ALSCAL

```
ALSCAL  VARIABLES=varlist

 [/FILE=file]  [CONFIG  [({INITIAL**})]]  [ROWCONF [({INITIAL**})]]
                          {FIXED  }                  {FIXED  }

               [COLCONF [({INITIAL**})]]  [SUBJWGHT[({INITIAL**})]]
                          {FIXED  }                  {FIXED  }


               [STIMWGHT[({INITIAL**})]]
                          {FIXED  }

 [/INPUT=ROWS ({ALL**})]
              { n   }

 [/SHAPE={SYMMETRIC**}]
         {ASYMMETRIC }
         {RECTANGULAR}

 [/LEVEL={ORDINAL**  [([UNTIE] [SIMILAR])]}]
         {INTERVAL[({1**})]                 }
         {        {n}                       }
         {RATIO[({1**})]                    }
         {      {n}                         }
         {NOMINAL                           }

 [/CONDITION={MATRIX**     }]
             {ROW          }
             {UNCONDITIONAL}

 [/{MODEL }={EUCLID**}]
   {METHOD} {INDSCAL }
            {ASCAL   }
            {AINDS   }
            {GEMSCAL }

 [/CRITERIA=[NEGATIVE] [CUTOFF({0**})] [CONVERGE({.001})]
                               { n }             { n  }

         [ITER({30})] [STRESSMIN({.005})] [NOULB]
               {n }               { n  }

         [DIMENS({2**      })] [DIRECTIONS(n)]
                 {min[,max]}

         [CONSTRAIN]  [TIESTORE(n)]]

 [/PRINT=[DATA] [HEADER]]    [/PLOT=[DEFAULT] [ALL]]

 [/OUTFILE=file]

 [/MATRIX=IN({file})]
             {*   }
```

**Default if the subcommand or keyword is omitted.

Example:

```
ALSCAL VARIABLES=ATLANTA TO TAMPA.
```

ALSCAL was originally designed and programmed by Forrest W. Young, Yoshio Takane, and Rostyslaw J. Lewyckyj of the Psychometric Laboratory, University of North Carolina.

Overview

ALSCAL uses an alternating least-squares algorithm to perform multidimensional scaling (MDS) and multidimensional unfolding (MDU). You can select one of the five models to obtain stimulus coordinates and/or weights in multidimensional space.

Options

Data Input. You can read inline data matrices, including all types of two- or three-way data, such as a single matrix or a matrix for each of several subjects, using the INPUT subcommand. You can read square (symmetrical or asymmetrical) or rectangular matrices of proximities with the SHAPE subcommand and proximity matrices created by PROXIMITIES and CLUSTER with the MATRIX subcommand. You can also read a file of coordinates and/or weights to provide initial or fixed values for the scaling process with the FILE subcommand.

Methodological Assumptions. You can specify data as matrix-conditional, row-conditional, or unconditional on the CONDITION subcommand. You can treat data as nonmetric (nominal or ordinal) or as metric (interval or ratio) using the LEVEL subcommand. You can also use LEVEL to identify ordinal-level proximity data as measures of similarity or dissimilarity and can specify tied observations as untied (continuous) or leave them tied (discrete).

Model Selection. You can specify most commonly used multidimensional scaling models by selecting the correct combination of ALSCAL subcommands, keywords, and criteria. In addition to the default Euclidean distance model, the MODEL subcommand offers the individual differences (weighted) Euclidean distance model (INDSCAL), the asymmetric Euclidean distance model (ASCAL), the asymmetric individual differences Euclidean distance model (AINDS), and the generalized Euclidean metric individual differences model (GEMSCAL).

Output. You can produce output that includes raw and scaled input data, missing-value patterns, normalized data with means, squared data with additive constants, each subject's scalar product and individual weight space, plots of linear or nonlinear fit, and plots of the data transformations using the PRINT and PLOT subcommands.

Basic Specification

The basic specification is VARIABLES followed by a variable list. By default, ALSCAL produces a two-dimensional nonmetric Euclidean multidimensional scaling solution. Input is assumed to be one or more square symmetric matrices with data elements that are dissimilarities at the ordinal level of measurement. Ties are not untied, and conditionality is by subject. Values less than 0 are treated as missing. The default output includes the improvement in Young's S-stress for successive iterations, two measures of fit for each input matrix (Kruskal's stress and the squared correlation, RSQ), and the derived configurations for each of the dimensions.

Subcommand Order

Subcommands can be named in any order.

Operations

- ALSCAL calculates the number of input matrices by dividing the total number of observations in the data set by the number of rows in each matrix. All matrices must contain the same number of rows. This number is determined by the settings on SHAPE and INPUT (if used). For square matrix data, the number of rows in the matrix equals the number of variables. For rectangular matrix data, it equals the number of rows specified or implied. For additional information, see the INPUT and SHAPE subcommands below.
- ALSCAL ignores user-missing specifications in all variables in the configuration/weights file (see the FILE subcommand on p. 6). The system-missing value is converted to 0.
- With split-file data, ALSCAL reads initial or fixed configurations from the configuration/weights file for each split-file group (see the FILE subcommand on p. 6). If there is only one initial configuration in the file, ALSCAL rereads these initial or fixed values for successive split-file groups.
- By default, ALSCAL estimates upper and lower bounds on missing values in the working data file in order to compute the initial configuration. To prevent this, specify CRITERIA=NOULB. Missing values are always ignored during the iterative process.

Limitations

- Maximum 100 variables on the VARIABLES subcommand.
- Maximum six dimensions can be scaled.
- ALSCAL does not recognize data weights created by the WEIGHT command.
- ALSCAL analyses can include no more than 32,767 values in each of the input matrices. Large analyses may require significant computing time.

Example

```
* Air distances among U.S. cities.
* Data are from Johnson and Wichern (1982), page 563.
DATA LIST
 /ATLANTA BOSTON CINCNATI COLUMBUS DALLAS INDNPLIS
   LITTROCK LOSANGEL MEMPHIS STLOUIS SPOKANE TAMPA 1-60.
BEGIN DATA
   0
1068    0
 461  867    0
 549  769  107    0
 805 1819  943 1050    0
 508  941  108  172  882    0
 505 1494  618  725  325  562    0
2197 3052 2186 2245 1403 2080 1701    0
 366 1355  502  586  464  436  137 1831    0
 558 1178  338  409  645  234  353 1848  294    0
2467 2747 2067 2131 1891 1959 1988 1227 2042 1820    0
 467 1379  928  985 1077  975  912 2480  779 1016 2821   0
END DATA.

ALSCAL VARIABLES=ATLANTA TO TAMPA
 /PLOT.
```

- By default, ALSCAL assumes a symmetric matrix of dissimilarities for ordinal-level variables. Only values below the diagonal are used. The upper triangle can be left blank. The 12 cities form the rows and columns of the matrix.
- The result is a classical MDS analysis that reproduces a map of the United States when the output is rotated to a north-south by east-west orientation.

VARIABLES Subcommand

VARIABLES identifies the columns in the proximity matrix or matrices that ALSCAL reads.

- VARIABLES is required and can name only numeric variables.
- Each matrix must have at least four rows and four columns.

INPUT Subcommand

ALSCAL reads data row by row, with each case in the working data file representing a single row in the data matrix. (VARIABLES specifies the columns.) Use INPUT when reading rectangular data matrices to specify how many rows are in each matrix.

- The specification on INPUT is ROWS. If INPUT is not specified or is specified without ROWS, the default is ROWS(ALL). ALSCAL assumes that each case in the working data file represents one row of a single input matrix, and the result is a square matrix.
- You can specify the number of rows (*n*) in each matrix in parentheses after the keyword ROWS. The number of matrices equals the number of observations divided by the number specified.
- The number specified on ROWS must be at least 4 and must divide evenly into the total number of rows in the data.
- With split-file data, *n* refers to the number of cases in each split-file group. All split-file groups must have the same number of rows.

Example

```
ALSCAL VARIABLES=V1 to V7 /INPUT=ROWS(8).
```

- INPUT indicates that there are eight rows per matrix, with each case in the working data file representing one row.
- The total number of cases must be divisible by 8.

SHAPE Subcommand

Use SHAPE to specify the structure of the input data matrix or matrices.

- You can specify one of the three keywords listed below.
- Both SYMMETRIC and ASYMMETRIC refer to square matrix data.

SYMMETRIC *Symmetric data matrix or matrices.* For a symmetric matrix, ALSCAL looks only at the values below the diagonal. Values on and above the diagonal can be omitted. This is the default.

ASYMMETRIC *Asymmetric data matrix or matrices.* The corresponding values in the upper and lower triangles are not all equal. The diagonal is ignored.

RECTANGULAR *Rectangular data matrix or matrices.* The rows and columns represent different sets of items.

Example

```
ALSCAL VAR=V1 TO V8 /SHAPE=RECTANGULAR.
```

- ALSCAL performs a classical MDU analysis, treating the rows and columns as separate sets of items.

LEVEL Subcommand

LEVEL identifies the level of measurement for the values in the data matrix or matrices. You can specify one of the keywords defined below.

ORDINAL *Ordinal-level data.* This specification is the default. It treats the data as ordinal, using Kruskal's (1964) least-squares monotonic transformation. The analysis is nonmetric. By default, the data are treated as discrete dissimilarities. Ties in the data remain tied throughout the analysis. To change the default, specify UNTIE and/or SIMILAR in parentheses. UNTIE treats the data as continuous and resolves ties in an optimal fashion; SIMILAR treats the data as similarities. UNTIE and SIMILAR cannot be used with the other levels of measurement.

INTERVAL(n) *Interval-level data*. This specification produces a metric analysis of the data using classical regression techniques. You can specify any integer from 1 to 4 in parentheses for the degree of polynomial transformation to be fit to the data. The default is 1.

RATIO(n) *Ratio-level data.* This specification produces a metric analysis. You can specify an integer from 1 to 4 in parentheses for the degree of polynomial transformation. The default is 1.

NOMINAL *Nominal-level data.* This specification treats the data as nominal by using a least-squares categorical transformation (Takane et al., 1977). This option produces a nonmetric analysis of nominal data. It is useful when there are few observed categories, when there are many observations in each category, and when the order of the categories is not known.

Example

```
ALSCAL VAR=ATLANTA TO TAMPA /LEVEL=INTERVAL(2).
```

- This example identifies the distances between U.S. cities as interval-level data. The 2 in parentheses indicates a polynomial transformation with linear and quadratic terms.

CONDITION Subcommand

CONDITION specifies which numbers in a data set are comparable.

MATRIX *Only numbers within each matrix are comparable.* If each matrix represents a different subject, this specification makes comparisons conditional by subject. This is the default.

ROW *Only numbers within the same row are comparable.* This specification is appropriate only for asymmetric or rectangular data. They cannot be used when ASCAL or AINDS is specified on MODEL.

UNCONDITIONAL *All numbers are comparable.* Comparisons can be made among any values in the input matrix or matrices.

Example

```
ALSCAL VAR=V1 TO V8 /SHAPE=RECTANGULAR /CONDITION=ROW.
```

- ALSCAL performs a Euclidean MDU analysis conditional on comparisons within rows.

FILE Subcommand

ALSCAL can read proximity data from the working data file or, with the MATRIX subcommand, from a matrix data file created by PROXIMITIES or CLUSTER. The FILE subcommand reads a file containing additional data: an initial or fixed configuration for the coordinates of the stimuli and/or weights for the matrices being scaled. This file can be created with the OUTFILE subcommand on ALSCAL or with an SPSS input program.

- The minimum specification is the file that contains the configurations and/or weights.
- FILE can include additional specifications that define the structure of the configuration/weights file.
- The variables in the configuration/weights file that correspond to successive ALSCAL dimensions must have the names *DIM1*, *DIM2*,...*DIMr*, where *r* is the maximum number of ALSCAL dimensions. The file must also contain the short string variable *TYPE_* to identify the types of values in all rows.
- Values for the variable *TYPE_* can be CONFIG, ROWCONF, COLCONF, SUBJWGHT, and STIMWGHT, in that order. Each value can be truncated to the first three letters. Stimulus coordinate values are specified as CONFIG; row stimulus coordinates as ROWCONF; column stimulus coordinates as COLCONF; and subject and stimulus weights as SUBJWGHT and STIMWGHT, respectively. ALSCAL accepts CONFIG and ROWCONF interchangeably.
- ALSCAL skips unneeded types as long as they appear in the file in their proper order. Generalized weights (GEM) and flattened subject weights (FLA) cannot be initialized or fixed and will always be skipped. (These weights can be generated by ALSCAL but cannot be used as input.)

The following list summarizes the optional specifications that can be used on FILE to define the structure of the configuration/weights file:

- Each specification can be further identified with option INITIAL or FIXED in parentheses.

- INITIAL is the default. INITIAL indicates that the external configuration or weights are to be used as initial coordinates and are to be modified during each iteration.
- FIXED forces ALSCAL to use the externally defined structure without modification to calculate the best values for all unfixed portions of the structure.

CONFIG *Read stimulus configuration.* The configuration/weights file contains initial stimulus coordinates. Input of this type is appropriate when SHAPE=SYMMETRIC or SHAPE=ASYMMETRIC, or when the number of variables in a matrix equals the number of variables on the ALSCAL command. The value of the *TYPE_* variable must be either CON or ROW for all stimulus coordinates for the configuration.

ROWCONF *Read row stimulus configuration.* The configuration/weights file contains initial row stimulus coordinates. This specification is appropriate if SHAPE=RECTANGULAR and if the number of ROWCONF rows in the matrix equals the number of rows specified on the INPUT subcommand (or, if INPUT is omitted, the number of cases in the working data file). The value of *TYPE_* must be either ROW or CON for the set of coordinates for each row.

COLCONF *Read column stimulus configuration.* The configuration/weights file contains initial column stimulus coordinates. This kind of file can be used only if SHAPE=RECTANGULAR and if the number of COLCONF rows in the matrix equals the number of variables on the ALSCAL command. The value of *TYPE_* must be COL for the set of coordinates for each column.

SUBJWGHT *Read subject (matrix) weights.* The configuration/weights file contains subject weights. The number of observations in a subject-weights matrix must equal the number of matrices in the proximity file. Subject weights can be used only if the model is INDSCAL, AINDS, or GEMSCAL. The value of *TYPE_* for each set of weights must be SUB.

STIMWGHT *Read stimulus weights.* The configuration/weights file contains stimulus weights. The number of observations in the configuration/weights file must equal the number of matrices in the proximity file. Stimulus weights can be used only if the model is AINDS or ASCAL. The value of *TYPE_* for each set of weights must be STI.

If the optional specifications for the configuration/weights file are not specified on FILE, ALSCAL sequentially reads the *TYPE_* values appropriate to the model and shape according to the defaults in Table 1.

Example

```
ALSCAL VAR=V1 TO V8 /FILE=ONE CON(FIXED) STI(INITIAL).
```

- ALSCAL reads the configuration/weights file *ONE*.
- The stimulus coordinates are read as fixed values, and the stimulus weights are read as initial values.

Table 1 Default specifications for the FILE subcommand

Shape	Model	Default specifications
SYMMETRIC	EUCLID	CONFIG (or ROWCONF)
	INDSCAL	CONFIG (or ROWCONF) SUBJWGHT
	GEMSCAL	CONFIG (or ROWCONF) SUBJWGHT
ASYMMETRIC	EUCLID	CONFIG (or ROWCONF)
	INDSCAL	CONFIG (or ROWCONF) SUBJWGHT
	GEMSCAL	CONFIG (or ROWCONF) SUBJWGHT
	ASCAL	CONFIG (or ROWCONF) STIMWGHT
	AINDS	CONFIG (or ROWCONF) SUBJWGHT STIMWGHT
RECTANGULAR	EUCLID	ROWCONF (or CONFIG) COLCONF
	INDSCAL	ROWCONF (or CONFIG) COLCONF SUBJWGHT
	GEMSCAL	ROWCONF (or CONFIG) COLCONF SUBJWGHT

MODEL Subcommand

MODEL (alias METHOD) defines the scaling model for the analysis. The only specification is MODEL (or METHOD) and any one of the five scaling and unfolding model types. EUCLID is the default.

EUCLID *Euclidean distance model.* This model can be used with any type of proximity matrix and is the default.

INDSCAL *Individual differences (weighted) Euclidean distance model.* ALSCAL scales the data using the weighted individual differences Euclidean distance model proposed by Carroll and Chang (1970). This type of analysis can be specified only if the analysis involves more than one data matrix and more than one dimension is specified on CRITERIA.

ASCAL *Asymmetric Euclidean distance model.* This model (Young, 1975) can be used only if SHAPE=ASYMMETRIC and more than one dimension is requested on CRITERIA.

AINDS *Asymmetric individual differences Euclidean distance model.* This option combines Young's (1975) asymmetric Euclidean model with the individual

differences model proposed by Carroll and Chang (1970). This model can be used only when SHAPE=ASYMMETRIC, the analysis involves more than one data matrix, and more than one dimension is specified on CRITERIA.

GEMSCAL *Generalized Euclidean metric individual differences model.* The number of directions for this model is set with the DIRECTIONS option on CRITERIA. The number of directions specified can be equal to but cannot exceed the group space dimensionality. By default, the number of directions equals the number of dimensions in the solution.

Example

```
ALSCAL VARIABLES = V1 TO V6
 /SHAPE = ASYMMETRIC
 /CONDITION = ROW
 /MODEL = GEMSCAL
 /CRITERIA = DIM(4) DIRECTIONS(4).
```

- In this example, the number of directions in the GEMSCAL model is set to 4.

CRITERIA Subcommand

Use CRITERIA to control features of the scaling model and to set convergence criteria for the solution. You can specify one or more of the following:

CONVERGE(n) *Stop iterations if the change in S-stress is less than* n. S-stress is a goodness-of-fit index. By default, *n*=0.001. To increase the precision of a solution, specify a smaller value, for example, 0.0001. To obtain a less precise solution (perhaps to reduce computing time), specify a larger value, for example, 0.05. Negative values are not allowed. If *n*=0, the algorithm will iterate 30 times unless a value is specified with the ITER option.

ITER(n) *Set the maximum number of iterations to* n. The default value is 30. A higher value will give a more precise solution but will take longer to compute.

STRESSMIN(n) *Set the minimum stress value to* n. By default, ALSCAL stops iterating when the value of S-stress is 0.005 or less. STRESSMIN can be assigned any value from 0 to 1.

NEGATIVE *Allow negative weights in individual differences models.* By default, ALSCAL does not permit the weights to be negative. Weighted models include INDSCAL, ASCAL, AINDS, and GEMSCAL. The NEGATIVE option is ignored if the model is EUCLID.

CUTOFF(n) *Set the cutoff value for treating distances as missing to* n. By default, ALSCAL treats all negative similarities (or dissimilarities) as missing, and 0 and positive similarities as nonmissing (*n*=0). Changing the CUTOFF value causes ALSCAL to treat similarities greater than or equal to that value as nonmissing. User- and system-missing values are considered missing regardless of the CUTOFF specification.

NOULB *Do not estimate upper and lower bounds on missing values.* By default, ALSCAL estimates the upper and lower bounds on missing values in order to compute the initial configuration. This specification has no effect during the iterative process, when missing values are ignored.

