<tr>
<td>_vm_B</td>
<td>matrix, linear equality constraint constants</td>
</tr>
<tr>
<td>_vm_Bounds</td>
<td>matrix, upper and lower bounds on parameter estimates</td>
</tr>
<tr>
<td>_vm_C</td>
<td>matrix, linear inequality constraint coefficients</td>
</tr>
<tr>
<td>_vm_D</td>
<td>matrix, linear inequality constraint constants</td>
</tr>
<tr>
<td>_vm_DirTol</td>
<td>scalar, convergence tolerance for the gradient</td>
</tr>
<tr>
<td>_vm_EqProc</td>
<td>function, used to specify nonlinear equality constraints.</td>
</tr>
<tr>
<td>_vm_FeasibleTest</td>
<td>scalar flag, test for parameter feasibility</td>
</tr>
<tr>
<td>_vm_Hessian</td>
<td>matrix, the estimated Hessian from ecm or varmax.</td>
</tr>
<tr>
<td>_vm_IndEquations</td>
<td>function, used to specify nonlinear inequality constraints.</td>
</tr>
<tr>
<td>_vm_IneqProc</td>
<td>vector, Lagrange coefficients for the constraints</td>
</tr>
<tr>
<td>_vm_Lagrange</td>
<td>scalar, lags over which the ACF and Diagnostics are defined.</td>
</tr>
<tr>
<td>_vm_lags</td>
<td>scalar, maximum number of iterations</td>
</tr>
<tr>
<td>_vm_MaxIters</td>
<td>scalar, controls the constant term in the Johansen tests.</td>
</tr>
<tr>
<td>_vm_NoDet</td>
<td>scalar, the Newey-West truncation lag</td>
</tr>
<tr>
<td>_vm_Output</td>
<td>scalar, determines the output to be printed</td>
</tr>
<tr>
<td>_vm_PrintIters</td>
<td>scalar flag, to print each iteration’s information</td>
</tr>
<tr>
<td>_vm_RandRadius</td>
<td>scalar, radius of random search if STEPBT line search fails.</td>
</tr>
<tr>
<td>_vm_scale</td>
<td>vector, used to scale time series</td>
</tr>
</tbody>
</table>
3. VARMA

VM_SETCONSTRAINTS scalar flag, impose stationarity and invertibility
VM_START vector, set start values
VM_TRUSTRADIUS scalar, max. amount of direction vector at each iteration.
VMCRITL scalar, the significance level for ACF indicator matrices

3.1.2 Printing Output

Three global variables, __output, __vm_output, and __vm_PrintIters determine the output that is displayed from the ecm, varmax, sqpsolve, and subordinate procedures.

1. Set __output = 0 to suppress all printing from the sqpsolve procedure.
2. Set __vm_PrintIters = 0 (__output is not equal to zero) to print an Executing... message while starting values are calculated for each dependent variable during the sqpsolve operation.
3. Set __vm_PrintIters > 0 (__output is not equal to zero) to print sqpsolve iteration information. This information includes the value of the objective function and the gradient at each estimated coefficient. It is useful in finding where and why convergence might fail.
4. Set __output > 0 to print sqpsolve results.

__vm_output is either a scalar or a 6 × 1 vector. Set __vm_output = 0 to suppress all printing from the ecm and varmax estimations. Set __vm_output > 0 to print all ecm and varmax output. Define __vm_output as a 6 × 1 vector to control the printing of various parts of ecm and varmax output.

1. Set element [1] of __vm_output to a non-zero value to print the model’s header
2. Set element [2] of __vm_output to a non-zero value to print a variety of unitroot tests, and, if a multivariate model, cointegration tests from the unitroots procedure.
3. Set element [3] of __vm_output to a non-zero value to print summary statistics for each estimated equation
4. Set element [4] of __vm_output to a non-zero value to print the estimated coefficients and their standard errors
5. Set element [5] of __vm_output to a non-zero value to print the roots of the AR and MA characteristic equations
6. Set element [6] of __vm_output to a non-zero value to print the autocorrelation function and portmanteau statistics
3. VARMA

3.2 VARMA Models

A stationary and centered (means-removed) VARMAX model may be written as:

$$Y_t - \sum_{j=1}^{p} \Phi_j Y_{t-j} + \beta X_t = \varepsilon_t - \sum_{i=1}^{q} \Theta_i \varepsilon_{t-i}$$

for $t = 1 \cdots T$ where $Y_t$ has dimension $L \times 1$, $\varepsilon_t$ is a zero mean covariance stationary process that is normally distributed with positive definite covariance matrix $\Sigma$, and $X_t$ is a $K \times 1$ vector of fixed explanatory variables. The $\Phi$ and $\Theta$ matrices have dimension $L \times L$. The $\beta$ coefficients have dimension $L \times K$.

Another way to write the same system is using the backshift operator, $B$:

$$\Phi_p(B)Y_t + \beta X_t = \Theta(B)\varepsilon_t$$

(3.1)

where $\Phi_p(B) = \Phi_0 - \Phi_1 B - \ldots - \Phi_p B^p$ and $\Theta_q(B) = \Theta_0 - \Theta_1 B - \ldots - \Theta_q B^q$ are matrix polynomials and $\Phi_0$ and $\Theta_0$ are nonsingular matrices of dimension $L \times L$ (often assumed to be the identity matrices).

3.2.1 Stationarity and Invertibility

The VARMAX process is stationary if the roots of $\det(\Phi_p(B))$ are greater than one in modulus. The VARMAX process is invertible if the roots of $\det(\Theta(B))$ are greater than one in modulus. The vmroots procedure finds the AR and MA characteristic roots and their moduli. The roots and their moduli are printed if _vm_output[5] is nonzero.

3.3 Unit Root and Cointegration Tests

Much applied research tests whether theoretically predicted relationships among variables are confirmed in the real world. Other research involves forecasting, whether from a naive time series model or using a structural model based on behavior. In all cases it is important to work with stationary or cointegrated variables. Spurious correlation may result if the relationships between nonstationary series are examined. In addition, forecast variances for nonstationary series increase without bound.

Model building involves first testing for unit roots. The vmadf procedure performs Dickey-Fuller (DF) and Augmented Dickey-Fuller (ADF) unit root tests. The vmpp procedure performs Phillips-Perron (PP) unit root tests.
Cointegration tests follow, to ward off spurious estimated relationships. The `vmcadf` procedure performs ADF cointegration tests. The `vmsj` procedure performs Johansen’s Trace and Maximum Eigenvalue cointegration tests.

The `COINT` module, written by Sam Ouliaris and Peter C.B. Phillips and sold by Aptech Systems, Inc., contains numerous other unit root and cointegration tests, including Park-Choi (1988) G(p,q) and J(p,q) tests, the Stock and Watson (1988) common trends test, and the Phillips-Ouliaris (1990) P(u) and P(z) tests. The `COINT` module also contains numerous (time and frequency domain) methods for estimating the cointegrating vector, ARMA models, various model selection criteria, spectral density estimation, and long-run variance estimation.

3.3.1 Univariate Unit Root Tests

The `unitroots` procedure calls a variety of unit root and cointegration tests and prints the results. The univariate unit root test statistics calculated are the Dickey-Fuller, Augmented Dickey-Fuller (both called with the `vmadf` procedure) and Phillips-Perron (called with `vmpp`) statistics.

DF and ADF Unit Root Tests

The `vmadf` procedure calculates Dickey-Fuller and Augmented Dickey-Fuller unit root test statistics, returning the statistic, its \( \tau \) statistic, and a \( 6 \times 1 \) vector of critical values. Three specifications are typically analyzed, a random walk with drift and trend, a random walk with drift, and a random walk:

\[
\Delta Y_t = \alpha + \beta t + (\rho - 1)Y_{t-1} + \sum_{i=1,2,\ldots} \rho_i \Delta Y_{t-i} + \varepsilon_t \quad (3.2)
\]
\[
\Delta Y_t = \alpha + (\rho - 1)Y_{t-1} + \sum_{i=1,2,\ldots} \rho_i \Delta Y_{t-i} + \varepsilon_t \quad (3.3)
\]
\[
\Delta Y_t = (\rho - 1)Y_{t-1} + \sum_{i=1,2,\ldots} \rho_i \Delta Y_{t-i} + \varepsilon_t \quad (3.4)
\]

The time polynomial input argument to `vmadf` determines which of the above models will be estimated.

The Dickey-Fuller test assumes independent and identically distributed errors. This assumption precludes models with lagged dependent variables, (i.e. the lagged dependent variable terms in specifications (3.2) to (3.4) are not estimated) since lags induce dependency in the errors.

The Augmented Dickey-Fuller test eliminates serial correlation in the residuals by including lagged dependent variables in the specification. The `__vm__adforder` global variable sets the number of AR terms to include in the Augmented Dickey-Fuller test statistic calculations. The default is `__vm__adforder = 2`. 
3. VARMA

Phillips-Perron Unit Root Tests

Phillips (1987) and Phillips and Perron (1988) test for unit roots by adjusting the OLS estimate of an AR(1) coefficient for serial correlation in the OLS residuals. Three specifications are considered, an AR(1) model without a drift, an AR(1) with a drift, and an AR(1) model with a drift and linear trend:

\begin{align*}
Y_t &= \rho Y_{t-1} + \varepsilon_t \quad (3.5) \\
Y_t &= \alpha + \rho Y_{t-1} + \varepsilon_t \quad (3.6) \\
Y_t &= \alpha + \delta t + \rho Y_{t-1} + \varepsilon_t \quad (3.7)
\end{align*}

The unit root null hypothesis is \( H_0: (\rho - 1) = 0 \).

Hamilton (1994, pp. 506-511) tests this hypothesis using two statistics that are analogs of the Phillips and Perron (1988) \( Z_\alpha \) and \( Z_t \) statistics. Hamilton’s statistics are based on OLS estimation of (3.5) to (3.7). They allow an identical formula for each statistic to be used for all three cases.

The \texttt{vmpp} procedure returns the \( Z_t \) statistic as calculated by Hamilton and critical values. Suppose any one of the above equations is estimated by OLS, returning \( \hat{\rho}_T \) and \( \hat{\sigma}_{\varepsilon_t}^2 \) (the OLS estimates of \( \rho \) and the standard error of \( \hat{\rho}_T \) respectively), \( t_T = (\rho - 1) / \hat{\sigma}_{\varepsilon_t}^2 \) (the usual OLS \( t \) statistic for testing \( H_0 \)), \( \hat{\varepsilon}_t \) (the OLS residuals), and \( s_T \) (the estimated standard error of the regression).

Hamilton's \( Z_t \) statistic is:

\[
Z_t = (\hat{\gamma}_0 / \hat{\lambda}^2)^{1/2} t_T - \left\{ \frac{1}{2} (\hat{\lambda}^2 - \hat{\gamma}_0) / \hat{\lambda} \right\} \{ T(\hat{\sigma}_{\varepsilon_t}^2 / s_T) \}
\]

\( \hat{\lambda}^2 \) is an estimate of the asymptotic variance of the sample mean of \( \varepsilon_t \). In the \texttt{vmpp} procedure \( \hat{\lambda}^2 \) is estimated using the Newey-West (1987) estimator,

\[
\hat{\lambda}^2 = \hat{\gamma}_0 + 2 \sum_{j=1}^q [1 - j/(q + 1)] \hat{\gamma}_j
\]

where \( \hat{\gamma}_j = T^{-1} \sum_{t=j+1}^T \hat{\varepsilon}_t \hat{\varepsilon}_{t-j} \) are the sample autocovariances of \( \varepsilon_t \).

A global variable, \texttt{_vm_nwtrunc}, sets the number of autocorrelations to use in calculating the Newey-West correction (\( q \) in the above equation). The default setting, \texttt{_vm_nwtrunc} = 0, causes \texttt{GAUSS} to use a truncation lag given by Newey and West, \( q = 4(T/100)^{2/9} \).

Under the null hypothesis, the \( Z_t \) statistics has the same asymptotic distribution as Dickey-Fuller statistics.
3.3.2 Cointegration Tests

Residual Based Cointegration Tests

Cointegration tests fit into two categories, those based on single-equation estimation methods and those based on estimating systems of equations. Single equation tests involve testing for a unit root in the residuals that result from regressing one series on another. The Augmented Dickey-Fuller (ADF) test may be used for this purpose. The `vmcadf` procedure implements the ADF test for cointegration. The `vmrztcrit` procedure returns critical values for ADF cointegration tests.

System Based Cointegration Tests

Maddala and Kim (1998, p 211) note that single equation cointegration test results depend on the variable used to normalize the cointegrating relationship. In addition, the number of cointegrating relationships cannot be determined using single equation tests. These problems are avoided using tests based on systems of equations. System based cointegration tests examine the dimension of the cointegrating space across two or more variables.

The `vmsj` procedure implements Johansen’s (1988) Trace and Maximum Eigenvalue system-based cointegration tests, using an ECM model. See chapter 4 for further details. The null hypothesis under the Trace test is that the cointegrating space has dimension less than or equal to \( r \). The alternative hypothesis is that there are more than \( r \) cointegrating vectors. The null hypothesis under the Maximum Eigenvalue test is that there are \( r + 1 \) cointegrating vectors versus the alternative that there are \( r \) cointegrating vectors.

The `vmsj` procedure returns the Trace and Maximum Eigenvalue test statistics. The `vmc_sjt` procedure returns Trace critical values at the 1%, 5%, 10%, 90%, 95%, and 99% levels. The `vmc_sja` procedure returns Maximum Eigenvalue critical values at the 1%, 5%, 10%, 90%, 95%, and 99% levels.

3.4 Identification

The first step in time series analysis is the identification of a stationary time series process. The `identify` procedure returns a number of statistics that are useful in identification. For univariate models, `identify` returns ACF and PACF functions and Ljung-Box statistics. The ACF and PACF functions examine individual autocorrelations across different lags while the Ljung-Box statistics summarize all autocorrelations over a given number of lags. All are calculated across the number of lags specified in `_vm_lags`. The default is `vm_lags = 12`. 
3. VARMA

The Ljung-Box statistics (see Ljung and Box (1978)) test:

\[ H_0 : \rho_1 = \rho_2 = \ldots = \rho_s = 0 \]

where \( \rho_j \) is the population correlation between the ARMAX disturbances at time \( t \) and the ARMAX disturbances at time \( t - j \). The statistics are defined by:

\[
Q_s = T(T + 2) \sum_{j=1}^{s} \left[ \frac{r_j^2}{T-j} \right]
\]

where \( r_j \) is the sample correlation between the ARMAX residuals at time \( t \) and the ARMAX residuals at time \( t - j \). Under \( H_0 \), \( Q_s \) has a chi-squared distribution with \( (s - \text{the number of parameters estimated}) \) degrees of freedom.

The `sumstat` procedure has two returns, a \( 2 \times L \) matrix containing the sum of squares \( Y (SS_{yy}) \) and the sum of squared errors for each dependent variable and a \( 4 \times L \) matrix containing information criteria for each dependent variable, in the following order:

**row 1** The minimized log-likelihood value, \( F \).

**row 2** The Akaike Information Criterion (AIC) = \( 2 \times (F + \text{the number of estimated parameters}) \).

**row 3** The Schwarz Bayesian Information Criterion (BIC) = \( 2 \times F + (\text{the number of estimated parameters}) \times \ln(\text{number of observations}) \).

**row 4** The Likelihood Ratio Statistic (LRS) = \( 2 \times F \).

Identification information is printed if `vm_output[6]` is nonzero. Summary statistics are printed if `vm_output[3]` is nonzero. For univariate model output (i.e. `vm_output[6]` is nonzero), identify flags, with * and ** symbols, ACF and PACF values that are significant at the 5% and 1% levels (using Bartlett’s large sample approximation for the standard errors, \( 1/\sqrt{T} \)).

