3 Data Transformation and Selection

In an ideal situation, your raw data are perfectly suitable for the type of analysis you want to perform, and any relationships between variables are either conveniently linear or neatly orthogonal. If this is the case, you can proceed directly from basic data definition to complex statistical analysis. However, you will probably find that this is rarely the case. Preliminary analysis may reveal inconvenient coding schemes or coding errors; complex data transformations may be required to coax out the true relationship between variables; or you may find that only a particular subset of cases is relevant to your analysis.

With SPSS you can perform data transformation ranging from simple tasks, such as collapsing categories for analysis, to creating new variables based on complex equations and conditional statements. You can also select cases for analysis based on an equally complex set of conditions or choose a simple random sample.

This chapter provides an overview of the data transformation and selection commands available with SPSS. For a complete discussion of these commands, see the *SPSS Reference Guide.*

3.1 RECODE COMMAND

The RECODE command tells SPSS to change the values for a variable as the data are being read. The command

```
RECODE X (0=9).
```

instructs SPSS to change all 0's found for variable X to 9's.

The variable or variables to be recoded must already exist on the active SPSS system file. You can specify as many value specifications as needed, enclosing each specification within parentheses, as in:

```
RECODE ITEM1 (0=1) (1=0) (2=-1).
```

You can use multiple input values in a single specification but only one output value following the equals sign, as in:

```
RECODE ITEM2 (8,9=1) (4 THRU 7=2) (1,2=3).
```

The RECODE command is evaluated left to right, and the values for a case are recoded only once per RECODE command. For example, if a case has an input value of 0 for variable ITEM1, the command

```
RECODE ITEM1 (0=1) (1=0) (2=-1).
```

recodes ITEM1 to 1 and SPSS then moves on. The value 1 is *not* recoded back to 0 by the second value specification. Input values not mentioned on the RECODE command are left unchanged.

You can name multiple variables for the same value specifications, as in:

```
RECODE ITEM1 TO ITEM3 (0=1) (1=0) (2=-1).
```

In addition, you can specify different values for different variables on the same RECODE command by separating the specifications with a slash, as in:

```
RECODE AGE (0=9)
 /ITEM1 TO ITEM3 (0=1) (1=0) (2=-1).
```

These rules apply to both numeric and string variables. See Section 3.8 for more information on recoding string variables.

3.2 THRU, LOWEST, and HIGHEST Keywords

To recode ranges of values for numeric variables into a single value, use keyword THRU. Use keyword LO (LOWEST) or HI (HIGHEST) to specify the lowest or highest input value for the variable. For example, to recode all individuals below the United States voting age to 0 and leave all other ages unchanged, specify:

```
RECODE AGE (LO THRU 17=0).
```

Keywords LOWEST and HIGHEST do not include the system-missing value. However, user-missing values are included.

3.3 ELSE Keyword

To recode all values not previously mentioned into a single catchall category, use the keyword ELSE. For example, to recode AGE to a dichotomous (two-valued) variable with 0 representing individuals below the voting age and 1 representing potential voters, specify:

```
RECODE AGE (LO THRU 17=0) (ELSE=1).
```

ELSE should be the last specification for the variable. Otherwise all subsequent value specifications for that variable are ignored. Keyword ELSE *does* include the system-missing value.

3.4 MISSING and SYSMIS Keywords

To recode a variable that may have missing values, use keyword MISSING or SYSMIS. For example, if −98 and −99 were declared missing for variable AGE, the command

```
RECODE AGE (MISSING=9).
```

recodes −98, −99, and any system-missing values (perhaps from input errors) for variable AGE to 9. The command

```
RECODE AGE (MISSING=SYSMIS).
```

recodes all missing values to the system-missing value.

You can use keyword MISSING only as an input value. MISSING refers to all missing values including the system-missing value. The output value from a MISSING input specification is not automatically missing; use the MISSING VALUES command to declare the new value missing (see Chapter 2). You can use keyword SYSMIS for either input or output. As an input value, SYSMIS refers to system-missing values. As an output value specification, SYSMIS recodes all values named on the left side of the equals sign to the system-missing value.