DIMENS(min[,max]) *Set the minimum and maximum number of dimensions in the scaling solution.* By default, ALSCAL calculates a solution with two dimensions. To obtain solutions for more than two dimensions, specify the minimum and the maximum number of dimensions in parentheses after DIMENS. The minimum and maximum can be integers between 2 and 6. A single value represents both the minimum and the maximum. For example, DIMENS(3) is equivalent to DIMENS(3,3). The minimum number of dimensions can be set to 1 only if MODEL=EUCLID.

DIRECTIONS(n) *Set the number of principal directions in the generalized Euclidean model to* n. This option has no effect for models other than GEMSCAL. The number of principal directions can be any positive integer between 1 and the number of dimensions specified on the DIMENS option. By default, the number of directions equals the number of dimensions.

TIESTORE(n) *Set the amount of storage needed for ties to* n. This option estimates the amount of storage needed to deal with ties in ordinal data. By default, the amount of storage is set to 1000 or the number of cells in a matrix, whichever is smaller. Should this be insufficient, ALSCAL terminates and displays a message that more space is needed.

CONSTRAIN *Constrain multidimensional unfolding solution.* This option can be used to keep the initial constraints throughout the analysis.

PRINT Subcommand

PRINT requests output not available by default. You can specify the following:

DATA *Display input data.* The display includes both the initial data and the scaled data for each subject according to the structure specified on SHAPE.

HEADER *Display a header page.* The header includes the model, output, algorithmic, and data options in effect for the analysis.

- Data options listed by PRINT=HEADER include the number of rows and columns, number of matrices, measurement level, shape of the data matrix, type of data (similarity or dissimilarity), whether ties are tied or untied, conditionality, and data cutoff value.
- Model options listed by PRINT=HEADER are the type of model specified (EUCLID, INDSCAL, ASCAL, AINDS, or GEMSCAL), minimum and maximum dimensionality, and whether or not negative weights are permitted.
- Output options listed by PRINT=HEADER indicate whether the output includes the header page and input data, whether ALSCAL plotted configurations and transformations, whether an output data set was created, and whether initial stimulus coordinates, initial column stimulus coordinates, initial subject weights, and initial stimulus weights were computed.

- Algorithmic options listed by PRINT=HEADER include the maximum number of iterations permitted, the convergence criterion, the maximum S-stress value, whether or not missing data are estimated by upper and lower bounds, and the amount of storage allotted for ties in ordinal data.

Example

```
ALSCAL VAR=ATLANTA TO TAMPA /PRINT=DATA.
```

- In addition to scaled data, ALSCAL will display initial data.

PLOT Subcommand

PLOT controls the display of plots. The minimum specification is simply PLOT to produce the defaults.

DEFAULT *Default plots.* Default plots include plots of stimulus coordinates, matrix weights (if the model is INDSCAL, AINDS, or GEMSCAL), and stimulus weights (if the model is AINDS or ASCAL). The default also includes a scatterplot of the linear fit between the data and the model and, for certain types of data, scatterplots of the nonlinear fit and the data transformation. If the SET command specifies HIGHRES=ON, ALSCAL sends all stimulus dimensions to the graphic editor and shows a 3-D plot with the first three dimensions if the solution has three or more dimensions. If HIGHRES=OFF, ALSCAL generates $d*(d-1)/2$ pages of plots for the stimulus space, where d is the number of dimensions in the solution. When appropriate, the same is true for the weight space.

ALL *Transformation plots in addition to the default plots.* SPSS produces a separate plot for each subject if CONDITION=MATRIX and a separate plot for each row if CONDITION=ROW. For interval and ratio data, PLOT=ALL has the same effect as PLOT=DEFAULT. This option can generate voluminous output, particularly when CONDITION=ROW.

Example

```
ALSCAL VAR=V1 TO V8 /INPUT=ROWS(8) /PLOT=ALL.
```

- This command produces all the default plots (the number may be different depending on the setting of HIGHRES). It also produces a separate plot for each subject's data transformation and a plot of *V1* through *V8* in a two-dimensional space for each subject.

OUTFILE Subcommand

OUTFILE saves coordinate and weight matrices to an SPSS data file. The only specification is a name for the output file.

- The output data file has an alphanumeric (short string) variable named *TYPE_* that identifies the kind of values in each row, a numeric variable *DIMENS* that specifies the number of dimensions, a numeric variable *MATNUM* that indicates the subject (matrix) to which each set of coordinates corresponds, and variables *DIM1*, *DIM2*,...*DIMn* that correspond to the *n* dimensions in the model.
- The values of any split-file variables are also included in the output file.
- The file created by OUTFILE can be used by subsequent ALSCAL commands as initial data.

The following are the types of configurations and weights that can be included in the output file:

CONFIG *Stimulus configuration coordinates*.

ROWCONF *Row stimulus configuration coordinates*.

COLCONF *Column stimulus configuration coordinates*.

SUBJWGHT *Subject (matrix) weights*.

FLATWGHT *Flattened subject (matrix) weights*.

GEMWGHT *Generalized weights*.

STIMWGHT *Stimulus weights*.

Only the first three characters of each identifier are written to variable *TYPE_* in the file. For example, CONFIG becomes CON. The structure of the file is determined by the SHAPE and MODEL subcommands, as shown in Table 2.

Table 2 Types of configurations and/or weights in output files

Shape	Model	TYPE_
SYMMETRIC	EUCLID	CON
	INDSCAL	CON SUB FLA
	GEMSCAL	CON SUB FLA GEM
ASYMMETRIC	EUCLID	CON
	INDSCAL	CON SUB FLA
	GEMSCAL	CON SUB FLA GEM
	ASCAL	CON STI
	AINDS	CON SUB FLA STI
RECTANGULAR	EUCLID	ROW COL
	INDSCAL	ROW COL SUB FLA
	GEMSCAL	ROW COL SUB FLA GEM

Example

```
ALSCAL VAR=ATLANTA TO TAMPA /OUTFILE=ONE.
```

- OUTFILE creates the SPSS configuration/weights file *ONE* from the example of air distances between cities.

MATRIX Subcommand

MATRIX reads SPSS matrix data files. It can read a matrix written by either PROXIMITIES or CLUSTER.

- Generally, data read by ALSCAL are already in matrix form. If the matrix materials are in the working data file, you do not need to use MATRIX to read them. Simply use the VARIABLES subcommand to indicate the variables (or columns) to be used. However, if the matrix materials are not in the working data file, MATRIX must be used to specify the matrix data file that contains the matrix.
- The proximity matrices ALSCAL reads have *ROWTYPE_* values of PROX. No additional statistics should be included with these matrix materials.
- ALSCAL ignores unrecognized *ROWTYPE_* values in the matrix file. In addition, it ignores variables present in the matrix file that are not specified on the VARIABLES subcommand in ALSCAL. The order of rows and columns in the matrix is unimportant.
- Since ALSCAL does not support case labeling, it ignores values for the *ID* variable (if present) in a CLUSTER or PROXIMITIES matrix.
- If split-file processing was in effect when the matrix was written, the same split file must be in effect when ALSCAL reads that matrix.
- The specification on MATRIX is the keyword IN and the matrix file in parentheses.
- MATRIX=IN cannot be used unless a working data file has already been defined. To read an existing matrix data file at the beginning of a session, first use GET to retrieve the matrix file and then specify IN(*) on MATRIX.

IN (filename) *Read a matrix data file.* If the matrix data file is the working data file, specify an asterisk in parentheses (*). If the matrix data file is another file, specify the filename in parentheses. A matrix file read from an external file does not replace the working data file.

Example

```
PROXIMITIES V1 TO V8 /ID=NAMEVAR /MATRIX=OUT(*).
ALSCAL VAR=CASE1 TO CASE10 /MATRIX=IN(*).
```

- PROXIMITIES uses *V1* through *V8* in the working data file to generate a matrix file of Euclidean distances between each pair of cases based on the eight variables. The number of rows and columns in the resulting matrix equals the number of cases. MATRIX=OUT then replaces the working data file with this new matrix data file.
- MATRIX=IN on ALSCAL reads the matrix data file, which is the new working data file. In this instance, MATRIX is optional because the matrix materials are in the working data file.
- If there were 10 cases in the original working data file, ALSCAL performs a multidimensional scaling analysis in two dimensions on *CASE1* through *CASE10*.

Example

```
GET FILE PROXMTX.
ALSCAL VAR=CASE1 TO CASE10 /MATRIX=IN(*).
```

- GET retrieves the matrix data file *PROXMTX*.

- MATRIX=IN specifies an asterisk because the working data file is the matrix. MATRIX is optional, however, since the matrix materials are in the working data file.

Example

```
GET FILE PRSNNL.
FREQUENCIES VARIABLE=AGE.
ALSCAL VAR=CASE1 TO CASE10 /MATRIX=IN(PROXMTX).
```

- This example performs a frequencies analysis on file *PRSNNL* and then uses a different file containing matrix data for ALSCAL. The file is an existing matrix data file.
- MATRIX=IN is required because the matrix data file, *PROXMTX*, is not the working data file. *PROXMTX* does not replace *PRSNNL* as the working data file.

Specification of Analyses

Table 3 summarizes the analyses that can be performed for the major types of proximity matrices you can use with ALSCAL, Table 4 lists the specifications needed to produce these analyses for nonmetric models, and Table 5 lists the specifications for metric models. You can include additional specifications to control the precision of your analysis with CRITERIA.

Table 3 Models for types of matrix input

Matrix mode	Matrix form	Model class	Single matrix	Replications of single matrix	Two or more individual matrices
Object by object	Symmetric	Multidimensional scaling	CMDS Classical multidimensional scaling	RMDS Replicated multidimensional scaling	WMDS(INDSCAL) Weighted multidimensional scaling
	Asymmetric single process	Multidimensional scaling	CMDS(row conditional) Classical row conditional multidimensional scaling	RMDS(row conditional) Replicated row conditional multidimensional scaling	WMDS(row conditional) Weighted row conditional multidimensional scaling
	Asymmetric multiple process	Internal asymmetric multidimensional scaling	CAMDS Classical asymmetric multidimensional scaling	RAMDS Replicated asymmetric multidimensional scaling	WAMDS Weighted asymmetric multidimensional scaling
		External asymmetric multidimensional scaling	CAMDS(external) Classical external asymmetric multidimensional scaling	RAMDS(external) Replicated external asymmetric multidimensional scaling	WAMDS(external) Weighted external asymmetric multidimensional scaling
Object by attribute	Rectangular	Internal unfolding	CMDU Classical internal multidimensional unfolding	RMDU Replicated internal multidimensional unfolding	WMDU Weighted internal multidimensional unfolding
		External unfolding	CMDU(external) Classical external multidimensional unfolding	RMDU(external) Replicated external multidimensional unfolding	WMDU(external) Weighted external multidimensional unfolding

Table 4 ALSCAL specifications for nonmetric models

Matrix mode	Matrix form	Model class	Single matrix	Replications of single matrix	Two or more individual matrices
Object by object	Symmetric	Multi-dimensional scaling	ALSCAL VAR= varlist.	ALSCAL VAR= varlist.	ALSCAL VAR= varlist /MODEL=INDSCAL.
	Asymmetric single process	Multi-dimensional scaling	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /CONDITION=ROW.	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /CONDITION=ROW.	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /CONDITION=ROW /MODEL=INDSCAL.
	Asymmetric multiple process	Internal asymmetric multi-dimensional scaling	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /MODEL=ASCAL.	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /MODEL=ASCAL.	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /MODEL=AINDS.
		External asymmetric multi-dimensional scaling	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /MODEL=ASCAL /FILE=file COLCONF(FIX).	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /MODEL=ASCAL /FILE=file COLCONF(FIX).	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /MODEL=AINDS /FILE=file COLCONF(FIX).
Object by attribute	Rectangular	Internal unfolding	ALSCAL VAR= varlist /SHAPE=REC /INP=ROWS /CONDITION=ROW.	ALSCAL VAR= varlist /SHAPE=REC /INP=ROWS /CONDITION(ROW).	ALSCAL VAR= varlist /SHAPE=REC /INP=ROWS /CONDITION=ROW /MODEL=INDSCAL.
		External unfolding	ALSCAL VAR= varlist /SHAPE=REC /INP=ROWS /CONDITION=ROW /FILE=file ROWCONF(FIX).	ALSCAL VAR= varlist /SHAPE=REC /INP=ROWS /CONDITION=ROW /FILE=file ROWCONF(FIX).	ALSCAL VAR= varlist /SHAPE=REC /INP=ROWS /CONDITION=ROW /FILE=file ROWCONF(FIX) /MODEL=INDSCAL.

Table 5 ALSCAL specifications for metric models

Matrix mode	Matrix form	Model class	Single matrix	Replications of single matrix	Two or more individual matrices
Object by object	Symmetric	Multi-dimensional scaling	ALSCAL VAR= varlist /LEVEL=INT.	ALSCAL VAR= varlist /LEVEL=INT.	ALSCAL VAR= varlist /LEVEL=INT /MODEL=INDSCAL.
	Asymmetric single process	Multi-dimensional scaling	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /CONDITION=ROW /LEVEL=INT.	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /CONDITION=ROW /LEVEL=INT.	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /CONDITION=ROW /LEVEL=INT /MODEL=INDSCAL.
	Asymmetric multiple process	Internal asymmetric multi-dimensional scaling	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /LEVEL=INT /MODEL=ASCAL.	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /LEVEL=INT /MODEL=ASCAL.	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /LEVEL=INT /MODEL=AINDS.
		External asymmetric multi-dimensional scaling	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /LEVEL=INT /MODEL=ASCAL /FILE=file COLCONF(FIX).	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /LEVEL=INT /MODEL=ASCAL /FILE=file COLCONF(FIX).	ALSCAL VAR= varlist /SHAPE=ASYMMETRIC /LEVEL=INT /MODEL=AINDS /FILE=file COLCONF(FIX).
Object by attribute	Rectangular	Internal unfolding	ALSCAL VAR= varlist /SHAPE=REC /INP=ROWS /CONDITION=ROW /LEVEL=INT.	ALSCAL VAR= varlist /SHAPE=REC /INP=ROWS /CONDITION=ROW /LEVEL=INT.	ALSCAL VAR= varlist /SHAPE=REC /INP=ROWS /CONDITION=ROW /LEVEL=INT /MODEL=INDSCAL.
		External unfolding	ALSCAL VAR= varlist /SHAPE=REC /INP=ROWS /CONDITION=ROW /LEVEL=INT /FILE=file ROWCONF(FIX).	ALSCAL VAR= varlist /SHAPE=REC /INP=ROWS /CONDITION=ROW /LEVEL=INT /FILE=file ROWCONF(FIX).	ALSCAL VAR= varlist /SHAPE=REC /INP=ROWS /CONDITION=ROW /LEVEL=INT /FILE=file ROWCONF(FIX) /MODEL=INDSCAL.

LOGISTIC REGRESSION

```
LOGISTIC REGRESSION [VARIABLES =] dependent var
        [WITH independent varlist [BY var [BY var] ... ]]

[/CATEGORICAL = var1, var2, ... ]

[/CONTRAST (categorical var) = [{INDICATOR [(refcat)]    }]]
                                {DEVIATION [(refcat)]    }
                                {SIMPLE [(refcat)]       }
                                {DIFFERENCE              }
                                {HELMERT                 }
                                {REPEATED                }
                                {POLYNOMIAL[({1,2,3...})]}
                                           {metric  }
                                {SPECIAL (matrix)        }

 [/METHOD = {ENTER**       }   [{ALL    }]]
            {BSTEP [{COND}]}    {varlist}
                   {LR  }
                   {WALD}
            {FSTEP [{COND}]}
                   {LR  }
                   {WALD}

 [/SELECT = {ALL**                 }]
            {varname relation value}

 [/{NOORIGIN**}]
   {ORIGIN    }

 [/ID = [variable]]

 [/PRINT = [DEFAULT**] [SUMMARY] [CORR] [ALL] [ITER [({1})]] [GOODFIT]]
                                                    {n}
           [CI(level)]

 [/CRITERIA = [BCON ({0.001**})] [ITERATE({20**})] [LCON({0.01**})]
                     {value  }            {n   }          {value }

              [PIN({0.05**})] [POUT({0.10**})] [EPS({.00000001**})]]
                   {value }         {value }        {value      }

              [CUT[{0.5**  }]]
                   [value  }

 [/CLASSPLOT]

 [/MISSING = {EXCLUDE **}]
             {INCLUDE   }

 [/CASEWISE = [tempvarlist]  [OUTLIER({2**  })]]
                                      {value}

 [/SAVE = tempvar[(newname)] tempvar[(newname)]...]

 [/EXTERNAL]
```

** Default if the subcommand or keyword is omitted.

Temporary variables created by LOGISTIC REGRESSION are:

PRED	*LEVER*	*COOK*
PGROUP	*LRESID*	*DFBETA*
RESID	*SRESID*	
DEV	*ZRESID*	

Example:

```
LOGISTIC REGRESSION PROMOTED WITH AGE, JOBTIME, JOBRATE.
```

Overview

LOGISTIC REGRESSION regresses a dichotomous dependent variable on a set of independent variables (Aldrich and Nelson, 1984; Fox, 1984; Hosmer and Lemeshow, 1989; McCullagh and Nelder, 1989; Agresti, 1990). Categorical independent variables are replaced by sets of contrast variables, each set entering and leaving the model in a single step.

Options

Processing of Independent Variables. You can specify which independent variables are categorical in nature on the CATEGORICAL subcommand. You can control treatment of categorical independent variables by the CONTRAST subcommand. Seven methods are available for entering independent variables into the model. You can specify any one of them on the METHOD subcommand. You can also use the keyword BY between variable names to enter interaction terms.

Selecting Cases. You can use the SELECT subcommand to define subsets of cases to be used in estimating a model.

Regression through the Origin. You can use the ORIGIN subcommand to exclude a constant term from a model.

Specifying Termination and Model-Building Criteria. You can further control computations when building the model by specifying criteria on the CRITERIA subcommand.

Adding New Variables to the Working Data File. You can save the residuals, predicted values, and diagnostics generated by LOGISTIC REGRESSION in the working data file.

Output. You can use the PRINT subcommand to print optional output, use the CASEWISE subcommand to request analysis of residuals, and use the ID subcommand to specify a variable whose values or value labels identify cases in output. You can request plots of the actual and predicted values for each case with the CLASSPLOT subcommand.

Basic Specification

- The minimum specification is the VARIABLES subcommand with one dichotomous dependent variable. You must specify a list of independent variables either following the keyword WITH on the VARIABLES subcommand or on a METHOD subcommand.

- The default output includes goodness-of-fit tests for the model (–2 log-likelihood, goodness-of-fit statistic, Cox and Snell R^2, and NagelKerke R^2) and a classification table for the predicted and observed group memberships. The regression coefficient, standard error of the regression coefficient, Wald statistic and its significance level, and a multiple correlation coefficient adjusted for the number of parameters (Atkinson, 1980) are displayed for each variable in the equation.