### 3.4.1 Multivariate Identification

For multivariate processes, `identify` returns ACF matrices and a multivariate portmanteau lack of fit statistic, \( Q_s \). PACF values, returned for univariate processes, are not returned for multivariate models. `sumstat` returns sums of squares and the information vector for multivariate models.
Multivariate Portmanteau Statistic

A multivariate portmanteau statistic (see Hosking (1980), Poksitt and Tremayne (1982), Li and McLeod (1981)) described in Reinsel (1993, p 133) is used to examine the residual autocorrelation matrices en toto.

Let the residuals be $\varepsilon_t$. Define the residual covariance matrix as

$$C_\varepsilon(l) = \frac{1}{T} \sum_{t=1}^{T-l} \varepsilon_t \varepsilon_t'$$

$$C_\varepsilon(0) = \frac{1}{T} \sum_{t=1}^{T} \varepsilon_t \varepsilon_t'$$

The residual autocorrelation matrix is

$$\hat{\rho}_\varepsilon(l) \equiv \hat{V}^{-1/2} C_\varepsilon(l) \hat{V}^{-1/2} = \hat{\rho}_{ij}(l)$$

$$Q_s = T^2 \sum_{l=1}^{s} (T-l)^{-1} \sum_{i=1}^{k} \sum_{j=1}^{k} C_{ij}(l) r_{ji}(-l)$$

$$Q_s = T^2 \sum_{l=1}^{s} (T-l)^{-1} tr\{C_\varepsilon(l) \hat{\Sigma}^{-1} C_\varepsilon(-l) \hat{\Sigma}^{-1}\}$$

$$Q_s = T^2 \sum_{l=1}^{s} (T-l)^{-1} tr\{\hat{\rho}_\varepsilon(l) \hat{\rho}_\varepsilon(0)^{-1} \hat{\rho}_\varepsilon(-l) \hat{\rho}_\varepsilon(0)^{-1}\}$$

Under the null hypothesis:

$H_0 : Y_t$ is an ARMA$(p,q)$ process

$H_1 : \text{not } H_0$

and assuming that $s$ is large, the $Q_s$ statistic has approximately a $\chi^2(L^2(s-p-q))$ distribution.

Wei (1990) notes that without further information a VARMA process may not be uniquely identified from its ACF function. Hannan (1969, 1970, 1976, and 1979) describes additional restrictions needed to identify a VARMA process.

If _vm_output[3] is nonzero, ACF and indicator matrices are printed, together with the portmanteau statistic. The indicator matrices contain + and - symbols, depending whether the individual autocorrelations are significantly positive or negative at the level specified in _vmcritl, using Bartlett’s approximation, $1/\sqrt{T}$, as the large sample standard error of each autocorrelation. The macf procedure calculates the ACF matrices.
3. **VARMA**

### 3.5 Estimation

**VARMA** and **ecm** use a full information maximum likelihood (FIML, exact, unconditional) estimation procedure adapted from code developed by Jose Alberto Mauricio of the Universidad Complutense de Madrid. The code was published as Algorithm AS311 in Applied Statistics. It is also described in “Exact maximum likelihood estimation of stationary vector ARMA models”, JASA, 90:282-264. Sample means are removed from all data prior to estimation and errors are assumed to be distributed $N(0, \Sigma)$.

Linear and non-linear constraints may be imposed on the coefficient estimates, invoking the **sqpsolve** procedure. For example, setting _vm__SetConstraints to a nonzero value enforces the stationarity required by the estimation procedure, by constraining the roots of the characteristic equation

$$I - \Phi_1 z - \Phi_2 z^2 - \cdots - \Phi_p z^p$$

to be outside the unit circle (where $\Phi_i$, $i = 1, \ldots, p$ are the $AR$ coefficient matrices).

As noted earlier, the **vmroots** procedure returns roots of the $AR$ and $MA$ characteristic equations. The roots are printed if _vm__output[5] is nonzero.

If any estimated parameters are on a constraint boundary, the Lagrangeans associated with these parameters will be nonzero. These Lagrangeans are stored in the global _vm__Lagrange, a compact matrix created using **vput**. The **vread** procedure is used to retrieve these estimates. Standard errors are generally not available for parameters on constraint boundaries.

#### 3.5.1 Quasi-Maximum Likelihood Covariance Matrix of Parameters

**VARMA** and **ecm** compute a QML covariance matrix of the parameters when requested. Let $F$ be the log-likelihood function. Define $B = (\partial F_A/\partial \theta)'(\partial F_A/\partial \theta)$ evaluated at the estimates. The covariance matrix of the parameters is $\Omega^{-1} B \Omega^{-1}$ where $\Omega$ is $(\partial F_A^2/\partial \theta' \partial \theta)$.

To request the QML covariance matrix, set _vctype equal to one. The default, _vctypet = 0, is ML estimation of the covariance matrix.

#### 3.5.2 Starting Values

The time that **sqpsolve** needs to reach a solution is often reduced significantly when starting values are specified. The **ecm** and **VARMA** procedures fit univariate $ARMA$
models to generate starting values for each $Y$ variable in the model, unless the user supplies their own starting values in the `vm_start` global variable. Starting values must be specified by the user when the computed starting point fails or when there are inequality constraints. The latter case requires a starting point which is feasible, i.e. one that satisfies the inequality constraints.

Starting values are entered into `vm_start` in a well-defined order:

1. The $AR$ coefficient matrices, if any, stored row-wise.
2. The $MA$ coefficient matrices, if any, stored row-wise.
3. The lower left nonredundant portion of the covariance matrix of the residuals, if the model is multivariate, stored row-wise (for univariate models, the log-likelihood function has the variance concentrated out of it).
4. The matrix of regression coefficients, if there are exogenous variables, stored row-wise.

### 3.6 SQPSolve and Newton’s Method

`ecm` and `varmax` minimize a log-likelihood function. When constraints exist (see section 3.7 for a discussion of constraints and how to place them), `sqpsolve` uses Newton’s method to minimize the log-likelihood function. This section provides a summary of the `sqpsolve` method. The reader is referred to Han (1977) for further details.

Newton’s method minimizes functions iteratively. Each iteration involves evaluating the function and determining the direction to move in the domain of the function that results in the greatest increase in the function’s value. Given the direction, the STEPBFE line search method determines the `step length` that results in a lower objective function. See Dennis and Schnabel (1983) for a discussion of the STEPBFE line search method.

Initial values for the unknown coefficients are required for the first iteration. These are generated automatically by `ecm` or `varmax` if `vm_start` is a missing value (the default). They may also be set by the user in `vm_start`, in the column vector order given earlier:

1. The $AR$ coefficient matrices, if any, stored row-wise.
2. The $MA$ coefficient matrices, if any, stored row-wise.
3. The lower left nonredundant portion of the covariance matrix of the residuals, if the model is multivariate, stored row-wise (for univariate models, the log-likelihood function has the variance concentrated out of it).
3. VARMA

4. The matrix of regression coefficients, if there are exogenous variables, stored row-wise.

Let $F$ be the log-likelihood function. \texttt{sqpsolve} minimizes $F$ within the context of a standard nonlinear programming problem:

$$\min F(\theta)$$

subject to the linear constraints,

$$A\theta = B$$
$$C\theta \geq D$$

the nonlinear constraints,

$$G(\theta) = 0$$
$$H(\theta) \geq 0$$

and bounds,

$$\theta_l \leq \theta \leq \theta_u$$

$G(\theta)$ and $H(\theta)$ are functions provided by the user and must be differentiable at least once with respect to $\theta$. $F(\theta)$ must have first and second derivatives with respect to the parameters, and the matrix of second derivatives (the Hessian, $\Sigma$ below) must be positive semi-definite.

Without loss of generality, we assume that the linear constraints and bounds have been incorporated into $G$ and $H$. However, in practice, linear constraints are specified separately from $G$ and $H$ because their Jacobians are known and easy to compute. Bounds constraints are also more easily handled separately from the linear inequality constraints.

Successive parameter values are defined by:

$$\theta_{t+1} = \theta_t + \rho d$$

where $\theta_t$ are the parameter values at time $t$, $d$, the direction, is an $NP \times 1$ vector ($NP$ is the number of coefficients) and $\rho$ is the step length, a scalar that applies equally to each element of $d$.

The direction, $d$, solves the quadratic program

$$\minimize \frac{1}{2} d^\top \Sigma(\theta_t)d + \Psi(\theta_t)d$$
subject to \[ \dot{G}(\theta_t)d + G(\theta_t) = 0 \]
\[ \dot{H}(\theta_t)d + H(\theta_t) \geq 0 \]

where \( \Sigma \) is positive semi-definite. The \( \Sigma(\theta) \) and \( \Psi(\theta) \) matrices are given by:

\[ \Sigma(\theta) = \frac{\partial^2 F}{\partial \theta \partial \theta'} \]
\[ \Psi(\theta) = \frac{\partial F}{\partial \theta} \]

and the Jacobians are:

\[ \dot{G}(\theta) = \frac{\partial G(\theta)}{\partial \theta} \]
\[ \dot{H}(\theta) = \frac{\partial H(\theta)}{\partial \theta} \]

sqpsolve computes the Hessian \( \Sigma \), \( \Psi \), various gradients, and the Jacobians, \( \dot{G}(\theta) \), and \( \dot{H}(\theta) \) using numerical methods.

Given \( \theta_t \) and \( d \), the STEPBT line search method finds the step length, \( \rho \), by minimizing the merit function:

\[ m(\theta_t + \rho d) = F + \max | \kappa | \sum_j g_j(\theta_t + \rho d) | - \max | \lambda | \sum_\ell \min(0, h_\ell(\theta_t + \rho d)) \]

as a scalar function of \( \rho \), where \( g_j \) is the \( j \)-th row of \( G \), \( h_\ell \) is the \( \ell \)-th row of \( H \), \( \kappa \) is the vector of Lagrangean coefficients of the equality constraints, and \( \lambda \) the Lagrangean coefficients of the inequality constraints.

STEPBT first approximates \( m \) as a quadratic function, and computes \( \rho \) to minimize the quadratic. If a feasible \( \rho \) does not exist, it attempts to fit a cubic function. If the cubic function fails and \( \_vm\_RandRadius = 0 \), sqpsolve stops iterating, without a solution.

Set \( \_vm\_RandRadius > 0 \) to have sqpsolve enter a random search in case the cubic loss function fitting fails. In a random search, sqpsolve chooses a random direction from the current point, within the radius set by \( \_vm\_RandRadius \). If the \( \_vm\_RandRadius \) global is set to zero, a random search will not be attempted and the iterations will terminate.

A poor starting point and an excessively large direction can often put the sqpsolve iterations into an ill-defined regions, from which the iterations cannot escape. To avoid this, a “trust region” can be defined to limit the direction (see Fletcher (1985)).

Setting \( \_vm\_TrustRadius \) imposes boundary constraints on the direction, relative to the starting position of the iterations. The direction is constrained to be no greater than \( \_vm\_TrustRadius \) in absolute value.

20
3. **VARMA**

### 3.7 Settings Constraints

General constraints may be placed on parameters of `VARMA` models. There are five types of constraints: linear equality, linear inequality, nonlinear equality, nonlinear inequality, and bounds. These are not exclusive categories (i.e., there are several ways most constraints can be placed.) Below we give examples of specifying constrained parameters.

#### 3.7.1 Constraints and the Coefficient vector

Log-likelihood optimization is conducted by the `sqpsolve` function. `sqpsolve` “sees” all parameters in the model as a single vector. This parameter vector must be used to place constraints. It has the same order as the `vm_start` vector, i.e.

1. The `AR` coefficient matrices, if any, stored row-wise.
2. The `MA` coefficient matrices, if any, stored row-wise.
3. The lower left nonredundant portion of the covariance matrix of the residuals, if the model is multivariate, stored row-wise (for univariate models, the log-likelihood function has the variance concentrated out of it).
4. The matrix of regression coefficients, if there are exogenous variables, stored row-wise.

Some attention may have to be paid to the starting point when there are inequality constraints placed on the parameters. In general, `sqpsolve` requires a starting point that satisfies inequality constraints. You may need to provide starting values if the ones computed by `varmax` do not satisfy the inequality constraints. It is not necessary for starting points to satisfy equality constraints.

To force a starting point on `sqpsolve`, assign the selected vector of starting values to the global `vm_start`. For example, for an AR(1) model

```plaintext
_vm_start = { .5, 0, 0, .5, 1, 0, 1 }
```

#### 3.7.2 Linear Equality Constraints

For computational convenience linear equality constraints are treated separately from general nonlinear constraints. Let \( \theta \) be the coefficient vector. Linear constraints are described as:

\[ A\theta = B \]
To place a linear equality constraint, $A$ is assigned to the global \texttt{vm\_A} and $B$ is assigned to global \texttt{vm\_B}. For example suppose we wish to constrain the first AR coefficient matrix of a bivariate AR(2) model to equal zero. The coefficient vector looks like this:

\[
\begin{align*}
\phi_{111} \\
\phi_{121} \\
\phi_{211} \\
\phi_{221} \\
\phi_{112} \\
\phi_{122} \\
\phi_{212} \\
\phi_{222} \\
\sigma_{11} \\
\sigma_{21} \\
\sigma_{22}
\end{align*}
\]

Then in the command file we define the globals:

\[
\texttt{\_vm\_A} = \{ 1 0 0 0 0 0 0 0 0 0, \\
0 1 0 0 0 0 0 0 0 0, \\
0 0 1 0 0 0 0 0 0 0, \\
0 0 0 1 0 0 0 0 0 0 \};
\]

\[
\texttt{\_vm\_B} = \{ 0, 0, 0, 0 \} ;
\]

This constrains the first four elements of the parameter vector to zero.

### 3.7.3 Linear Inequality Constraints

Linear inequality constraints are described as:

\[
C\theta \geq D
\]

To place a linear inequality constraint, $C$ is assigned to the global \texttt{vm\_C} and $D$ is assigned to global \texttt{vm\_D}. For example, suppose a bivariate AR(1) model is specified. We wish to constrain the diagonal elements of the AR coefficient matrix to be greater than the off diagonal elements. The coefficient vector looks like this:

\[
\begin{align*}
\phi_{11} \\
\phi_{12} \\
\phi_{21} \\
\phi_{22} \\
\sigma_{11} \\
\sigma_{21} \\
\sigma_{22}
\end{align*}
\]

In the command file we define the globals:

\[
\texttt{\_vm\_B} = \{ 0, 0, 0, 0 \} ;
\]
3. VARMA

\_vm\_C = \{ 1 -1 0 0 0 0 0,
1 0 -1 0 0 0 0,
0 -1 0 1 0 0 0,
0 0 -1 1 0 0 0 \};

\_vm\_D = \{ 0, 0, 0, 0 \};

3.7.4 Nonlinear Equality Constraints

Nonlinear equality constraints are defined as

\[ G(\theta) = 0 \]

i.e. values for \( \theta \) are found such that \( G(\theta) = 0 \).

Nonlinear constraints are placed by supplying a GAUSS procedure for \( G \). sqpsolve finds parameter estimates, \( \hat{\theta} \) such that \( G(\hat{\theta}) = 0 \).

To place a nonlinear equality constraint, write a procedure taking the parameter vector as an input argument and returning a vector. Each element of the return vector represents a different constraint.

The following code, added to the command file, constrains the singular values of a bivariate AR(2) model coefficient matrices to be equal:

```
proc eqp(b);
  local phi1,phi2,s1,s2;
  phi1 = reshape(b[1:4],2,2);
  phi2 = reshape(b[5:8],2,2);
  s1 = svd(phi1);
  s2 = svd(phi2);
  retp(s1-s2);
endp;

\_vm\_EqProc = &eqp;
```

3.7.5 Nonlinear Inequality Constraints

Nonlinear inequality constraints are defined as:

\[ H(\theta) \geq 0 \]
Nonlinear inequality constraints are placed by providing a \texttt{GAUSS} procedure for \( H \). \texttt{sqpsolve} finds parameter estimates, \( \hat{\theta} \) such that \( H(\hat{\theta}) \geq 0 \).

