3

# Data Modifications

## 3.1 Introduction

This chapter deals with commands for modifying data files. Most of these commands are common applications and are widely used in research. They will be addressed here under 26 different sections. This way, more experienced users can find specific commands quickly and easily. For readers who are not familiar with the commands, it is recommended to read the entire chapter to appreciate the full potential of SPSS and working with syntax. The commands that are discussed here are summarized at the end of the chapter. For those who want to reproduce the results in any section, please always execute the command in Section 3.2 that opens the data set together with the commands that are listed under **example** and/or **results** in the particular section(s).

## 3.2 GET

### Function

The GET command is used to open an SPSS data file with the extension *.sav*. In general, this will be the first command in a syntax file. It is also possible to exclude a number of variables within this command or to rename variables.

### Structure

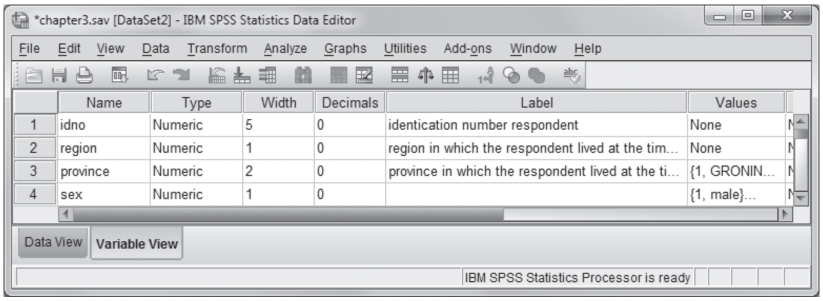
The GET command is followed by the FILE subcommand and is specified as a string, specifying the directory (c:\data); for Macintosh computers, it is only \data and the name of the data file (chapter3.sav) (Figure 3.1).

### Example

```
GET FILE "c:/data/chapter3.sav".
* Macintosh users:.
GET FILE "/data/chapter3.sav".
```

### Results

**Figure 3.1** The File "chapter3.sav" Is Opened With GET


*chapter3.sav [DataSet2] - IBM SPSS Statistics Data Editor
File Edit View Data Transform Analyze Graphs Utilities Add-ons Window Help
Data View
Variable View
IBM SPSS Statistics Processor is ready

| | Name | Type | Width | Decimals | Label | Values |
|---|---|---|---|---|---|---|
| 1 | idno | Numeric | 5 | 0 | identication number respondent | None |
| 2 | region | Numeric | 1 | 0 | region in which the respondent lived at the tim... | None |
| 3 | province | Numeric | 2 | 0 | province in which the respondent lived at the ti... | {1, GRONIN... |
| 4 | sex | Numeric | 1 | 0 | | {1, male}... |

### Optional Subcommands

The KEEP subcommand allows you to specify the variables that you want to include in the file that is to be opened. You must list these variables directly (after a space) after KEEP. The remaining variables in the data file that are not listed are removed from future analyses. If this subcommand is omitted, all variables from the file are included.

Similarly, the DROP subcommand ignores the specified variables and includes those not listed instead.

The RENAME command is, unsurprisingly, used to rename variables. This command is followed by the name of an existing variable, an equal sign (=), and the new variable name. If more variables need renaming, you can then leave a space and specify the next variable that needs to be renamed.

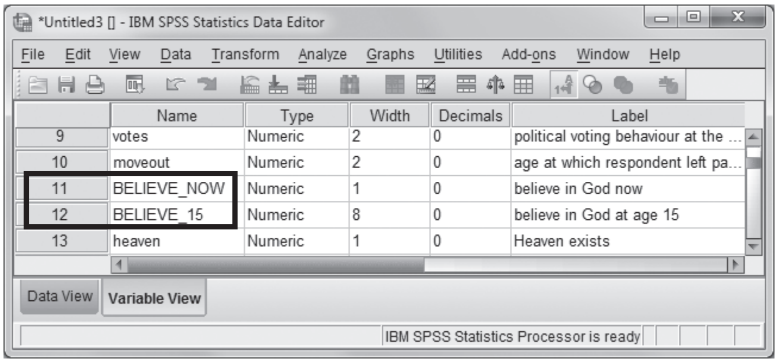
### Extensive Example

```
GET FILE "c:/data/chapter3.sav"
 /DROP LIST FUTURE
 /RENAME GOD2 = BELIEVE_NOW GOD1 = BELIEVE_15.
EXECUTE.
```

This command opens the "chapter3.sav" file. The variables LIST and FUTURE are removed, and the variables GOD1 and GOD2 are given more meaningful names (i.e., BELIEVE_NOW and BELIEVE_15; note the use of underscores) (Figure 3.2). Finally, the syntax ends with the EXECUTE command. When using the KEEP and DROP subcommands, some versions of SPSS display the information in the data window only *after* the EXECUTE command is executed, which can be irritating for some users. See Section 3.4 for more information (p. 26).