Subcommand Order

- Subcommands can be named in any order. If the VARIABLES subcommand is not specified first, a slash (/) must precede it.
- The ordering of METHOD subcommands determines the order in which models are estimated. Different sequences may result in different models.

Syntax Rules

- Only one dependent variable can be specified for each LOGISTIC REGRESSION.
- Any number of independent variables may be listed. The dependent variable may not appear on this list.
- The independent variable list is required if any of the METHOD subcommands are used without a variable list or if the METHOD subcommand is not used. The keyword TO cannot be used on any variable list.
- If you specify the keyword WITH on the VARIABLES subcommand, all independent variables must be listed.
- If the keyword WITH is used on the VARIABLES subcommand, interaction terms do not have to be specified on the variable list, but the individual variables that make up the interactions must be listed.
- Multiple METHOD subcommands are allowed.
- The minimum truncation for this command is LOGI REG.

Operations

- Independent variables specified on the CATEGORICAL subcommand are replaced by sets of contrast variables. In stepwise analyses, the set of contrast variables associated with a categorical variable is entered or removed from the model as a single step.
- Independent variables are screened to detect and eliminate redundancies.
- If the linearly dependent variable is one of a set of contrast variables, the set will be reduced by the redundant variable or variables. A warning will be issued, and the reduced set will be used.
- For the forward stepwise method, redundancy checking is done when a variable is to be entered into the model.
- When backward stepwise or direct-entry methods are requested, all variables for each METHOD subcommand are checked for redundancy before that analysis begins.

Limitations

- The dependent variable must be dichotomous for each split-file group. Specifying a dependent variable with more or less than two nonmissing values per split-file group will result in an error.

Example

```
LOGISTIC REGRESSION PASS WITH GPA, MAT, GRE.
```

- *PASS* is specified as the dependent variable.
- *GPA*, *MAT*, and *GRE* are specified as independent variables.
- LOGISTIC REGRESSION produces the default output for the logistic regression of *PASS* on *GPA*, *MAT*, and *GRE*.

VARIABLES Subcommand

VARIABLES specifies the dependent variable and, optionally, all independent variables in the model. The dependent variable appears first on the list and is separated from the independent variables by the keyword WITH.

- One VARIABLES subcommand is allowed for each Logistic Regression procedure.
- The dependent variable must be dichotomous—that is, it must have exactly two values other than system-missing and user-missing values for each split-file group.
- The dependent variable may be a string variable if its two values can be differentiated by their first eight characters.
- You can indicate an interaction term on the variable list by using the keyword BY to separate the individual variables.
- If all METHOD subcommands are accompanied by independent variable lists, the keyword WITH and the list of independent variables may be omitted.
- If the keyword WITH is used, *all* independent variables must be specified. For interaction terms, only the individual variable names that make up the interaction (for example, `X1, X2`) need to be specified. Specifying the actual interaction term (for example, `X1 BY X2`) on the VARIABLES subcommand is optional if you specify it on a METHOD subcommand.

Example

```
LOGISTIC REGRESSION PROMOTED WITH AGE,JOBTIME,JOBRATE,
    AGE BY JOBTIME.
```

- *PROMOTED* is specified as the dependent variable.
- *AGE*, *JOBTIME*, *JOBRATE*, and the interaction *AGE* by *JOBTIME* are specified as the independent variables.
- Because no METHOD is specified, all three single independent variables and the interaction term are entered into the model.
- LOGISTIC REGRESSION produces the default output.

CATEGORICAL Subcommand

CATEGORICAL identifies independent variables that are nominal or ordinal. Variables that are declared to be categorical are automatically transformed to a set of contrast variables as specified on the CONTRAST subcommand. If a variable coded as $0 – 1$ is declared as categorical, its coding scheme will be changed to deviation contrasts by default.

- Independent variables not specified on CATEGORICAL are assumed to be at least interval level, except for string variables.
- Any variable specified on CATEGORICAL is ignored if it does not appear either after WITH on the VARIABLES subcommand or on any METHOD subcommand.
- Variables specified on CATEGORICAL are replaced by sets of contrast variables. If the categorical variable has *n* distinct values, there will be $n – 1$ contrast variables generated. The set of contrast variables associated with a categorical variable is entered or removed from the model as a step.
- If any one of the variables in an interaction term is specified on CATEGORICAL, the interaction term is replaced by contrast variables.
- All string variables are categorical. Only the first eight characters of each value of a string variable are used in distinguishing between values. Thus, if two values of a string variable are identical for the first eight characters, the values are treated as though they were the same.

Example

```
LOGISTIC REGRESSION PASS WITH GPA, GRE, MAT, CLASS, TEACHER
 /CATEGORICAL = CLASS,TEACHER.
```

- The dichotomous dependent variable *PASS* is regressed on the interval-level independent variables *GPA*, *GRE*, and *MAT* and the categorical variables *CLASS* and *TEACHER*.

CONTRAST Subcommand

CONTRAST specifies the type of contrast used for categorical independent variables. The interpretation of the regression coefficients for categorical variables depends on the contrasts used. The default is INDICATOR. The categorical independent variable is specified in parentheses following CONTRAST. The closing parenthesis is followed by one of the contrast-type keywords.

- If the categorical variable has *n* values, there will be $n – 1$ rows in the contrast matrix. Each contrast matrix is treated as a set of independent variables in the analysis.
- Only one categorical independent variable can be specified per CONTRAST subcommand, but multiple CONTRAST subcommands can be specified.

The following contrast types are available. See Finn (1974) and Kirk (1982) for further information on a specific type. For illustration of contrast types, see the appendix "Categorical Variable Coding Schemes" in *SPSS Professional Statistics 7.5*.

INDICATOR(refcat) *Indicator variables.* Contrasts indicate the presence or absence of category membership. By default, refcat is the last category (represented in the contrast matrix as a row of zeros). To omit a category other than

the last, specify the sequence number of the omitted category (which is not necessarily the same as its value) in parentheses after the keyword INDICATOR.

DEVIATION(refcat) *Deviations from the overall effect.* This is the default. The effect for each category of the independent variable except one is compared to the overall effect. Refcat is the category for which parameter estimates are not displayed (they must be calculated from the others). By default, refcat is the last category. To omit a category other than the last, specify the sequence number of the omitted category (which is not necessarily the same as its value) in parentheses after the keyword DEVIATION.

SIMPLE(refcat) *Each category of the independent variable except the last is compared to the last category.* To use a category other than the last as the omitted reference category, specify its sequence number (which is not necessarily the same as its value) in parentheses following the keyword SIMPLE.

DIFFERENCE *Difference or reverse Helmert contrasts.* The effects for each category of the independent variable except the first are compared to the mean effects of the previous categories.

HELMERT *Helmert contrasts.* The effects for each category of the independent variable except the last are compared to the mean effects of subsequent categories.

POLYNOMIAL(metric) *Polynomial contrasts.* The first degree of freedom contains the linear effect across the categories of the independent variable, the second contains the quadratic effect, and so on. By default, the categories are assumed to be equally spaced; unequal spacing can be specified by entering a metric consisting of one integer for each category of the independent variable in parentheses after the keyword POLYNOMIAL. For example, `CONTRAST(STIMULUS)=POLYNOMIAL(1,2,4)` indicates that the three levels of STIMULUS are actually in the proportion 1:2:4. The default metric is always (1,2,...,*k*), where *k* categories are involved. Only the relative differences between the terms of the metric matter: (1,2,4) is the same metric as (2,3,5) or (20,30,50) because the difference between the second and third numbers is twice the difference between the first and second in each instance.

REPEATED *Comparison of adjacent categories.* Each category of the independent variable except the first is compared to the previous category.

SPECIAL(matrix) *A user-defined contrast.* After this keyword, a matrix is entered in parentheses with $k - 1$ rows and *k* columns (where *k* is the number of categories of the independent variable). The rows of the contrast matrix contain the special contrasts indicating the desired comparisons between categories. If the special contrasts are linear combinations of each other, LOGISTIC REGRESSION reports the linear dependency and stops processing. If *k* rows are entered, the first row is discarded and only the last $k - 1$ rows are used as the contrast matrix in the analysis.

Example

```
LOGISTIC REGRESSION PASS WITH GRE, CLASS
 /CATEGORICAL = CLASS
 /CONTRAST(CLASS)=HELMERT.
```

- A logistic regression analysis of the dependent variable *PASS* is performed on the interval independent variable *GRE* and the categorical independent variable *CLASS*.
- *PASS* is a dichotomous variable representing course pass/fail status and *CLASS* identifies whether a student is in one of three classrooms. A HELMERT contrast is requested.

Example

```
LOGISTIC REGRESSION PASS WITH GRE, CLASS
 /CATEGORICAL = CLASS
 /CONTRAST(CLASS)=SPECIAL(2 -1 -1
                          0  1 -1).
```

- In this example, the contrasts are specified with the keyword SPECIAL.

METHOD Subcommand

METHOD indicates how the independent variables enter the model. The specification is the METHOD subcommand followed by a single method keyword. The keyword METHOD can be omitted. Optionally, specify the independent variables and interactions for which the method is to be used. Use the keyword BY between variable names of an interaction term.

- If no variable list is specified or if the keyword ALL is used, all of the independent variables following the keyword WITH on the VARIABLES subcommand are eligible for inclusion in the model.
- If no METHOD subcommand is specified, the default method is ENTER.
- Variables specified on CATEGORICAL are replaced by sets of contrast variables. The set of contrast variables associated with a categorical variable is entered or removed from the model as a single step.
- Any number of METHOD subcommands can appear in a Logistic Regression procedure. METHOD subcommands are processed in the order in which they are specified. Each method starts with the results from the previous method. If BSTEP is used, all remaining eligible variables are entered at the first step. All variables are then eligible for entry and removal unless they have been excluded from the METHOD variable list.
- The beginning model for the first METHOD subcommand is either the constant variable (by default or if NOORIGIN is specified) or an empty model (if ORIGIN is specified).

The available METHOD keywords are:

ENTER *Forced entry.* All variables are entered in a single step. This is the default if the METHOD subcommand is omitted.

FSTEP *Forward stepwise.* The variables (or interaction terms) specified on FSTEP are tested for entry into the model one by one, based on the significance level of the score statistic. The variable with the smallest significance less than PIN is entered into the model. After each entry, variables that are already in the model are tested for possible removal, based on the significance of the conditional statistic, the Wald sta-

tistic, or the likelihood-ratio criterion. The variable with the largest probability greater than the specified POUT value is removed and the model is reestimated. Variables in the model are then evaluated again for removal. Once no more variables satisfy the removal criterion, covariates not in the model are evaluated for entry. Model building stops when no more variables meet entry or removal criteria, or when the current model is the same as a previous one.

BSTEP *Backward stepwise.* As a first step, the variables (or interaction terms) specified on BSTEP are entered into the model together and are tested for removal one by one. Stepwise removal and entry then follow the same process as described for FSTEP until no more variables meet entry or removal criteria, or when the current model is the same as a previous one.

The statistic used in the test for removal can be specified by an additional keyword in parentheses following FSTEP or BSTEP. If FSTEP or BSTEP is specified by itself, the default is COND.

COND *Conditional statistic.* This is the default if FSTEP or BSTEP is specified by itself.

WALD *Wald statistic.* The removal of a variable from the model is based on the significance of the Wald statistic.

LR *Likelihood ratio.* The removal of a variable from the model is based on the significance of the change in the log-likelihood. If LR is specified, the model must be reestimated without each of the variables in the model. This can substantially increase computational time. However, the likelihood-ratio statistic is the best criterion for deciding which variables are to be removed.

Example

```
LOGISTIC REGRESSION PROMOTED WITH AGE JOBTIME JOBRATE RACE SEX AGENCY
 /CATEGORICAL RACE SEX AGENCY
 /METHOD ENTER AGE  JOBTIME
 /METHOD BSTEP (LR) RACE SEX JOBRATE AGENCY.
```

- *AGE*, *JOBTIME*, *JOBRATE*, *RACE*, *SEX*, and *AGENCY* are specified as independent variables. *RACE*, *SEX*, and *AGENCY* are specified as categorical independent variables.
- The first METHOD subcommand enters *AGE* and *JOBTIME* into the model.
- Variables in the model at the termination of the first METHOD subcommand are included in the model at the beginning of the second METHOD subcommand.
- The second METHOD subcommand adds the variables *RACE*, *SEX*, *JOBRATE*, and *AGENCY* to the previous model.
- Backward stepwise logistic regression analysis is then done with only the variables on the BSTEP variable list tested for removal using the LR statistic.
- The procedure continues until all variables from the BSTEP variable list have been removed or the removal of a variable will not result in a decrease in the log-likelihood with a probability larger than POUT.

SELECT Subcommand

By default, all cases in the working data file are considered for inclusion in LOGISTIC REGRESSION. Use the optional SELECT subcommand to include a subset of cases in the analysis.

- The specification is either a logical expression or keyword ALL. ALL is the default. Variables named on VARIABLES, CATEGORICAL, or METHOD subcommands cannot appear on SELECT.
- In the logical expression on SELECT, the relation can be EQ, NE, LT, LE, GT, or GE. The variable must be numeric and the value can be any number.
- Only cases for which the logical expression on SELECT is true are included in calculations. All other cases, including those with missing values for the variable named on SELECT, are unselected.
- Diagnostic statistics and classification statistics are reported for both selected and unselected cases.
- Cases deleted from the working data file with the SELECT IF or SAMPLE command are not included among either the selected or unselected cases.

Example

```
LOGISTIC REGRESSION VARIABLES=GRADE WITH GPA,TUCE,PSI
 /SELECT SEX EQ 1 /CASEWISE=RESID.
```

- Only cases with the value 1 for *SEX* are included in the logistic regression analysis.
- Residual values generated by CASEWISE are displayed for both selected and unselected cases.

ORIGIN and NOORIGIN Subcommands

ORIGIN and NOORIGIN control whether or not the constant is included. NOORIGIN (the default) includes a constant term (intercept) in all equations. ORIGIN suppresses the constant term and requests regression through the origin. (NOCONST can be used as an alias for ORIGIN.)

- The only specification is either ORIGIN or NOORIGIN.
- ORIGIN or NOORIGIN can be specified only once per Logistic Regression procedure, and it affects all METHOD subcommands.

Example

```
LOGISTIC REGRESSION VARIABLES=PASS WITH GPA,GRE,MAT /ORIGIN.
```

- ORIGIN suppresses the automatic generation of a constant term.

ID Subcommand

ID specifies a variable whose values or value labels identify the casewise listing. By default, cases are labeled by their case number.

- The only specification is the name of a single variable that exists in the working data file.

- Only the first eight characters of the variable's value labels are used to label cases. If the variable has no value labels, the values are used.
- Only the first eight characters of a string variable are used to label cases.

PRINT Subcommand

PRINT controls the display of optional output. If PRINT is omitted, DEFAULT output (defined below) is displayed.

- The minimum specification is PRINT followed by a single keyword.
- If PRINT is used, only the requested output is displayed.

DEFAULT *Goodness-of-fit tests for the model, classification tables, and statistics for the variables in and not in the equation at each step.* Tables and statistics are displayed for each split file and METHOD subcommand.

SUMMARY *Summary information.* Same output as DEFAULT, except that the output for each step is not displayed.

CORR *Correlation matrix of parameter estimates for the variables in the model.*

ITER(value) *Iterations at which parameter estimates are to be displayed.* The value in parentheses controls the spacing of iteration reports. If the value is *n*, the parameter estimates are displayed for every *n*th iteration starting at 0. If a value is not supplied, intermediate estimates are displayed at each iteration.

GOODFIT *Hosmer-Lemeshow goodness-of-fit statistic* (Hosmer and Lemeshow, 1989).

CI(level) *Confidence interval for exp(B).* The value in parentheses must be an integer between 1 and 99.

ALL *All available output.*

Example

```
LOGISTIC REGRESSION VARIABLES=PASS WITH GPA,GRE,MAT
 /METHOD FSTEP
 /PRINT CORR SUMMARY ITER(2).
```

- A forward stepwise logistic regression analysis of *PASS* on *GPA*, *GRE*, and *MAT* is specified.
- The PRINT subcommand requests the display of the correlation matrix of parameter estimates for the variables in the model (CORR), classification tables and statistics for the variables in and not in the equation for the final model (SUMMARY), and parameter estimates at every second iteration (ITER(2)).

CRITERIA Subcommand

CRITERIA controls the statistical criteria used in building the logistic regression models. The way in which these criteria are used depends on the method specified on the METHOD sub-

command. The default criteria are noted in the description of each keyword below. Iterations will stop if the criterion for BCON, LCON, or ITERATE is satisfied.

BCON(value) *Change in parameter estimates to terminate iteration.* Iteration terminates when the parameters change by less than the specified value. The default is 0.001. To eliminate this criterion, specify a value of 0.

ITERATE *Maximum number of iterations.* The default is 20.

LCON(value) *Percentage change in the log-likelihood ratio for termination of iterations.* If the log-likelihood decreases by less than the specified value, iteration terminates. The default is 0.01. To eliminate this criterion, specify a value of 0.

PIN(value) *Probability of score statistic for variable entry.* The default is 0.05. The larger the specified probability, the easier it is for a variable to enter the model.

POUT(value) *Probability of conditional, Wald, or LR statistic to remove a variable.* The default is 0.1. The larger the specified probability, the easier it is for a variable to remain in the model.

EPS(value) *Epsilon value used for redundancy checking.* The specified value must be less than or equal to 0.05 and greater than or equal to 10^{-12}. The default is 10^{-8}. Larger values make it harder for variables to pass the redundancy check—that is, they are more likely to be removed from the analysis.

CUT(value) *Cutoff value for classification.* A case is assigned to a group when the predicted event probability is greater than or equal to the cutoff value. The cutoff value affects the value of the dichotomous derived variable in the classification table, the predicted group (PGROUP on CASEWISE), and the classification plot (CLASSPLOT). The default cutoff value is 0.5. You can specify a value between 0 and 1 ($0 < \text{value} < 1$).

Example

```
LOGISTIC REGRESSION PROMOTED WITH AGE JOBTIME RACE
 /CATEGORICAL RACE
 /METHOD BSTEP
 /CRITERIA BCON(0.01) PIN(0.01) POUT(0.05).
```

- A backward stepwise logistic regression analysis is performed for the dependent variable *PROMOTED* and the independent variables *AGE*, *JOBTIME*, and *RACE*.
- CRITERIA alters four of the statistical criteria that control the building of a model.
- BCON specifies that if the change in the absolute value of all of the parameter estimates is less than 0.01, the iterative estimation process should stop. Larger values lower the number of iterations required. Notice that the ITER and LCON criteria remain unchanged and that if either of them is met before BCON, iterations will terminate. (LCON can be set to 0 if only BCON and ITER are to be used.)
- POUT requires that the probability of the statistic used to test whether a variable should remain in the model be smaller than 0.05. This is more stringent than the default value of 0.1.
- PIN requires that the probability of the score statistic used to test whether a variable should be included be smaller than 0.01. This makes it more difficult for variables to be included in the model than the default value of 0.05.