3.5 Recoding Continuous Variables

If a numeric variable has noninteger values, some values may not be recoded unless you make certain they are included in a value range. For example, if AGE had noninteger values, the command

```
RECODE AGE (LO THRU 17=0) (18 THRU HI=1).
```

would not recode values between 17 and 18, such as 17.5. You can avoid this problem by using overlapping endpoint values, as in:

```
RECODE AGE (18 THRU HI=1) (LO THRU 18=0).
```

Note that the order of the recode specifications has been reversed. Since a value is recoded only once, any cases with a value of exactly 18 will be recoded to a value of 1.

3.6 INTO Keyword

To recode the values of one variable and store them in another variable, use the keyword INTO, as in:

```
RECODE AGE (18 THRU HI=1) (LO THRU 18=0) INTO VOTER.
```

The recoded AGE values are stored in the *target variable* VOTER, leaving AGE unchanged.

Target variables can be existing or new variables. If you use an existing variable, cases with values not mentioned in the recode specification are not changed. If you recode a variable into a new variable, cases with values not specified for recoding are assigned the system-missing value.

New numeric variables have default print and write formats of F8.2 (see Chapter 2) or the format you specify using the SET command (see Chapter 5).

3.7 COPY Keyword

When you recode a variable into a new variable or use keyword ELSE as a cleanup category, you may want to retain a set of input values. The command

```
RECODE ITEM1 TO ITEM3 (0=1) (1=0) (2=-1) (ELSE=COPY)
  INTO DEFENSE WELFARE HEALTH.
```

creates three new variables. Input values other than 0, 1, or 2 are retained. In other words, if a case has value 9 for variable ITEM1, it will have value 9 for variable DEFENSE, and so forth.

Keyword COPY is an output specification only. Input values to be copied can be a range of values, keywords SYSMIS or MISSING, or keyword ELSE. User-missing values are copied, but their missing-value status is not. The MISSING VALUES command should be used to redeclare missing values for the new variables (see Chapter 2).

3.8 Recoding String Variables

If you are recoding a string variable, enclose each value specification in apostrophes or quotation marks, as in:

```
RECODE STATE ('IO'='IA').
```

The following additional rules apply to recoding string variables:

- The keywords THRU, HIGHEST, LOWEST, MISSING, and SYSMIS cannot be used.

- If a value specification applies to more than one variable, all the named variables must have string values of equal length.
- New string values cannot be longer than the variable length as defined on the DATA LIST or STRING command (see Section 3.23).
- If you specify fewer characters than the defined string variable length, SPSS right-pads the value with blanks to the defined length. For example, if the defined length is A3, and you specify a value of 'NO' on the RECODE command, SPSS reads the value as 'NO '.
- Target variables specified with the INTO keyword must already exist on the active system file. To create a new variable, use the STRING command (see Section 3.23) before the RECODE command to declare new string target variables.

3.9 COMPUTE COMMAND

The COMPUTE command creates new variables through numeric transformations of existing variables. COMPUTE names the variable you want to create (the *target variable*) followed by an *expression* defining the variable. For example, the command

```
COMPUTE TOTSCORE=MIDTERM+FINAL+HOMEWORK.
```

defines the new variable TOTSCORE as the sum of the variables MIDTERM, FINAL, and HOMEWORK.

The target variable can be a variable that already exists or a new variable. If the target variable already exists, its values are replaced with those produced by the specified transformation. If it is a new variable, it is added to the end of the dictionary in your active system file.

The expression on the COMPUTE command can use existing numeric variables, constants, arithmetic operators (such as + and −), and functions such as SQRT (square root) and TRUNC (truncate). For example, the command

```
COMPUTE GRADESCR=.35*MIDTERM+.45*FINAL+.2*HOMEWORK.
```

creates a new variable, GRADESCR, that is the weighted average of the variables MIDTERM, FINAL, and HOMEWORK.

3.10 Arithmetic Operators

The following arithmetic operators are available for transforming numeric variables with COMPUTE:

+ *Addition.*
− *Subtraction.*
* *Multiplication.*
/ *Division.*
** *Exponentiation.*

Arithmetic operators must be explicitly specified. You cannot, for example, write (PROPTAX)(100) instead of (PROPTAX)*100.

