Econometrics Handout
William M. Boal
(Rev.4/94)

BASICS OF SAS

SAS, short for "Statistical Analysis System," is a widely-used computer package for data management and analysis, written and licensed by the SAS Institute. It is available on a wide variety of computers including mainframes, minicomputers, workstations, and DOS (IBM-compatible) personal computers. This handout summarizes a few very basic features of SAS useful in elementary econometric work. SAS is only one of many statistical packages currently available. (See the article by MacKie-Mason listed in the references below for a comparison of some popular packages.)

SAS is especially good at data manipulation: reading data in a wide variety of formats, merging data sets in elaborate ways, modifying the data, and writing new data sets of virtually unlimited size. However, these capabilities are too complex to be documented in this brief handout. To find out more about SAS, consult the voluminous documentation published by the SAS Institute. The name "SAS" is a trademark of the SAS Institute.

1. SAS jobs

Every SAS job, or program, consists of a sequence of DATA steps and PROC steps. The DATA steps create and modify data sets, while the PROC steps display or analyze the data. A step ends when either another step begins or the end of the job is reached. For readability, most SAS programmers used indentation or skipped lines to divide steps from each other.

Every SAS step consists of a sequence of statements, each ending in a semicolon (;). A statement can spill across several lines, if necessary, and several statements can appear on the same line. SAS ignores case. You can type a SAS program in lower case, upper case, or a combination. In the examples in this handout, for clarity, standard SAS words will be written all-upper case, while user-specified names will be written in lower case. Note, however, that user-specified names must begin with a letter and consist of no more than eight characters. Italics will be used to indicate user-specified phrases.

SAS jobs produce two kinds of output. The SAS *log* is an annotated list of all statements in the job. The annotations report good news (the size of data sets created or analyzed, the number of missing values, etc.) and bad news (syntax errors, problems in reading data, etc.). The SAS *listing* gives the results of SAS PROC steps (estimates, standard errors, etc.). On some computers the *log* and the *listing* are output separately, on other computers together.

A few statements can be used anywhere in a SAS job. These include the following:

Example of SAS DATA step statement	Meaning
TITLE1 'Great Statistical Results';	Prints the phrase "Great Statistical Results" as the first title line at the top of SAS listing output for the current and subsequent PROCs. Phrases can be up to 60 characters long, but must not include a quote mark. To change the title in mid-job, just include another TITLE1 statement.
TITLEn 'May Be the Result of Faked Data';	Prints the phrase "May Be the Result of Faked Data" as the nth title line at the top of SAS listing output for the current and subsequent PROCs. Here n can be an integer from 1 to 10, indicating the row of the title. Changing the title for a particular row deletes all titles beneath it (having higher n).

Example of SAS DATA step statement	Meaning
* Now compute average annual earnings;	Any statement beginning with an asterisk (*) is treated as a comment and ignored by SAS. Good programmers sprinkle their programs liberally with comments, describing what their SAS jobs are supposed to do.
ENDSAS;	Stops SAS processing. Later statements are ignored. This can be handy for debugging.

2. The DATA step

The DATA step begins with a DATA statement, usually followed immediately by statements specifying the source for the input data. The most common ways to specify input data use either the SET statement (for an existing SAS dataset created earlier in the same job) or the pair of statements INFILE and INPUT (for an external file residing on a disk drive). These are usually followed by assignment statements, which create new variables out of the old ones. Here are examples of statements used in the DATA step.