CLASSPLOT Subcommand

CLASSPLOT generates a classification plot of the actual and predicted values of the dichotomous dependent variable at each step.

- Keyword CLASSPLOT is the only specification.
- If CLASSPLOT is not specified, plots are not generated.

Example

```
LOGISTIC REGRESSION PROMOTED WITH JOBTIME RACE
 /CATEGORICAL RACE
 /CLASSPLOT.
```

- A logistic regression model is constructed for the dichotomous dependent variable *PROMOTED* and the independent variables *JOBTIME* and *RACE*.
- CLASSPLOT produces a classification plot for the dependent variable *PROMOTED*. The vertical axis of the plot is the frequency of the variable *PROMOTED*. The horizontal axis is the predicted probability of membership in the second of the two levels of *PROMOTED*.

CASEWISE Subcommand

CASEWISE produces a casewise listing of the values of the temporary variables created by LOGISTIC REGRESSION.

The following keywords are available for specifying temporary variables (see Fox, 1984). When CASEWISE is specified by itself, the default lists *PRED*, *PGROUP*, *RESID*, and *ZRESID*. If a list of variable names is given, only those named temporary variables are displayed.

PRED *Predicted probability.* For each case, the predicted probability of having the second of the two values of the dichotomous dependent variable.

PGROUP *Predicted group.* The group to which a case is assigned based on the predicted probability.

RESID *Difference between observed and predicted probabilities.*

DEV *Deviance values.* For each case, a log-likelihood-ratio statistic, which measures how well the model fits the case, is computed.

LRESID *Logit residual.* Residual divided by the product of *PRED* and 1–šš*PRED*.

SRESID *Studentized residual.*

ZRESID *Normalized residual.* Residual divided by the square root of the product of *PRED* and 1–šš*PRED*.

LEVER *Leverage value.* A measure of the relative influence of each observation on the model's fit.

COOK *Analog of Cook's influence statistic.*

DFBETA *Difference in beta.* The difference in the estimated coefficients for each independent variable if the case is omitted.

The following keyword is available for restricting the cases to be displayed, based on the absolute value of *SRESID*:

OUTLIER (value) *Cases with absolute values of SRESID greater than or equal to the specified value are displayed.* If OUTLIER is specified with no value, the default is 2.

Example

```
LOGISTIC REGRESSION PROMOTED WITH JOBTIME SEX RACE
 /CATEGORICAL SEX RACE
 /METHOD ENTER
 /CASEWISE SRESID LEVER DFBETA.
```

- CASEWISE produces a casewise listing of the temporary variables *SRESID*, *LEVER*, and *DFBETA*.
- There will be one *DFBETA* value for each parameter in the model. The continuous variable *JOBTIME*, the two-level categorical variable *SEX*, and the constant each require one parameter while the four-level categorical variable *RACE* requires three parameters. Thus, six values of *DFBETA* will be produced for each case.

MISSING Subcommand

LOGISTIC REGRESSION excludes all cases with missing values on any of the independent variables. For a case with a missing value on the dependent variable, predicted values are calculated if it has nonmissing values on all independent variables. The MISSING subcommand controls the processing of user-missing values. If the subcommand is not specified, the default is EXCLUDE.

EXCLUDE *Delete cases with user-missing values as well as system-missing values.* This is the default.

INCLUDE *Include user-missing values in the analysis.*

SAVE Subcommand

SAVE saves the temporary variables created by LOGISTIC REGRESSION. To specify variable names for the new variables, assign the new names in parentheses following each temporary variable name. If new variable names are not specified, LOGISTIC REGRESSION generates default names.

- Assigned variable names must be unique in the working data file. Scratch or system variable names (that is, names that begin with # or $) cannot be used.
- A temporary variable can be saved only once on the same SAVE subcommand.

Example

```
LOGISTIC REGRESSION PROMOTED WITH JOBTIME AGE
 /SAVE PRED (PREDPRO) DFBETA (DF).
```

- A logistic regression analysis of *PROMOTED* on the independent variables *JOBTIME* and *AGE* is performed.
- SAVE adds four variables to the working data file: one variable named *PREDPRO*, containing the predicted value from the specified model for each case, and three variables named *DF0*, *DF1*, and *DF2*, containing, respectively, the *DFBETA* values for each case of the constant, the independent variable *JOBTIME*, and the independent variable *AGE*.

EXTERNAL Subcommand

EXTERNAL indicates that the data for each split-file group should be held in an external scratch file during processing. This can help conserve memory resources when running complex analyses or analyses with large data sets.

- The keyword EXTERNAL is the only specification.
- Specifying EXTERNAL may result in slightly longer processing time.
- If EXTERNAL is not specified, all data are held internally and no scratch file is written.

NLR

```
MODEL PROGRAM parameter=value [parameter=value ...]
transformation commands

[DERIVATIVES
transformation commands]

[CLEAR MODEL PROGRAMS]
```

Procedure CNLR (Constrained Nonlinear Regression):

```
[CONSTRAINED FUNCTIONS
transformation commands]

CNLR dependent var

 [/FILE=file]   [/OUTFILE=file]

 [/PRED=varname]

 [/SAVE [PRED] [RESID[(varname)]] [DERIVATIVES] [LOSS]]

 [/CRITERIA=[ITER n] [MITER n] [CKDER {0.5**}]
                                      {n    }
            [ISTEP {1E+20**}] [FPR n] [LFTOL n]
                   {n      }
            [LSTOL n] [STEPLIMIT {2**}] [NFTOL n]
                                 {n  }
            [FTOL n] [OPTOL n] [CRSHTOL {.01**}]]
                                        {n    }

 [/BOUNDS=expression, expression, ...]

 [/LOSS=varname]

 [/BOOTSTRAP [=n]]
```

Procedure NLR (Nonlinear Regression):

```
NLR dependent var

 [/FILE=file]   [/OUTFILE=file]

 [/PRED=varname]

 [/SAVE [PRED] [RESID [(varname)] [DERIVATIVES]]

 [/CRITERIA=[ITER {100**}] [CKDER {0.5**}]
                  {n    }        {n    }
            [SSCON {1E-8**}]  [PCON {1E-8**}]  [RCON {1E-8**}]]
                   {n     }         {n     }         {n     }
```

**Default if the subcommand or keyword is omitted.

Example:

```
MODEL PROGRAM A=.6.
COMPUTE PRED=EXP(A*X).

NLR Y.
```

Overview

Nonlinear regression is used to estimate parameter values and regression statistics for models that are not linear in their parameters. SPSS has two procedures for estimating nonlinear equations. CNLR (constrained nonlinear regression), which uses a sequential quadratic programming algorithm, is applicable for both constrained and unconstrained problems. NLR (nonlinear regression), which uses a Levenberg-Marquardt algorithm, is applicable only for unconstrained problems.

CNLR is more general. It allows linear and nonlinear constraints on any combination of parameters. It will estimate parameters by minimizing any smooth loss function (objective function), and can optionally compute bootstrap estimates of parameter standard errors and correlations. The individual bootstrap parameter estimates can optionally be saved in a separate SPSS data file.

Both programs estimate the values of the parameters for the model and, optionally, compute and save predicted values, residuals, and derivatives. Final parameter estimates can be saved in an SPSS data file and used in subsequent analyses.

CNLR and NLR use much of the same syntax. Some of the following sections discuss features common to both procedures. In these sections, the notation [C]NLR means that either the CNLR or NLR procedure can be specified. Sections that apply only to CNLR or only to NLR are clearly identified.

Options

The Model. You can use any number of transformation commands under MODEL PROGRAM to define complex models.

Derivatives. You can use any number of transformation commands under DERIVATIVES to supply derivatives.

Adding Variables to Working Data File. You can add predicted values, residuals, and derivatives to the working data file with the SAVE subcommand.

Writing Parameter Estimates to a New Data File. You can save final parameter estimates as an external SPSS data file using the OUTFILE subcommand and retrieve them in subsequent analyses using the FILE subcommand.

Controlling Model-Building Criteria. You can control the iteration process used in the regression with the CRITERIA subcommand.

Additional CNLR Controls. For CNLR, you can impose linear and nonlinear constraints on the parameters with the BOUNDS subcommand. Using the LOSS subcommand, you can specify a loss function for CNLR to minimize and, using the BOOTSTRAP subcommand, you can provide bootstrap estimates of the parameter standard errors, confidence intervals, and correlations.

Basic Specification

The basic specification requires three commands: MODEL PROGRAM, COMPUTE (or any other computational transformation command), and [C]NLR.

- The MODEL PROGRAM command assigns initial values to the parameters and signifies the beginning of the model program.
- The computational transformation command generates a new variable to define the model. The variable can take any legitimate name, but if the name is not *PRED*, the PRED subcommand will be required.
- The [C]NLR command provides the regression specifications. The minimum specification is the dependent variable.
- By default, the residual sum of squares and estimated values of the model parameters are displayed for each iteration. Statistics generated include regression and residual sums of squares and mean squares, corrected and uncorrected total sums of squares, R^2, parameter estimates with their asymptotic standard errors and 95% confidence intervals, and an asymptotic correlation matrix of the parameter estimates.

Command Order

- The model program, beginning with the MODEL PROGRAM command, must precede the [C]NLR command.
- The derivatives program (when used), beginning with the DERIVATIVES command, must follow the model program but precede the [C]NLR command.
- The constrained functions program (when used), beginning with the CONSTRAINED FUNCTIONS command, must immediately precede the CNLR command. The constrained functions program cannot be used with the NLR command.
- The CNLR command must follow the block of transformations for the model program and the derivatives program when specified; the CNLR command must also follow the constrained functions program when specified.
- Subcommands on [C]NLR can be named in any order.

Syntax Rules

- The FILE, OUTFILE, PRED, and SAVE subcommands work the same way for both CNLR and NLR.
- The CRITERIA subcommand is used by both CNLR and NLR, but iteration criteria are different. Therefore, the CRITERIA subcommand is documented separately for CNLR and NLR.
- The BOUNDS, LOSS, and BOOTSTRAP subcommands can be used only with CNLR. They cannot be used with NLR.

Operations

- By default, the predicted values, residuals, and derivatives are created as temporary variables. To save these variables, use the SAVE subcommand.

Weighting Cases

- If case weighting is in effect, [C]NLR uses case weights when calculating the residual sum of squares and derivatives. However, the degrees of freedom in the ANOVA table are always based on unweighted cases.
- When the model program is first invoked for each case, the weight variable's value is set equal to its value in the working data file. The model program may recalculate that value. For example, to effect a robust estimation, the model program may recalculate the weight variable's value as an inverse function of the residual magnitude. [C]NLR uses the weight variable's value after the model program is executed.

Missing Values

Cases with missing values for any of the dependent or independent variables named on the [C]NLR command are excluded.

- Predicted values, but not residuals, can be calculated for cases with missing values on the dependent variable.
- [C]NLR ignores cases that have missing, negative, or zero weights. The procedure displays a warning message if it encounters any negative or zero weights at any time during its execution.
- If a variable used in the model program or the derivatives program is omitted from the independent variable list on the [C]NLR command, the predicted value and some or all of the derivatives may be missing for every case. If this happens, SPSS generates an error message.

Example

```
MODEL PROGRAM A=.5 B=1.6.
COMPUTE PRED=A*SPEED**B.

DERIVATIVES.
COMPUTE D.A=SPEED**B.
COMPUTE D.B=A*LN(SPEED)*SPEED**B.

NLR STOP.
```

- MODEL PROGRAM assigns values to the model parameters *A* and *B*.
- COMPUTE generates the variable *PRED* to define the nonlinear model using parameters *A* and *B* and the variable *SPEED* from the working data file. Because this variable is named *PRED*, the PRED subcommand is not required on NLR.
- DERIVATIVES indicates that calculations for derivatives are being supplied.

- The two COMPUTE statements on the DERIVATIVES transformations list calculate the derivatives for the parameters *A* and *B*. If either one had been omitted, NLR would have calculated it numerically.
- NLR specifies *STOP* as the dependent variable. It is not necessary to specify *SPEED* as the independent variable since it has been used in the model and derivatives programs.

MODEL PROGRAM Command

The MODEL PROGRAM command assigns initial values to the parameters and signifies the beginning of the model program. The model program specifies the nonlinear equation chosen to model the data. There is no default model.

- The model program is required and must precede the [C]NLR command.
- The MODEL PROGRAM command must specify all parameters in the model program. Each parameter must be individually named. Keyword TO is not allowed.
- Parameters can be assigned any acceptable SPSS variable name. However, if you intend to write the final parameter estimates to a file with the OUTFILE subcommand, do not use the name *SSE* or *NCASES* (see the OUTFILE subcommand on p. 39).
- Each parameter in the model program must have an assigned value. The value can be specified on MODEL PROGRAM or read from an existing parameter data file named on the FILE subcommand.
- Zero should be avoided as an initial value because it provides no information on the scale of the parameters. This is especially true for CNLR.
- The model program must include at least one command that uses the parameters and the independent variables (or preceding transformations of these) to calculate the predicted value of the dependent variable. This predicted value defines the nonlinear model. There is no default model.
- By default, the program assumes that *PRED* is the name assigned to the variable for the predicted values. If you use a different variable name in the model program, you must supply the name on the PRED subcommand (see the PRED subcommand on p. 40).
- In the model program, you can assign a label to the variable holding predicted values and also change its print and write formats, but you should not specify missing values for this variable.
- You can use any computational commands (such as COMPUTE, IF, DO IF, LOOP, END LOOP, END IF, RECODE, or COUNT) or output commands (WRITE, PRINT, or XSAVE) in the model program, but you cannot use input commands (such as DATA LIST, GET, MATCH FILES, or ADD FILES).
- Transformations in the model program are used only by [C]NLR, and they do not affect the working data file. The parameters created by the model program do not become a part of the working data file. Permanent transformations should be specified before the model program.

Caution

The selection of good initial values for the parameters in the model program is very important to the operation of [C]NLR. The selection of poor initial values can result in no solution, a local rather than a general solution, or a physically impossible solution.

Example

```
MODEL PROGRAM A=10 B=1 C=5 D=1.
COMPUTE PRED= A*exp(B*X) + C*exp(D*X).
```

- The MODEL PROGRAM command assigns starting values to the four parameters *A, B, C,* and *D*.
- COMPUTE defines the model to be fit as the sum of two exponentials.

DERIVATIVES Command

The optional DERIVATIVES command signifies the beginning of the derivatives program. The derivatives program contains transformation statements for computing some or all of the derivatives of the model. The derivatives program must follow the model program but precede the [C]NLR command.

If the derivatives program is not used, [C]NLR numerically estimates derivatives for all the parameters. Providing derivatives reduces computation time and, in some situations, may result in a better solution.

- The DERIVATIVES command has no further specifications but must be followed by the set of transformation statements that calculate the derivatives.
- You can use any computational commands (such as COMPUTE, IF, DO IF, LOOP, END LOOP, END IF, RECODE, or COUNT) or output commands (WRITE, PRINT, or XSAVE) in the derivatives program, but you cannot use input commands (such as DATA LIST, GET, MATCH FILES, or ADD FILES).
- To name the derivatives, specify the prefix *D.* before each parameter name. For example, the derivative name for the parameter *PARM1* must be *D.PARM1*.
- Once a derivative has been calculated by a transformation, the variable for that derivative can be used in subsequent transformations.
- You do not need to supply all of the derivatives. Those that are not supplied will be estimated by the program. During the first iteration of the nonlinear estimation procedure, derivatives calculated in the derivatives program are compared with numerically calculated derivatives. This serves as a check on the supplied values (see the CRITERIA subcommand on p. 42).
- Transformations in the derivatives program are used by [C]NLR only and do not affect the working data file.
- For NLR, the derivative of each parameter must be computed with respect to the predicted function. (For computation of derivatives in CNLR, see the LOSS subcommand on p. 46.)

Example

```
MODEL PROGRAM A=1, B=0, C=1, D=0
COMPUTE PRED = Ae^Bx + Ce^Dx
DERIVATIVES.
COMPUTE D.A = exp (B * X).
COMPUTE D.B = A * exp (B * X) * X.
COMPUTE D.C = exp (D * X).
COMPUTE D.D = C * exp (D * X) * X.
```

- The derivatives program specifies derivatives of the PRED function for the sum of the two exponentials in the model described by the following equation:

$$Y = Ae^{Bx} + Ce^{Dx}$$

Example

```
DERIVATIVES.
COMPUTE D.A = exp (B * X).
COMPUTE D.B = A * X * D.A.
COMPUTE D.C = exp (D * X).
COMPUTE D.D = C * X * D.C.
```

- This is an alternative way to express the same derivatives program specified in the previous example.

CONSTRAINED FUNCTIONS Command

The optional CONSTRAINED FUNCTIONS command signifies the beginning of the constrained functions program, which specifies nonlinear constraints. The constrained functions program is specified after the model program and the derivatives program (when used). It can only be used with, and must precede, the CNLR command. For more information, see the BOUNDS subcommand on p. 45.

Example

```
MODEL PROGRAM A=.5 B=1.6.
COMPUTE PRED=A*SPEED**B.

CONSTRAINED FUNCTIONS.
COMPUTE CF=A-EXP(B).

CNLR STOP
  /BOUNDS CF LE 0.
```

CLEAR MODEL PROGRAMS Command

CLEAR MODEL PROGRAMS deletes all transformations associated with the model program, the derivative program, and/or the constrained functions program previously submitted. It is primarily used in interactive mode to remove temporary variables created by these programs without affecting the working data file or variables created by other transformation programs or temporary programs. It allows you to specify new models, derivatives, or constrained functions without having to run [C]NLR.

It is not necessary to use this command if you have already executed the [C]NLR procedure. Temporary variables associated with the procedure are automatically deleted.

CNLR/NLR Command

Either the CNLR or the NLR command is required to specify the dependent and independent variables for the nonlinear regression.

- For either CNLR or NLR, the minimum specification is a dependent variable.
- Only one dependent variable can be specified. It must be a numeric variable in the working data file and cannot be a variable generated by the model or the derivatives program.

OUTFILE Subcommand

OUTFILE stores final parameter estimates for use on a subsequent [C]NLR command. The only specification on OUTFILE is the target file. Some or all of the values from this file can be read into a subsequent [C]NLR procedure with the FILE subcommand. The parameter data file created by OUTFILE stores the following variables:

- All of the split-file variables. OUTFILE writes one case of values for each split-file group in the working data file.
- All of the parameters named on the MODEL PROGRAM command.
- The labels, formats, and missing values of the split-file variables and parameters defined for them previous to their use in the [C]NLR procedure.
- The sum of squared residuals (named *SSE*). *SSE* has no labels or missing values. The print and write format for *SSE* is F10.8.
- The number of cases on which the analysis was based (named *NCASES*). *NCASES* has no labels or missing values. The print and write format for *NCASES* is F8.0.

When OUTFILE is used, the model program cannot create variables named *SSE* or *NCASES*.

Example

```
MODEL PROGRAM A=.5 B=1.6.
COMPUTE PRED=A*SPEED**B.
NLR STOP /OUTFILE=PARAM.
```

- OUTFILE generates a parameter data file containing one case for four variables: *A*, *B*, *SSE*, and *NCASES*.