### Results

**Figure 3.2** Variables "God1" and "God2" Renamed and LIST/FUTURE Omitted


*Untitled3 [] - IBM SPSS Statistics Data Editor
File Edit View Data Transform Analyze Graphs Utilities Add-ons Window Help
Data View
Variable View
IBM SPSS Statistics Processor is ready

| | Name | Type | Width | Decimals | Label |
|---|---|---|---|---|---|
| 9 | votes | Numeric | 2 | 0 | political voting behaviour at the ... |
| 10 | moveout | Numeric | 2 | 0 | age at which respondent left pa... |
| 11 | BELIEVE_NOW | Numeric | 1 | 0 | believe in God now |
| 12 | BELIEVE_15 | Numeric | 8 | 0 | believe in God at age 15 |
| 13 | heaven | Numeric | 1 | 0 | Heaven exists |

## 3.3 SAVE △

### Function

The SAVE command saves the data file. This is used after recoding variables, constructing scales, or other data operations. It is advisable not to overwrite the original data file (see Figure 3.3 on page 24). As mistakes are bound to occur, it is better to keep the original file as a backup—so rename the file you want to save. SAVE can also be used to save a file that contains only relevant variables from the current file. This makes the data set easier for others to use, and it also reduces the size (in bytes) of the file, which makes it easier for direct e-mailing. We like to add that there are free transfer programs available on the Internet (up to 2 GB or more) in case the .sav file is still too large.

### Structure

The SAVE command is always followed by the OUTFILE subcommand, which in turn is followed by the directory and file name as a string.

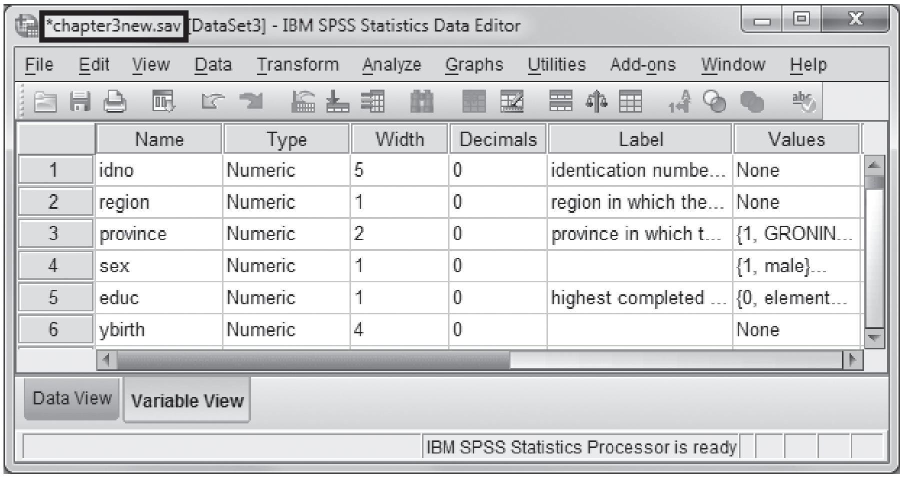
## Example

```
SAVE OUTFILE "c:/data/chapter3new.sav".
```

## Results

```
GET FILE "c:/data/chapter3new.sav".
```

Figure 3.3 The File "chapter3new.sav" Is Opened With GET


*chapter3new.sav [DataSet3] - IBM SPSS Statistics Data Editor
File Edit View Data Transform Analyze Graphs Utilities Add-ons Window Help
Data View
Variable View
IBM SPSS Statistics Processor is ready

| | Name | Type | Width | Decimals | Label | Values |
|---|---|---|---|---|---|---|
| 1 | idno | Numeric | 5 | 0 | identication numbe... | None |
| 2 | region | Numeric | 1 | 0 | region in which the... | None |
| 3 | province | Numeric | 2 | 0 | province in which t... | {1, GRONIN... |
| 4 | sex | Numeric | 1 | 0 | | {1, male}... |
| 5 | educ | Numeric | 1 | 0 | highest completed ... | {0, element... |
| 6 | ybirth | Numeric | 4 | 0 | | None |

## Optional Subcommands

As with the GET command, the KEEP subcommand can be added to save a limited number of variables. The KEEP subcommand should be followed by a list of variables that you wish to save.