Example SAS DATA step statement	Meaning
DATA mydata;	Begins a DATA step which will create the SAS dataset entitled "mydata."
INFILE mystuff; INPUT myvar1 myvar2 myvar3;	Reads input data from external file "mystuff." (The location of this external file must be described at the beginning of the SAS job in a form which depends on the computer. For example, MVS computers require that "mystuff" be described in a DD statement in the job control language.) The data in this file are arranged in three columns and will be henceforth called "myvar1," "myvar2," and "myvar3."
SET myold;	Reads input data from existing SAS dataset "myold" created earlier in the same SAS job.
myx = myvar1 + myvar2 * myvar3;	Creates a new variable to be called "myx," equal to the sum of myvar1 and the product of myvar2 and myvar3. The standard order of operation applies: multiplication and division precede addition and subtraction. (Statements creating new variables, or assigning new values to old ones, are called "assignment statements." Spaces, shown here for clarity, are not required in assignment statements.)
IF myx LE 23 THEN myy = myvar3;	Creates a new variable to be called "myy," equal to myvar3 if myx is less than or equal to than 23. If myx is greater than 23 then myy is recorded as a missing value in the new dataset. Note that in this example, the clause following "THEN" is an assignment statement.
IF myvar2 GT 5 THEN myz = SQRT(myvar2); ELSE myz = 64;	Creates a new variable to be called "myz," equal to the square root of myvar2 if myvar2 is greater than 5, and 64 otherwise. The new variable will thus have no missing values.

Example SAS DATA step statement	Meaning
IF myvar1 NE 17;	Includes in the new dataset only those observations for which myvar1 is not equal to 17. Other observations are discarded. (This is the so-called "subsetting IF" statement, whose function is very different than the "IF-THEN" statement described above.)

Assignment statements are the most common statements in a DATA step. SAS permits the usual arithmetic operators to be used on the right-hand side (+ for addition, - for subtraction, * for multiplication, / for division) as well as ** for exponentiation. SAS also supplies a huge number of built-in functions. A few very useful ones are given below. The arguments of functions can be algebraic expressions, not just a single variable as shown here, except for the lag functions.

SAS function	Meaning
LOG(myvar)	Natural logarithm (logarithm to base e = 2.71828... Produces missing value if argument is nonpositive.
EXP(myvar)	Exponential function. (Inverse of LOG.)
SQRT(myvar)	Square root. Produces missing value if argument is negative.
LAG1(myvar)	First lag: previous observation of myvar. (Do not use any lag function in an IF or ELSE statement. Strange results may occur.)
LAG5(myvar)	Fifth lag: observation of myvar, five observations back.
PROBNORM(myvar)	$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp(-z^2/2)\, dz$ Cumulative standard normal probability distribution function:

So-called comparison operators are used in "IF-THEN" and "subsetting IF" statements. The most widely-used comparison operators in SAS are the following. All can be written either as two-letter mnemonics or as symbols, but the "not-equal" symbols should be avoided because they vary across computers (use "NE" instead). (For comparison purposes, any missing value is treated as being less than the smallest negative number that SAS can store.)

Two-letter mnemonic	Symbol	Meaning
EQ	=	Equal to.
NE	^= or ~=	Not equal to.
GT	>	Greater than.
LT	<	Less than.

Two-letter mnemonic	Symbol	Meaning
GE	>=	Greater than or equal to.
LE	<=	Less than or equal to.

EXAMPLE:
```
DATA grbgout;
    SET grbgin;
    * This step creates SAS file "grbgout" out of SAS file "grbgin,"
    which contains data on earnings, age, schooling, and gender;
    lwage = LOG(earnings);
    * Have no data on work experience, so impute it using age and
    years of schooling;
    exper = age - school - 6;
    * Create dummy (zero-one) numerical variable for gender,
    called "gendum," using existing character variable "gender";
    IF gender = 'F' THEN gendum = 1;
        ELSE gendum = 0;
```

3. The PROC step

The PROC step begins with a PROC statement, taking the following general form:

PROC GEEWHIZZ DATA=mydata (WHERE=(*condition*)) *options*;

The *options* are, of course, optional, and vary with the particular PROC used. The WHERE expression in parentheses (one of SAS's many "dataset options") modifies the "DATA=" and is not required either. It serves to limit the observations included in the analysis to those which satisfy the *condition*, which might be take a form like "myvar1 LT 4" or "myvar2 EQ myvar3." The "DATA=" expression indicates which SAS dataset is to be used. It is required if a dataset option such as WHERE is used. If no "DATA=" expression is used, then the PROC uses the SAS dataset created most recently.

Of course, "GEEWHIZZ" is not a real SAS PROC. Some useful real PROCs are described below.