FILE Subcommand

FILE reads starting values for the parameters from a parameter data file created by an OUTFILE subcommand from a previous [C]NLR procedure. When starting values are read from a file, they do not have to be specified on the MODEL PROGRAM command. Rather, the MODEL PROGRAM command simply names the parameters that correspond to the parameters in the data file.

- The only specification on FILE is the file that contains the starting values.

- Some new parameters may be specified for the model on the MODEL PROGRAM command while others are read from the file specified on the FILE subcommand.
- You do not have to name the parameters on MODEL PROGRAM in the order in which they occur in the parameter data file. In addition, you can name a partial list of the variables contained in the file.
- If the starting value for a parameter is specified on MODEL PROGRAM, the specification overrides the value read from the parameter data file.
- If split-file processing is in effect, the starting values for the first subfile are taken from the first case of the parameter data file. Subfiles are matched with cases in order until the starting value file runs out of cases. All subsequent subfiles use the starting values for the last case.
- To read starting values from a parameter data file and then replace those values with the final results from [C]NLR, specify the same file on the FILE and OUTFILE subcommands. The input file is read completely before anything is written in the output file.

Example

```
MODEL PROGRAM A B C=1 D=3.
COMPUTE PRED=A*SPEED**B + C*SPEED**D.
NLR STOP /FILE=PARAM /OUTFILE=PARAM.
```

- MODEL PROGRAM names four of the parameters used to calculate *PRED*, but assigns values to only *C* and *D*. The values of *A* and *B* are read from the existing data file *PARAM*.
- After NLR computes the final estimates of the four parameters, OUTFILE writes over the old input file. If, in addition to these new final estimates, the former starting values of *A* and *B* are still desired, specify a different file on the OUTFILE subcommand.

PRED Subcommand

PRED identifies the variable holding the predicted values.

- The only specification is a variable name, which must be identical to the variable name used to calculate predicted values in the model program.
- If the model program names the variable *PRED*, the PRED subcommand can be omitted. Otherwise, the PRED subcommand is required.
- The variable for predicted values is not saved in the working data file unless the SAVE subcommand is used.

Example

```
MODEL PROGRAM A=.5 B=1.6.
COMPUTE PSTOP=A*SPEED**B.
NLR STOP /PRED=PSTOP.
```

- COMPUTE in the model program creates a variable named *PSTOP* to temporarily store the predicted values for the dependent variable *STOP*.
- PRED identifies *PSTOP* as the variable used to define the model for the NLR procedure.

SAVE Subcommand

SAVE is used to save the temporary variables for the predicted values, residuals, and derivatives created by the model and the derivatives programs.

- The minimum specification is a single keyword.
- The variables to be saved must have unique names on the working data file. If a naming conflict exists, the variables are not saved.
- Temporary variables, for example, variables created after a TEMPORARY command and parameters specified by the model program, are not saved in the working data file. They will not cause naming conflicts.

The following keywords are available and can be used in any combination and in any order. The new variables are always appended to the working data file in the order in which these keywords are presented here:

PRED *Save the predicted values.* The variable's name, label, and formats are those specified for it (or assigned by default) in the model program.

RESID [(varname)] *Save the residuals variable.* You can specify a variable name in parentheses following the keyword. If no variable name is specified, the name of this variable is the same as the specification you use for this keyword. For example, if you use the three-character abbreviation RES, the default variable name will be *RES*. The variable has the same print and write format as the predicted values variable created by the model program. It has no variable label and no user-defined missing values. It is system-missing for any case in which either the dependent variable is missing or the predicted value cannot be computed.

DERIVATIVES *Save the derivative variables.* The derivative variables are named with the prefix *D.* to the first six characters of the parameter names. Derivative variables use the print and write formats of the predicted values variable and have no value labels or user-missing values. Derivative variables are saved in the same order as the parameters named on MODEL PROGRAM. Derivatives are saved for all parameters, whether or not the derivative was supplied in the derivatives program.

LOSS *Save the user-specified loss function variable.* This specification is available only with CNLR and only if the LOSS subcommand has been specified.

Asymptotic standard errors of predicted values and residuals, and special residuals used for outlier detection and influential case analysis are not provided by the [C]NLR procedure. However, for a squared loss function, the asymptotically correct values for all these statistics can be calculated using the SAVE subcommand with [C]NLR and then using the REGRESSION procedure. In REGRESSION, the dependent variable is still the same, and derivatives of the model parameters are used as independent variables. Casewise plots, standard errors of prediction, partial regression plots, and other diagnostics of the regression are valid for the nonlinear model.

Example

```
MODEL PROGRAM A=.5 B=1.6.
COMPUTE PSTOP=A*SPEED**B.
NLR STOP /PRED=PSTOP
  /SAVE=RESID(RSTOP) DERIVATIVES PRED.
REGRESSION VARIABLES=STOP D.A D.B /ORIGIN
  /DEPENDENT=STOP /ENTER D.A D.B /RESIDUALS.
```

- The SAVE subcommand creates the residuals variable *RSTOP* and the derivative variables *D.A* and *D.B*.
- Because the PRED subcommand identifies *PSTOP* as the variable for predicted values in the nonlinear model, keyword PRED on SAVE adds the variable *PSTOP* to the working data file.
- The new variables are added to the working data file in the following order: *PSTOP, RSTOP, D.A,* and *D.B*.
- The subcommand RESIDUALS for REGRESSION produces the default analysis of residuals.

CRITERIA Subcommand

CRITERIA controls the values of the cutoff points used to stop the iterative calculations in [C]NLR.

- The minimum specification is any of the criteria keywords and an appropriate value. The value can be specified in parentheses after an equals sign, a space, or a comma. Multiple keywords can be specified in any order. Defaults are in effect for keywords not specified.
- Keywords available for CRITERIA differ between CNLR and NLR and are discussed separately. However, with both CNLR and NLR, you can specify the critical value for derivative checking.

Checking Derivatives for CNLR and NLR

Upon entering the first iteration, [C]NLR always checks any derivatives calculated on the derivatives program by comparing them with numerically calculated derivatives. For each comparison, it computes an agreement score. A score of 1 indicates agreement to machine precision; a score of 0 indicates definite disagreement. If a score is less than 1, either an incorrect derivative was supplied or there were numerical problems in estimating the derivative. The lower the score, the more likely it is that the supplied derivatives are incorrect. Highly correlated parameters may cause disagreement even when a correct derivative is supplied. Be sure to check the derivatives if the agreement score is not 1.

During the first iteration, [C]NLR checks each derivative score. If any score is below 1, it begins displaying a table to show the worst (lowest) score for each derivative. If any score is below the critical value, the program stops.

To specify the critical value, use the following keyword on CRITERIA:

CKDER n *Critical value for derivative checking.* Specify a number between 0 and 1 for *n*. The default is 0.5. Specify 0 to disable this criterion.

Iteration Criteria for CNLR

The CNLR procedure uses NPSOL (Version 4.0) Fortran Package for Nonlinear Programming (Gill et al., 1986). The CRITERIA subcommand of CNLR gives the control features of NPSOL. The following section summarizes the NPSOL documentation.

CNLR uses a sequential quadratic programming algorithm, with a quadratic programming subproblem to determine the search direction. If constraints or bounds are specified, the first step is to find a point that is feasible with respect to those constraints. Each major iteration sets up a quadratic program to find the search direction, *p*. Minor iterations are used to solve this subproblem. Then, the major iteration determines a steplength α by a line search, and the function is evaluated at the new point. An optimal solution is found when the optimality tolerance criterion is met.

The CRITERIA subcommand has the following keywords when used with CNLR:

ITER n — *Maximum number of major iterations.* Specify any positive integer for *n*. The default is $\max(50, 3(p + m_L) + 10m_N)$, where *p* is the number of parameters, m_L is the number of linear constraints, and m_N is the number of nonlinear constraints. If the search for a solution stops because this limit is exceeded, CNLR issues a warning message.

MINORITERATION n — *Maximum number of minor iterations.* Specify any positive integer. This is the number of minor iterations allowed within each major iteration. The default is $\max(50, 3(n + m_L + m_N))$.

CRSHTOL n — *Crash tolerance.* CRSHTOL is used to determine if initial values are within their specified bounds. Specify any value between 0 and 1. The default value is 0.01. A constraint of the form $a'X \geq l$ is considered a valid part of the working set if $|a'X - l| < \text{CRSHTOL}(1 + |l|)$.

STEPLIMIT n — *Step limit.* The CNLR algorithm does not allow changes in the length of the parameter vector to exceed a factor of *n*. The limit prevents very early steps from going too far from good initial estimates. Specify any positive value. The default value is 2.

FTOLERANCE n — *Feasibility tolerance.* This is the maximum absolute difference allowed for both linear and nonlinear constraints for a solution to be considered feasible. Specify any value greater than 0. The default value is the square root of your machine's epsilon.

LFTOLERANCE n — *Linear feasibility tolerance.* If specified, this overrides FTOLERANCE for linear constraints and bounds. Specify any value greater than 0. The default value is the square root of your machine's epsilon.

NFTOLERANCE n — *Nonlinear feasibility tolerance.* If specified, this overrides FTOLERANCE for nonlinear constraints. Specify any value greater than 0. The default value is the square root of your machine's epsilon.

LSTOLERANCE n — *Line search tolerance.* This value must be between 0 and 1 (but not including 1). It controls the accuracy required of the line search that forms the innermost search loop. The default value, 0.9, specifies an inaccurate search. This is appropriate for many problems, particularly if nonlinear constraints are involved. A smaller positive value, corre-

sponding to a more accurate line search, may give better performance if there are no nonlinear constraints, all (or most) derivatives are supplied in the derivatives program, and the data fit in memory.

OPTOLERANCE n *Optimality tolerance.* If an iteration point is a feasible point and the next step will not produce a relative change in either the parameter vector or the objective function of more than the square root of OPTOLERANCE, an optimal solution has been found. OPTOLERANCE can also be thought of as the number of significant digits in the objective function at the solution. For example, if OPTOLERANCE=10^{-6}, the objective function should have approximately six significant digits of accuracy. Specify any number between the FPRECISION value and 1. The default value for OPTOLERANCE is epsilon**0.8.

FPRECISION n *Function precision.* This is a measure of the accuracy with which the objective function can be checked. It acts as a relative precision when the function is large, and an absolute precision when the function is small. For example, if the objective function is larger than 1, and six significant digits are desired, FPRECISION should be 1E – 6 . If, however, the objective function is of the order 0.001, FPRECISION should be 1E – 9 to get six digits of accuracy. Specify any number between 0 and 1. The choice of FPRECISION can be very complicated for a badly scaled problem. Chapter 8 of Gill et al. (1981) gives some scaling suggestions. The default value is epsilon**0.9.

ISTEP n *Infinite step size.* This value is the magnitude of the change in parameters that is defined as infinite. That is, if the change in the parameters at a step is greater than ISTEP, the problem is considered unbounded, and estimation stops. Specify any positive number. The default value is 1E + 20 .

Iteration Criteria for NLR

The NLR procedure uses an adaptation of subroutine LMSTR from the MINPACK package by Garbow et al. Because the NLR algorithm differs substantially from CNLR, the CRITERIA subcommand for NLR has a different set of keywords.

NLR computes parameter estimates using the Levenberg-Marquardt method. At each iteration, NLR evaluates the estimates against a set of control criteria. The iterative calculations continue until one of five cutoff points is met, at which point the iterations stop and the reason for stopping is displayed.

The CRITERIA subcommand has the following keywords when used with NLR:

ITER n *Maximum number of major and minor iterations allowed.* Specify any positive integer for *n*. The default is 100 iterations per parameter. If the search for a solution stops because this limit is exceeded, NLR issues a warning message.

SSCON n *Convergence criterion for the sum of squares.* Specify any non-negative number for *n*. The default is 1E – 8. If successive iterations fail to reduce the sum of squares by this proportion, the procedure stops. Specify 0 to disable this criterion.

PCON n *Convergence criterion for the parameter values.* Specify any non-negative number for *n*. The default is 1E – 8. If successive iterations fail to change any of the parameter values by this proportion, the procedure stops. Specify 0 to disable this criterion.

RCON n *Convergence criterion for the correlation between the residuals and the derivatives.* Specify any non-negative number for *n*. The default is 1E – 8. If the largest value for the correlation between the residuals and the derivatives equals this value, the procedure stops because it lacks the information it needs to estimate a direction for its next move. This criterion is often referred to as a gradient convergence criterion. Specify 0 to disable this criterion.

Example

```
MODEL PROGRAM A=.5 B=1.6.
COMPUTE PRED=A*SPEED**B.
NLR STOP /CRITERIA=ITER(80) SSCON=.000001.
```

- CRITERIA changes two of the five cutoff values affecting iteration, ITER and SSCON, and leaves the remaining three, PCON, RCON, and CKDER, at their default values.

BOUNDS Subcommand

The BOUNDS subcommand can be used to specify both linear and nonlinear constraints. It can be used only with CNLR; it cannot be used with NLR.

Simple Bounds and Linear Constraints

BOUNDS can be used to impose bounds on parameter values. These bounds can involve either single parameters or a linear combination of parameters and can be either equalities or inequalities.

- All bounds are specified on the same BOUNDS subcommand and separated by semicolons.
- The only variables allowed on BOUNDS are parameter variables (those named on MODEL PROGRAM).
- Only * (multiplication), + (addition), – (subtraction), = or EQ, >= or GE, and <= or LE can be used. When two relational operators are used (as in the third bound in the example below), they must both be in the same direction.

Example

```
/BOUNDS 5 >= A;
        B >= 9;
        .01 <= 2*A + C <= 1;
        D + 2*E = 10
```

- BOUNDS imposes bounds on the parameters *A, B, C,* and *D.* Specifications for each parameter are separated by a semicolon.

Nonlinear Constraints

Nonlinear constraints on the parameters can also be specified with the BOUNDS subcommand. The constrained function must be calculated and stored in a variable by a constrained functions program directly preceding the CNLR command. The constraint is then specified on the BOUNDS subcommand.

In general, nonlinear bounds will not be obeyed until an optimal solution has been found. This is different from simple and linear bounds, which are satisfied at each iteration. The constrained functions must be smooth near the solution.

Example

```
MODEL PROGRAM A=.5 B=1.6.
COMPUTE PRED=A*SPEED**B.

CONSTRAINED FUNCTIONS.
COMPUTE DIFF=A-10**B.

CNLR STOP /BOUNDS DIFF LE 0.
```

- The constrained function is calculated by a constrained functions program and stored in variable *DIFF*. The constrained functions program immediately precedes CNLR.
- BOUNDS imposes bounds on the function (less than or equal to 0).
- CONSTRAINED FUNCTIONS variables and parameters named on MODEL PROGRAM cannot be combined in the same BOUNDS expression. For example, you *cannot* specify $(DIFF + A) > = 0$ on the BOUNDS subcommand.

LOSS Subcommand

LOSS specifies a loss function for CNLR to minimize. By default, CNLR minimizes the sum of squared residuals. LOSS can be used only with CNLR; it cannot be used with NLR.

- The loss function must first be computed in the model program. LOSS is then used to specify the name of the computed variable.
- The minimizing algorithm may fail if it is given a loss function that is not smooth, such as the absolute value of residuals.
- If derivatives are supplied, the derivative of each parameter must be computed with respect to the loss function, rather than the predicted value. The easiest way to do this is in two steps: first compute derivatives of the model, and then compute derivatives of the loss function with respect to the model and multiply by the model derivatives.
- When LOSS is used, the usual summary statistics are not computed. Standard errors, confidence intervals, and correlations of the parameters are available only if the BOOTSTRAP subcommand is specified.

Example

```
MODEL PROGRAM  A=1 B=1.
COMPUTE PRED=EXP(A+B*T)/(1+EXP(A+B*T)).
COMPUTE LOSS=-W*(Y*LN(PRED)+(1-Y)*LN(1-PRED)).

DERIVATIVES.
COMPUTE D.A=PRED/(1+EXP(A+B*T)).
COMPUTE D.B=T*PRED/(1+EXP(A+B*T)).
COMPUTE D.A=(-W*(Y/PRED - (1-Y)/(1-PRED)) * D.A).
COMPUTE D.B=(-W*(Y/PRED - (1-Y)/(1-PRED)) * D.B).

CNLR Y /LOSS=LOSS.
```

- The second COMPUTE command in the model program computes the loss functions and stores its values in the variable *LOSS*, which is then specified on the LOSS subcommand.
- Because derivatives are supplied in the derivatives program, the derivatives of all parameters are computed with respect to the loss function, rather than the predicted value.

BOOTSTRAP Subcommand

BOOTSTRAP provides bootstrap estimates of the parameter standard errors, confidence intervals, and correlations. BOOTSTRAP can be used only with CNLR; it cannot be used with NLR.

Bootstrapping is a way of estimating the standard error of a statistic, using repeated samples from the original data set. This is done by sampling with replacement to get samples of the same size as the original data set.

- The minimum specification is the subcommand keyword. Optionally, specify the number of samples to use for generating bootstrap results.
- By default, BOOTSTRAP generates bootstrap results based on $10*p*(p+1)/2$ samples, where p is the number of parameters. That is, 10 samples are drawn for each statistic (standard error or correlation) to be calculated.
- When BOOTSTRAP is used, the nonlinear equation is estimated for each sample. The standard error of each parameter estimate is then calculated as the standard deviation of the bootstrapped estimates. Parameter values from the original data are used as starting values for each bootstrap sample. Even so, bootstrapping is computationally expensive.
- If the OUTFILE subcommand is specified, a case is written to the output file for each bootstrap sample. The first case in the file will be the actual parameter estimates, followed by the bootstrap samples. After the first case is eliminated (using SELECT IF), other SPSS procedures (such as FREQUENCIES) can be used to examine the bootstrap distribution.

Example

```
MODEL PROGRAM A=.5 B=1.6.
COMPUTE PSTOP=A*SPEED**B.
CNLR STOP /BOOTSTRAP /OUTFILE=PARAM.
GET FILE=PARAM.
LIST.
COMPUTE ID=$CASENUM.
SELECT IF (ID > 1).
FREQUENCIES A B /FORMAT=NOTABLE /HISTOGRAM.
```

- CNLR generates the bootstrap standard errors, confidence intervals, and parameter correlation matrix. OUTFILE saves the bootstrap estimates in the file *PARAM*.
- GET retrieves the system file *PARAM*.
- LIST lists the different sample estimates along with the original estimate. *NCASES* in the listing (see the OUTFILE subcommand on p. 39) refers to the number of distinct cases in the sample because cases are duplicated in each bootstrap sample.
- FREQUENCIES generates histograms of the bootstrapped parameter estimates.