To place a nonlinear inequality constraint, write a procedure taking the parameter vector as an input argument and returning a vector. Each element of the return vector represents a different constraint.

For example, for a bivariate AR(2) model, the following constrains the absolute value of the eigenvalues of the first coefficient matrix to be greater than the eigenvalues of the second coefficient matrix:

```plaintext
proc ineqp(b);
    local phi1,phi2,l1,l2;
    phi1 = reshape(b[1:4],2,2);
    phi2 = reshape(b[5:8],2,2);
    l1 = abs(eig(phi1));
    l2 = abs(eig(phi2));
    retp(l1-l2);
endp;

_vm_IneqProc = &ineqp;
```

### 3.7.6 Bounds Constraints

Bounds are a type of linear inequality constraint but are treated separately by \texttt{sqpsolve} for computational convenience. To place bounds on parameters, lower and upper values are entered into the global \texttt{vm.bounds}. For example, to bound the coefficients of an AR(1) model to be between -.5 and +.5 define

```plaintext
_vm_bounds = { -.5 .5,
               -.5 .5,
               -.5 .5,
               -.5 .5,
               -1e256 1e256,
               -1e256 1e256,
               -1e256 1e256};
```

The first column of \texttt{vm.bounds} corresponds to the lower boundaries and the second column the upper boundaries. The first four rows correspond to the \( AR \) coefficients in the parameter vector, and the last three rows to the elements of the covariance matrix of the residuals which we choose not to constrain.
3. VARMA

3.8 SQPSolve and Managing Optimization

The critical elements in optimization are scaling, the starting point, and the condition of the model. When the starting point is reasonably close to the solution and the model is well-specified and reasonably scaled, the iterations converge quickly and without difficulty.

3.8.1 Scaling

For best performance, the diagonal elements of the Hessian matrix should be roughly equal (the \texttt{vm\_Hessian} global contains the estimated Hessian). \texttt{sqpsolve} has difficulty converging when some diagonal elements contain numbers that are very large and/or very small with respect to the others. It may not be obvious how to scale the diagonal elements of the Hessian. However, ensuring that the data are of the same magnitude may help.

The \texttt{vm\_scale} global variable, used to scale the data, is either a scalar or an $L \times 1$ If \texttt{vm\_scale} is a scalar, the data in all series are multiplied by the value. If \texttt{vm\_scale} is an $L \times 1$ vector, each series is multiplied by the corresponding element of \texttt{vm\_scale}. The default scale value is $4/\text{standard deviation of each series}$ (found to be best by experimentation).

3.8.2 Condition

A well-conditioned problem has a Hessian for which the columns are linearly independent and the diagonal elements are roughly the same size, i.e. the data are properly scaled. In this case, the condition number of the Hessian, the ratio of the largest eigenvalue to the smallest eigenvalue, is close to unity. The condition number will be large when the data are improperly scaled or the extent to which the Hessian columns exhibit linear dependencies. Users may examine the estimated Hessian. It’s in the \texttt{vm\_Hessian} global variable.

The \texttt{sqpsolve} solution is found by searching for parameter values for which the gradient is zero. However, \texttt{sqpsolve} has difficulty determining optimal values when the Jacobian of the gradient (i.e., the Hessian) is very small for a particular parameter. In this case, a large region of the function appears virtually flat to \texttt{sqpsolve}. When the Hessian has very small elements, the inverse of the Hessian has very large elements, and the search direction gets buried in the large numbers.

Poor condition can be caused by bad scaling, poor model specification, or bad data. Bad models and bad data are two sides of the same coin. If the problem is highly nonlinear, it is important that data are available to describe the curve over all relevant
regions. For example, one of the parameters of the Weibull function describes the shape of the curve as it approaches the upper asymptote. If data are not available for that portion of the curve, the corresponding parameter is poorly estimated. The gradient of the function with respect to the parameter is very flat; elements of the Hessian associated with that parameter are very small, and the inverse of the Hessian contains very large numbers. In this case, if the underlying behavioral theory allows it, the model should be respecified to exclude the parameter.

3.8.3 Starting Point

As noted in section 3.7.1, the \texttt{ecm} and \texttt{varmax} procedures fit univariate \textit{ARMA} models to generate starting values for each $Y$ variable in the model, unless the user supplies their own starting values in \texttt{_vm_start}. User-defined \texttt{_vm_start} values are required when the automatically generated starting values fail or when there are inequality constraints in the model. The latter case requires a starting point that satisfies the inequality constraints.

The starting point can be critical in finding a solution to a model that is not well-defined. Try different starting points when the optimization doesn’t seem to work. If the underlying behavioral theory allows it, a simpler problem with the same parameters might be specified. This could lead to a closed form solution. For example, ordinary least squares estimates might be used for nonlinear least squares problems or nonlinear regressions like probit or logit. There are no general methods for computing start values, and it may be necessary to attempt the estimation from a variety of starting points.

3.9 Diagnostic Checking

Identification and diagnostic checking go hand in hand. The earlier Identification section discussed the ACF, PACF, portmanteau, and information criteria from the \texttt{identify} and \texttt{sumstat} procedures.

3.10 Forecasting

The \texttt{vmforecast} procedure calculates $t$ step ahead forecasts for a \textit{VARMAX} model. Users must specify the coefficients involved, the dependent variable data set, residuals from the \texttt{varmax} estimation, the $AR$ and $MA$ orders, and the number of periods to forecast. A $t \times K$ matrix of fixed explanatory variables, covering only the forecast horizon, is also entered if \texttt{beta} coefficients were estimated.

\texttt{vmforecast} returns a $t \times (L + 1)$ matrix. The first column is the forecast horizon, i.e. the $t$ in $T + t$. Subsequent columns contain the forecast $Y$ values.
3. **VARMA**

### 3.11 References


Chapter 4

Error Correction Models

Error Correction Models are often used to estimate long-run and short-run relationships and to test for cointegration.

A stationary (means-removed) $VAR(p)$ model is written as:

$$
\phi(L)Y_t = Y_t - \sum_{j=1}^{p} \phi_j Y_{t-j} + \beta X_t = \epsilon_t
$$

(4.1)

where $Y_t$ is an $L$ dimensioned covariance stationary time series process, the $\epsilon_t$ are i.i.d. $N(0, \Omega_n)$, $\Omega_n$ is a positive definite matrix of order $L$, and $X_t$ is a $K \times 1$ vector of fixed explanatory variables, has the error correction form:

$$
\Delta Y_t = \Pi Y_{t-1} + \sum_{i=1}^{k} \Gamma_i \Delta Y_{t-i} + \beta X_t + \epsilon_t, \quad t = 1, ..., T
$$

(4.2)

where the $\Pi$ and $\Gamma$ matrices have dimension $L \times L$. The $\beta$ coefficients have dimension $L \times K$.

The `ecm` procedure estimates this model using FIML (exact, unconditional - Mauricio’s procedure, discussed in section 3.5). It has a number of returns, including:

- $A$ $L \times r$ matrix of coefficients, such that $AB = \Pi$
- $B$ $r \times L$ matrix, eigenvectors spanning the cointegrating space of dimension $r$
- $\nu$ $r \times 1$ vector, eigenvalues
- $Pi$ $L \times L$ matrix of cointegration coefficients

*Note that $\Pi$ is a reserved word in GAUSS. Users will need to assign this to a different variable name.*
4. ERROR CORRECTION MODELS

4.0.1 Cointegration Tests

Given the above ECM model, the degree of cointegration (the dimension of the cointegrating space) may be examined using Johansen’s likelihood ratio Trace and Maximum Eigenvalue statistics, returned by the \texttt{vmsj} procedure. If the \texttt{ecmflag} input argument equals one, two sets of Trace and Maximum Eigenvalue statistics are returned. The first set is based on Johansen’s estimation procedure, specifically on his method for calculating eigenvalues of the $\Pi$ matrix. The second set is based on the $\Pi$ eigenvalues returned from Full Information Maximum Likelihood estimation of the ECM model.

If $\Pi$ has full rank then all the variables in $Y_t$ are stationary. If $\Pi$ has less than full rank, say $r$, then $r$ of the variables are cointegrated. The Trace statistic tests the null hypothesis that the rank of $\Pi$ is less than or equal to $r$ versus the alternative that it is greater than $r$. The Maximum Eigenvalue statistic tests the null hypothesis that the rank of $\Pi$ is equal to $r$ versus the alternative that the rank of $\Pi$ is $r+1$. These statistics are given in Johansen (1995):

\[ \text{Trace} = -T \sum_{i=r+1}^{L} \ln(1 - \hat{\lambda}_i) \]
\[ \lambda_{\text{max}} = -T \ln(1 - \hat{\lambda}_{r+1}) \]

where $\hat{\lambda}_{r+1}, ..., \hat{\lambda}_L$ are the smallest $L - r$ eigenvalues of $S_{11}^{-1} S_{10} S_{00}^{-1} S_{01}$ and the $S_{ij}$ matrices represent sums of squares from two regressions, $\Delta Y_t$ on $\Delta Y_{t-1}, ..., \Delta Y_{t-p+1}$ (returning residuals $R_{0t}$) and $Y_{t-1}$ on $\Delta Y_{t-1}, ..., \Delta Y_{t-p+1}$ (returning residuals $R_{1t}$).

Asymptotic critical values for the Trace and Maximum Eigenvalue statistics, based on Johansen’s method of calculating eigenvalues and given that the correlations are estimated rather than observed, are returned by \texttt{vmc\_sja} and \texttt{vmc\_sjt}. The former returns Maximum Eigenvalue critical values and the latter returns Trace critical values.

4.0.2 Cointegration Coefficients and $\Pi$

Occasionally the $A \ast B$ calculation will not match the returned $\Pi$ matrix. This is because the eigenvalues close to zero are associated with eigenvectors not in the cointegrating space. $A \ast B$ will always equal $\Pi$ if $r$ equals zero, i.e. if all eigenvectors are in the cointegrating space. If $r$ equals one, $A \ast B$ will equal $\Pi$ only if the eigenvalue associated with the removed eigenvector is zero. If the eigenvalue is close to zero, $A \ast B$ will almost equal $\Pi$. If the eigenvalue is not close to zero, $A \ast B$ will be quite different from $\Pi$.

4.1 Printing ECM Results

Three global variables, \texttt{~output}, \texttt{~vm\_output}, and \texttt{~vm\_PrintIters} determine the output that is displayed from the \texttt{ecm}, \texttt{varmax}, \texttt{sqpsolve}, and subordinate procedures.
4. ERROR CORRECTION MODELS

1. Set output = 0 to suppress all printing from the sqpsolve procedure.

2. Set _vm_PrintIters = 0 (output is not equal to zero) to print an
   Executing... message while starting values are calculated for each
   dependent variable during the sqpsolve operation.

3. Set _vm_PrintIters > 0 (output is not equal to zero) to print sqpsolve
   iteration information. This information includes the value of the objective
   function and the gradient at each estimated coefficient. It is useful in finding
   where and why convergence might fail.

4. Set output > 0 to print sqpsolve results.

_vm_output is either a scalar or a 6 × 1 vector. Set _vm_output = 0 to suppress all
printing from the ecm and varmax estimations. Set _vm_output > 0 to print all ecm
and varmax output. Define _vm_output as a 6 × 1 vector to control the printing of
various parts of ecm and varmax output.

1. Set element [1] of _vm_output to a non-zero value to print the model’s
   header

2. Set element [2] of _vm_output to a non-zero value to print a variety of
   unitroot tests, and, if a multivariate model, cointegration tests from the
   unitroots procedure.

   statistics for each estimated equation

4. Set element [4] of _vm_output to a non-zero value to print the estimated
   coefficients and their standard errors

5. Set element [5] of _vm_output to a non-zero value to print the roots of the
   AR and MA characteristic equations

6. Set element [6] of _vm_output to a non-zero value to print the
   autocorrelation function and portmanteau statistics

4.2 References

Johansen, S.J. and Juselius, K. “Maximum Likelihood Estimation and Inference on
Cointegration, with Applications for the Demand for Money,” Oxford Bulletin of

the Maximum Likelihood Cointegration Rank Test Statistics: Four Cases,” Oxford
4. ERROR CORRECTION MODELS
Chapter 5

Panel Data

The *TIME SERIES* module includes procedures for the computation of estimates for the “pooled times-series cross-section” (TSCS) regression model.

### 5.1 Introduction

In order to use these procedures the *tscs* library must be active. This is done by including *tscs* in the *library* statement at the top of your program:

```gauss
library tscs, quantal, pgraph;
```

This enables *GAUSS* to find the *TSCS* procedures. If you plan to make any right-hand references to the global variables (described under the *tscs* function definition in chapter 8), you will also need the statement:

```gauss
#include tscs.ext;
```

Finally, to reset global variables in succeeding executions of the program the following instruction can be used:

```gauss
tscsset;
```

This could be included with the earlier statements without harm and would insure the proper definition of the global variables for all executions of the program.
5.2 Pooled Time-Series Cross-Section Regression Model

This program provides procedures to compute estimates for “pooled time-series cross-sectional” models. The assumption is that there are multiple observations over time on a set of cross-sectional units (e.g., people, firms, countries). For example, the analyst may have data for a cross-section of individuals each measured over 10 time periods. While these models were devised to study a cross-section of units over multiple time periods, they also correspond to models in which there are data for groups such as schools or firms with measurements on multiple observations within the groups (e.g., students, teachers, employees).

The specific model that can be estimated with this program is a regression model with variable intercepts, i.e., a model with individual-specific effects. The regression parameters for the exogenous variables are assumed to be constant across cross-sectional units. The intercept varies across individuals.

This program provides three estimators:

- the fixed-effects OLS estimator (analysis of covariance estimator),
- the constrained OLS estimator (individual-specific effects are excluded from the equation) and
- the random effects estimator using GLS.

A Hausman test is computed to show whether the error components (random effects) model is the correct specification.

In addition to providing the analysis of covariance and GLS estimates, two multiple partial-squared correlations are computed. The first partial correlation (squared correlation) shows the percentage of variation in the dependent variable that can be explained by the set of independent variables while holding constant the group variable. The second estimate shows the extent to which variation in the dependent variable can be accounted for by the group variable after the other independent variables have been statistically held constant.

A feature of this program is that it allows for a variable number of time-series observations per cross-sectional unit. For instance, there might be 5 time-series observations for the first individual, 10 for the second, and so on. This is useful, for example, if there are missing values.

5.3 References


Chapter 6

ARIMA

The TIME SERIES module includes procedures for the computation of estimates and forecasts for the autoregressive integrated moving average model. The model may include fixed regressors such as linear or quadratic time trends, or other explanatory variables which are predetermined. Forecasts are computed using the estimated parameters and errors.

6.1 Introduction

In order to use these procedures the arima library must be active. This is done by including arima in the library statement at the top of your program:

```
library arima.pgraph;
```

This enables GAUSS to find the arima procedures.

Finally, to reset global variables in succeeding executions of the program the following instruction can be used:

```
arimaset;
```

This could be included with the earlier statements without harm and would insure the proper definition of the global variables for all executions of the program.
6.2 ARIMA Models

This program will compute estimates of the parameters and standard errors for a time series model with ARMA errors. If the model contains only autoregressive parameters, then arima gives the same estimates as autoreg. arima reports standard errors, parameters estimates, model selection criteria, roots of the parameters, the Ljung-Box portmanteau statistic and the covariance and correlation matrices.