4. Some PROCs for displaying data

4.1 PROC PRINT prints all the data in the dataset. If a statement like

VAR myvar1 myvar2;

is included in the PROC, then only the variables myvar1 and myvar2 are printed.

EXAMPLE:
```
PROC PRINT DATA=mydata;
    TITLE1 'Look At These Numbers!';
    VAR wage school exper gender;
```

4.2 PROC MEANS computes and prints descriptive statistics (means, variances, minima, maxima, etc.) for all variables in the dataset. Just as with PROC PRINT, inclusion of a VAR statement limits the computation to the specified variables. This PROC takes very little computer time, and it is a good idea to include it before any more elaborate statistical analysis is performed.

4.3 PROC PLOT produces crude scatter plots on a line printer. It requires at least one PLOT statement (not to be confused with PROC PLOT, which begins the PROC) of the following form:

```
PLOT mvert*myhor;
```

where myvert is the variable plotted on the vertical axis and myhor is the variable on the horizontal axis. Many plots can be requested in a single PLOT statement, as shown by the following examples:

```
PLOT myvert1*myhor1 myvert2*myhor2 myvert3*myhor3;
PLOT myvert1*(myhor1 myhor2);
PLOT (myvert1 myvert2)*(myhor1 myhor2);
```

The first statement requests three plots with three different pairs of variables. The second statement requests two plots, both having myvert1 on the vertical axis. The third statement requests four plots, using alternately myvert1 and myvert2 on the vertical axis, and myhor1 and myhor2 on the horizontal axis.

EXAMPLE:
```
PROC PLOT  DATA=mydata;
     TITLE1 'Scatter Plot of Earnings Against Age';
     TITLE2 'To Give You Something to Look Forward To';
     PLOT earn*age;
```

5. Some PROCs for estimating linear and qualitative models

Many different PROCs exist for estimating linear and qualitative econometric models. Most require at least one model statement of the following form:

```
mylabel: MODEL mydepvar = myvar1 myvar2 myvar3 / options;
```

The label clause ("mylabel:") is optional but useful if more than one model is to be estimated. The label appears at the top of the results listing for that model. The variable to the left of the equal sign is the explained variable, while the variables to the right are the explanatory variables. A constant term is present by default unless excluded by the option "NOINT." The entire estimated variance-covariance matrix of the coefficient estimates is printed if the option "COVB" is present. Many other options are available, specific to the particular PROC. More than one model statement can be included in the same PROC if more than one model is to be estimated on the same data.

Several other optional statements, common to the PROCs described in this section, are shown in the following table.

Example SAS PROC step statement	Meaning
WEIGHT mywtvar;	Specifies a variable "mywtvar" in the data set used to weight each observation. For weighted least-squares, the value of mywtvar should be equal or proportional to the reciprocal of the variance of the error term. (This statement is not available in PROC AUTOREG.)
OUTPUT OUT=newfile P=myfitted R=myresid;	Outputs a new SAS dataset called "newfile," having all the original data, plus the predicted or fitted values (given the name myfitted here) and the residuals (given the name myresid here). Multiple OUTPUT statements are permitted. Each applies only to the preceding MODEL statement. If the "OUT=" clause is omitted, the output dataset replaces and takes the name of the input dataset. (For PROC SYSLIN, the "OUT=" clause appears as an option on the PROC statement, not here.) The "P=" and "R=" clauses are optional. (The "R=" clause is not available in PROC PROBIT.)
TEST myvar2=5;	Computes an F-statistic for the hypothesis that the coefficient of myvar2 equals five. (This statement is not available in PROC AUTOREG or PROC PROBIT.)
TEST myvar2=2*myvar3, 3*INTERCEPT+myvar3=0;	Computes an F-statistic for the joint hypothesis that both: (1) the coefficient of myvar2 equals twice the coefficient of myvar3, and (2) 3 times the intercept, plus the coefficient of myvar3, equals zero. (This statement is not available in PROC AUTOREG or PROC PROBIT.)