You can include blanks in an arithmetic expression to improve readability, as in the command

```
COMPUTE TAXTOTAL = PROPTAX + FICA + STATETAX + FEDTAX.
```

Since fairly complex expressions are possible, it is important to keep in mind the order in which operations are performed. Functions (see Sections 3.11 through 3.13) are evaluated first, then exponentiation, then multiplication and division, and, finally, addition and subtraction. Thus, if you specify

```
COMPUTE NEWRATE=SQRT(RATE1)/SQRT(RATE1)+SQRT(RATE3).
```

the square roots (SQRT) are calculated first, then the division is performed, and then the addition.

You can control the order in which operations are performed by enclosing the operation you want executed first in parentheses. Thus, the command

```
COMPUTE NEWRATE=SQRT(RATE1)/(SQRT(RATE1)+SQRT(RATE3)).
```

produces different results than the previous command, since the added parentheses cause the addition to be performed before division. Operations at the same level, as far as order of execution is concerned, are evaluated from left to right. If you are uncertain about the order of execution, you should use parentheses to make the order you want explicit.

3.11 Numeric Functions

Many numeric functions are available with the COMPUTE command. Numeric functions always return numbers (or the system-missing value).

The expression to be transformed by a function is called the *argument*. Most functions have a variable name or variable list as arguments. In numeric functions with two or more arguments, each argument must be separated by a comma. You cannot use blanks alone to separate the variable names, expressions, or constants used as arguments.

For example, to generate the square root of variable X, specify variable X as the argument to the SQRT function, as in SQRT(X). Enclose arguments in parentheses, as in

```
COMPUTE INCOME=TRUNC(INCOME).
```

where the TRUNC function returns the integer portion of variable INCOME. Separate multiple arguments with commas, as in

```
COMPUTE SCALE=MEAN(Q1,Q2,Q3).
```

where the MEAN function returns the mean of variables Q1, Q2, and Q3.

Sections 3.12 through 3.14 discuss arithmetic, statistical, and other functions for the COMPUTE command. These functions can also be used with the IF, SELECT IF, DO IF, ELSE IF, LOOP IF, and END LOOP IF commands, which are discussed later in this chapter.

3.12 Arithmetic Functions

The following arithmetic functions are available:

ABS(arg) *Absolute value.* ABS(SCALE) is 4.7 when SCALE equals 4.7 or −4.7.

RND(arg) *Round the absolute value to an integer and reaffix the sign.* RND(SCALE) is −5 when SCALE equals −4.7.

TRUNC(arg) *Truncate to an integer.* TRUNC(SCALE) is −4 when SCALE equals −4.7.

MOD(arg,arg) *Remainder (modulo) of the first argument divided by the second.* MOD(YEAR,100) is 83 when YEAR equals 1983.

SQRT(arg) *Square root.* SQRT(SIBS) is 1.41 when SIBS equals 2.

EXP(arg) *Exponential. e* is raised to the power of the argument. EXP(VARA) is 7.39 when VARA equals 2.

LG10(arg) *Base 10 logarithm.* LG10(VARB) is .48 when VARB equals 3.

LN(arg) *Natural or Naperian logarithm (base* e*).* LN(VARC) is 2.30 when VARC equals 10.

ARSIN(arg) *Arcsine.* The result is given in radians (alias ASIN). ARSIN(ANG) is 1.57 when ANG equals 1.

ARTAN(arg) *Arctangent.* The result is given in radians (alias ATAN). ARTAN(ANG2) is .79 when ANG2 equals 1.

SIN(arg) *Sine.* The argument must be specified in radians. SINE(VARC) is .84 when VARC equals 1.

COS(arg) *Cosine.* The argument must be specified in radians. COS(VARD) is .54 when VARD equals 1.

All arithmetic functions except MOD have single arguments; MOD has two. The arguments to MOD must be separated by a comma. Arguments can be numeric expressions, as in RND(A**2/B).

3.13 Statistical Functions

Each argument to a statistical function (expression, variable name, or constant) must be separated by a comma. The available statistical functions are:

SUM(arg list) *Sum of the values across the argument list.*

MEAN(arg list) *Mean of the values across the argument list.*

SD(arg list) *Standard deviation of the values across the argument list.*

VARIANCE(arg list) *Variance of the values across the argument list.*

CFVAR(arg list) *Coefficient of variation of the values across the argument list.* The coefficient of variation is the standard deviation divided by the mean.