The model estimated is of the general form:

\[ \phi(L) [(1 - L)^d y_t - x_t \beta] = \theta(L) \epsilon_t \]

where

\[ L^j y_t = y_{t-j} \]
\[ \phi(L) = 1 - \phi_1 L - \phi_2 L^2 - \cdots - \phi_p L^p \]
\[ \theta(L) = 1 - \theta_1 L - \theta_2 L^2 - \cdots - \theta_q L^q \]

where it is assumed that \( \epsilon_t \) is a white noise error term, distributed as \( N(0, \sigma^2) \). Such models are referred to as arima(\( p, d, q \)), where \( p \) is the autoregressive order, \( d \) is the difference order and \( q \) is the moving average order.

The parameters to be estimated are thus: \( \phi \ (P \times 1) \), \( \theta \ (Q \times 1) \), \( \beta \ (M \times 1) \) and \( \sigma^2 \ (1 \times 1) \).

The arima procedure computes starting values or allows the user to specify starting values. User specified starting values are useful when the user wants to determine whether the parameters estimates computed by arima correspond to the global maximum of the log likelihood function and not just a local maximum. Finally, the tsforecast procedure computes forecasts for the series \( h \) steps ahead using the estimated parameters and errors returned by the arima procedure.

6.3 References


Chapter 7

Autoregression

The *TIME SERIES* module includes procedures for the computation of estimates for the autoregression model with autoregressive errors of any specified order, and the computation of autocorrelations and autocovariances.

7.1 Introduction

In order to use these procedures the *autoreg* library must be active. This is done by including `auto` in the `library` statement at the top of your program:

```
library auto,quantal,pgraph;
```

This enables *GAUSS* to find the *autoreg* procedures. If you plan to make any right-hand references to the global variables (described under the *autoreg* function definition in chapter 8), you will also need the statement:

```
#include auto.ext;
```

Finally, to reset global variables in succeeding executions of the program the following instruction can be used:

```
autoset;
```

This could be included with the earlier statements without harm and would insure the proper definition of the global variables for all executions of the program.
7. AUTOREGRESSION

7.2 Autoregression Models

This program will compute estimates of the parameters and standard errors for a regression model with autoregressive errors. Thus, it can be used for models for which the Cochrane-Orcutt or similar procedure can be used. It is also similar to the SAS autoreg procedure except that this routine will compute the maximum likelihood estimates based upon the exact likelihood function.

The model estimated is of the general form:

\[ y_t = x_t \beta + u_t \]
\[ u_t - \phi_1 u_{t-1} - \ldots - \phi_p u_{t-p} = e_t \]

where it is assumed that \( e_t \) is a white noise error term, distributed as \( N(0, \sigma^2) \).

The parameters to be estimated are thus: \( \beta \) (\( K \times 1 \)), \( \phi \) (\( L \times 1 \)) and \( \sigma^2 \) (a scalar). The order of the process is \( L \).

In addition, this program will estimate the autocovariances and autocorrelations of the error term \( u \). It produces initial estimates of these based upon the residuals of an OLS regression. Then it computes the maximum likelihood estimates of these based upon the maximum likelihood estimates of the other parameters.

7.3 References

Chapter 8

Command Reference
acf

- **Library**
  arima

- **Purpose**
  Computes sample autocorrelations for a univariate time series.

- **Format**
  
  \[ a = \text{acf}(x,l,d); \]

- **Input**
  
  \[ x \quad N \times 1 \text{ vector. The mean is subtracted automatically.} \]

  \[ l \quad \text{scalar, the maximum lags to compute.} \]

  \[ d \quad \text{scalar, the difference order.} \]

- **Output**
  
  \[ a \quad l \times 1 \text{ vector, sample autocorrelations.} \]

- **Remarks**
  This function is similar to autocor, however, acf allows the users to compute the autocorrelations for the differenced data.

- **Source**
  tsutil.src
8. COMMAND REFERENCE

■ Library

arima

■ Purpose

Estimates coefficients of a univariate time series model with autoregressive-moving average errors. Model may include fixed regressors.

■ Format

\{ \text{coefs,ll,e,vcb,aic,sbc} \} = \text{arima}(\text{startv},y,p,d,q,\text{const})

■ Input

\begin{itemize}
  \item \textit{startv} \hspace{1cm} \text{scalar, 0, then arima computes starting values.}\n    \hspace{1cm} \text{– or –}\n    \hspace{1cm} K \times 1 \text{ vector, starting values.}\n  \item \textit{y} \hspace{1cm} \text{N} \times 1 \text{ vector, data.}\n  \item \textit{p} \hspace{1cm} \text{scalar, the autoregressive order.}\n  \item \textit{d} \hspace{1cm} \text{scalar, the order of differencing.}\n  \item \textit{q} \hspace{1cm} \text{scalar, the moving average order.}\n  \item \textit{const} \hspace{1cm} \text{scalar, if 1, a constant is estimated, 0 otherwise.}\n    \hspace{1cm} \text{– or –}\n    \hspace{1cm} N \times 1 \text{ matrix, fixed regressors.}\n\end{itemize}

The number of rows in the fixed regressor matrix must be equal the number of rows for \textit{y} after differencing.

■ Output

\begin{itemize}
  \item \textit{coefs} \hspace{1cm} K \times 1 \text{ vector, estimated model coefficients.}\n  \item \textit{ll} \hspace{1cm} \text{scalar, the value of the log likelihood function.}\n  \item \textit{e} \hspace{1cm} \text{N} \times 1 \text{ vector, residual from fitted model.}\n  \item \textit{vcb} \hspace{1cm} K \times K \text{ matrix, the covariance matrix of estimated model coefficients.}\n  \item \textit{aic} \hspace{1cm} \text{scalar, value of the Akaike information criterion.}\n  \item \textit{sbc} \hspace{1cm} \text{scalar, value of the Schwartz Bayesian criterion.}\n\end{itemize}
arima

8. COMMAND REFERENCE

 Globals

__am__itol 3x1 vector, controls the convergence criterion.

   Default = 1e-8.
   = 1e-6.

__output__ scalar, controls printing of output

   0 Nothing will be printed by arima.
   1 Final results are printed.
   2 Final results, iterations results, residual autocorrelations,
      Box-Ljung statistic, and Covariance and correlation matrices
      are printed,

__am__varn 1x(M+1) vector of parameter names. This is used for models with fixed
regressors. The first element contains the name of the independent
variable; the second through M\(^{th}\) elements contain the variable names for
the fixed regressors. If __am__varn = 0, the fixed regressors labeled as
\(X_0, X_1, \ldots, X_M\). Default __am__varn = 0.

 Remarks

There are other global variables which are used by arima’s likelihood function. These
are __am__b, __am__y, __am__p, __am__d, __am__q, __am__const, __am__n, __am__e,
__am__k, __am__m, __am__inter.

This program will only handle data sets that fit in memory.

All autoregressive and moving average parameters are estimated up to the specified lag.
You cannot estimate only the first and fourth lag, for instance.

arima forces the autoregressive coefficients to be invertible (in other words, the
autoregressive roots have modulus greater than one). The moving average roots will
have modulus one or greater. If a moving average root is one, arima reports a missing
value for the moving average coefficient’s standard deviation, t-statistic and p-value.
This is because these values are meaningless when one of the moving average roots is
equal to one. A moving average root equal to one suggests that the data may have been
over-differenced.

 Source

arima.src

42
Library

arima

Purpose

Initializes arima global values to default values.

Format

arimaset;

Input

None

Output

None

Remarks

Putting this instruction at the top of all programs that invoke arima is generally good practice. This will prevent globals from being inappropriately defined when a program is run either several times or after another program that also call arima.

Source

arima.src
armanames

- **Library**
  varma

- **Purpose**
  Returns the names of the AR, MA and x variables.

- **Format**
  
  \[ names = \text{armanames}(p,q,y,x,xnames,ecmflag); \]

- **Input**
  
  - \( p \) scalar, AR order
  - \( y \) \( T \times L \) matrix, dependent variables
  - \( x \) \( T \times K \) matrix, independent variables. Enter 0 if there is no \( x \) matrix.
  - \( xnames \) \( K \times 1 \) character vector, names of variables in \( x \) or 0. Enter 0 if there are no \( xnames \) given
  - \( ecmflag \) scalar, Enter 1 if an ecm model is estimated. Enter 0 otherwise.

- **Output**
  
  \[ names \ (p + q + K) \times 1 \) character vector, names of ARMA and x variables

- **Remarks**
  
  `armanames` returns AR and MA variable names in the order \( Phi - 1, Phi - 2,..., Phi - p, Theta - 1,..., Theta - q \), concatenated onto the \( x \) variable names (either as given by the user or \( X1,...,XK \)). Only the ARMA terms are returned if there are no \( x \) variables.

- **Source**
  
  varma.src
8. COMMAND REFERENCE

- **Library**
  
  auto

- **Purpose**
  
  Computes specified autocorrelations for each column of a matrix.

- **Format**
  
  \[ a = \text{autocor}(x,f,l); \]

- **Input**
  
  \( x \): \( N \times K \) matrix. Autocorrelations will be computed for each column separately. \( x \) is assumed to have 0 mean.

  \( f \): scalar, in range \([0, \text{rows}(x)-1]\), denoting the first autocorrelation to compute.

  \( l \): scalar, in range \([0, \text{rows}(x)-1]\), denoting the last autocorrelation to compute. It must be that \( f \leq l \); if \( l = 0 \) and \( f = 0 \), then \( l \) is set to \( \text{rows}(x)-1 \) and all autocorrelations from \( f \) to \( l \) are computed. If \( l = 0 \) and \( f < 0 \), then only the 0\(^{th}\) order autocorrelation is computed (this equals \( x'x \)).

- **Output**
  
  \( c \): \( G \times K \) matrix, where \( G = l - f + 1 \), containing the autocorrelations of order \( f, f+1, ..., l \) for each of the columns of \( x \). If the variance of any variable is 0, missings will be returned for that variable.

- **Remarks**
  
  The 0\(^{th}\) autocorrelation will always be 1.

  The data are assumed to have 0 mean. Thus, use

  \[ x = x - \text{meanc}(x)'; \]

  prior to the use of this function if the mean is not 0.

- **Source**
  
  autoreg.src
autocov

■ Library
auto

■ Purpose
Computes specified autocovariances for each column of a matrix.

■ Format
\[ a = \text{autocov}(x,f,l); \]

■ Input
\[ x \quad N \times K \text{ matrix. Autocovariances will be computed for each column separately. } x \text{ is assumed to have 0 mean.} \]
\[ f \quad \text{scalar, in range } [0, \text{rows}(x)−1], \text{ denoting the first autocovariance to compute.} \]
\[ l \quad \text{scalar, in range } [0, \text{rows}(x)−1], \text{ denoting the last autocovariance to compute. It must be that } f \leq l; \text{ if } l = 0 \text{ and } f = 0, \text{ then } l \text{ is set to } \text{rows}(x)−1 \text{ and all autocovariances are computed. If } l = 0 \text{ and } f < 0, \text{ then only the 0th order autocovariance is computed (this equals } x'x). \]

■ Output
\[ a \quad \text{GxK matrix, where } G = l − f + 1, \text{ containing the autocovariances of order } f, f+1, ..., l \text{ for each of the columns of } x. \]

■ Remarks
The 0th autocovariance is just the variance of the variable. The divisor for each autocovariance is the number of elements involved in its computation. Thus, the \( p^{th} \) order cross product is divided by \( N - P \), where \( N = \text{rows}(x) \), to obtain the \( p^{th} \) order autocovariance.

The data are assumed to have 0 mean. Thus, use
\[ x = x - \text{meanc}(x)'; \]

prior to the use of this function if mean is not 0.

■ Source
\text{autoreg.src}
8. COMMAND REFERENCE

■ Library

auto

■ Purpose

Estimates coefficients of a regression model with autoregressive errors of any specified order.

■ Format

\[
\{ \text{coefs, vcb, phi, vcphi, sigsq, acov, acor} \} = \text{autoreg}(\text{dataset, depvar, indvars, lagvars, order})
\]

■ Input

\textit{dataset} \hspace{1em} \text{string, name of GAUSS data set.} \\
\hspace{1em} \text{or} \\
\hspace{1em} N \times K \text{ matrix, data}

\textit{depvar} \hspace{1em} \text{string, the name of the dependent variable} \\
\hspace{1em} \text{or} \\
\hspace{1em} \text{scalar, the index of the dependent variable.}

If \textit{dataset} is a matrix and if variable names have been provided using \____\text{altnam}, then \textit{depvar} may be a string or character variable containing a variable label.

\textit{indvars} \hspace{1em} K \times 1 \text{ character vector, names of the independent variables} \\
\hspace{1em} \text{or} \\
\hspace{1em} K \times 1 \text{ numeric vector, indices of the independent variables.}

\textit{indvars} can include repeated entries of the independent variables and the dependent variable as long as the corresponding entries in \textit{lagvars} are lagged uniquely.

If \textit{dataset} is a matrix and if variable names have been provided using \____\text{altnam}, then \textit{indvars} may be a character vector containing variable labels.

\textit{lagvars} \hspace{1em} K \times 1 \text{ vector, the number of periods to lag the variables in \textit{indvars}. If there are no lagged variables, set to scalar 0.}

The variables in \textit{indvars} will be lagged the number of periods indicated in the corresponding entries in \textit{lagvars}. \textit{indvars} may contain the dependent variable in one of its columns as long as the corresponding entry in \textit{lagvars} is not 0; also, the independent variables can be repeated if the corresponding entries in \textit{lagvars} are unique.
autoreg

order scalar, order of the autoregressive process; must be greater than 0 and less than the number of observations.

Output

coefs $K \times 1$ vector, estimated regression coefficients

$vcb$ $K \times K$ matrix, covariance matrix of estimated regression coefficients

$phi$ $L \times 1$ vector, lag coefficients

$vcphi$ $L \times L$ matrix, covariance matrix of $phi$

$sigsq$ scalar, variance of white noise error

$acov$ $(L+1) \times 1$ vector, autocovariances

$acor$ $(L+1) \times 1$ vector, autocorrelations

Globals

__arinit scalar. If 1, only initial estimates will be computed. Default = 0.

__ariter scalar.

0 Nothing will be printed by autoreg.

1 Results will be printed at end of iterations.

2 Results will be printed at all iterations.

Default = 2.

__altnam $K \times 1$ vector, alternate names for variables when a matrix is passed to autoreg. These names will be used in place of the names set by autoreg (X1, X2, ...). When a data matrix is passed to autoreg and the user is selecting from that matrix, __altnam, if used, must contain names for the original matrix.

__con scalar integer. If 1, constant will be used in model. Default = 1.

__header string, specifies the format for the output header. __header can contain zero or more of the following characters:

- t print title (see __title)
- l bracket title with lines
- d print date and time
- v print procedure name and version number
- f print file name being analyzed

Example:
8. COMMAND REFERENCE

```plaintext
autoreg

__header = "tld";
If __header == "", no header is printed. Default = "tldvf".

__output scalar, if nonzero, results are printed to screen. Under UNIX, default = 1; under DOS, default = 2.

__row scalar. Specifies how many rows of the data set will be read per iteration of the read loop. By default, the number of rows to be read will be calculated by autoreg.

__rowfac scalar, "row factor". If autoreg fails due to insufficient memory while attempting to read a GAUSS data set, __rowfac may be set to some value between 0 and 1 to read a proportion of the original number of rows of the GAUSS data set. For example, setting

    __rowfac = 0.8;
causes GAUSS to read in 80% of the rows of the GAUSS data set that were read when the failure due to insufficient memory occurred. Default = 1.

__rowfac has an effect only when __row = 0.
Default = 1.

__title string, a title to be printed at the top of the output header (see __header). By default, no title is printed (__title = "").

__tol scalar, convergence tolerance. Default = 1e−5.

__vpad scalar. If dataset is a matrix in memory, the variable names are automatically created by autoreg. Two types of names can be created:

    0 Variable names are not padded to give them equal length. For example, X1, X2 ... X10, X11, ....
    1 Variable names are padded with zeros to give them an equal number of characters. For example, X01, X02 ... X10, X11, ....
This is useful if you want the variable names to sort properly.
Default = 1.

Global Output

__arvsig scalar, variance of sigsq (variance of the variance of white noise error).
__archisq scalar, −2 * log-likelihood.
__artobs scalar, number of observations.
```
Remarks

This program will only handle data sets that fit in memory.