5.1 PROC REG computes ordinary least squares regression. Useful options on the MODEL statement for this PROC are shown in the following table.

MODEL statement option for PROC REG	Meaning
CLI	Prints predicted values and 95% confidence intervals for *individual* prediction for all observations in the sample with nonmissing explanatory variables.
CLM	Prints predicted values and 95% confidence intervals for *mean* prediction for all observations in the sample with nonmissing explanatory variables.
DW	Prints Durbin-Watson statistic.
ACOV	Prints White's alternative estimate of the variance-covariance matrix for the coefficients which is consistent in the presence of heteroskedasticity. The square roots of the diagonal elements are White's alternative standard errors.
SPEC	Prints White's test statistic for heteroskedasticity.

EXAMPLE: PROC REG DATA=ohiodata (WHERE=(AGE>25));

```
TITLE1 'Standard Human Capital Equation';
eqwage:  MODEL  lwage = school exper /SPEC;
* Suspect heteroskedasticity here;
```

5.2 PROC GLM is similar to PROC REG, but has some different capabilities, such as the ability to handle discrete regressors represented as character (non-numeric) data, which are automatically interpreted as dummy variables. However, PROC GLM allows only one MODEL statement, so in general PROC REG is preferable for econometric work.

5.3 PROC AUTOREG computes linear regression corrected for serial correlation. Useful options on the MODEL statement for this PROC are shown in the following table.

MODEL statement option for PROC AUTOREG	Meaning
METHOD=ML	Uses the maximum-likelihood method.
METHOD=ULS	Uses the "unconditional least squares" method, that is, nonlinear least squares.
METHOD=YW	Uses the Yule-Walker method, a generalization of the Prais-Winsten method for autocorrelation of arbitrary order. If no METHOD option is present, the Yule-Walker method is used by default.
NLAG=n	Specifies the order of the autocorrelation to be correct for. If n=1 the error term is assumed to be AR(1), while if n=2 the error term is assumed to be AR(2), etc. If this option is not present, then ordinary least squares estimates are computed.
ITER	Specifies that the Yule-Walker estimates are to be iterated until convergence. If this option is not present, then no iteration takes place and two-step estimates are computed. This option is ignored if METHOD=ML or METHOD=ULS is specified.

5.4 PROC SYSLIN computes simultaneous-equation estimates using all equations together. The method to be used is specified as an option on the PROC statement. Some useful options on the PROC statement are shown in the following table.

PROC statement option for PROC SYSLIN	Meaning
OUT=newfile	Outputs a new SAS dataset called "newfile," having all the original data, plus any variables (such as residuals or predicted values) specified in OUTPUT statements later in the PROC.
2SLS	Uses the two-stage least-squares method.
3SLS	Uses the three-stage least-squares method.
LIML	Uses the limited-information maximum-likelihood method.
FIML	Use the full-information maximum-likelihood method.

PROC statement option for PROC SYSLIN	Meaning
SUR	Use Zellner's "seemingly unrelated regressions" method, also called joint generalized least squares.
FIRST	Prints the first-stage regression results for 2SLS or 3SLS.

If no estimation method is selected, this PROC will produce ordinary least squares estimates. If 2SLS, 3SLS, LIML or FIML estimates are desired then two additional statements must be present:

Statements for PROC SYSLIN	Meaning
ENDOGENOUS myend1 myend2;	Lists the endogenous regressors in the model, i.e. the variables which appear on the left-hand side in the first stage.
INSTRUMENTS myins1 myins2;	Lists all the predetermined variables in the model, i.e. the variables which appear on the right-hand side in the first stage.

EXAMPLE: The following statements compute two-stage least-squares estimates for the regression of q on p and y (a demand curve), where p is endogenous, and w and r are additional instruments (factor prices from the supply curve):

```
PROC SYSLIN 2SLS;
        TITLE1 'Estimate of Demand Curve";
        demandeq: MODEL q = p y;
        ENDOGENOUS p;
        INSTRUMENTS y w r;
```

5.5 PROC PROBIT estimates a probit and/or logit model. Three types of statements are most commonly used in this PROC: the CLASS statement, the MODEL statement, and the OUTPUT statement.