PROBIT

```
PROBIT response-count varname OF observation-count varname
       WITH varlist [BY varname(min,max)]

 [/MODEL={PROBIT**}]
         {LOGIT   }
         {BOTH    }

 [/LOG=[{10** }]
        {2.718}
        {value}
        {NONE }

 [/CRITERIA=[{OPTOL   }({epsilon**0.8})][P({0.15**})][STEPLIMIT({0.1**})]
             {CONVERGE} {n           }     {p     }             {n    }

            [ITERATE({max(50,3(p+1)**})]]
                     {n               }

 [/NATRES[=value]]

 [/PRINT={[CI**] [FREQ**] [RMP**]} [PARALL] [NONE] [ALL]]
         {DEFAULT**               }

 [/MISSING=[{EXCLUDE**}]  ]
           {INCLUDE  }
```

**Default if the subcommand or keyword is omitted.

Example:

```
PROBIT  R OF N BY ROOT(1,2) WITH X
  /MODEL = BOTH.
```

Overview

PROBIT can be used to estimate the effects of one or more independent variables on a dichotomous dependent variable (such as dead or alive, employed or unemployed, product purchased or not). The program is designed for dose-response analyses and related models, but PROBIT can also estimate logistic regression models.

Options

The Model. You can request a probit or logit response model, or both, for the observed response proportions with the MODEL subcommand.

Transform Predictors. You can control the base of the log transformation applied to the predictors or request no log transformation with the LOG subcommand.

Natural Response Rates. You can instruct PROBIT to estimate the natural response rate (threshold) of the model or supply a known natural response rate to be used in the solution with the NATRES subcommand.

Algorithm Control Parameters. You can specify values of algorithm control parameters, such as the limit on iterations, using the CRITERIA subcommand.

Statistics. By default, PROBIT calculates frequencies, fiducial confidence intervals, and the relative median potency. It also produces a plot of the observed probits or logits against the values of a single independent variable. Optionally, you can use the PRINT subcommand to request a test of the parallelism of regression lines for different levels of the grouping variable or to suppress any or all of these statistics.

Basic Specification

- The basic specification is the response-count variable, keyword OF, the observation-count variable, keyword WITH, and at least one independent variable.
- PROBIT calculates maximum-likelihood estimates for the parameters of the default probit response model and automatically displays estimates of the regression coefficient and intercept terms, their standard errors, a covariance matrix of parameter estimates, and a Pearson chi-square goodness-of-fit test of the model.

Subcommand Order

- The variable specification must be first.
- Subcommands can be named in any order.

Syntax Rules

- The variables must include a response count, an observation count, and at least one predictor. A categorical grouping variable is optional.
- All subcommands are optional and each can appear only once.
- Generally, data should not be entered for individual observations. PROBIT expects predictor values, response counts, and the total number of observations as the input case.
- If the data are available only in a case-by-case form, use AGGREGATE first to compute the required response and observation counts.

Operations

- The transformed response variable is predicted as a linear function of other variables using the nonlinear-optimization method. Note that the previous releases used the iteratively weighted least-squares method, which has a different way of transforming the response variables. See the MODEL subcommand on p. 53.

- If individual cases are entered in the data, PROBIT skips the plot of transformed response proportions and predictor values.
- If individual cases are entered, the degrees of freedom for the chi-square goodness-of-fit statistic are based on the individual cases.

Limitations

- Only one prediction model can be tested on a single PROBIT command, although both probit and logit response models can be requested for that prediction.
- Confidence limits, the plot of transformed response proportions and predictor values, and computation of relative median potency are necessarily limited to single-predictor models.

Example

```
PROBIT  R OF N BY ROOT(1,2) WITH X
  /MODEL = BOTH.
```

- This example specifies that both the probit and logit response models be applied to the response frequency *R*, given *N* total observations and the predictor *X*.
- By default, the predictor is log transformed.

Example

```
* Using data in a case-by-case form

DATA LIST FREE / PREPARTN DOSE RESPONSE.
BEGIN DATA
1 1.5 0
 ...
4 20.0 1
END DATA.
COMPUTE SUBJECT = 1.
PROBIT RESPONSE OF SUBJECT BY PREPARTN(1,4) WITH DOSE.
```

- This dose-response model (Finney, 1971) illustrates a case-by-case analysis. A researcher tests four different preparations at varying doses and observes whether each subject responds. The data are individually recorded for each subject, with 1 indicating a response and 0 indicating no response. The number of observations is always 1 and is stored in variable *SUBJECT*.
- PROBIT warns that the data are in a case-by-case form and that the plot is therefore skipped.
- Degrees of freedom for the goodness-of-fit test are based on individual cases, not dosage groups.
- PROBIT displays predicted and observed frequencies for all individual input cases.

Example

```
* Aggregating case-by-case data

DATA LIST FREE/PREPARTN DOSE RESPONSE.
BEGIN DATA
     1.00     1.50      .00
     ...
     4.00    20.00     1.00
END DATA.
AGGREGATE OUTFILE=*
  /BREAK=PREPARTN DOSE
  /SUBJECTS=N(RESPONSE)
  /NRESP=SUM(RESPONSE).
PROBIT NRESP OF SUBJECTS BY PREPARTN(1,4) WITH DOSE.
```

- This example analyzes the same dose-response model as the previous example, but the data are first aggregated.
- AGGREGATE summarizes the data by cases representing all subjects who received the same preparation (*PREPARTN*) at the same dose (*DOSE*).
- The number of cases having a nonmissing response is recorded in the aggregated variable *SUBJECTS*.
- Because *RESPONSE* is coded 0 for no response and 1 for a response, the sum of the values gives the number of observations with a response.
- PROBIT requests a default analysis.
- The parameter estimates for this analysis are the same as those calculated for individual cases in the example above. The chi-square test, however, is based on the number of dosages.

Variable Specification

The variable specification on PROBIT identifies the variables for response count, observation count, groups, and predictors. The variable specification is required.

- The variables must be specified first. The specification must include the response-count variable, followed by the keyword OF and then the observation-count variable.
- If the value of the response-count variable exceeds that of the observation-count variable, a procedure error occurs and PROBIT is not executed.
- At least one predictor (covariate) must be specified following the keyword WITH. The number of predictors is limited only by available workspace. All predictors must be continuous variables.
- You can specify a grouping variable (factor) after the keyword BY. Only one variable can be specified. It must be numeric and can contain only integer values. You must specify, in parentheses, a range indicating the minimum and maximum values for the grouping variable. Each integer value in the specified range defines a group.
- Cases with values for the grouping variable that are outside the specified range are excluded from the analysis.

- Keywords BY and WITH can appear in either order. However, both must follow the response- and observation-count variables.

Example

```
PROBIT R OF N WITH X.
```

- The number of observations having the measured response appears in variable *R*, and the total number of observations is in *N*. The predictor is *X*.

Example

```
PROBIT  R OF N BY ROOT(1,2) WITH X.

PROBIT  R OF N WITH X BY ROOT(1,2).
```

- Because keywords BY and WITH can be used in either order, these two commands are equivalent. Each command specifies *X* as a continuous predictor and *ROOT* as a categorical grouping variable.
- Groups are identified by the levels of variable *ROOT*, which may be 1 or 2.
- For each combination of predictor and grouping variables, the variable *R* contains the number of observations with the response of interest, and *N* contains the total number of observations.

MODEL Subcommand

MODEL specifies the form of the dichotomous-response model. Response models can be thought of as transformations (*T*) of response rates, which are proportions or probabilities (*p*). Note the difference in the transformations between the current version and the previous versions.

- A **probit** is the inverse of the cumulative standard normal distribution function. Thus, for any proportion, the probit transformation returns the value below which that proportion of standard normal deviates is found. For the probit response model, the program uses $T(p) = \text{PROBIT}(p)$. Hence:

  $T(0.025) = \text{PROBIT}(0.025) = -1.96$

  $T(0.400) = \text{PROBIT}(0.400) = -0.25$

  $T(0.500) = \text{PROBIT}(0.500) = 0.00$

  $T(0.950) = \text{PROBIT}(0.950) = 1.64$

- A **logit** is simply the natural log of the odds ratio, $p/(1-p)$. In the Probit procedure, the response function is given as $T(p) = \log_e(p/(1-p))$. Hence:

  $T(0.025) = \text{LOGIT}(0.025) = -3.66$

  $T(0.400) = \text{LOGIT}(0.400) = -0.40$

  $T(0.500) = \text{LOGIT}(0.500) = 0.00$

  $T(0.950) = \text{LOGIT}(0.950) = 2.94$

You can request one or both of the models on the MODEL subcommand. The default is PROBIT if the subcommand is not specified or is specified with no keyword.

PROBIT *Probit response model.* This is the default.

LOGIT *Logit response model.*

BOTH *Both probit and logit response models.* PROBIT displays all the output for the logit model followed by the output for the probit model.

- If subgroups and multiple-predictor variables are defined, PROBIT estimates a separate intercept, a_j, for each subgroup and a regression coefficient, b_i, for each predictor.

LOG Subcommand

LOG specifies the base of the logarithmic transformation of the predictor variables or suppresses the default log transformation.

- LOG applies to all predictors.
- To transform only selected predictors, use COMPUTE commands before the Probit procedure. Then specify NONE on the LOG subcommand.
- If LOG is omitted, a logarithm base of 10 is used.
- If LOG is used without a specification, the natural logarithm base *e* (2.718) is used.
- If you have a control group in your data and specify NONE on the LOG subcommand, the control group is included in the analysis. See the NATRES subcommand on p. 55.

You can specify one of the following on LOG:

value *Logarithm base to be applied to all predictors.*

NONE *No transformation of the predictors.*

Example

```
PROBIT R OF N BY ROOT (1,2) WITH X
  /LOG = 2.
```

- LOG specifies a base-2 logarithmic transformation.

CRITERIA Subcommand

Use CRITERIA to specify the values of control parameters for the PROBIT algorithm. You can specify any or all of the keywords below. Defaults remain in effect for parameters that are not changed.

OPTOL(n) *Optimality tolerance.* Alias CONVERGE. If an iteration point is a feasible point and the next step will not produce a relative change in either the parameter vector or the log-likelihood function of more than the square root of *n*, an optimal solution has been found. OPTOL can also be thought of as the number of significant digits in the log-likelihood function at the solution. For example, if OPTOL=10^{-6}, the log-likelihood function should have approxi-

mately six significant digits of accuracy. The default value is machine epsilon**0.8.

ITERATE(n) *Iteration limit.* Specify the maximum number of iterations. The default is $\max(50, 3(p + 1))$, where p is the number of parameters in the model.

P(p) *Heterogeneity criterion probability*. Specify a cutoff value between 0 and 1 for the significance of the goodness-of-fit test. The cutoff value determines whether a heterogeneity factor is included in calculations of confidence levels for effective levels of a predictor. If the significance of chi-square is greater than the cutoff, the heterogeneity factor is not included. If you specify 0, this criterion is disabled; if you specify 1, a heterogeneity factor is automatically included. The default is 0.15.

STEPLIMIT(n) *Step limit.* The PROBIT algorithm does not allow changes in the length of the parameter vector to exceed a factor of *n*. This limit prevents very early steps from going too far from good initial estimates. Specify any positive value. The default value is 0.1.

CONVERGE(n) *Alias of OPTOL.*

NATRES Subcommand

You can use NATRES either to supply a known natural response rate to be used in the solution or to instruct PROBIT to estimate the natural (or threshold) response rate of the model.

- To supply a known natural response rate as a constraint on the model solution, specify a value less than 1 on NATRES.
- To instruct PROBIT to estimate the natural response rate of the model, you can indicate a control group by giving a 0 value to any of the predictor variables. PROBIT displays the estimate of the natural response rate and the standard error and includes the estimate in the covariance/correlation matrix as NAT RESP.
- If no control group is indicated and NATRES is specified without a given value, PROBIT estimates the natural response rate from the entire data and informs you that no control group has been provided. The estimate of the natural response rate and the standard error are displayed and NAT RESP is included in the covariance/correlation matrix.
- If you have a control group in your data and specify NONE on the LOG subcommand, the control group is included in the analysis.

Example

```
DATA LIST FREE / SOLUTION DOSE NOBSN NRESP.
BEGIN DATA
1  5 100 20
1 10  80 30
1  0 100 10
...
END DATA.

PROBIT NRESP OF NOBSN BY SOLUTION(1,4) WITH DOSE
  /NATRES.
```

- This example reads four variables and requests a default analysis with an estimate of the natural response rate.
- The predictor variable, *DOSE*, has a value of 0 for the third case.
- The response count (10) and the observation count (100) for this case establish the initial estimate of the natural response rate.
- Because the default log transformation is performed, the control group is not included in the analysis.

Example

```
DATA LIST FREE / SOLUTION DOSE NOBSN NRESP.
BEGIN DATA
1  5 100 20
1 10  80 30
1  0 100 10
  ...
END DATA.

PROBIT NRESP OF NOBSN BY SOLUTION(1,4) WITH DOSE
  /NATRES = 0.10.
```

- This example reads four variables and requests an analysis in which the natural response rate is set to 0.10. The values of the control group are ignored.
- The control group is excluded from the analysis because the default log transformation is performed.

PRINT Subcommand

Use PRINT to control the statistics calculated by PROBIT.

- PROBIT always displays the plot (for a single-predictor model) and the parameter estimates and covariances for the probit model.
- If PRINT is used, the requested statistics are calculated and displayed in addition to the parameter estimates and plot.
- If PRINT is not specified or is specified without any keyword, FREQ, CI, and RMP are calculated and displayed in addition to the parameter estimates and plot.

DEFAULT *FREQ, CI, and RMP*. This is the default if PRINT is not specified or is specified by itself.

FREQ *Frequencies.* Display a table of observed and predicted frequencies with their residual values. If observations are entered on a case-by-case basis, this listing can be quite lengthy.

CI *Fiducial confidence intervals.* Print Finney's (1971) fiducial confidence intervals for the levels of the predictor needed to produce each proportion of responses. PROBIT displays this default output for single-predictor models only. If a categorical grouping variable is specified, PROBIT produces a table of confidence intervals for each group. If the Pearson chi-square goodness-of-fit test is significant ($p < 0.15$ by default), PROBIT uses a heterogeneity factor to calculate the limits.

RMP *Relative median potency.* Display the relative median potency (RMP) of each pair of groups defined by the grouping variable. PROBIT displays this default output for single-predictor models only. For any pair of groups, the RMP is the ratio of the stimulus tolerances in those groups. **Stimulus tolerance** is the value of the predictor necessary to produce a 50% response rate. If the derived model for one predictor and two groups estimates that a predictor value of 21 produces a 50% response rate in the first group, and that a predictor value of 15 produces a 50% response rate in the second group, the relative median potency would be 21/15 = 1.40. In biological assay analyses, RMP measures the comparative strength of preparations.

PARALL *Parallelism test.* Produce a test of the parallelism of regression lines for different levels of the grouping variable. This test displays a chi-square value and its associated probability. It requires an additional pass through the data and, thus, additional processing time.

NONE *Display only the unconditional output.* This option can be used to override any other specification on the PRINT subcommand for PROBIT.

ALL *All available output.* This is the same as requesting FREQ, CI, RMP, and PARALL.

MISSING Subcommand

PROBIT always deletes cases having a missing value for any variable. In the output, PROBIT indicates how many cases it rejected because of missing data. This information is displayed with the DATA Information that prints at the beginning of the output. You can use the MISSING subcommand to control the treatment of user-missing values.

EXCLUDE *Delete cases with user-missing values.* This is the default. You can also make it explicit by using the keyword DEFAULT.

INCLUDE *Include user-missing values.* PROBIT treats user-missing values as valid. Only cases with system-missing values are rejected.

RELIABILITY

```
RELIABILITY VARIABLES={varlist}
                      {ALL    }

 [/SCALE(scalename)=varlist [/SCALE... ]]

 [/MODEL={ALPHA         }] [/VARIABLES...]
         {SPLIT[(n)]    }
         {GUTTMAN       }
         {PARALLEL      }
         {STRICTPARALLEL}

 [/STATISTICS=[DESCRIPTIVE]   [SCALE]      [{ANOVA         }]  [ALL]]
              [COVARIANCES]   [TUKEY]       {ANOVA FRIEDMAN}
              [CORRELATIONS]  [HOTELLING]   {ANOVA COCHRAN }

 [/SUMMARY=[MEANS] [VARIANCE] [COV] [CORR] [TOTAL] [ALL]]

[/ICC=[MODEL({MIXED**})][TYPE({CONSISTENCY**})][CIN={95**}]]
           {RANDOM}      {ABSOLUTE   }     {n   }
     [TESTVAL={0**}]
              {p }


 [/METHOD=COVARIANCE]

 [/FORMAT={NOLABELS**}]
          {LABELS    }

 [/MISSING={EXCLUDE**}]
           {INCLUDE  }

 [/MATRIX =[IN({*   })] [OUT({*   })] [NOPRINT]]
                {file}        {file}
```

**Default if the subcommand or keyword is omitted.

Example:

```
RELIABILITY  VARIABLES=SCORE1 TO SCORE10
  /SCALE (OVERALL) = ALL
  /MODEL = ALPHA
  /SUMMARY = MEANS TOTAL.
```

Overview

RELIABILITY estimates reliability statistics for the components of multiple-item additive scales. It uses any one of five models for reliability analysis and offers a variety of statistical displays. RELIABILITY can also be used to perform a repeated measures analysis of variance, a two-way factorial analysis of variance with one observation per cell, Tukey's test for additivity, Hotelling's *T*-square test for equality of means in repeated measures designs, and Friedman's two-way analysis of variance on ranks. For more complex repeated measures designs, use the MANOVA procedure (available in the SPSS Advanced Statistics option).

Options

Model Type. You can specify any one of five models on the MODEL subcommand.

Statistical Display. Statistics available on the STATISTICS subcommand include descriptive statistics, correlation and covariance matrices, a repeated measures analysis of variance table, Hotelling's *T*-square, Tukey's test for additivity, Friedman's chi-square for the analysis of ranked data, and Cochran's *Q*.

Computational Method. You can force RELIABILITY to use the covariance method, even when you are not requesting any output that requires it, by using the METHOD subcommand.

Matrix Input and Output. You can read data in the form of correlation matrices and you can write correlation-type matrix materials to a data file using the MATRIX subcommand.

Basic Specification

The basic specification is VARIABLES and a variable list. By default, RELIABILITY displays the number of cases, number of items, and Cronbach's alpha. Whenever possible, it uses an algorithm that does not require the calculation of the covariance matrix.

Subcommand Order

- VARIABLES must be specified first.
- The remaining subcommands can be named in any order.

Operations

- STATISTICS and SUMMARY are cumulative. If you enter them more than once, all requested statistics are produced for each scale.
- If you request output that is not available for your model or for your data, RELIABILITY ignores the request.
- RELIABILITY uses an economical algorithm whenever possible but calculates a covariance matrix when necessary (see the METHOD subcommand on p. 63).

Limitations

- Maximum 10 VARIABLES subcommands.
- Maximum 50 SCALE subcommands.
- Maximum 500 variables on the combined VARIABLES subcommands. Each occurrence of a variable counts as 1 toward this limit.
- Maximum 500 variables on one SCALE subcommand.
- Maximum 1000 variables on all SCALE subcommands combined. Each mention of a variable counts one toward this limit.
- If the available workspace is insufficient to handle multiple VARIABLES subcommands, RELIABILITY deletes them in the reverse order of specification until the workspace is sufficient.