All autoregressive parameters are estimated up to the specified lag. You cannot estimate only the first and fourth lags, for instance.

The algorithm will fail if the model is not stationary at the estimated parameters. Thus, in that sense it automatically tests for stationarity.

Source

autoreg.src
8. COMMAND REFERENCE

**Library**

auto

**Purpose**

Initializes autoreg global variables to default values.

**Format**

autoset;

**Input**

None

**Output**

None

**Remarks**

Putting this instruction at the top of all programs that invoke autoreg is generally good practice. This will prevent globals from being inappropriately defined when a program is run either several times or after another program that also calls autoreg.

autoset calls gausset.

**Source**

autoreg.src
coefficient

8. COMMAND REFERENCE

- **Library**
  - varma

- **Purpose**
  - Print the coefficient estimates and standard errors from an ecm or varmax call.

- **Format**
  - coeffprt(coeffs, x, xnames, ynames, p, q, ecmflag);

- **Input**
  - **coeffs** compact matrix created using vput. Read it using vread. It contains:
    - **phi** \( p \times (L \times L) \) matrix of AR coefficient estimates stacked in the order \( AR(1), \ldots, AR(p) \)
    - **phi_se** \( p \times (L \times L) \) matrix of AR standard errors stacked in the order \( AR(1), \ldots, AR(p) \)
    - **theta** \( q \times (L \times L) \) matrix of MA coefficient estimates stacked in the order \( MA(1), \ldots, MA(q) \)
    - **theta_se** \( q \times (L \times L) \) matrix of MA standard errors stacked in the order \( MA(1), \ldots, MA(q) \)
    - **beta** \( L \times K \) matrix of \( x \) coefficient estimates
    - **beta_se** \( L \times K \) matrix of \( x \) coefficient standard errors stacked in the order \( MA(1), \ldots, MA(q) \)
  - **x** \( T \times K \) matrix of independent variables or scalar, equals zero if there are no independent variables
  - **xnames** \( K \times 1 \) vector of names for the \( x \) matrix variables
  - **ynames** \( L \times 1 \) vector of names for the dependent variables in \( y \)
  - **p** scalar, order of the AR process
  - **q** scalar, order of the MA process
  - **ecmflag** scalar, 1 if an ecm model was estimated, 0 otherwise

- **Output**
  - None
8. COMMAND REFERENCE

## Globals

**coeffprt**

___output, __vm_output

__vm_output is either a scalar or a $6 \times 1$ vector. Set __vm_output = 0 to suppress all printing from the ecm and varmax estimations. Set __vm_output > 0 to print all ecm and varmax output.

Define __vm_output as a $6 \times 1$ vector to control the printing of various parts of ecm and varmax output.

1. Set element [1] of __vm_output to a non-zero value to print the model’s header

2. Set element [4] of __vm_output to a non-zero value to print the estimated coefficients and their standard errors
corm

- **Library**
  varma

- **Purpose**
  Prints, with labels, the correlation matrix of parameters for univariate models.

- **Format**
  
  `corm(covb, names);`

- **Input**
  
  - `covb` \( K \times K \) matrix, covariances of estimated parameters
  - `names` \( K \times 1 \) character vector, names of the parameters

- **Output**
  
  none

- **Globals**
  
  none

- **Source**
  
  varma.src
8. COMMAND REFERENCE

**Library**

varma

**Purpose**

Prints, with labels, the covariance matrix of parameters for univariate models.

**Format**

\texttt{covm(covb,names);} 

**Input**

- \texttt{covb} $K \times K$ matrix, covariances of estimated parameters 
- \texttt{names} $K \times 1$ character vector, names of the parameters

**Output**

none

**Globals**

none

**Source**

varma.src
ecm

Library
varma

Purpose
Calculates and returns parameter estimates for an error correction model.

Format
\[
\{ A, B, va, coeffs, vc, covb, res, statret, retc \} = \\
ecm(dataset, depvars, indvars, ynames, xnames, p, r, vctype)
\]

Input

dataset string, name of dataset.
If this is a null string, , the procedure assumes that the actual data has been passed in the depvar and indvars arguments.
depvars dependent variables.

1. If dataset is a null string, “”, this is interpreted as a $T \times L$ matrix, the dependent variables.
2. If dataset contains the name of a dataset, this is interpreted as:
   $L \times 1$ character vector, names of dependent variables in the dataset
   
   or

   $L \times 1$ numeric vector, indices of dependent variables in the dataset.
   These can be any subset of the variables in the dataset, and can be in any order.

NOTE: Each column must be ordered as $y_1, y_2, ..., y_T$. The top row is the first observation and the last row is the most current observation.

indvars independent variables.

1. Enter a 0 if there are no independent variables.
2. If dataset is a null string, , this is interpreted as a $T \times K$ matrix, the independent variables
3. If dataset contains the name of a dataset, this is interpreted as:
   $K \times 1$ character vector, names of independent variables in the dataset.
8. COMMAND REFERENCE

ecm

– or –

$K \times 1$ numeric vector, indices of independent variables in the dataset.
These can be any subset of the variables in the dataset, and can be in any order.

NOTE: Each column must be ordered as $x_1, x_2, ..., x_T$. The top row is the first observation and the last row is the most current observation.

ynames $L \times 1$ character matrix of names for the variables in depvars, or scalar 0. GAUSS will supply variable names $Y_1, ..., Y_L$ if 0 is entered.

p scalar, order of AR process.

r scalar, number of cointegrating relations. Set to -1 to have GAUSS estimate this value.

vcType scalar, set to 1 for ML covariance matrix of parameters set to 2 for QML covariance matrix of parameters

■ Output

$A$ $L \times r$ matrix of coefficients, such that $AB = \Pi$ (see remarks below)

$B$ $r \times L$ matrix, eigenvectors spanning the cointegrating space of dimension $r$

va $r \times 1$ vector, eigenvalues

coeffs compact matrix created using vput. Read it using vread. It contains:

Pi $L \times L$ matrix of cointegration coefficients
Note that $\Pi$ is a reserved word in GAUSS. Users will need to assign this to a different variable name.

Pi.se $L \times L$ matrix of corresponding standard errors

phi $p \times (L \times L)$ matrix of AR coefficient estimates stacked in the order $AR(1), ..., AR(p)$

phi.se $p \times (L \times L)$ matrix of AR standard errors stacked in the order $AR(1), ..., AR(p)$

vc $L \times L$ matrix, covariance matrix of residuals

covb $Q \times Q$ matrix of estimated parameters where $Q$ is the number of estimated parameters. The parameters are in the row-major order: $\Pi$, $AR(1)$ to $AR(p)$, beta (if $x$ variables were present in the estimation), and the constants.
 commod

res  $T \times L$ matrix, residuals

statret compact matrix created using \texttt{vput}. Read it using \texttt{vread}. It contains:

- \texttt{ss} $L \times 2$ matrix, the sum of squares for $Y$ in column one and the sum of squared error in column two
- \texttt{info} $L \times 4$ matrix
  - \texttt{row one} - the likelihood value
  - \texttt{row two} - the Akaike Information Criterion
  - \texttt{row three} - the Schwarz Bayesian Information Criterion
  - \texttt{row four} - the Likelihood Ratio Statistic
- \texttt{arroots} $p \times 1$ vector of AR roots, possibly complex
- \texttt{maroots} $q \times 1$ vector of MA roots, possibly complex
- \texttt{acfm} $L \times (p \times L)$ matrix, the autocorrelation function.
  The first $L$ columns are the lag 1 ACF, The last $L$ columns are the lag $p$ ACF.
- \texttt{pacfm} $L \times (p \times L)$ matrix, the partial autocorrelation function, only computed if a univariate model is estimated. The first $L$ columns are the lag 1 ACF, The last $L$ columns are the lag $p$ ACF.
- \texttt{portman} $pv\_lags - (p + q) \times 3$ matrix of portmanteau statistics for the multivariate model and Ljung-Box statistics for the univariate model. The time period is in column one, the $Q_s$ (portmanteau) statistic in column two and the p-value in column three

retn  $2 \times 1$ vector, return code

- first element of \texttt{retn}
  - 0 normal convergence
  - 1 forced exit
  - 2 maximum number of iterations exceeded
  - 3 function calculation failed
  - 4 gradient calculation failed
  - 5 Hessian calculation failed
  - 6 line search failed
  - 7 error with constraints

- second element of \texttt{retn}
  - 0 covariance matrix of parameters fails
  - 1 ML covariance matrix computed
2 QML covariance matrix
3 Cross-Product covariance matrix

- **Globals**

  - `_vmcritl`  scalar, the significance levels defining p-values
  - `_vm_DirTol`  scalar, convergence tolerance for gradient of estimated coefficients. Default = 1e-5. When this criterion has been satisfied SQPSolve (the iteration procedure beneath `varma` and `ecm`) will exit the iterations.
  - `_vm_FeasibleTest`  scalar, if nonzero, parameters are tested for feasibility before computing function in line search. If function is defined outside inequality boundaries, then this test can be turned off.
  - `_vm_IndEquations`  \(K \times L\) matrix of zeros and ones. Used to set zero restrictions on the \(x\) variables to be estimated. Only used if the number of equations, \(_vm_L\) is greater than one. Elements set to one indicate the coefficients to be estimated. If \(_vm_L = 1\), all coefficients will be estimated. If \(_vm_L > 1\) and \(_vm_IndEquations\) is set to a missing value (the default), all coefficients will be estimated.
  - `_vm_Lagrange`  compact matrix created using `vput`. Contains the Lagrangean coefficients for the constraints. They may be extracted with the `vread` procedure using the following strings:
    
    | String   | Description                  |
    |----------|------------------------------|
    | "lineq"  | linear equality constraints  |
    | "nlineq" | nonlinear equality constraints|
    | "linineq"| linear inequality constraints |
    | "nlinineq"| nonlinear inequality constraints |
    | "bounds" | boundaries                    |

  When an inequality or bounds constraint is active, its associated Lagrangean is nonzero. The linear Lagrangeans precede the nonlinear Lagrangeans in the covariance matrices.

  - `_vm_lags`  scalar, No. of lags over which ACF and Diagnostics are calculated.
  - `_vm_MaxIters`  scalar, maximum number of iterations. Default = 1e+5. Termination can be forced by pressing C on the keyboard
  - `_vm_Output`  scalar or a \(6 \times 1\) matrix. Set \(_vm_output = 0\) to suppress all printing from the `ecm` and `varmax` estimations. Set \(_vm_output > 0\) to print all `ecm` and `varmax` output.
    
    Set \(_vm_output\) to a \(6 \times 1\) vector to control the printing of various parts of `ecm` and `varmax` output.
1. Set element \([1] \) of \( \text{vm\_output} \) to a non-zero value to print the model’s header
2. Set element \([2] \) of \( \text{vm\_output} \) to a non-zero value to print a variety of unitroot tests, and, if a multivariate model, cointegration tests from the unitroots procedure.
3. Set element \([3] \) of \( \text{vm\_output} \) to a non-zero value to print summary statistics for each estimated equation
4. Set element \([4] \) of \( \text{vm\_output} \) to a non-zero value to print the estimated coefficients and their standard errors
5. Set element \([5] \) of \( \text{vm\_output} \) to a non-zero value to print the roots of the AR and MA characteristic equations
6. Set element \([6] \) of \( \text{vm\_output} \) to a non-zero value to print the autocorrelation function and portmanteau statistics

\( \text{vm\_PrintIters} \)

1. Set \( \text{vm\_PrintIters} = 0 \) (\( \text{output} \) is not equal to zero) to print an \texttt{Executing...} message while starting values are calculated for each dependent variable during the \texttt{sqpsolve} operation.
2. Set \( \text{vm\_PrintIters} > 0 \) (\( \text{output} \) is not equal to zero) to print \texttt{sqpsolve} iteration information. This information includes the value of the objective function and the gradient at each estimated coefficient. It is useful in finding where and why convergence might fail.

\( \text{vm\_RandRadius} \)

Scalar, if nonzero gives the radius of random search taken when the \texttt{STEPBT} line search fails. If zero, no random search occurs and SQPSolve returns with an error code. Default = .01.

\( \text{vm\_scale} \)

Scalar, scalar or an Lx1 vector, scales the time series. If scalar, all series are multiplied by the value. If an Lx1 vector, each series is multiplied by the corresponding element of \( \text{vm\_scale} \). Default = 4 / standard deviation (found to be best by experimentation).

\( \text{vm\_SetConstraints} \)

Scalar, set to a nonzero value to impose stationarity and invertibility by constraining roots of the AR and MA characteristic equations to be outside the unit circle. Set to zero (the default) to estimate an unconstrained model.

\( \text{vm\_Start} \)

\((Q - L) \times 1\) vector of starting values, in the row major order, \(AR(1)\) to \(AR(p)\), \(MA(1)\) to \(MA(q)\), \(beta\), and the covariance matrix of these parameters.

\( \text{vm\_TrustRadius} \)

Scalar, gives the radius of the trust region if nonzero, i.e., the maximum amount in absolute value for the direction vector at each iteration. If zero, the trust region method inactivated.
8. COMMAND REFERENCE

- Remarks
  Errors are assumed to be distributed N(0,Q).

- Source
  varma.src
identify

Library
varma

Purpose
Computes and prints ACF and PACF functions and portmanteau test statistics.

Format
\{ acfm, pacfm, Qs \} = identify(p, q, res);

Input
p \quad \text{AR order}
q \quad \text{MA order}
res \quad T \times L \text{ matrix of residuals}

Output
acfm \quad (_\text{vm\_lags}*L) \times 1 \text{ vector of univariate scalar autocorrelations, in order from lag(1) to lag(_\text{vm\_lags})}

\quad \text{– or –}
\quad L \times (_\text{vm\_lags}*L) \text{ matrix, the autocorrelation function matrices in order from lag(1) to lag(_\text{vm\_lags})}

pacfm \quad _\text{vm\_lags} \times 1 \text{ vector of univariate scalar partial autocorrelations, in order from lag(1) to lag(_\text{vm\_lags})}

Qs \quad _\text{vm\_lags} \times 1 \text{ vector of Ljung-Box statistics, in order from lag(1) to lag(_\text{vm\_lags}), for the univariate case}

\quad \text{– or –}
\quad _\text{vm\_lags} \times 1 \text{ vector of } Qs \text{ Portmanteau statistics (see remarks below) in order from lag(1) to lag(_\text{vm\_lags}), for the multivariate case}

Globals

_\text{vm\_output} \text{ is either a scalar or a } 6 \times 1 \text{ vector. Set } _\text{vm\_output} = 0 \text{ to suppress all printing from the ecm and varmax estimations. Set } _\text{vm\_output} > 0 \text{ to print all ecm and varmax output.}

Define _\text{vm\_output} \text{ as a } 6 \times 1 \text{ vector to control the printing of various parts of ecm and varmax output.}
8. **COMMAND REFERENCE**

1. Set element [1] of `vm_output` to a non-zero value to print the model’s header
2. Set element [6] of `vm_output` to a non-zero value to print the autocorrelation function and portmanteau statistics

`vm_lags` The `vm_lags` global variable sets the number of lags over which ACF, PACF, and Portmanteau statistics are calculated. The default value for `vm_lags` is 12.