The CLASS statement flags categorical data. Like PROC GLM, PROC PROBIT can handle character as well as numeric data. All character variables and the dependent variable (if it is a dummy [zero-one] variable) must first be listed in a CLASS statement.

The MODEL statement, as usual, lists the dependent and independent variables. With micro-data, the dependent data is typically a dummy variable, so it should have already been listed in the preceding CLASS statement. However, PROC PROBIT can also handle aggregate data, where one (integer) variable gives the number of successes and another (integer) variable gives the number of trials. In this case both variables are listed on the left-hand-side of the equal sign, separated by a "/". Useful options on the MODEL statement for this PROC are shown in the following table.

MODEL statement option for PROC PROBIT	Meaning
D=LOGISTIC	Estimates a logit model.
D=NORMAL	Estimates a probit model. If no "D=" option is present, a probit model is estimated by default.

The OUTPUT statement as usual lists the results to be saved and their destination. Three commonly used options are given below. Note that the "P=" option saves the predicted probability, while the XBETA saves the estimated coefficients times the independent variables.

OUTPUT statement option for PROC PROBIT	Meaning
OUT=newfile	Outputs a new SAS dataset called "newfile."
P=probs	Includes predicted probabilities in output file as a new variable called "probs." Warning: if the dependent variable is a dummy (zero-one) variable, the predictions are for the probability that it equals *zero!*
XBETA=z	Includes the inner product of the estimated coefficients and the independent variables in the output file as a new variable called "z."

It should be noted that if the dependent variable is a dummy (zero-one) variable, the PROC PROBIT coefficient estimates correspond to the probability that the dependent variable equals *zero.* That is, the signs are reversed from usual econometric convention.

EXAMPLE 1: Micro-data on individual workers, dummy dependent variable.

```
PROC PROBIT  DATA=people;
        TITLE1 'Probit & Logit Models of Labor Force Participation: Micro-Data';
        CLASS working;
        * Note: the variable 'working' = 1 if working, = 0 otherwise;
        labor1:  working = school health income /D=NORMAL;
        OUTPUT  OUT=myfile  P=ppredict  XBETA=zprobit;
        labor2:  working = school health income /D=LOGISTIC;
        OUTPUT  OUT=myfile  P=lpredict  XBETA=zlogit;
```

EXAMPLE 2: Aggregate data, dependent variables show number of success and number of trials in each observation (here a state).

```
PROC PROBIT  DATA=states;
        TITLE1 'Probit Model of Cancer Deaths: State-Level Data';
        cancer:  deaths/pop = avginc avgage povrate/D=NORMAL;
        * Note:  The variables 'deaths' and 'population' are state totals;
        * The variables 'avginc,' 'avgage,' and 'povrate' are state averages;
```

6. PROC NLIN for nonlinear least squares

PROC NLIN estimates models which are nonlinear in their parameters, using nonlinear least squares.

6.1 Specifying the equation to be estimated: A PROC NLIN step looks very different from a PROC for estimating linear models, like PROC REG. For one thing, the MODEL statement does not merely list the variables. It specifies the model explicitly. The general form of the model statement is:

MODEL mydepvar = *expression*;

Obviously, explicit expressions (not just lists of regressors) are necessary in PROC NLIN, since the functional form cannot be assumed to be linear, as in PROC REG. However, in reading these expressions SAS must distinguish somehow between parameters to be estimated and data variables to be read from the DATA set. To that end, SAS requires a **PARAMETERS** statement to name the parameters, whose typical form is illustrated as follows.

PARAMETERS myparm1 myparm2 myparm3;

Sometimes MODELS can be quite complicated to write explicitly. PROC NLIN therefore permits assignment statements before the PARAMETERS and MODEL statements to define intermediate or temporary variables. Both assignment statements and MODEL statements can use almost any SAS operator or function which is permissible in a DATA step (except lag functions).