Example

```
RELIABILITY  VARIABLES=SCORE1 TO SCORE10
  /SCALE (OVERALL) = ALL
  /SCALE (ODD) = SCORE1 SCORE3 SCORE5 SCORE7 SCORE9
  /SUMMARY = MEANS TOTAL.
```

- This example analyzes two additive scales.
- One scale (labeled *OVERALL* in the output) includes all 10 items. Another (labeled *ODD*) includes every other item.
- Summary statistics are displayed for each scale, showing item means and the relationship of each item to the total scale.

Example

```
RELIABILITY  VARIABLES=SCORE1 TO SCORE10.
```

- This example analyzes one scale (labeled *ALL* in the display output) that includes all 10 items.
- Because there is no SUMMARY subcommand, no summary statistics are displayed.

VARIABLES Subcommand

VARIABLES specifies the variables to be used in the analysis. Only numeric variables can be used.

- VARIABLES is required and must be specified first.
- You can use keyword ALL to refer to all user-defined variables in the working data file.
- You can specify VARIABLES more than once on a single RELIABILITY command. A reliability analysis is performed on each set of variables.

SCALE Subcommand

SCALE defines a scale for analysis, providing a label for the scale and specifying its component variables. If SCALE is omitted, all variables named on VARIABLES are used, and the label for the scale is *ALL*.

- The label is specified in parentheses after SCALE. It can have a maximum of eight characters and can use only the letters A to Z and the numerals 0 to 9.
- RELIABILITY does not add any new variables to the working data file. The label is used only to identify the output. If the analysis is satisfactory, use COMPUTE to create a new variable containing the sum of the component items.
- Variables named on SCALE must have been named on the previous VARIABLES subcommand. Use the keyword ALL to refer to all variables named on the preceding VARIABLES subcommand.
- To analyze different groups of component variables, specify SCALE more than once following a VARIABLES subcommand.

Example

```
RELIABILITY VARIABLES = ITEM1 TO ITEM20
  /SCALE (A) = ITEM1 TO ITEM10
  /SCALE (B) = ITEM1 ITEM3 ITEM5 ITEM16 TO ITEM20
  /SCALE (C) = ALL.
```

- This command analyzes three different scales: scale A has 10 items, scale B has 8 items, and scale C has 20 items.

MODEL Subcommand

MODEL specifies the type of reliability analysis for the scale named on the preceding SCALE subcommand.

ALPHA *Cronbach's* α. Standardized item α is displayed. This is the default.

SPLIT [(n)] *Split-half coefficients.* You can specify a number in parentheses to indicate how many items should be in the second half. For example, MODEL SPLIT (6) uses the last six variables for the second half and all others for the first. By default, each half has an equal number of items, with the odd item, if any, going to the first half.

GUTTMAN *Guttman's lower bounds for true reliability.*

PARALLEL *Maximum-likelihood reliability estimate under parallel assumptions.* This model assumes that items have the same variance but not necessarily the same mean.

STRICTPARALLEL *Maximum-likelihood reliability estimate under strictly parallel assumptions.* This model assumes that items have the same means, the same true score variances over a set of objects being measured, and the same error variance over replications.

STATISTICS Subcommand

STATISTICS displays optional statistics. There are no default statistics.

- STATISTICS is cumulative. If you enter it more than once, all requested statistics are produced for each scale.

DESCRIPTIVES *Item means and standard deviations.*

COVARIANCES *Inter-item variance-covariance matrix.*

CORRELATIONS *Inter-item correlation matrix.*

SCALE *Scale means and scale variances.*

TUKEY *Tukey's test for additivity.* This helps determine whether a transformation of the items is needed to reduce nonadditivity. The test displays an estimate of the power to which the items should be raised in order to be additive.

HOTELLING *Hotelling's* T-*square*. This is a test for equality of means among the items.

ANOVA *Repeated measures analysis of variance table.*

FRIEDMAN *Friedman's chi-square and Kendall's coefficient of concordance.* These apply to ranked data. You must request ANOVA in addition to FRIEDMAN; Friedman's chi-square appears in place of the usual *F* test.

COCHRAN *Cochran's* Q. This applies when all items are dichotomies. You must request ANOVA in addition to COCHRAN; the *Q* statistic appears in place of the usual *F* test.

ALL *All applicable statistics.*

ICC Subcommand

ICC displays intraclass correlation coefficients for single measure and average measure. Single measure applies to single measurements, for example, the rating of judges, individual item scores, or the body weights of individuals. Whereas, average measure applies to average measurements, for example, the average rating of *k* judges, or the average score for a *k*-item test.

MODEL *Model.*You can speficy the model for the computation of ICC. There are three keywords for this option. ONEWAY is the one-way random effects model (people effects are random). RANDOM is the two-way random effect model (people effects and the item effects are random). MIXED is the two-way mixed (people effects are random and the item effects are fixed). MIXED is the default. Only one model can be specified.

TYPE *Type of definition.* There are two keywords for this option. CONSISTENCY is the consistency definition and ABSOLUTE is the absolute agreement definition. For the consistency coefficient, the between measures variance is excluded from the denominator variance, and for absolute agreement, it is not.

CIN *The value of the percent for confidence interval and significance leve of the hypothesis testing.*

TESTVAL *The value with which an estimate of ICC is compared.* The value should be between 0 and 1.

SUMMARY Subcommand

SUMMARY displays summary statistics for each individual item in the scale.

- SUMMARY is cumulative. If you enter it more than once, all requested statistics are produced for each scale.
- You can specify one or more of the following:

MEANS *Statistics on item means.* The average, minimum, maximum, range, ratio of maximum to minimum, and variance of the item means.

VARIANCE *Statistics on item variances.* This displays the same statistics as for MEANS.

COVARIANCES *Statistics on item covariances.* This displays the same statistics as for MEANS.

CORRELATIONS *Statistics on item correlations.* This displays the same statistics as for MEANS.

TOTAL *Statistics comparing each individual item to the scale composed of the other items.* The output includes the scale mean, variance, and Cronbach's α without the item, and the correlation between the item and the scale without it.

ALL *All applicable summary statistics.*

METHOD Subcommand

By default, RELIABILITY uses a computational method that does not require the calculation of a covariance matrix wherever possible. METHOD forces RELIABILITY to calculate the covariance matrix. Only a single specification applies to METHOD:

COVARIANCE *Calculate and use the covariance matrix, even if it is not needed.*

If METHOD is not specified, RELIABILITY computes the covariance matrix for all variables on each VARIABLES subcommand only if any of the following is true:

- You specify a model other than ALPHA or SPLIT.
- You request COV, CORR, FRIEDMAN, or HOTELLING on the STATISTICS subcommand.
- You request anything other than TOTAL on the SUMMARY subcommand.
- You write the matrix to a matrix data file, using the MATRIX subcommand.

FORMAT Subcommand

FORMAT controls the initial display of variable names and labels before the analysis.

NOLABELS *Do not display names and labels before the analysis.* This is the default.

LABELS *Display names and labels for all items before the analysis.*

MISSING Subcommand

MISSING controls the deletion of cases with user-missing data.

- RELIABILITY deletes cases from analysis if they have a missing value for any variable named on the current VARIABLES subcommand. By default, both system-missing and user-missing values are excluded.

EXCLUDE *Exclude user- and system-missing values.* This is the default.

INCLUDE *Treat user-missing values as valid.* Only system-missing values are excluded.

MATRIX Subcommand

MATRIX reads and writes SPSS matrix data files.

- Either IN or OUT and the matrix file in parentheses are required. When both IN and OUT are used on the same RELIABILITY procedure, they can be specified on separate MATRIX subcommands or on the same subcommand.
- If both IN and OUT are used on the same RELIABILITY command and there are grouping variables, these variables are treated as if they were split variables. Values of the grouping variables in the input matrix are passed on to the output matrix (see "Split Files" on p. 65).

OUT (filename) *Write a matrix data file.* Specify either a filename or an asterisk (*), enclosed in parentheses. If you specify a filename, the file is stored on disk and can be retrieved at any time. If you specify an asterisk, the matrix file replaces the working data file but is not stored on disk unless you use SAVE or XSAVE.

IN (filename) *Read a matrix data file.* If the matrix data file is the working data file, specify an asterisk (*) in parentheses. If it is another file, specify the filename in parentheses. A matrix file read from an external file does not replace the working data file.

Matrix Output

- RELIABILITY writes correlation-type matrices that include the number of cases, means, and standard deviations with the matrix materials (see "Format of the Matrix Data File" below for a description of the file). These matrix materials can be used as input to RELIABILITY or other procedures.
- Any documents contained in the working data file are not transferred to the matrix file.
- RELIABILITY displays the scale analyses when it writes matrix materials. To suppress the display of scale analyses, specify keyword NOPRINT on MATRIX.

Matrix Input

- RELIABILITY can read a matrix data file created by a previous RELIABILITY command or by another SPSS procedure. The matrix input file must have records of type N, MEAN, STDDEV, and CORR for each split-file group. For more information, see the Universals section in the *SPSS Base Syntax Reference Guide*.
- SPSS reads variable names, variable and value labels, and print and write formats from the dictionary of the matrix data file.
- MATRIX=IN cannot be used unless a working data file has already been defined. To read an existing matrix data file at the beginning of a session, use GET to retrieve the matrix file and then specify IN(*) on MATRIX.

Format of the Matrix Data File

- The matrix data file includes two special variables created by SPSS: *ROWTYPE_* and *VARNAME_*. Variable *ROWTYPE_* is a short string variable having values N, MEAN, STDDEV, and CORR. Variable *VARNAME_* is a short string variable whose values are the names of the variables used to form the correlation matrix.
- When *ROWTYPE_* is CORR, *VARNAME_* gives the variable associated with that row of the correlation matrix.
- The remaining variables in the matrix file are the variables used to form the correlation matrix.

Split Files

- When split-file processing is in effect, the first variables in the matrix data file will be the split variables, followed by *ROWTYPE_*, *VARNAME_*, and the dependent variable(s).
- If grouping variables are in the matrix input file, their values are between *ROWTYPE_* and *VARNAME_*. The grouping variables are treated like split-file variables.
- A full set of matrix materials is written for each split-file group defined by the split variables.
- A split variable cannot have the same variable name as any other variable written to the matrix data file.
- If split-file processing is in effect when a matrix is written, the same split file must be in effect when that matrix is read by any procedure.

Missing Values

Missing-value treatment affects the values written to a matrix data file. When reading a matrix data file, be sure to specify a missing-value treatment on RELIABILITY that is compatible with the treatment that was in effect when the matrix materials were generated.

Example

```
DATA LIST / TIME1 TO TIME5 1-10.
BEGIN DATA
 0 0 0 0 0
 0 0 1 1 0
 0 0 1 1 1
 0 1 1 1 1
 0 0 0 0 1
 0 1 0 1 1
 0 0 1 1 1
 1 0 0 1 1
 1 1 1 1 1
 1 1 1 1 1
END DATA.
RELIABILITY  VARIABLES=TIME1 TO TIME5
  /MATRIX=OUT(RELMTX).
LIST.
```

- RELIABILITY reads data from the working data file and writes one set of matrix materials to file *RELMTX*.
- The working data file is still the file defined by DATA LIST. Subsequent commands are executed in this file.

Example

```
DATA LIST  / TIME1 TO TIME5 1-10.
BEGIN DATA
 0 0 0 0 0
 0 0 1 1 0
 0 0 1 1 1
 0 1 1 1 1
 0 0 0 0 1
 0 1 0 1 1
 0 0 1 1 1
 1 0 0 1 1
 1 1 1 1 1
 1 1 1 1 1
END DATA.
RELIABILITY  VARIABLES=TIME1 TO TIME5
  /MATRIX=OUT(*) NOPRINT.
LIST.
```

- RELIABILITY writes the same matrix as in the previous example. However, the matrix data file replaces the working data file. The LIST command is executed in the matrix file, not in the file defined by DATA LIST.
- Because NOPRINT is specified on MATRIX, scale analyses are not displayed.

Example

```
GET FILE=RELMTX.
RELIABILITY VARIABLES=ALL
  /MATRIX=IN(*).
```

- This example assumes that you are starting a new session and want to read an existing matrix data file. GET retrieves the matrix data file *RELMTX*.
- MATRIX=IN specifies an asterisk because the matrix data file is the working data file. If MATRIX=IN(RELMTX) is specified, SPSS issues an error message.
- If the GET command is omitted, SPSS issues an error message.

Example

```
GET FILE=PRSNNL.
FREQUENCIES VARIABLE=AGE.

RELIABILITY VARIABLES=ALL
  /MATRIX=IN(RELMTX).
```

- This example performs a frequencies analysis on file *PRSNNL* and then uses a different file containing matrix data for RELIABILITY. The file is an existing matrix data file. In order for this to work, the analysis variables named in *RELMTX* must also exist in *PRSNNL*.
- *RELMTX* must have records of type N, MEAN, STDDEV, and CORR for each split-file group.
- *RELMTX* does not replace *PRSNNL* as the working data file.

Example

```
GET FILE=PRSNNL.
CORRELATIONS VARIABLES=V1 TO V5
  /MATRIX=OUT(*).
RELIABILITY VARIABLES=V1 TO V5
  /MATRIX=IN(*).
```

- RELIABILITY uses matrix input from procedure CORRELATIONS. An asterisk is used to specify the working data file for both the matrix output from CORRELATIONS and the matrix input for RELIABILITY.

2SLS

```
2SLS [EQUATION=]dependent variable WITH predictor variable

 [/[EQUATION=]dependent variable...]

  /INSTRUMENTS=varlist

 [/ENDOGENOUS=varlist]

 [/{CONSTANT**}
   {NOCONSTANT}

 [/PRINT=COV]

 [/SAVE = [PRED] [RESID]]

 [/APPLY[='model name']]
```

**Default if the subcommand or keyword is omitted.

Example:

```
2SLS VAR01 WITH VAR02 VAR03
  /INSTRUMENTS VAR03 LAGVAR01.
```

Overview

2SLS performs two-stage least-squares regression to produce consistent estimates of parameters when one or more predictor variables might be correlated with the disturbance. This situation typically occurs when your model consists of a system of simultaneous equations wherein endogenous variables are specified as predictors in one or more of the equations. The two-stage least-squares technique uses instrumental variables to produce regressors that are not contemporaneously correlated with the disturbance. Parameters of a single equation or a set of simultaneous equations can be estimated.

Options

New Variables. You can change NEWVAR settings on the TSET command prior to 2SLS to evaluate the regression statistics without saving the values of predicted and residual variables, or save the new values to replace the values saved earlier, or save the new values without erasing values saved earlier (see the TSET command in the *SPSS Base Syntax Reference Guide*). You can also use the SAVE subcommand on 2SLS to override the NONE or the default CURRENT settings on NEWVAR.

Covariance Matrix. You can obtain the covariance matrix of the parameter estimates in addition to all of the other output by specifying PRINT=DETAILED on the TSET command prior to 2SLS. You can also use the PRINT subcommand to obtain the covariance matrix regardless of the setting on PRINT.

Basic Specification

The basic specification is at least one EQUATION subcommand and one INSTRUMENTS subcommand.

- For each equation specified, 2SLS estimates and displays the regression analysis-of-variance table, regression standard error, mean of the residuals, parameter estimates, standard errors of the parameter estimates, standardized parameter estimates, *t* statistic significance tests and probability levels for the parameter estimates, tolerance of the variables, the parameter estimates, and correlation matrix of the parameter estimates.
- If the setting on NEWVAR is either ALL or the default CURRENT, two new variables containing the predicted and residual values are automatically created for each equation. The variables are labeled and added to the working data file.

Subcommand Order

- Subcommands can be specified in any order.

Syntax Rules

- The INSTRUMENTS subcommand must specify at least as many variables as are specified after WITH on the longest EQUATION subcommand.
- If a subcommand is specified more than once, the effect is cumulative (except for the APPLY subcommand, which only executes the last occurrence).

Operations

- 2SLS cannot produce forecasts beyond the length of any regressor series.
- 2SLS honors the SPSS WEIGHT command.
- 2SLS uses listwise deletion of missing data. Whenever a variable is missing a value for a particular observation, that observation will not be used in any of the computations.

EQUATION Subcommand

EQUATION specifies the structural equations for the model and is required. The actual keyword EQUATION is optional.

- An equation specifies a single dependent variable, followed by keyword WITH and one or more predictor variables.
- You can specify more than one equation. Multiple equations are separated by slashes.

Example

```
2SLS EQUATION=Y1 WITH X1 X2
  /INSTRUMENTS=X1 LAGX2 X3.
```

- In this example, *Y1* is the dependent variable and *X1* and *X2* are the predictors. The instruments used to predict the *X2* values are *X1*, *LAGX2*, and *X3*.

INSTRUMENTS Subcommand

INSTRUMENTS specifies the instrumental variables. These variables are used to compute predicted values for the endogenous variables in the first stage of 2SLS.

- At least one INSTRUMENTS subcommand must be specified.
- If more than one INSTRUMENTS subcommand is specified, the effect is cumulative. All variables named on INSTRUMENTS subcommands are used as instruments to predict all the endogenous variables.
- Any variable in the working data file can be named as an instrument.
- Instrumental variables can be specified on the EQUATION subcommand, but this is not required.
- The INSTRUMENTS subcommand must name at least as many variables as are specified after WITH on the longest EQUATION subcommand.
- If all the predictor variables are listed as the only INSTRUMENTS, the results are the same as results from ordinary least-squares regression.

Example

```
2SLS DEMAND WITH PRICE, INCOME
  /PRICE WITH DEMAND, RAINFALL, LAGPRICE
  /INSTRUMENTS=INCOME, RAINFALL, LAGPRICE.
```

- The endogenous variables are PRICE and DEMAND.
- The instruments to be used to compute predicted values for the endogenous variables are INCOME, RAINFALL, and LAGPRICE.

ENDOGENOUS Subcommand

All variables not specified on the INSTRUMENTS subcommand are used as endogenous variables by 2SLS. The ENDOGENOUS subcommand simply allows you to document what these variables are.

- Computations are not affected by specifications on the ENDOGENOUS subcommand.

Example

```
2SLS Y1 WITH X1 X2 X3
  /INSTRUMENTS=X2 X4 LAGY1
  /ENDOGENOUS=Y1 X1 X3.
```

- In this example, the ENDOGENOUS subcommand is specified to document the endogenous variables.

CONSTANT and NOCONSTANT Subcommands

Specify CONSTANT or NOCONSTANT to indicate whether a constant term should be estimated in the regression equation. The specification of either subcommand overrides the CONSTANT setting on the TSET command for the current procedure.

- CONSTANT is the default and specifies that the constant term is used as an instrument.
- NOCONSTANT eliminates the constant term.

SAVE Subcommand

SAVE saves the values of predicted and residual variables generated during the current session to the end of the working data file. The default names *FIT_n* and *ERR_n* will be generated, where *n* increments each time variables are saved for an equation. SAVE overrides the NONE or the default CURRENT setting on NEWVAR for the current procedure.

PRED *Save the predicted value.* The new variable is named *FIT_n*, where *n* increments each time a predicted or residual variable is saved for an equation.

RESSID *Save the residual value.* The new variable is named *ERR_n*, where *n* increments each time a predicted or residual variable is saved for an equation.