**Remarks**

The $Q_s$ multivariate portmanteau statistic is described in Reinsel (1993, p 133).

$$Q_s = T^2 \sum_{l=1}^{s} (T - l)^{-1} \sum_{i=1}^{k} \sum_{j=1}^{k} C_{ij}(l) r_{ji}(-l)$$

$$= T^2 \sum_{l=1}^{s} (T - l)^{-1} tr \{C_{z}(l) \hat{\Sigma}^{-1} C_{z}(-l) \hat{\Sigma}^{-1} \}$$

$$= T^2 \sum_{l=1}^{s} (T - l)^{-1} tr \{\hat{\rho}_{z}(l) \hat{\rho}_{z}(0)^{-1} \hat{\rho}_{z}(-l) \hat{\rho}_{z}(0)^{-1} \}$$

Under the null hypothesis:

$H_0$ : $Y_t$ is an ARMA($p$, $q$) process

$H_1$ : not $H_0$

and assuming that $s$ is large, the $Q_s$ statistic has approximately a $\chi^2(L^2(s - p - q))$ distribution.

**Source**

`varma.src`
**macf**

- **Library**
  
  `varma`

- **Purpose**
  
  Finds an autocorrelation function matrix for multiple dependent variables

- **Format**
  
  \[ x = \text{macf}(y, \text{lag}); \]

- **Input**
  
  \[ y \quad T \times L \text{ matrix of data} \]

  \[ \text{lag} \quad \text{scalar, the lag for which an autocorrelation matrix is desired. Specify 0 to obtain the initial correlation} \]

- **Output**
  
  \[ x \quad L \times L \text{ matrix of autocorrelations, } \text{res and res(-lag)} \]

- **Globals**
  
  None

- **Source**
  
  `varma.src`
8. COMMAND REFERENCE

**Library**

varma

**Purpose**

Finds the Newey-West Covariance matrix

**Format**

\[ x = \text{nw}(\text{covb}, \text{resid}); \]

**Input**

- **covb** \( Q \times Q \) matrix, covariance matrix for the AR parameters
- **resid** \( T \times L \) matrix of residuals

**Output**

- \( x \) \( Q \times Q \) matrix, Newey-West adjusted covariances.

**Globals**

- \_vm\_nwtrunc sets the number of autocorrelations to use in calculating the
  Newey-West correction (\( q \) in the Remarks section below). Setting \_vm\_nwtrunc = 0
  causes GAUSS to use a truncation lag given by Newey and West, \( q = 4(T/100)^{2/9}. \)

**Remarks**

The Newey-West correction is used to account for the effect of heteroskedasticity and
residual serial correlation on estimated parameter standard errors. The adjusted
parameter covariance matrix is \((X'X)^{-1}\Omega(X'X)^{-1}\) where

\[
\Omega = \sum_{t=1}^{T} \sum_{j=1}^{q} \left[ \frac{j}{q+1} \right] \sum_{t=j+1}^{T} (x_t \varepsilon_t \varepsilon_{t-j} x_{t-j} + x_t \varepsilon_t \varepsilon_{t-j} x_{t-j})
\]

(8.1)

**Source**

varma.src
**pacf**

- **Library**
  arima

- **Purpose**
  Computes partial autocorrelations for a univariate time series.

- **Format**
  \[ a = \texttt{pacf}(y,l,d); \]

- **Input**
  \[
  y \quad N \times 1 \text{ vector, data.} \\
  l \quad \text{scalar, number of partial autocorrelations to compute.} \\
  d \quad \text{scalar, order of differencing.}
  \]

- **Output**
  \[
  a \quad l \times 1 \text{ vector, partial autocorrelations.}
  \]

- **Source**
  tsutil.src
8. COMMAND REFERENCE

- **Library**
  
  varma

- **Purpose**
  
  Returns parameter estimates from `ecm` and `varmax`

- **Format**
  
  ```
  coeffs = paramconfig(p,q,coeffs,se,x,ecmflag);
  ```

- **Input**
  
  - `p` scalar, order of the AR process
  - `q` scalar, order of the MA process
  - `coeffs` \( L \times (p + q + K + 1) \times 1 \) vector of coefficient estimates in the order AR, MA, x, Constant
  - `se` \( L \times (p + q + K + 1) \times 1 \) vector of standard error estimates in the order AR, MA, x, Constant
  - `x` \( T \times K \) matrix of explanatory variables
  - `ecmflag` scalar, equals one if an ECM model was estimated, zero otherwise.

- **Output**
  
  - `coeffs` compact matrix created using `vput`. Read it using `vread`. It contains:
    - `pi` \( L \times L \) matrix, the impact matrix. Only returned if an `ecm` model was estimated.
      
      *Note that \( \Pi \) is a reserved word in GAUSS. Users will need to assign this to a different variable name.*
    - `pi_se` \( L \times L \) matrix of impact coefficient standard errors. Only returned if an `ecm` model was estimated.
    - `phi` \( p \times (L \times L) \) matrix of AR coefficient estimates stacked in the order AR(1), ..., AR(p)
    - `phi_se` \( p \times (L \times L) \) matrix of AR standard errors stacked in the order AR(1), ..., AR(p)
    - `theta` \( q \times (L \times L) \) matrix of MA coefficient estimates stacked in the order MA(1), ..., MA(q). Only returned if a `varmax` model was estimated.
theta_se  $q^* (L \times L)$ matrix of $MA$ standard errors stacked in the order $MA(1), ..., MA(q)$. Only returned if a varmax model was estimated.

beta  $L \times K$ matrix of $x$ coefficient estimates. Only returned if a varmax model was estimated.

beta_se  $L \times K$ matrix of $x$ coefficient standard errors. Only returned if a varmax model was estimated.

 Globals

None

 Source

varma.src
8. COMMAND REFERENCE

- **Library**
  - arima

- **Purpose**
  Simulate ARMA time series process.

- **Format**
  
  \[ y = \text{simarma}(b, p, q, \text{const}, n, k, \text{std}, seed); \]

- **Input**
  - \( b \): \( K \times 1 \) vector, coefficient values for theoretical ARMA process.
  - \( p \): scalar, the autoregressive order.
  - \( q \): scalar, the moving average order.
  - \( \text{const} \): scalar, value of the constant term.
  - or – \( N \times M \) matrix, fixed regressor matrix.
  - \( n \): scalar, the number of observations to generate.
  - \( k \): scalar, the number of replications to generate.
  - \( \text{std} \): scalar, the standard deviation of the error process.
  - \( \text{seed} \): scalar, the value of the seed. If \( \text{seed} = 0 \), then \text{rndn} is used, otherwise \text{rndns} is used.

- **Output**
  - \( y \): \( N \times K \) matrix, simulated ARMA process. Each column represents an independent realization of a univariate time series.

- **Remarks**
  \text{simarma} only simulates times series which are generated by normally distributed errors.

  If your simulation is large or if your available memory is limited, make several calls to \text{simarma} during a simulation. Keep in mind that there is some overhead computing the starting values with the desired multivariate distribution.

  If the process you are simulating lies on or near a boundary, try generating a longer time series, then trim the beginning observations. In general, \text{simarma} should give reasonable results since the starting values are normalized to have required multivariate normal distribution.

- **Source**
  - \text{simarma.src}
sumstat

- **Library**
  varma

- **Purpose**
  Return summary statistics from the **ecm** and **varmax** procedures

- **Format**
  \[
  \{ \text{ss, info} \} = \text{sumstat}(\text{res, y, f, ynames, ecmflag});
  \]

- **Input**
  - **res** \(T \times L\) matrix, residuals
  - **y** \(T \times L\) matrix of dependent variables
  - **f** value of the maximized likelihood function.
  - **ynames** \(L \times 1\) character matrix of dependent variable names
  - **ecmflag** scalar, equals one if an **ecm** model was estimated, zero otherwise.

- **Output**
  - **ss** \(2 \times L\) matrix, sum of squares of \(y\) in row 1 and sum of squared errors in row 2.
  - **info** \(4 \times L\) matrix of information criteria
    - **row 1** the value of the likelihood function
    - **row 2** the Akaike Information Criterion (AIC)
    - **row 3** the Schwarz Bayesian Information Criterion (BIC)
    - **row 4** the likelihood ratio statistic (LRS)

- **Globals**
  None

- **Source**
  varma.src

70
8. COMMAND REFERENCE

**Library**

arima

**Purpose**

Compute the theoretical autocovariances given the coefficient values from an ARMA($p,q$) process.

**Format**

$$g = \text{tautocov}(b,p,q);$$

**Input**

- $b$ $K \times 1$ vector, parameter coefficients.
- $p$ scalar, the autoregressive order.
- $q$ scalar, the moving average order.

**Output**

- $g$ $[\text{Max}(p,q)+1] \times 1$ vector, theoretical autocovariances.

**Remarks**

The theoretical autocorrelations are found by dividing $g$ by $g[1]$.

**Source**

tautocov.src
**tscs**

- **Library**
  
  tscs

- **Purpose**

  Estimates the parameters of the pooled time-series cross-section regression model.

- **Format**

  \[
  \{ \text{bdv, vcdv, mdv, bec, vcec, mec} \} = \text{tscs} (\text{dataset, depvar, indvars, grp})
  \]

- **Input**

  - **dataset**  
    
    string, name of the input GAUSS data set.
  
  - **depvar**  
    
    string, name of the dependent (endogenous) variable
    
    - or -
    
    scalar, index of the dependent (endogenous) variable.
  
  - **indvars**  
    
    \( K \times 1 \) character vector, names of the independent (exogenous) variables
    
    - or -
    
    \( K \times 1 \) numeric vector, indices of the independent (exogenous) variables.
  
  - **grp**  
    
    string, name of the group variable
    
    - or -
    
    scalar, index of the group variable.

- **Output**

  - **bdv**  
    
    \( K \times 1 \) vector, regression coefficients from the dummy effects model (excluding individual-variables regression model).
  
  - **vcdv**  
    
    \( K \times K \) matrix, variance-covariance matrix of the dummy variables regression model.
  
  - **mdv**  
    
    \((K+1)\times(K+1)\) matrix, moment matrix of the transformed variables (including a constant) from the dummy variables regression model.
  
  - **bec**  
    
    \( K \times 1 \) vector, regression coefficients from the random effects regression model.
  
  - **vcec**  
    
    \( K \times K \) matrix, variance-covariance matrix of the random effects regression model.
  
  - **mec**  
    
    \((K+1)\times(K+1)\) matrix, moment matrix of the transformed variables (including a constant) from the random effects regression model.
# Globals

**__tmodel**

Scalar, controls the type of models to be estimated. Possible values are:

- **0**: all models are estimated.
- **1**: the random effects (error components model) is not estimated.

Default: 0.

**__tsstnd**

Scalar. If 1, print standardized estimates of regression parameters.

Default: 1.

**__tsmeth**

Scalar. Possible values are:

- **0**: Uses the fixed effects estimates of the individual-specific effects to estimate the variance components of the random effects model. Use this option if there are a different number of observations for each cross-sectional unit. The chi-squared test for the individual error components equal to 0 may not be correct if there are a different number of observations for each individual.
- **1**: Uses regression on group means to estimate variance components.

Default: 0.

**__tsise**

Scalar. If 1, the individual-specific effects are not printed. Default = 0.

**__tsmnsfn**

String, the name of a file in which to save the group means of the data set. By default, __tsmnsfn = “”, so the means are not saved.

**__header**

String, specifies the format for the output header. __header can contain zero or more of the following characters:

- **t**: print title (see __title)
- **l**: bracket title with lines
- **d**: print date and time
- **v**: print procedure name and version number
- **f**: print file name being analyzed

Example:

```
__header = "tld";
```

If __header == “”, no header is printed. Default = “tldvf”.

**__output**

Scalar, if nonzero, results are printed to screen. Under UNIX, default = 1; under DOS, default = 2.
__row__ scalar. Specifies how many rows of the data set are to be read per iteration of the read loop. By default, the number of rows to be read is calculated by `tscs`.

__rowfac__ scalar, “row factor”. If `tscs` fails due to insufficient memory while attempting to read a GAUSS data set, __rowfac__ may be set to some value between 0 and 1 to read a proportion of the original number of rows of the GAUSS data set. For example, setting

```
__rowfac = 0.8;
```

causes GAUSS to read in 80% of the rows of the GAUSS data set that were read when the failure due to insufficient memory occurred.

__rowfac__ has an effect only when __row__ = 0.

Default = 1.

__title__ string, a title to be printed at the top of the output header (see __header__). By default, no title is printed (__title__ = “”).

**Remarks**

The data must be contained in a GAUSS data set cross-sectional unit by cross-sectional unit, with one variable containing an index for the units. From each cross-sectional unit all observations must be grouped together. For example, for the first cross-sectional unit there may be 10 rows in the data set, for the second cross-sectional unit there may be another 10 rows, and so on. Each row in the data set contains measurements on the endogenous and exogenous variables measured for each observation along with the index identifying the cross-sectional unit.

The index variable must be a series of integers. While all observations for each cross-sectional unit must be grouped together, they do not have to be sorted according to the index.

**Example**

The following example is taken from the program `tscs.e`, located in the `examples` subdirectory. The program uses the sample data in `jdata.dat`.

```
library tscs;
#include tscs.ext;
tscsset;
 lhs = { x2 };
 exog = { x3 };
 inname = "jdata";
 output file = jdata.out reset;
 grp = { x1 };
 _tsmeth = 1;
call tscs(inname,lhs,exog,grp);
output off;
```
8. COMMAND REFERENCE

■ Source

tscs.src
**tscsset**

- **Library**
  
  *tscs*

- **Purpose**
  
  Initializes TSCS global variables to default values.

- **Format**
  
  `tscsset;`

- **Input**
  
  None

- **Output**
  
  None

- **Remarks**
  
  Putting this instruction at the top of all programs that invoke *tscs* is generally good practice. This prevents globals from being inappropriately defined when a program is run either several times or after another program that also calls *tscs*.

  *tscsset* calls *gausset*.

- **Source**
  
  *tscs.src*
8. COMMAND REFERENCE

- **Library**
  
arima

- **Purpose**
  
  Estimate forecasts using estimation results obtained from arima.

- **Format**
  
  \[ f = \text{tsforecast}(b,y,p,d,q,\text{const},e,h); \]

- **Input**
  
  - \( b \): \( K \times 1 \) vector, estimated coefficients.
  - \( y \): \( N \times 1 \) vector, data.
  - \( p \): scalar, the autoregressive order.
  - \( d \): scalar, the order of differencing.
  - \( q \): scalar, the moving average order.
  - \( \text{const} \): scalar, if 1, a constant is estimated, 0 otherwise.
  - \( e \): \( N \times 1 \) vector, residuals reported by arima program.
  - \( h \): scalar, the number of step-ahead forecasts to compute.

- **Output**
  
  \( f \): \( h \times 3 \) matrix,
  
  - \([.,1]\) Lower forecast confidence bounds.
  - \([.,2]\) Forecasts.
  - \([.,3]\) Upper forecast confidence bounds.

- **Globals**
  
  - \texttt{amcritl} scalar, confidence level to compute for forecast confidence bounds. Default = 0.95.
  - \texttt{output} scalar
    
    - 0 Nothing is printed.
Forecasts, confidence bounds and forecast standard errors are printed.

**Remarks**

Data must be transformed before being sent to `tsforecast`.

`tsforecast` does not compute forecasts for models with fixed regressors.