6.2 Specifying the computational algorithm: Yet another difference between PROC NLIN and PROC REG reflects the fact finding the estimates is more difficult. Estimates for nonlinear models are solutions to systems of nonlinear equations. These estimates cannot be computed in one step. Instead, they must be found through some iterative technique that gets closer and closer to the exact solution with each iteration. An iteration rule is called an *algorithm*. The algorithm used for computing the estimates is specified by an option on the PROC NLIN statement.

PROC statement option for PROC NLIN	Meaning
METHOD=GAUSS	Uses Gauss-Newton algorithm. If no method is specified, this one is used by default.
METHOD=MARQUARDT	Uses Marquardt algorithm.
METHOD=NEWTON	Uses Newton-Raphson algorithm.
METHOD=GRADIENT	Uses "Steepest descent" algorithm.
METHOD=DUD	Uses the "doesn't use derivatives" algorithm of Ralston and Jennrich. Does not require "DER." statements (see below). Instead, computes estimates of derivatives from previous iterations.

Most computational algorithms use the partial derivatives of the objective function (the sum of squared residuals) with respect to the parameters to find the estimates. (Methods using or approximating these partial derivatives are collectively called "gradient algorithms" and generally converge much faster to

the estimates than alternative methods.) Although some statistical packages can differentiate the objective function automatically, SAS needs a little help. In particular, SAS requires the user to supply partial derivatives of the *model* as "DER." statements, whose general form is as follows.

DER.myparm = *expression*;

There must be one "DER." statement for each parameter to be estimated.

PROC NLIN also permits OUTPUT statements similar to those used in PROC REG."

EXAMPLE: Suppose we wish to estimate the following model by the Newton-Raphson method:

$$y_t = \beta^{x_t} + \varepsilon_t \qquad (1)$$

where y_t and x_t are data and $ is an unknown parameter. The derivative of the model with respect to $ is:

$$\frac{\partial y_t}{\partial \beta} = x_t \beta^{x_t - 1} \qquad (2)$$

(check this). To estimate this model in PROC NLIN we would use the following statements:

```
PROC NLIN  DATA=grbgin  METHOD=NEWTON;
        TITLE1 'Estimates of FNLM';
        TITLE2 '(FNLM = funny non-linear model)';
        PARAMETERS beta;
        MODEL y = beta**x;
        DER.beta = x * beta**(x-1);
        OUTPUT  OUT=grbgout  P=yhat  R=uhat;
```

Alternatively, we could use an assignment statement so that exponentiation is performed only once, saving a little computation time:

```
PROC SYSLIN  DATA=gooddata  METHOD=NEWTON;
        TITLE1 'Estimates of FNLM';
        TITLE2 '(FNLM = funny non-linear model)';
        PARAMETERS beta;
        btothex = beta**x;
        MODEL y = btothex;
        DER.beta = x * btothex/beta;
        OUTPUT  OUT=grbgout  P=yhat  R=uhat;
```

7. PROC ARIMA for Box-Jenkins time series analysis

Box and Jenkins (1976) propose three steps for analysis of data thought to be generated by an ARIMA (auto-regressive integrated moving average) statistical process, namely

(1) identification (also called specification)
(2) estimation
(3) forecasting

Corresponding to these three steps in the Box-Jenkins methodology are three statements used in PROC ARIMA.

7.1 Identification: The IDENTIFY statement computes and plots the sample autocovariances, inverse autocovariances, and partial autocovariances for a series. Typical examples of the IDENTIFY statement are:

```
IDENTIFY VAR=myvar;
IDENTIFY VAR=myvar(1);
IDENTIFY VAR=myvar(1,1);
```

The first statement analyzes the raw data. The second and third statements analyze the first-differenced data and the second-differenced data, respectively. (The "1"s could be replaced by higher integers if seasonal factors were thought to be present.) Typically, an initial computer run using PROC ARIMA includes several IDENTIFY statements using various degrees of differencing to determine the level of differencing required to produce a stationary series. The estimates and plots from the stationary series are then used to make guesses as to the number of autoregressive parameters (p) and the number of moving-average parameters (q).

