Time Series

Documentation Version: January 15, 2001

Contents

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. Use `cd` to make the directory containing **GAUSS** the current working directory.
2. Use `tar` to extract the files.

   `tar xvf` *device_name*

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

The following device names are suggestions. See your system administrator. If you are using Solaris 2.x, see Section 1.1.1.

Operating System	3.5-inch diskette	1/4-inch tape	DAT tape
Solaris 1.x SPARC	`/dev/rfd0`	`/dev/rst8`	
Solaris 2.x SPARC	`/dev/rfd0a` (vol. mgt. off)	`/dev/rst12`	`/dev/rmt/1l`
Solaris 2.x SPARC	`/vol/dev/aliases/floppy0`	`/dev/rst12`	`/dev/rmt/1l`
Solaris 2.x x86	`/dev/rfd0c` (vol. mgt. off)		`/dev/rmt/1l`
Solaris 2.x x86	`/vol/dev/aliases/floppy0`		`/dev/rmt/1l`
HP-UX	`/dev/rfloppy/c20Ad1s0`		`/dev/rmt/0m`
IBM AIX	`/dev/rfd0`	`/dev/rmt.0`	
SGI IRIX	`/dev/rdsk/fds0d2.3.5hi`		

1.1.1 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
```

1.2 DOS

1. Place the diskette in a floppy drive.
2. Log onto the root directory of the diskette drive. For example:

   ```
   A:<enter>
   cd\<enter>
   ```

3. 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

4. The screen output option used may require that the DOS screen driver ANSI.SYS be installed on your system. If ANSI.SYS is not already installed on your system, you can put the command like this one in your CONFIG.SYS file:

   ```
   DEVICE=C:\DOS\ANSI.SYS
   ```

   (This particular statement assumes that the file ANSI.SYS is on the subdirectory DOS; modify as necessary to indicate the location of your copy of ANSI.SYS.)

1.3 Differences Between the UNIX and DOS Versions

- In the DOS version, when the global **__output** = 2, information may be written to the screen using commands requiring the ANSI.SYS screen driver. These are not available in the current UNIX version, and therefore setting **__output** = 2 may have the same effect as setting **__output** = 1.
- 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 DOS 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 DOS.

Chapter 2

Pooled Time-Series Cross-Section Analysis

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

2.1 Getting Started

Run–Time Library version 3.2.37 is required to use these routines. See **_rtl_ver** in `src/gauss.dec`.

2.1.1 README Files

The file `README.ts` contains any last minute information on the **TSCS** procedures. Please read it before using them.

2.1.2 Setup

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:

```
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 5), you will also need the statement:

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

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

```
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.

The **TSCS** version number is stored in a global variable:

_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 **TSCS**.

2.2 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.

2.3 References

Judge, George C., R. Carter Hill, William E. Griffiths, Helmut Lütkepohl, and Tsoung-Chao Lee. 1988. *Introduction to the Theory and Practice of Econometrics. Second Edition*, New York: Wiley.

Hsiao, Cheng. 1986. *Analysis of Panel Data*. Cambridge: Cambridge University Press.

Chapter 3

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.

3.1 Getting Started

Run–Time Library version 3.2.37 is required to use these routines. See **_rtl_ver** in `src/gauss.dec`.

3.1.1 README Files

The file README.arm contains any last minute information on the **arima** procedures. Please read it before using them.

3.1.2 Setup

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 executiions 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 progam.

The **arima** version number is stored in a global variable:

_am_ver 3×1 matrix, the first element contains the major version number, the second element the minor versi on 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 **arima**.

3.2 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

$$\begin{aligned} L^j y_t &= y_{t-j} \\ \phi(L) &= 1 - \phi_1 L - \phi_2 L^2 - \cdots - \phi_p L^p \\ \theta(L) &= 1 - \theta_1 L - \theta_2 L^2 - \cdots - \theta_q L^q \end{aligned}$$

where it is assumed that e_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: ϕ (Px1), θ (Qx1), β (Mx1) and σ^2 (1x1).

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.

3.3 References

Granger, C.W.J. and Newbold, Paul. 1986. *Forecasting Economic Time Series. Second Edition*, San Diego: Academic Press.

Ansely, Craig F. 1979. "An Algorithm for the Exact Likelihood of a Mixed Autoregressive-Moving Average Process," *Biometrika* **66**, 59–65.

Chapter 4

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.

4.1 Getting Started

Run–Time Library version 3.2.37 is required to use these routines. See **_rtl_ver** in `src/gauss.dec`.

4.1.1 README Files

The file `README.ar` contains any last minute information on the **autoreg** procedures. Please read it before using them.

4.1.2 Setup

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 5), 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.

The **autoreg** version number is stored in a global variable:

_ar_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 **autoreg**.

4.2 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:

$$\begin{aligned} y_t &= x_t\beta + u_t \\ u_t - \phi_1 u_{t-1} - \ldots - \phi_p u_{t-p} &= e_t \end{aligned}$$

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: β (Kx1), ϕ (Lx1) and σ^2 (1x1). 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.

4.3 References

Judge, George C., R. Carter Hill, William E. Griffiths, Helmut Lütkepohl, and Tsoung-Chao Lee. 1988. *Introduction to the Theory and Practice of Econometrics. Second Edition*, New York: Wiley.

Chapter 5

Command Reference

Library

`arima`

Purpose

Computes sample autocorrelations for a univariate time series.

Format

a = **acf(**x,l,d**);**

Input

x	Nx1 vector. The mean is subtracted automatically.
l	scalar, the maximum lags to compute.
d	scalar, the difference order.

Output

a	lx1 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`

Library

`arima`

Purpose

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

Format

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

Input

startv	scalar, 0, then **arima** computes starting values. – or – Kx1 vector, starting values.
y	Nx1 vector, data.
p	scalar, the autoregressive order.
d	scalar, the order of differencing.
q	scalar, the moving average order.
const	scalar, if 1, a constant is estimated, 0 otherwise. – or – NxM matrix, fixed regressors. The number of rows in the fixed regressor matrix must be equal the number of rows for y after differencing.

Output

coefs	Kx1 vector, estimated model coefficients.
ll	scalar, the value of the log likelihood function.
e	Nx1 vector, residual from fitted model.
vcb	KxK matrix, the covariance matrix of estimated model coefficients.
aic	scalar, value of the Akaike information criterion.
sbc	scalar, value of the Schwartz Bayesian criterion.

■ Globals

_am_itol 3x1 vector, controls the convergence criterion.

- [**1**] Maximum number of iterations. Default = 100.
- [**2**] Minimum percentage change in the sum of squared errors. Default = 1e-8.
- [**3**] Minimum percentage change in the parameter values. Default = 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 Mth 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 autorgressive 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`

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`

Library

`auto`

Purpose

Computes specified autocorrelations for each column of a matrix.

Format

a = **autocor**(x,f,l);

Input

x	NxK matrix. Autocorrelations will be computed for each column separately. x is assumed to have 0 mean.
f	scalar, in range [0, **rows**(x)-1], denoting the first autocorrelation to compute.
l	scalar, in range [0, **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 **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	GxK 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 - meanc(x)';
```

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

Source

`autoreg.src`

Library

`auto`

Purpose

Computes specified autocovariances for each column of a matrix.

Format

a = **autocov(**x,f,l**);**

Input

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

Output

a	GxK matrix, where G $= l - f + 1$, containing the autocovariances of order f, f+1, ..., l for each of the columns of x.

Remarks

The 0^{th} 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 = **rows**(x), to obtain the p^{th} order autocovariance.

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

```
    x = x - meanc(x)';
```

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

Source

`autoreg.src`

Library

`auto`

Purpose

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

Format

{ *coefs*,*vcb*,*phi*,*vcphi*,*sigsq*,*acov*,*acor* } =
        autoreg(*dataset*,*depvar*,*indvars*,*lagvars*,*order***)**

Input

dataset	string, name of **GAUSS** data set. – or – NxK matrix, data
depvar	string, the name of the dependent variable – or – scalar, the index of the dependent variable. If *dataset* is a matrix and if variable names have been provided using **__altnam**, then *depvar* may be a string or character variable containing a variable label.
indvars	Kx1 character vector, names of the independent variables – or – Kx1 numeric vector, indices of the independent variables. *indvars* can include repeated entries of the independent variables and the dependent variable as long as the corresponding entries in *lagvars* are lagged uniquely. If *dataset* is a matrix and if variable names have been provided using **__altnam**, then *indvars* may be a character vector containing variable labels.
lagvars	Kx1 vector, the number of periods to lag the variables in *indvars*. If there are no lagged variables, set to scalar 0. The variables in *indvars* will be lagged the number of periods indicated in the corresponding entries in *lagvars*. *indvars* may contain the dependent variable in one of its columns as long as the corresponding entry in *lagvars* is not 0; also, the independent variables can be repeated if the corresponding entries in *lagvars* are unique.