**Source**

`forecast.src`
8. COMMAND REFERENCE

- **Library**
  varma

- **Purpose**
  Calls and prints unit root and cointegration tests

- **Format**
  unitroots(y, ynames)

- **Input**
  
  - \( y \) \( T \times L \) matrix of dependent variables
  
  - \( ynames \) \( L \times 1 \) character vector of dependent variable names

- **Globals**

  \_vm\_output is either a scalar or a \( 6 \times 1 \) vector. Set \_vm\_output = 0 to suppress all printing from the ecm and varmax estimations. Set \_vm\_output > 0 to print all ecm and varmax output.

  Define \_vm\_output as a \( 6 \times 1 \) vector to control the printing of various parts of ecm and varmax output.

  1. Set element [1] of \_vm\_output to a non-zero value to print the model’s header

  2. Set element [2] of \_vm\_output to a non-zero value to print a variety of unitroot tests, and, if a multivariate model, cointegration tests from the unitroots procedure.

- **Remarks**

  unitroots calls the unit root and cointegration procedures, vmadf, vmcadf, vmpp, and vmsj. These use the global variable \_vm\_adforder to define the number of AR lags to include in the unit root and ECM cointegration specifications.

- **Source**

  varma.src
varmaset

- **Library**
  varma

- **Purpose**
  Sets global variables in the VARMA library to their default values.

- **Format**
  varmaset;

- **Input**
  None

- **Output**
  None

- **Remarks**
  Putting this instruction at the top of all programs that call a VARMA library procedure is generally good practice. This will prevent globals from being inappropriately defined when a program is run either several times or after another program that also calls one of the VARMA procedures.

- **Source**
  varma.src
8. COMMAND REFERENCE

varmax

Library
varma

Purpose
Computes exact maximum likelihood parameter estimates for a varmax model.

Format

\{
  \texttt{coeffs, res, vc, ynames, xnames, covb, statret, retc}
\} =
\varmax(\texttt{dataset, depvars, indvars, ynames, xnames, p, d, q, vctype})

Input

dataset string, name of dataset.
   If this is a null string, , the procedure assumes that the actual data has
   been passed in the depvar and \texttt{indvars} arguments.

   dependent variables.

1. If \texttt{dataset} is a null string, , this is interpreted as:
   \( T \times L \) matrix, the dependent variables

2. If \texttt{dataset} contains the name of a dataset, this is interpreted as:
   \( L \times 1 \) character vector, names of dependent variables in the
   dataset
   – or –
   \( L \times 1 \) numeric vector, indices of dependent variables in the
   dataset
   These can be any subset of the variables in the dataset, and
   can be in any order.

NOTE: Each column must be ordered as \( y_1, y_2, \ldots, y_T \). The top row is the
first observation and the last row is the most current observation.

\texttt{indvars} independent variables.

1. Enter a 0 if there are no independent variables. This will result
in a vector ARMA model being estimated.

2. If \texttt{dataset} is a null string, , this is interpreted as a
   \( T \times K \) matrix, the independent variables

3. If \texttt{dataset} contains the name of a dataset, this is interpreted as
8. COMMAND REFERENCE

\textbf{varmax}

\hspace{1cm} K \times 1 \text{ character vector, names of independent variables in the dataset}
\hspace{1cm} K \times 1 \text{ numeric vector, indices of independent variables in the dataset.}
\hspace{1cm} These can be any subset of the variables in the dataset, and can be in any order.

\textbf{NOTE:} Each column must be ordered as \(x_1, x_2, ..., x_T\). The top row is the first observation and the last row is the most current observation.

\textbf{ynames} \hspace{1cm} L \times 1 \text{ character matrix}
\hspace{1cm} Names for the variables in \textit{depvars}, or scalar 0. GAUSS will supply variable names \(Y_1, ..., Y_L\) if 0 is entered.

\textbf{xnames} \hspace{1cm} K \times 1 \text{ character matrix}
\hspace{1cm} Names for the variables in \textit{indvars}, or scalar 0. GAUSS will supply variable names \(X_1, ..., X_k\) if 0 is entered.

\textbf{p} \hspace{1cm} scalar, number of \textit{AR} matrices to be estimated.

\textbf{d} \hspace{1cm} scalar, order of differencing to achieve stationarity

\textbf{q} \hspace{1cm} scalar, number of \textit{MA} matrices to be estimated.

\textbf{vcType} \hspace{1cm} scalar, set to 1 for ML covariance matrix of parameters set to 2 for QML covariance matrix of parameters

\section*{Output}

\textbf{coeffs} \hspace{1cm} compact matrix created using \texttt{vput}. Read it using \texttt{vread}. It contains:

\textbf{phi} \hspace{1cm} \(p \star (L \times L)\) matrix of \textit{AR} coefficient estimates stacked in the order \textit{AR}(1), ..., \textit{AR}(p)

\textbf{phi.se} \hspace{1cm} \(p \star (L \times L)\) matrix of \textit{AR} standard errors stacked in the order \textit{AR}(1), ..., \textit{AR}(p)

\textbf{theta} \hspace{1cm} \(q \star (L \times L)\) matrix of \textit{MA} coefficient estimates stacked in the order \textit{MA}(1), ..., \textit{MA}(q).

\textbf{theta.se} \hspace{1cm} \(q \star (L \times L)\) matrix of \textit{MA} standard errors stacked in the order \textit{MA}(1), ..., \textit{MA}(q).

\textbf{beta} \hspace{1cm} \(L \times K\) matrix of \(x\) coefficient estimates.

\textbf{beta.se} \hspace{1cm} \(L \times K\) matrix of \(x\) coefficient standard errors.

\textbf{res} \hspace{1cm} \(T \times L\) matrix, residuals

\textbf{vc} \hspace{1cm} \(L \times L\) matrix, residual covariance matrix

82
8. COMMAND REFERENCE

**ynames**  
$L \times 1$ character matrix of names for the variables in `depears`, or scalar 0. GAUSS will supply variable names $Y_1, \ldots, Y_L$ if 0 is entered.

**xnames**  
$(p + q + K) \times 1$ character matrix of corresponding variable names, in the order AR($p$), MA($q$), `indevars`.

**covb**  
$Q \times Q$ matrix of estimated parameters. The parameters are in the row-major order: AR(1) to AR($p$), MA(1) to MA($q$), `beta` (if $x$ variables were present in the estimation), and the constants.

**statret**  
compact matrix created using `vput`. Read it using `vread`. It contains:

- **ss**  
  $L \times 2$ matrix, the sum of squares for Y in column one and the sum of squared error in column two

- **info**  
  Lx4 matrix
  
  - **row one** - the likelihood value
  - **row two** - the Akaike Information Criterion
  - **row three** - the Schwarz Bayesian Information Criterion
  - **row four** - the Likelihood Ratio Statistic

- **arroots**  
  $p \times 1$ vector of AR roots, possibly complex

- **maroots**  
  $q \times 1$ vector of MA roots, possibly complex

- **acfm**  
  $L \times (p \times L)$ matrix, the autocorrelation function. The first $L$ columns are the lag 1 ACF, The last $L$ columns are the lag $p$ ACF.

- **pacfm**  
  $L \times (p \times L)$ matrix, the partial autocorrelation function, only returned if a univariate model is estimated. The first $L$ columns are the lag 1 ACF, The last $L$ columns are the lag $p$ ACF.

- **portman**  
  `(vmlags)\times3$ matrix of portmanteau statistics for the multivariate model and Ljung-Box statistics for the univariate model. The time period is in column one, the $Qs$ (portmanteau) statistic in column two and the p-value in column three

- **retc**  
  $2 \times 1$ vector, return code
  
  - first element of retc

  0 normal convergence
  1 forced exit
  2 maximum number of iterations exceeded
  3 function calculation failed
  4 gradient calculation failed
5 Hessian calculation failed
6 line search failed
7 error with constraints

second element of retc
0 covariance matrix of parameters fails
1 ML covariance matrix computed
2 QML covariance matrix
3 Cross-Product covariance matrix

- **Globals**

  ```
  __output
  1. Set __output = 0 to suppress all printing from the sqpsolve procedure.
  2. Set __output > 0 to print sqpsolve results.
  __vm_A
  \( M \times K \) matrix, linear equality constraint coefficients
  __vm_adforder
  scalar, number of AR lags in the ADF test statistic
  __vm_B
  \( M \times 1 \) vector, linear equality constraint constants These globals are used to specify linear equality constraints of the following type: __vm_A * X = __vm_B where X is the \( K \times 1 \) unknown parameter vector.
  __vm_Bounds
  \( K \times 2 \) matrix, bounds on parameters. The first column contains the lower bounds, and the second column the upper bounds. If the bounds for all the coefficients are the same, a 1x2 matrix may be used. Default = -1e256 1e256
  __vm_C
  \( M \times K \) matrix, linear inequality constraint coefficients
  __vm_critl
  scalar, the significance levels defining p-values
  __vm_D
  \( M \times 1 \) vector, linear inequality constraint constants These globals are used to specify linear inequality constraints of the following type: __vm_C * X >= __vm_D where X is the \( K \times 1 \) unknown parameter vector.
  __vm_DirTol
  scalar, convergence tolerance for gradient of estimated coefficients. Default = 1e-5. When this criterion has been satisfied SQPSolve (the iteration procedure beneath varma and ecm) will exit the iterations.
  __vm_FeasibleTest
  scalar, if nonzero, parameters are tested for feasibility before computing function in line search. If function is defined outside inequality boundaries, then this test can be turned off.
  ```
* _vm_IndEquations  $K \times L$ matrix of zeros and ones. Used to set zero restrictions on the $x$ variables to be estimated. Only used if the number of equations, _vm_ is greater than one. Elements set to one indicate the coefficients to be estimated. If _vm_ $= 1$, all coefficients will be estimated. If _vm_ $> 1$ and _vm_IndEquations is set to a missing value (the default), all coefficients will be estimated.

* _vm_Lagrange_ compact matrix created using *vput*. Contains the Lagrangean coefficients for the constraints. They may be extracted with the *vread* procedure using the following strings:

<table>
<thead>
<tr>
<th>String</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“lineq”</td>
<td>linear equality constraints</td>
</tr>
<tr>
<td>“nlineq”</td>
<td>nonlinear equality constraints</td>
</tr>
<tr>
<td>“linineq”</td>
<td>linear inequality constraints</td>
</tr>
<tr>
<td>“nilineq”</td>
<td>nonlinear inequality constraints</td>
</tr>
<tr>
<td>“bounds”</td>
<td>bounds</td>
</tr>
</tbody>
</table>

When an inequality or bounds constraint is active, its associated Lagrangean is nonzero. The linear Lagrangeans precede the nonlinear Lagrangeans in the covariance matrices.

* _vm_lags_ scalar, No. of lags over which ACF and Diagnostics are calculated.

* _vm_MaxIters_ scalar, maximum number of iterations. Default = 1e+5. Termination can be forced by pressing C on the keyboard.

* _vm_Output_ scalar or a $6 \times 1$ matrix. Set _vm_output_ $= 0$ to suppress all printing from the *ecm* and *varmax* estimations. Set _vm_output_ $> 0$ to print all *ecm* and *varmax* output.

Set _vm_output_ to a $6 \times 1$ vector to control the printing of various parts of *ecm* and *varmax* output.

1. Set element [1] of _vm_output_ to a non-zero value to print the model’s header
2. Set element [2] of _vm_output_ to a non-zero value to print a variety of unitroot tests, and, if a multivariate model, cointegration tests from the unitroots procedure.
3. Set element [3] of _vm_output_ to a non-zero value to print summary statistics for each estimated equation
4. Set element [4] of _vm_output_ to a non-zero value to print the estimated coefficients and their standard errors
5. Set element [5] of _vm_output_ to a non-zero value to print the roots of the AR and MA characteristic equations
6. Set element [6] of _vm_output_ to a non-zero value to print the autocorrelation function and portmanteau statistics
varmax

8. COMMAND REFERENCE

__vm__PrintIters 1. Set __vm__PrintIters = 0 (__output is not equal to zero) to print an executing... message while starting values are calculated for each dependent variable during the sqpsolve operation.
2. Set __vm__PrintIters > 0 (__output is not equal to zero) to print sqpsolve iteration information. This information includes the value of the objective function and the gradient at each estimated coefficient. It is useful in finding where and why convergence might fail.

__vm__RandRadius  scalar, if nonzero gives the radius of random search taken when the STEPBT line search fails. If zero, no random search occurs and SQPSolve returns with an error code. Default = .01.

__vm__scale  scalar, scalar or an Lx1 vector, scales the time series. If scalar, all series are multiplied by the value. If an Lx1 vector, each series is multiplied by the corresponding element of `vm scale. Default = 4 / standard deviation (found to be best by experimentation).

__vm__SetConstraints  scalar, set to a nonzero value to impose stationarity and invertibility by constraining roots of the AR and MA characteristic equations to be outside the unit circle. Set to zero (the default) to estimate an unconstrained model.

__vm__Start  \((Q - L) \times 1\) vector of starting values, in the row major order, AR(1) to AR(p), MA(1) to MA(q), beta, and the covariance matrix of these parameters.

__vm__TrustRadius  scalar, gives the radius of the trust region if nonzero, i.e., the maximum amount in absolute value for the direction vector at each iteration. If zero, the trust region method inactivated.

 Remarks

Errors are assumed to be distributed N(0,Q). The estimation procedure assumes that all series are stationary. Setting __vm__SetConstraints nonzero enforces stationarity, by constraining the roots of the characteristic equation

\[ 1 - \Phi_1 z - \Phi_2 z^2 - \cdots - \Phi_p z^p \]

to be outside the unit circle (where \(\Phi_i\), \(i = 1, ..., p\) are the AR coefficient matrices).

If any estimated parameters in the coefficient matrices are on a constraint boundary, the Lagrangeans associated with these parameters will be nonzero. These Lagrangeans are stored in the global __vm__Lagrange. Standard errors are generally not available for parameters on constraint boundaries.

 Source

varma.src

86
8. Command Reference

- **Library**
  varma

- **Purpose**
  Compute the Augmented Dickey Fuller statistic, allowing for deterministic polynomial time trends of an arbitrary order.

- **Format**
  \[
  \{ \alpha, tstat, vmzt\text{crit} \} = \text{vmadf}(x, p, l);
  \]

- **Input**
  - \(x\) matrix, time series variable
  - \(p\) scalar, order of the time-polynomial to include in the ADF regression. Set \(p = -1\) for no deterministic part.
  - \(l\) scalar, number of lagged changes of \(x\) to include in the fitted regression.

- **Output**
  - \(\alpha\) estimate of the autoregressive parameter;
  - \(tstat\) ADF t-statistic
  - \(vmzt\text{crit}\) \((6 \times 1)\) vector of critical values for the adf-t-statistic: 1% 5% 10% 90% 95% 99%

- **Source**
  varma.src
vmcdf

■ Library

varma

■ Purpose

Compute the Augmented Dickey Fuller statistic applied to the residuals of a cointegrating regression, allowing for deterministic polynomial time trends of an arbitrary order.

■ Format

\{ \alpha, tstat, vmrztcrit \} = \texttt{vmcdf}(y, x, p, l);

■ Input

\begin{itemize}
  \item \texttt{y} \quad \text{dependent variable}
  \item \texttt{x} \quad \text{explanatory variables}
  \item \texttt{p} \quad \text{order of the time-polynomial to include in the cointegrating regression. Set } p = -1 \text{ for no deterministic part.}
  \item \texttt{l} \quad \text{number of lagged changes of the residuals to include in the fitted regression.}
\end{itemize}

■ Output

\begin{itemize}
  \item \texttt{alpha} \quad \text{estimate of the autoregressive parameter;}
  \item \texttt{tstat} \quad \text{ADF t-statistic}
  \item \texttt{vmrztcrit} \quad 6 \times 1 \text{ vector of critical values for the adf-t-statistic: 1\% 5\% 10\% 90\% 95\% 99\%}
\end{itemize}

■ Source

varma.src
8. Command Reference

- **Library**
  varma

- **Purpose**
  Returns critical values for the Johansen Maximum Eigenvalue statistic. Computed using 8000 iterations and 500 observations.