As mentioned, the DROP subcommand is the counterpart of KEEP: The variables specified here will *not* be saved. This subcommand is used if you want to omit a number of variables through the syntax. Just open the file with the GET command and save it with the redundant variables specified with DROP.

The RENAME command can be used to save variables under a different name. The name of an existing variable, an equal sign (=), and the new variable name follows this RENAME command. If you need to rename more variables, you can leave a space and specify the next variable that needs to be renamed.

## Extensive Example

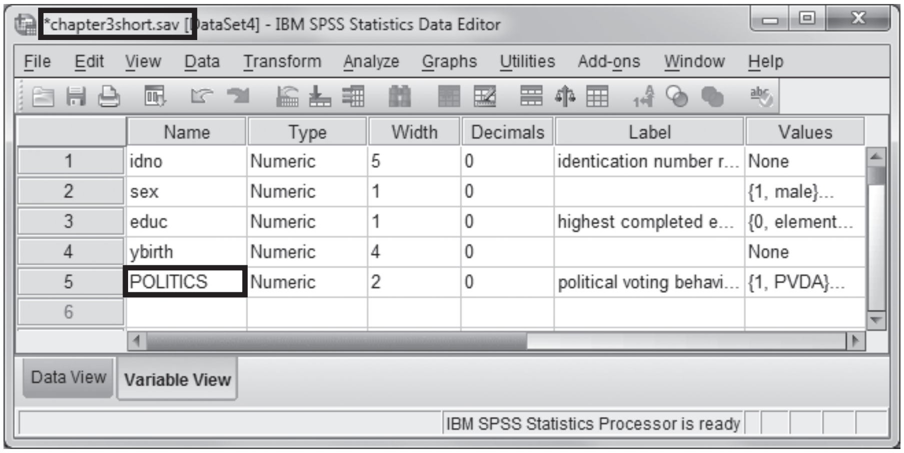
```
SAVE OUTFILE "c:/data/chapter3short.sav"
 /KEEP IDNO SEX EDUC YBIRTH VOTES
 /RENAME VOTES = POLITICS.
```

Following this command, five variables are saved from the old file into the new file "chapter3short.sav," and the variable VOTES is now renamed as POLITICS (Figure 3.4).

## Results

```
GET FILE "c:/data/chapter3short.sav".
```

**Figure 3.4** The File "chapter3short.sav" Opened With GET


*chapter3short.sav [DataSet4] - IBM SPSS Statistics Data Editor
File Edit View Data Transform Analyze Graphs Utilities Add-ons Window Help
Data View
Variable View
IBM SPSS Statistics Processor is ready

| | Name | Type | Width | Decimals | Label | Values |
|---|---|---|---|---|---|---|
| 1 | idno | Numeric | 5 | 0 | identication number r... | None |
| 2 | sex | Numeric | 1 | 0 | | {1, male}... |
| 3 | educ | Numeric | 1 | 0 | highest completed e... | {0, element... |
| 4 | ybirth | Numeric | 4 | 0 | | None |
| 5 | POLITICS | Numeric | 2 | 0 | political voting behavi... | {1, PVDA}... |
| 6 | | | | | | |

*Note:* If you want to proceed to the next section, the file "chapter3.sav" has to be reopened (so execute the GET command from Section 3.2 again).

## 3.4 EXECUTE

### Function

It is more efficient for SPSS to postpone some data transformations until they are actually required. As discussed in the GET command example, some versions of SPSS only display the data in the data window *after* you run the EXECUTE command. However, when a statistical procedure is run—such as FREQUENCIES—SPSS will run an (invisible) EXECUTE command instead. So if you use the EXECUTE command, it will release all SPSS operations that are still queued.

### Structure

EXECUTE does not have any subcommands, so you can just use the command together with a period (.) to end the command.

### Example

```
EXECUTE.
```

## 3.5 VARIABLE LABELS

### Function

VARIABLE LABELS allows you to make detailed descriptions ("labels") for each variable. These labels will appear in the output window, making the results more readable. It is therefore possible to clarify ambiguous variable names with a clear label using this command.

### Structure

The VARIABLE LABELS command (please note that commands can have two words) is followed by the name(s) of one or more variables. A string variable containing the label is placed after each variable.