PRINT Subcommand

PRINT can be used to produce an additional covariance matrix for each equation. The only specification on this subcommand is keyword COV. The PRINT subcommand overrides the PRINT setting on the TSET command for the current procedure.

APPLY Subcommand

APPLY allows you to use a previously defined 2SLS model without having to repeat the specifications.

- The only specification on APPLY is the name of a previous model. If a model name is not specified, the model specified on the previous 2SLS command is used.
- To change the series used with the model, enter new series names before or after the APPLY subcommand.
- To change one or more model specifications, specify the subcommands of only those portions you want to change after the APPLY subcommand.
- If no series are specified on the command, the series that were originally specified with the model being reapplied are used.

Example

```
2SLS Y1 WITH X1 X2 / X1 WITH Y1 X2
  /INSTRUMENTS=X2 X3.
2SLS APPLY
  /INSTRUMENTS=X2 X3 LAGX1.
```

- In this example, the first command requests 2SLS using *X2* and *X3* as instruments.
- The second command specifies the same equations but changes the instruments to *X2*, *X3*, and *LAGX1*.

WLS

```
WLS [VARIABLES=]dependent varname WITH independent varnames

 [/SOURCE=varname]

 [/DELTA=[{1.0**                 }]]
          {value list            }
          {value TO value BY value}

 [/WEIGHT=varname]

 [/{CONSTANT**}
   {NOCONSTANT}

 [/PRINT={BEST}]
         {ALL }

 [/SAVE = WEIGHT]

 [/APPLY[='model name']]
```

**Default if the subcommand or keyword is omitted.

Example:

```
WLS VARY WITH VARX VARZ
  /SOURCE=VARZ
  /DELTA=2.
```

Overview

WLS (weighted least squares) estimates regression models with different weights for different cases. Weighted least squares should be used when errors from an ordinary regression are heteroscedastic—that is, when the size of the residual is a function of the magnitude of some variable, termed the **source**.

The WLS model is a simple regression model in which the residual variance is a function of the source variable, up to some power transform indicated by a delta value. For fuller regression results, save the weights produced by WLS and specify that weight variable on the REGWGT subcommand in REGRESSION.

Options

Calculated and Specified Weights. WLS can calculate the weights based on a source variable and delta values (subcommands SOURCE and DELTA), or it can apply existing weights contained in a series (subcommand WEIGHT). If weights are calculated, each weight value is calculated as the source series value raised to the negative delta value.

New Variables. You can change NEWVAR settings on the TSET command prior to WLS to evaluate the regression coefficients and log-likelihood function without saving the weight variable, or save the new values to replace the values saved earlier, or save the new values without erasing values saved earlier (see the TSET command in the *SPSS Base Syntax Ref-*

erence Guide). You can also use the SAVE subcommand on WLS to override the NONE or the default CURRENT settings on NEWVAR for the current procedure.

Statistical Output. You can change the PRINT setting on the TSET command prior to WLS to display regression coefficients or the list of log-likelihood functions at each delta value, or to limit the output to only the regression statistics for the delta value at which the log-likelihood function is maximized (see the TSET command in the *SPSS Base Syntax Reference Guide*). You can also use the PRINT subcommand to override the PRINT setting on the TSET command for the current procedure and obtain regression coefficients at each value of delta in addition to the default output.

Basic Specification

- The basic specification is the VARIABLES subcommand specifying one dependent variable, the keyword WITH, and one or more independent variables. Weights are calculated using the first independent variable as the source variable and a default delta value of 1.
- The default output for calculated weights displays the log-likelihood function for each value of delta. For the value of delta at which the log-likelihood function is maximized, the displayed summary regression statistics include *R*, R^2, adjusted R^2, standard errors, analysis of variance, and *t* tests of the individual coefficients. A variable named *WGT#1* containing the calculated weights is automatically created, labeled, and added to the working data file.

Syntax Rules

- VARIABLES can be specified only once.
- DELTA can be specified more than once. Each specification will be executed.
- If other subcommands are specified more than once, only the last specification of each one is executed.
- You can specify either SOURCE and DELTA, or just the WEIGHT subcommand. You cannot specify all three, and you cannot specify WEIGHT with SOURCE or with DELTA.

Subcommand Order

- Subcommands can be specified in any order.

Operations

- If neither the WEIGHT subcommand nor the SOURCE and DELTA subcommands are specified, a warning is issued and weights are calculated using the default source and delta value.
- Only one *WGT#1* variable is created per procedure. If more than one delta value is specified, the weights used when the log-likelihood function is maximized are the ones saved as *WGT#1*.

- *WGT#1* is not created when the WEIGHT subcommand is used.
- The SPSS WEIGHT command specifies case replication weights, which are *not* the same as the weights used in weighted least squares. If the WEIGHT command and WLS WEIGHT subcommand are both specified, both types of weights are incorporated in WLS.
- WLS uses listwise deletion of missing values. Whenever one variable is missing a value for a particular observation, that observation will not be included in any computations.

Limitations

- Maximum one VARIABLES subcommand.
- Maximum one dependent variable on the VARIABLES subcommand. There is no limit on the number of independent variables.
- Maximum 150 values specified on the DELTA subcommand.

Example

```
WLS VARY WITH VARX VARZ
  /SOURCE=VARZ
  /DELTA=2.
```

- This command specifies a weighted least-squares regression in which *VARY* is the dependent variable and *VARX* and *VARZ* are the independent variables.
- *VARZ* is identified as the source of heteroscedasticity.
- Weights will be calculated using a delta value of 2. Thus, the weights will equal $VARZ^{-2}$.

VARIABLES Subcommand

VARIABLES specifies the variable list and is the only required subcommand. The actual keyword VARIABLES can be omitted.

SOURCE Subcommand

SOURCE is used in conjunction with the DELTA subcommand to compute weights. SOURCE names the variable that is the source of heteroscedasticity.

- The only specification on SOURCE is the name of a variable to be used as the source of heteroscedasticity.
- Only one source variable can be specified.
- If neither SOURCE nor WEIGHT is specified, the first independent variable specified on the VARIABLES subcommand is assumed to be the source variable.

DELTA Subcommand

DELTA, alias POWER, is used in conjunction with the SOURCE subcommand to compute weights. DELTA specifies the values to use in computing weights. The weights are equal to 1/(SOURCE raised to the DELTA power).

- The specification on DELTA is a list of possible delta values and/or value grids.
- Multiple values and grids can be specified on one DELTA subcommand.
- Delta values can be any value in the range of –6.5 to +7.5. Values below this range are assigned the minimum (–6.5), and values above are assigned the maximum (7.5).
- A grid is specified by naming the starting value, the keyword TO, an ending value, the keyword BY, and an increment value. Alternatively, the keyword BY and the increment value can be specified after the starting value.
- More than one DELTA subcommand can be specified; each subcommand will be executed.
- If DELTA is not specified, the delta value defaults to 1.0.

Example

```
WLS X1 WITH Y1 Z1
  /SOURCE=Z1
  /DELTA=0.5.
```

- In this example, weights are calculated using the source variable *Z1* and a delta value of 0.5. Thus, the weights are 1/(SQRT (Z1)).

Example

```
WLS SHARES WITH PRICE
  /DELTA=0.5 TO 2.5 BY 0.5.
```

- In this example, several regression equations will be fit, one for each value of delta.
- Weights are calculated using the source variable *PRICE* (the default).
- The delta values start at 0.5 and go up to 2.5, incrementing by 0.5. This specification is equivalent to 0.5 BY 0.5 TO 2.5.
- The weights that maximize the log-likelihood function will be saved as variable *WGT#1*.

WEIGHT Subcommand

WEIGHT specifies the variable containing the weights to be used in weighting the cases. WEIGHT is an alternative to computing the weights using the SOURCE and DELTA subcommands. If a variable containing weights is specified, the output includes the regression coefficients, log-likelihood function, and summary regression statistics such as R, R^2, adjusted R^2, standard errors, analysis of variance, and t tests of the coefficients. Since no new weights are computed, no new variable is created. For a description of the output when weights are calculated by WLS, see “Basic Specification” on p. 74.

- The only specification on WEIGHT is the name of the variable containing the weights. Typically, *WGT* variables from previous WLS procedures are used.
- Only one variable can be specified.

Example

```
WLS SHARES WITH PRICE
  /WEIGHT=WGT_1.
```

- This WLS command uses the weights contained in variable *WGT_1* to weight cases.

CONSTANT and NOCONSTANT Subcommands

Specify CONSTANT or NOCONSTANT to indicate whether a constant term should be estimated in the regression equation. The specification of either subcommand overrides the CONSTANT setting on the TSET command for the current procedure.

- CONSTANT is the default and specifies that the constant term is used as an instrument.
- NOCONSTANT eliminates the constant term.

SAVE Subcommand

SAVE saves the weight variable generated during the current session to the end of the working data file. The default name *WGT_n* will be generated, where *n* increments to make the variable name unique. The only specification on SAVE is WEIGHT. The specification overrides the NONE or the default CURRENT setting on NEWVAR for the current procedure.

PRINT Subcommand

PRINT can be used to override the PRINT setting on the TSET command for the current procedure. Two keywords are available.

BEST *Display coefficients for the best weight only.* This is the default.

ALL *Display coefficients for all weights.*

APPLY Subcommand

- The APPLY subcommand allows you to use a previously defined WLS model without having to repeat the specifications.
- The only specification on APPLY is the name of a previous model in quotes. If a model name is not specified, the model specified on the previous WLS command is used.
- To change one or more model specifications, specify the subcommands of only those portions you want to change after the APPLY subcommand.
- If no variables are specified on the command, the variables that were originally specified with the model being reapplied are used.

Example

```
WLS X1 WITH Y1
  /SOURCE=Y1
  /DELTA=1.5.
WLS APPLY
  /DELTA=2.
```

- The first command produces a weighted least-squares regression of *X1*, with *Y1* as the source variable and delta equal to 1.5.
- The second command uses the same variable and source but changes the delta value to 2.

Example

```
WLS X1 WITH Y1 Z1
  /SOURCE=Z1
  /DELTA=1 TO 3 BY 0.5
WLS APPLY
  /WEIGHT=WGT#1.
```

- The first command regresses *X1* on *Y1* and *Z1*, using *Z1* as the source variable. The delta values range from 1 to 3, incrementing by 0.5.
- The second command again regresses *X1* on *Y1* and *Z1*, but this time applies the values of *WGT#1* as the weights.

Appendix
Categorical Variable Coding Schemes

In many SPSS procedures, you can request automatic replacement of a categorical independent variable with a set of contrast variables, which will then be entered or removed from an equation as a block. You can specify how the set of contrast variables is to be coded, usually on the CONTRAST subcommand. This appendix explains and illustrates how different contrast types requested on CONTRAST actually work.

Deviation

Deviation from the grand mean. In matrix terms, these contrasts have the form:

mean	(1/*k*	1/*k*	...	1/*k*	1/*k*)
df(1)	(1-1/*k*	-1/*k*	...	-1/*k*	-1/*k*)
df(2)	(-1/*k*	1-1/*k*	...	-1/*k*	-1/*k*)
.		.			
.		.			
df(k-1)	(-1/*k*	-1/*k*	...	1-1/*k*	-1/*k*)

where *k* is the number of categories for the independent variable and the last category is omitted by default. For example, the deviation contrasts for an independent variable with three categories are as follows:

(1/3	1/3	1/3)
(2/3	-1/3	-1/3)
(-1/3	2/3	-1/3)

To omit a category other than the last, specify the number of the omitted category in parentheses after the DEVIATION keyword. For example, the following subcommand obtains the deviations for the first and third categories and omits the second:

```
/CONTRAST(FACTOR)=DEVIATION(2)
```

Suppose that *factor* has three categories. The resulting contrast matrix will be

```
(   1/3      1/3      1/3 )
(   2/3     -1/3     -1/3 )
(  -1/3     -1/3      2/3 )
```

Simple

Simple contrasts. Compares each level of a factor to the last. The general matrix form is

```
mean   (  1/k      1/k      ...     1/k      1/k )
df(1)  (   1        0       ...      0       -1 )
df(2)  (   0        1       ...      0       -1 )
.                   .
.                   .
df(k-1) (  0        0       ...      1       -1 )
```

where *k* is the number of categories for the independent variable. For example, the simple contrasts for an independent variable with four categories are as follows:

```
( 1/4      1/4      1/4      1/4 )
(   1        0        0       -1 )
(   0        1        0       -1 )
(   0        0        1       -1 )
```

To use another category instead of the last as a reference category, specify in parentheses after the SIMPLE keyword the sequence number of the reference category, which is not necessarily the value associated with that category. For example, the following CONTRAST subcommand obtains a contrast matrix that omits the second category:

```
/CONTRAST(FACTOR) = SIMPLE(2)
```

Suppose that *factor* has four categories. The resulting contrast matrix will be

```
( 1/4      1/4      1/4      1/4 )
(   1       -1        0        0 )
(   0       -1        1        0 )
(   0       -1        0        1 )
```

Helmert

Helmert contrasts. Compares categories of an independent variable with the mean of the subsequent categories. The general matrix form is

mean	(1/*k*	1/*k*	...	1/*k*	1/*k*)
df(1)	(1	-1/(*k*-1)	...	-1/(*k*-1)	-1/(*k*-1))
df(2)	(0	1	...	-1/(*k*-2)	-1/(*k*-2))
.		.			
.		.			
df(*k-2)*	(0	0	1	-1/2	-1/2)
df(k-1)	(0	0	...	1	-1)

where *k* is the number of categories of the independent variable. For example, an independent variable with four categories has a Helmert contrast matrix of the following form:

(1/4	1/4	1/4	1/4)
(1	-1/3	-1/3	-1/3)
(0	1	-1/2	-1/2)
(0	0	1	-1)

Difference

Difference or reverse Helmert contrasts. Compares categories of an independent variable with the mean of the previous categories of the variable. The general matrix form is

mean	(1/*k*	1/*k*	1/*k*	...	1/*k*)
df(1)	(-1	1	0	...	0)
df(2)	(-1/2	-1/2	1	...	0)
.		.			
.		.			
df(k-1)	(-1/(*k*-1)	-1/(*k*-1)	-1/(*k*-1)	...	1)

where *k* is the number of categories for the independent variable. For example, the difference contrasts for an independent variable with four categories are as follows:

(1/4	1/4	1/4	1/4)
(-1	1	0	0)
(-1/2	-1/2	1	0)
(-1/3	-1/3	-1/3	1)

Polynomial

Orthogonal polynomial contrasts. The first degree of freedom contains the linear effect across all categories; the second degree of freedom, the quadratic effect; the third degree of freedom, the cubic; and so on for the higher-order effects.

You can specify the spacing between levels of the treatment measured by the given categorical variable. Equal spacing, which is the default if you omit the metric, can be specified as consecutive integers from 1 to *k*, where *k* is the number of categories. If the variable *drug* has three categories, the subcommand

```
/CONTRAST(DRUG)=POLYNOMIAL
```

is the same as

```
/CONTRAST(DRUG)=POLYNOMIAL(1,2,3)
```

Equal spacing is not always necessary, however. For example, suppose that *drug* represents different dosages of a drug given to three groups. If the dosage administered to the second group is twice that to the first group and the dosage administered to the third group is three times that to the first group, the treatment categories are equally spaced, and an appropriate metric for this situation consists of consecutive integers:

```
/CONTRAST(DRUG)=POLYNOMIAL(1,2,3)
```

If, however, the dosage administered to the second group is four times that given the first group, and the dosage given the third group is seven times that to the first, an appropriate metric is

```
/CONTRAST(DRUG)=POLYNOMIAL(1,4,7)
```

In either case, the result of the contrast specification is that the first degree of freedom for *drug* contains the linear effect of the dosage levels and the second degree of freedom contains the quadratic effect.

Polynomial contrasts are especially useful in tests of trends and for investigating the nature of response surfaces. You can also use polynomial contrasts to perform nonlinear curve fitting, such as curvilinear regression.

Repeated

Compares adjacent levels of an independent variable. The general matrix form is

mean	(1/*k*	1/*k*	1/*k*	...	1/*k*	1/*k*)
df(1)	(1	-1	0	...	0	0)
df(2)	(0	1	-1	...	0	0)
.		.				
.		.				
df(k-1)	(0	0	0	...	1	-1)

where *k* is the number of categories for the independent variable. For example, the repeated contrasts for an independent variable with four categories are as follows:

(1/4	1/4	1/4	1/4)
(1	-1	0	0)
(0	1	-1	0)
(0	0	1	-1)

These contrasts are useful in profile analysis and wherever difference scores are needed.

Special

A user-defined contrast. Allows entry of special contrasts in the form of square matrices with as many rows and columns as there are categories of the given independent variable. For MANOVA and LOGLINEAR, the first row entered is always the mean, or constant, effect and represents the set of weights indicating how to average other independent variables, if any, over the given variable. Generally, this contrast is a vector of ones.

The remaining rows of the matrix contain the special contrasts indicating the desired comparisons between categories of the variable. Usually, orthogonal contrasts are the most useful. Orthogonal contrasts are statistically independent and are nonredundant. Contrasts are orthogonal if:

- For each row, contrast coefficients sum to zero.
- The products of corresponding coefficients for all pairs of disjoint rows also sum to zero.

For example, suppose that *treatment* has four levels and that you want to compare the various levels of treatment with each other. An appropriate special contrast is

(1	1	1	1)	weights for mean calculation
(3	-1	-1	-1)	compare 1st with 2nd through 4th
(0	2	-1	-1)	compare 2nd with 3rd and 4th
(0	0	1	-1)	compare 3rd with 4th

which you specify by means of the following CONTRAST subcommand for MANOVA, LOGISTIC REGRESSION, and COXREG:

```
/CONTRAST(TREATMNT)=SPECIAL( 1  1  1  1
                             3 -1 -1 -1
                             0  2 -1 -1
                             0  0  1 -1 )
```

For LOGLINEAR, you need to specify:

```
/CONTRAST(TREATMNT)=BASIS SPECIAL( 1  1  1  1
                                   3 -1 -1 -1
                                   0  2 -1 -1
                                   0  0  1 -1 )
```

Each row except the means row sums to zero. Products of each pair of disjoint rows sum to zero as well:

Rows 2 and 3: $(3)(0) + (-1)(2) + (-1)(-1) + (-1)(-1) = 0$

Rows 2 and 4: $(3)(0) + (-1)(0) + (-1)(1) + (-1)(-1) = 0$

Rows 3 and 4: $(0)(0) + (2)(0) + (-1)(1) + (-1)(-1) = 0$

The special contrasts need not be orthogonal. However, they must not be linear combinations of each other. If they are, the procedure reports the linear dependency and ceases processing. Helmert, difference, and polynomial contrasts are all orthogonal contrasts.

Indicator

Indicator variable coding. Also known as dummy coding, this is not available in LOGLINEAR or MANOVA. The number of new variables coded is $k - 1$. Cases in the reference category are coded 0 for all $k - 1$ variables. A case in the ith category is coded 0 for all indicator variables except the ith, which is coded 1.

Subject Index

Syntax Index