- **Format**
  
  $c$-values = \texttt{vmc\_sja}(n,p);

- **Input**
  
  - n \hspace{1cm} scalar, number of variables in the system
  - p \hspace{1cm} scalar, order of the time-polynomial in the fitted regression

- **Output**
  
  $c$-values A

- **Source**
  
  varma.src
**vmc_sjt**

- **Library**
  
  `varma`

- **Purpose**
  
  Returns critical values for the Johansen Trace statistic.

- **Format**
  
  \[
  \text{c-values} = \text{vmc_sjt}(n,p);
  \]

- **Input**
  
  - `n` scalar, number of variables in the system
  - `p` scalar, order of the time-polynomial in the fitted regression

- **Output**
  
  `vmrztcrit` 6x1 vector of critical values for the adf-t-statistic: 1 5 10 90 95 99

- **Source**
  
  `varma.src`
8. COMMAND REFERENCE

- **Library**
  varma

- **Purpose**
  Returns residuals from a regression of data on a time trend polynomial

- **Format**
  \[ res = \text{vmdetrend}(y, p); \]

- **Input**
  - \( y \) \( T \times L \) matrix of data
  - \( p \) scalar. If \( p = -1 \) returns the data. Use \( p = 0 \) for demeaning; \( p = 1 \) for regression against a constant term and trend; \( p > 1 \) for a higher order polynomial time trend.

- **Output**
  - \( res \) \( T \times L \) matrix of residuals

- **Source**
  varma.src
vmdiff

■ Library
  varma

■ Purpose
  Differences matrices

■ Format
  \[ y = \text{vmdiff}(x,d); \]

■ Input
  \[ x \quad \text{\(T \times K\) matrix} \]
  \[ d \quad \text{scalar, the number of periods over which differencing occurs} \]

■ Output
  \[ y \quad \text{\((T - d) \times K\) matrix, the differenced data.} \]

■ Source
  varma.src
8. COMMAND REFERENCE

- **Library**
  
  varma

- **Purpose**

  Calculates forecasts from a VARMAX model

- **Format**

  \[ f = \text{vmforecast}(\text{coeffs},p,q,y,x,\text{res},t); \]

- **Input**

  - `coeffs` compact matrix created using `vput`. Read it using `vread`. It contains:
    
    - `phi` \( p \times (L \times L) \) matrix of AR coefficient estimates stacked in the order \( AR(1),...,AR(p) \)
    - `phi_se` \( p \times (L \times L) \) matrix of AR standard errors stacked in the order \( AR(1),...,AR(p) \)
    - `theta` \( q \times (L \times L) \) matrix of MA coefficient estimates stacked in the order \( MA(1),...,MA(q) \)
    - `theta_se` \( q \times (L \times L) \) matrix of MA standard errors stacked in the order \( MA(1),...,MA(q) \)
    - `beta` \( L \times K \) matrix of \( x \) coefficient estimates
    - `beta_se` \( L \times K \) matrix of \( x \) coefficient standard errors stacked in the order \( MA(1),...,MA(q) \)
    - `b0` \( L \times 1 \) matrix of intercept estimates
    - `b0_se` \( L \times 1 \) matrix of intercept standard errors (missing values in `vm_ver = 1,0,0.

    - `p` scalar, AR order
    - `q` scalar, MA order
    - `y` \( T \times L \) matrix, the variables to be forecast
    - `x` \( t \times K \) matrix of \( x \) variables covering only the forecast horizon, in the order \( T + 1,...,T + t \) or the scalar zero if there are no \( x \) variables.
    - `res` \( T \times L \) matrix of residuals from the VARMA estimation
    - `t` scalar, the number of periods to forecast

- **Output**
\textbf{vmforecast}

\[ f \times (L + 1) \] matrix. Column one contains the period forecast, The remaining columns contain the forecast values.

\section*{Globals}

None

\section*{Remarks}

The \texttt{varmax} and \texttt{ecm} procedures estimate centered models and do not return intercepts. However, \texttt{vmforecast} allows intercepts, so that it might be used with the results of other estimation procedures.

\section*{Source}

\texttt{varma.src}
8. COMMAND REFERENCE

■ Library
varma

■ Purpose
Returns Phillips-Perron unit root test statistics and critical values

■ Format
{ ppb, ppt, pptcrit } = vmpp(y,p);

■ Input
y $T \times 1$ vector, a time series
p scalar, order of the time-polynomial to include in the regression. Set $p = -1$ for no deterministic part, $p = 0$ for a constant term, and $p = 1$ for a constant with trend.

■ Output
ppb scalar, estimate of the autoregressive parameter, the $\rho$ coefficient below.
ppt scalar, the adjusted $t$-statistic for testing $H_0 : \rho = 1$
pptcrit $6 \times 1$ vector of critical values, vector of critical values for the adjusted $t$ statistic, in the order 1%, 5%, 10%, 90%, 95%, 99%.

■ Globals
\_vm\_nwtrunc sets the number of autocorrelations to use in calculating the Newey-West correction ($q$ in the Remarks section below. Setting \_vm\_nwtrunc = 0 causes GAUSS to use a truncation lag given by Newey and West, $q = 4(T/100)^{2/9}$.

■ Remarks
Phillips (1987) and Phillips and Perron (1988) test for unit roots by adjusting the OLS estimate of an AR(1) coefficient for serial correlation in the OLS residuals. Three specifications are considered, an AR(1) model without a drift, an AR(1) with a drift, and an AR(1) model with a drift and linear trend:

\[
\begin{align*}
Y_t &= \rho Y_{t-1} + \varepsilon_t \\
Y_t &= \alpha + \rho Y_{t-1} + \varepsilon_t \\
Y_t &= \alpha + \delta t + \rho Y_{t-1} + \varepsilon_t
\end{align*}
\]
The unit root null hypothesis is $H_0 : (\rho - 1) = 0$.

Hamilton (1994, pp. 506-511) tests this hypothesis using two statistics that are analogs of the Phillips and Perron (1988) $Z_\alpha$ and $Z_t$ statistics. Hamilton's statistics are based on OLS estimation of the above equations. They allow an identical formula for each statistic to be used for all three cases.

The `vmpp` procedure returns the $Z_t$ statistic as calculated by Hamilton and critical values. Suppose any of the equations is estimated by OLS, returning $\hat{\rho}_T$ and $\hat{\sigma}_{\hat{\rho}_T}$ (the OLS estimates of $\rho$ and the standard error of $\hat{\rho}_T$ respectively), $t_T = (\rho - 1)/\hat{\sigma}_{\hat{\rho}_T}$ (the usual OLS $t$ statistic for testing $H_0$), $\hat{\varepsilon}_t$ (the OLS residuals), and $s_T$ (the estimated standard error of the regression).

Hamilton’s $Z_t$ statistic is:

$$Z_t = (\hat{\gamma}_0 / \hat{\lambda}^2) t_T - \{1/2 (\hat{\lambda}^2 - \hat{\gamma}_0)/\hat{\lambda}\} \{T(\hat{\sigma}_{\hat{\rho}_T}/s_T)\}$$

$\hat{\lambda}^2$ is an estimate of the asymptotic variance of the sample mean of $\varepsilon_t$. In the `vmpp` procedure $\hat{\lambda}^2$ is estimated using the Newey-West (1987) estimator,

$$\hat{\lambda}^2 = \hat{\gamma}_0 + 2 \sum_{j=1}^q \{1 - j/(q + 1)\} \hat{\gamma}_j$$

where $\hat{\gamma}_j = T^{-1} \sum_{t=j+1}^T \varepsilon_t \varepsilon_{t-j}$ are the sample autocovariances of $\varepsilon_t$.

A global variable, `vm_nwtrunc`, sets the number of autocorrelations to use in calculating the Newey-West correction ($q$ in the above equation). The default setting, `vm_nwtrunc = 0`, causes `GAUSS` to use a truncation lag given by Newey and West, $q = 4(T/100)^{2/9}$.

Under the null hypothesis, the $Z_t$ statistics has the same asymptotic distribution as a Dickey-Fuller statistic.

References


Source

`varma.src`
8. COMMAND REFERENCE

- **Purpose**
  Computes and prints the roots of the AR and MA characteristic equations.

- **Format**
  \[
  \{ \text{arroots}, \text{maroots} \} = \text{vmroots}(p, q, \text{coeffs});
  \]

- **Library**
  varma

- **Input**
  - \( p \) scalar, AR order
  - \( q \) scalar, MA order
  - \( \text{coeffs} \) compact matrix created using \text{vput}. Read it using \text{vread}. The contents used in \text{vmroots} are:
    - \( \phi \) \( p \times (L \times L) \) matrix of AR coefficient estimates stacked in the order \( AR(1), ..., AR(p) \)
    - \( \theta \) \( q \times (L \times L) \) matrix of MA coefficient estimates stacked in the order \( MA(1), ..., MA(q) \). Only returned if a \text{varmax} model was estimated.

- **Output**
  - \( \text{arroots} \) \( p \times 1 \) vector of AR roots, possibly complex
  - \( \text{maroots} \) \( q \times 1 \) vector of MA roots, possibly complex

- **Globals**
  \_vm\_output is either a scalar or a 6 \times 1 vector. Set \_vm\_output = 0 to suppress all printing from the \text{ecm} and \text{varmax} estimations. Set \_vm\_output > 0 to print all \text{ecm} and \text{varmax} output.

  Define \_vm\_output as a 6 \times 1 vector to control the printing of various parts of \text{ecm} and \text{varmax} output.
  1. Set element [1] of \_vm\_output to a non-zero value to print the model’s header
  2. Set element [5] of \_vm\_output to a non-zero value to print the roots of the AR and MA characteristic equations

- **Remarks**
  Calls the polymroot procedure.

- **Source**
  varma.src
vmrztcrit

- **Library**
  
  varma

- **Purpose**
  
  Returns $\tau$ critical values for the Augmented Dickey-Fuller statistic, derived from the residuals of a cointegrating regression. Depends on $p$, the $AR$ order in the fitted regression, the number of observations, and the number of explanatory variables.

- **Format**
  
  $c-values = \text{vmrztcrit}(nobs,n,p);$  

- **Input**
  
  $nobs$  
  scalar, number of observations in the series.
  
  $n$  
  scalar, column dimension of $x$;
  
  $p$  
  scalar, order of the time-polynomial in the null hypothesis

- **Source**
  
  varma.src
8. Command Reference

- **Library**
  
  varma

- **Purpose**
  
  Compute Johansen’s (1988) ML Trace and Maximum Eigenvalue statistics

- **Format**
  
  \[
  \{ ev, evec, lr1, lr2 \} = vmsj(x, p, k);
  \]

- **Input**
  
  \[
  x \quad T \times L \text{ matrix}
  \]
  
  \[
  p \quad \text{scalar, order of the time polynomial in the fitted regression. Set } p = -1 \\
  \text{for no deterministic part, } p = 0 \text{ for a constant term, and } p = 1 \text{ for a} \\
  \text{constant with trend.}
  \]
  
  \[
  k \quad \text{scalar, number of lagged difference terms to use when computing the} \\
  \text{estimator}
  \]

- **Output**
  
  \[
  ev \quad L \times 1 \text{ vector of eigenvalues}
  \]
  
  \[
  evec \quad L \times L \text{ matrix of eigenvectors. The first } r \text{ columns are the unnormalized} \\
  \text{cointegrating vectors.}
  \]
  
  \[
  lr1 \quad L \times 1 \text{ vector of Johansen’s likelihood ratio Trace statistics for the null} \\
  \text{hypotheses of } H_0: \text{ at most } r \text{ cointegrating vectors versus } H_1: \text{ not } H_0, \\
  r = 0, ..., L - 1
  \]
  
  \[
  lr2 \quad L \times 1 \text{ vector of Johansen’s Maximum Eigenvalue statistics for the null} \\
  \text{hypotheses of } H_0: r \text{ cointegrating vectors versus } H_1: r+1 \text{ cointegrating} \\
  \text{vectors, } r = 0, ..., L - 1
  \]

- **Globals**
  
  Set `vm_NoDet = 1` to suppress the constant term from the fitted regression and 
  include it in the co-integrating regression. Remember to set `vm_NoDet = 0` after the 
  procedure call to ensure that subsequent procedures are not affected.

- **Source**
  
  varma.src
vmztcrit

■ Library

varma

■ Purpose

Returns \( \tau \) critical values for the Augmented Dickey-Fuller test statistic, depending on the number of observations and \( p \), the AR order in the fitted regression. Computed using 10000 iterations.

■ Format

\[ c-values = \text{vmztcrit}(nobs,p); \]

■ Input

\( nobs \) scalar, number of observations in the series.

\( p \) scalar, order of the time-polynomial in the null hypothesis

■ Output

\( c-values \) 6x1 vector of critical values in order 1%, 5%, 10%, 90%, 95%, 99%.

■ Source

varma.src
Index

__vm__Hessian, 25

A ____________________________

acf, 40
__altnam, 47, 48
__am__itol, 42
__am__varn, 42
__amcritl, 78
__archisq, 49
arima, 41, 43
ARIMA, 35
arima.src, 43
arimaset, 35
__arinit, 48
__ariter, 48
armanames, 44
__arrsq, 50
__artobs, 49
__arvsig, 49
autocor, 45
autocorrelations, 37, 45
autocov, 46
autocovariances, 37, 46
autoreg, 47
autoreg.src, 50
arima.src, 42
autoreg.src, 45, 46, 51
autoregression, 35, 37
autoset, 37, 51

B ____________________________

bounds constraints, 9, 19, 24

C ____________________________

coeffprt, 52
__con, 48
condition of Hessian, 25
constraints, 9, 17, 21
corm, 54
covm, 55

E ____________________________

ecm, 56
equality constraints, 21, 23

F ____________________________

forecast, 35, 77
forecast.src, 78
forecasting, 26

G ____________________________

gausset, 51

H ____________________________

__header, 48, 73
Hessian, 19, 20, 25

I ____________________________

identify, 62
inequality constraints, 22, 23
Installation, 1

L ____________________________

Lagrange coefficients, 59, 85
library, 8, 33, 35, 37
linear constraints, 19, 21, 22
INDEX

M
macf, 64

N
Newton’s method, 18
nonlinear constraints, 9, 19, 23
nonlinear constraints, 23
nw, 65

O
___output, 42, 49, 73, 78

P
pacf, 66
panel data, 33
paramconfig, 67

Q
QML covariance matrix, 17

R
regression, autoregression, 37
regression, error components, 33
regression, fixed effects, 33
regression, TSCS, 33
___row, 49, 74
___rowfac, 49, 74

S
scaling, 25
simarma, 69
simarma.src, 69
starting point, 26
starting values, 17, 26
sumstat, 70

T
tautocov, 71
tautcov.src, 71
time series, 37
time-series cross-section, 33, 34, 72

___title, 49, 74
___tol, 49
tscs, 72
TSCS model, 33
tscs.src, 75, 76
tscsset, 33, 75, 76
tsforecast, 36, 77
___tsise, 73
___tsmeth, 73
___tsmnsfn, 73
___tmodel, 73
___tstnd, 73

U
unitroots, 79
UNIX, 1, 3

V
varma.src, 80
varmaset, 80
varmax, 81
__vm__Lagrange, 59, 85
vmadf, 87
vmc_sja, 89
vmc_sjt, 90
vmadf, 88
vmdetrend, 91
vmdiff, 92
vmforecast, 93
vmpp, 95
vmroots, 97
vmrztcrit, 98
vmsj, 99
vmztcrit, 100
__vpad, 49

W
Windows/NT/2000, 3