### Example

```
VARIABLE LABELS
LEAVECHURCH "age at which the respondent left the church"
YBIRTH "year of birth of the respondent".
```

To see if the variable labels were applied properly, you may run a frequency table of the variables (Table 3.1):

### Results

```
FREQUENCIES
 /VARIABLES LEAVECHURCH YBIRTH.
```

**Table 3.1** Variable Labels for LEAVECHURCH and YBIRTH

**Statistics**

| | | leavechurch age at which the respondent left the church | ybirth year of birth of the respondent |
|---|---|---|---|
| N | Valid | 1974 | 1974 |
| | Missing | 0 | 0 |

## 3.6 VALUE LABELS △

### Function

The VALUE LABELS command is similar to the VARIABLE LABELS command in that it also assigns a label. With this command, however, the label is not assigned to a variable but to the categories of that variable. This is especially useful for variables with a nominal or ordinal measurement level, where the meaning of the categories cannot be deduced from the numbers of the categories. The categories are named in such a way that their meaning is clear from reading the results in the output file.

### Structure

The VALUE LABELS command is followed by the name of one or more variables. A list follows this command with a value and the label for that value in quotes. After a slash, more variables can be specified with the values and labels.

### Example

```
VALUE LABELS SEX 1 "Male" 2 "Female"
 /REGION 1 "North" 2 "East" 3 "West" 4 "South".
```

To see if the value labels were applied properly, you may run a frequency table of the two variables (Table 3.2):

### Results

```
FREQUENCIES
 /VARIABLES SEX REGION.
```

**Table 3.2** Value Labels of SEX and REGION

**Sex**

| | | Frequency | Percent | Valid Percent | Cumulative Percent |
|---|---|---|---|---|---|
| Valid | 1 Male | 945 | 47.9 | 47.9 | 47.9 |
| | 2 Female | 1029 | 52.1 | 52.1 | 100.0 |
| | Total | 1974 | 100.0 | 100.0 | |

**Region where the respondent was interviewed**

| | | Frequency | Percent | Valid Percent | Cumulative Percent |
|---|---|---|---|---|---|
| Valid | 1 North | 217 | 11.0 | 11.0 | 11.0 |
| | 2 East | 441 | 22.3 | 22.3 | 33.3 |
| | 3 West | 891 | 45.1 | 45.1 | 78.5 |
| | 4 South | 425 | 21.5 | 21.5 | 100.0 |
| | Total | 1974 | 100.0 | 100.0 | |

## △ 3.7 MISSING VALUES

### Function

The units of analysis (often respondents) can have missing (unfilled) scores on variables and/or invalid scores (scores that may not be useful in the analysis). In SPSS, missing scores (i.e., no score at all) are referred to as "system missing." They occur as blanks in your data file and are automatically left out of any analysis. Invalid scores (labeled "missing values"), however, must be defined by the user. The MISSING VALUES command transforms scores into missing values (invalid scores). For example, the scores that fall into the categories "do not know" or "will not say" are often transformed into missing values first. Otherwise, SPSS will use them in analysis, which may be senseless (e.g., when calculating means).

### Structure

The MISSING VALUES command is followed by the name of the variable (or multiple variables) and the values of the missing categories in parentheses. If there are several variables that share the same missing value(s), it is possible to specify the variables first and then specify the values that must be treated as invalid (see the example below). If you have many variables, instead of typing all the variable names, it is more efficient to use SPSS syntax statements like LOWEST, HIGHEST, or THRU (see Section 3.9). If you want to undo the missing values (but not the "system missings"), include parentheses without a value (i.e., (); see our example for HEAVEN in the box below).

### Example

```
MISSING VALUES VOTES (14 15 16)
LEAVECHURCH CHURCHMEMBER (99).
MISSING VALUES HEAVEN ().
```

In this example, the categories/values 14, 15, and 16 ("will not say," "do not know," "does not vote") of the VOTES variable are specified as invalid, while the value 99 ("does not apply") is made invalid for the LEAVECHURCH and CHURCHMEMBER variables. Finally, all invalid values of HEAVEN are turned into valid observations (Table 3.3).

### Results

```
FREQUENCIES
 /VARIABLES LEAVECHURCH CHURCHMEMBER VOTES HEAVEN.
```

**Table 3.3** Missing Values for LEAVECHURCH, CHURCHMEMBER, VOTES and Validation of Initially Invalid Values for HEAVEN

**Statistics**

| | | **leavechurch age at which the respondent left the church** | **churchmember church membership** | **votes political voting behavior at the time of interview** | **heaven heaven exists** |
|---|---|---|---|---|---|
| N | Valid | 443 | 1966 | 1647 | 1974 |
| | Missing | 1531 | 8 | 327 | 0 |

## 3.8 DISPLAY

### Function

The DISPLAY command displays information about variables, including, among other things, variable labels to gain insights into the meaning of variables and their measurement levels, and so on.

### Structure

The DISPLAY command is followed by an instruction to indicate what information is desired. DISPLAY can be followed by the VARIABLES subcommand and a list of selected variables. Leaving out this subcommand displays all available information. The LABELS command lists all variable labels, and the DICTIONARY command displays label categories, information on missing values, and the measurement levels of the variables, as well as the labels of the variables.