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

Output

coefs	Kx1 vector, estimated regression coefficients
vcb	KxK matrix, covariance matrix of estimated regression coefficients
phi	Lx1 vector, lag coefficients
vcphi	LxL matrix, covariance matrix of *phi*
sigsq	scalar, variance of white noise error
acov	(L+1)x1 vector, autocovariances
acor	(L+1)x1 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 Kx1 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:

```
__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.

_arrsq	scalar, explained variance.

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
```

■ 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`

Library

`arima`

Purpose

Estimate forecasts using estimation results obtained from arima.

Format

f = tsforecast(b,y,p,d,q,*const*,e,h**);**

Input

b	Kx1 vector, estimated coefficients.
y	Nx1 vector, data.
p	scalar, the autoregressive order.
d	scalar, the order of differencing.
q	scalar, the moving average order.
const	scalar, if 1, a constant is estimated, 0 otherwise.
e	Nx1 vector, residuals reported by arima program.
h	scalar, the number of step-ahead forecasts to compute.

Output

f	hx3 matrix,
	[.,**1**] Lower forecast confidence bounds.
	[.,**2**] Forecasts.
	[.,**3**] Upper forecast confidence bounds.

Globals

_amcritl	scalar, confidence level to compute for forecast confidence bounds. Default = 0.95.
__output	scalar
	0 Nothing is printed.

1 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`

Library

`arima`

Purpose

Computes partial autocorrelations for a univariate time series.

Format

a = pacf(y**,**l**,**d**);**

Input

y	Nx1 vector, data.
l	scalar, number of partial autocorrelations to compute.
d	scalar, order of differencing.

Output

a	lx1 vector, partial autocorrelations.

Source

`tsutil.src`

Library

`arima`

Purpose

Simulate ARMA time series process.

Format

y = simarma(b,p,q,*const*,n,k,*std*,*seed***);**

Input

b	Kx1 vector, coefficient values for theoretical ARMA process.
p	scalar, the autoregressive order.
q	scalar, the moving average order.
const	scalar, value of the constant term. – or – NxM matrix, fixed regressor matrix.
n	scalar, the number of observations to generate.
k	scalar, the number of replications to generate.
std	scalar, the standard deviation of the error process.
seed	scalar, the value of the seed. If *seed* $= 0$, then **rndn** is used, otherwise **rndns** is used.

Output

y	NxK matrix, simulated ARMA process. Each column represents an independent realization of a univariate time series.

Remarks

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 **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, **simarma** should give reasonable results since the starting values are normalized to have required multivariate normal distribution.

Source

`simarma.src`

Library

`arima`

Purpose

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

Format

g = tautocov(b,p,q**);**

Input

b	Kx1 vector, parameter coefficients.
p	scalar, the autoregressive order.
q	scalar, the moving average order.

Output

g	[Max(p,q)+1]x1 vector, theoretical autocovariances.

Remarks

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

Source

`tautocov.src`

■ Library

`tscs`

■ Purpose

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

■ Format

{ *bdv*,*vcdv*,*mdv*,*bec*,*vcec*,*mec* } = **tscs(***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	Kx1 character vector, names of the independent (exogenous) variables – or – Kx1 numeric vector, indices of the independent (exogenous) variables.
grp	string, name of the group variable – or – scalar, index of the group variable.

■ Output

bdv	Kx1 vector, regression coefficients from the dummy effects model (excluding individual-variables regression model).
vcdv	KxK matrix, variance-covariance matrix of the dummy variables regression model.
mdv	(K+1)x(K+1) matrix, moment matrix of the transformed variables (including a constant) from the dummy variables regression model.
bec	Kx1 vector, regression coefficients from the random effects regression model.
vcec	KxK matrix, variance-covariance matrix of the random effects regression model.
mec	(K+1)x(K+1) matrix, moment matrix of the transformed variables (including a constant) from the random effects regression model.

Globals

_tsmodel 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.

Two more globals are used internally by **tscs**, **_ts_mn** and **_ts_ver**.

■ 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;
```

■ Source

`tscs.src`

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`

Index