MIN(arg list) *Minimum value across the argument list.*

MAX(arg list) *Maximum value across the argument list.*

3.14 Other Functions

Other available functions include:

RANGE(arg,arg list) *Return 1 (true) if the value of the first argument is in the inclusive range(s); otherwise, return 0 (false).* The arguments must be separated by commas. The first argument is usually a variable, and the list usually contains pairs of values. For example, NONWORK =RANGE(AGE,1,17,62,99) returns a value of 1 for ages 1 through 17 and ages 62 through 99. The value of NONWORK is 0 for any other value of AGE.

ANY(arg, arglist) *Return 1 (true) if the value of the first argument matches one of the arguments in the list. Otherwise, return 0 (false).* The arguments must be separated by commas. The first argument is usually a variable. For example, PARTIC=ANY(PROJECT, 3, 4, 7, 9) returns a value of 1 if the value of PROJECT is 3, 4, 7, or 9. If PROJECT is any other value, PARTIC has a value of 0.

UNIFORM(arg) *A uniform pseudo-random number.* The random number is uniformly distributed with values varying between 0 and the value of the argument. For example, SAMP1=UNIFORM(150) assigns random values between 1 and 150 to the variable SAMP1 for each case in the active system file.

NORMAL(arg) *A normal pseudo-random number.* The random number is normally distributed, with a mean of approximately 0 and a standard deviation equal to the value of the argument.

VALUE(arg) *Ignore user-missing values.* User-missing values are treated as valid observations and included in any specified calculations.

MISSING(arg) *Return 1 (true) if the value of the argument is missing; otherwise return 0 (false).* The argument is a variable name, and the missing values include both user- and system-missing values.

SYSMIS(arg) *Return 1 (true) if the value of the argument is system-missing; otherwise 0 (false).*

There are also numerous functions that enable you to convert and extract dates and times. For example, you can calculate the number of days between two date variables (see Chapter 2) with the command:

```
COMPUTE DAYDIFF=CTIME.DAYS(VISIT2-VISIT1).
```

SPSS stores all dates as the number of seconds since October 14, 1582. The function CTIME.DAYS converts dates into the number of days. The argument (VISIT2–VISIT1) calculates the difference, in days, between the date variables VISIT2 and VISIT1.

For a complete list of functions, including date and time functions, see the *SPSS Reference Guide*.

3.15 Using Functions in Complex Expressions

You can specify more than one function in an argument as well as combine functions with arithmetic operators. Such arguments will be evaluated in the order described in Section 3.10 or in the order specified by parentheses. For example, if the command

```
COMPUTE PCTTAXES=RND((TAXES/INCOME)*100).
```

is used, TAXES is first divided by INCOME, the result is multiplied by 100, and this result is rounded off to the nearest integer to get the new variable PCTTAXES.

3.16 Missing Values

If a case has missing values for any of the variables used in a COMPUTE arithmetic expression, the case is assigned the system-missing value for the computed variable. For example, if the command

```
COMPUTE AGECUBE=AGE**3.
```

is used, the AGECUBE variable will be system-missing for any case with a missing value for AGE.

A case is also assigned the system-missing value for a computed variable when the specified operation is not defined for that case. For example, if the command

```
COMPUTE PCTTAXES=(TAXES/INCOME)*100.
```

is used, a case with the value 0 for INCOME is assigned the system-missing value for PCTTAXES because division by 0 is not defined. If the result of an expression cannot be represented on the computer (even when valid values are used in the expression itself), the system-missing value is assigned to the new variable.

The assignment of missing values is treated differently for numeric functions. For example, the command

```
COMPUTE MEANSCOR=(SCORE1+SCORE2+SCORE3)/3.
```

will return a missing value for MEANSCOR if a case has a missing value for any one of the variables SCORE1, SCORE2, or SCORE3. However, the command

```
COMPUTE MEANSCOR=MEAN(SCORE1, SCORE2, SCORE3).
```

will return a numeric value unless a case has missing values for all three specified variables.

For a complete discussion of the treatment of missing values with numeric functions, see the *SPSS Reference Guide*.