7.2 Estimation: The ESTIMATE statement then computes estimates of a particular model. Typical examples of the ESTIMATE statement are:

```
ESTIMATE P=1 PLOT;
ESTIMATE Q=2 PLOT;
ESTIMATE P=2 Q=1 PLOT MAXIT=25;
```

The first statement estimates an AR(1) model. The second statement estimates an MA(2) model. The third statement estimates an ARMA(2,1) model. The PLOT option is not required but recommended. It causes plots of the autocorrelations and partial autocorrelations of the residuals to be plotted. These should be close to white noise for a good model. The MAXIT option is also optional, but recommended for models of moderate order (p+q>2). It permits numerical iterations beyond the default number of 15.

The position of the ESTIMATE statement determines the level of differencing. The level of differencing used is simply that of the last preceding IDENTIFY statement. Typically, the second computer run using PROC ARIMA inserts ESTIMATE statements after that IDENTIFY statement which specifies the minimum level of differencing apparently required to produce a stationary series.

In addition to parameter estimates, the ESTIMATE statement computes standard errors, t-statistics, and goodness-of-fit criteria that can help determine the best model.

7.3 Forecasting: The FORECAST statement computes forecasts of the series for as many periods into the future as desired. Typical examples of the FORECAST statement are:

```
FORECAST;
```

```
        FORECAST LEAD=12;
```

The first statement forecasts the default number of periods (24). The second statement forecasts 12 periods. The output always includes within-sample forecasts as well.

The position of the FORECAST statement determines which model is to be used. The model used is simply that of the last preceding ESTIMATE statement. Thus a forecast cannot be produced without all three statements: IDENTIFY, ESTIMATE, and FORECAST. Typically, the third computer run using PROC ARIMA inserts FORECAST statements after the ESTIMATE statements which specify the most promising model.

EXAMPLE: The following example computes autocorrelations and partial autocorrelations for the S&P 500 index of stock prices, estimates AR(1), MA(1), and ARMA(1,1) models for the first-differenced data, and forecasts on the basis of each model. Indentation is used to emphasize the hierarchy of IDENTIFY, ESTIMATE, and FORECAST statements.

```
PROC ARIMA  DATA=indices;
        TITLE1 'Forecasting Project to Get Rich Quick';
        IDENTIFY SP500;
        IDENTIFY SP500 (1);
                ESTIMATE P=1 PLOT;
                        FORECAST LEAD=6;
                ESTIMATE Q=1 PLOT;
                        FORECAST LEAD=6;
                ESTIMATE P=1 Q=1 PLOT;
                        FORECAST LEAD=6;
        IDENTIFY SP500 (1,1);
```

References

Box, George E. P., and Jenkins, Gwilym M., "Time Series Analysis: Forecasting and Control," Oakland, California: Holden-Day, 1976.

Mackie-Mason, Jeffrey K., "Econometric Software: A User's View," *Journal of Economic Perspectives,* Vol. 6, No. 4 (Fall 1992), pp.165-188. [Reviews and compares six statistical packages: LIMDEP, RATS, SAS, SST, STATA, and TSP.]

SAS Institute, *SAS Language: Reference, Version 6, First Edition,* Cary, NC: SAS Institute, 1989. [Documents all statements used in the DATA step, dataset options, and many other detailed features of SAS.]

SAS Institute, *SAS Procedures: Reference, Version 6, First Edition,* Cary, NC: SAS Institute, 1989. [Documents PROC PRINT, PROC MEANS, PROC PLOT, and other nonstatistical PROCs.]

SAS Institute, *SAS/STAT User's Guide, Version 6, Fourth Edition,* Volumes 1 and 2, Cary, NC: SAS Institute, 1989. [Documents PROC REG, PROC GLM, PROC PROBIT and other statistical PROCs.]

SAS Institute, *SAS/ETS User's Guide, Version 6, First Edition,* Cary, NC: SAS Institute, 1989.

[Documents PROC ARIMA, PROC AUTOREG, PROC SYSLIN and other statistical PROCs used by primarily by Econometricians and Time Series analysts.]