### Example

```
DISPLAY DICTIONARY
 /VARIABLES SEX MARITALSTAT EDUC.
```

### Results

The output begins with a table displaying the names of the variables followed by its position in the data file (in the above example, MARITALSTAT is the 14th variable in the file). Displayed next are label, measurement level,[1] and features of the format in which the variable is stored. The line "Missing Values" indicates what values are defined as missing (see Section 3.7, p. 28). A list of values and labels follows this table. Values that are defined as missing are presented as such (Table 3.4).

**Table 3.4** Information About SEX, MARTIALSTAT, and EDUC

**Variable Information**

| Variable | Position | Label | Measurement Level | Column Width | Alignment | Print Format | Missing Values |
|---|---|---|---|---|---|---|---|
| sex | 4 | <None> | Nominal | 8 | Right | F1 | |
| maritalstat | 14 | Marital status | Nominal | 8 | Right | F1 | 8, 9 |
| educ | 5 | Highest completed education | Ordinal | 8 | Right | F1 | |

*Note:* Variables in the working file.

**Variable Values**

| Value | | Label |
|---|---|---|
| sex | 1 | Male |
| | 2 | Female |
| maritalstat | 1 | Not married |
| | 2 | Married |
| | 3 | Divorced |
| | 4 | Widow |
| | 8[a] | Won't say |
| | 9[a] | Unknown |
| educ | 1 | Elementary school |
| | 2 | Middle school |
| | 3 | Junior high school |
| | 4 | Senior high school |
| | 5 | Vocational school |
| | 6 | College |
| | 7 | Bachelor degrees |
| | 8 | Master degrees |

*Note:* a. Missing value.

[1]The measurement level in SPSS is "scale" (=ratio/interval) by default and may be adjusted by the user if this is not correct! For the analysis and modification of data, it is not relevant which measurement level is assigned to the variable, with the exception of certain graphs and tables.

## △ 3.9 RECODE

### Function

RECODE is used to give variables new codes or values. For each value or set of values, a new value is specified. The new variable will usually be a nominal or an ordinal variable, while the original variable can be of any measurement level.

### Structure

The RECODE command is followed by the name of the original variable. This is then followed by a value (or set of values) in parentheses, an equal sign (=), and then the new value. We recommend using INTO at the end of the command followed by a new name for the recoded variable. If this is not done, the old variable will be overwritten and the old values will be lost! This kind of recoding is thus discouraged. Please note that when using INTO the variable labels are not copied. You will have to add them yourself with the VALUE LABELS command (Section 3.6).

To give a set of values of the original variable the same value, you can put a number of values in a row—separated by spaces—before the equal sign (=). For example, when you use (2 3 4 = 0), respondents with a value of 2, 3, or 4 will get the value 0 in the new recoded variable. You can also assign a new value to a whole interval (which is vital when recoding most interval or ratio variables) by using the command THRU. In addition, you can use the commands LOWEST and HIGHEST to indicate the lowest or highest values of the original variable. The statement "LOWEST THRU 5 = 3" means that respondents with values lesser than or equal to 5 on the original value will get a value 3 in the new variable. When recoding interval variables, it is customary to equate the upper and lower limits so that no values are missed: (0 THRU 100=1) (100 THRU 200=2). All numbers from 0 to 99.999 are recoded to 1, whereas values from 100 to 200 are recoded into 2.

The MISSING and SYSMIS commands can be used to recode invalid or missing values. SYSMIS means that the respondent does not have a score ("a blank") on the variable ("a system missing") and is excluded from statistical analyses. MISSING indicates that the respondent does have a score, but an invalid one, and it is thus again excluded from the analysis. So "MISSING=999" and "SYSMIS=999" in a recode command mean that all missing or invalid values are recoded to the value 999.

All values that have not been recoded by the RECODE command are specified as system missing by SPSS. You can use the operator ELSE to prevent this. ELSE refers to all values that are not recoded by previous lines in the command (see the example below). There are also special cases for the value after the equal sign (=). COPY allows the variable to keep a certain old value. For example, "16 THRU HIGHEST = COPY" will leave values of 16 and higher unchanged in the new variable.

## Example

```
RECODE VOTES (1 5 12 = 1) (2 4 9 10 = 2)
(3 6 THRU 8 11 = 3) (ELSE = 999) INTO POLITICS.
```

The variable VOTES is made less ambiguous with this command. The Dutch political parties are divided into left wing (1), center of politics (2), and right wing (3). All other values ("other party," "will not say," "do not know," and "does not vote") are coded (valid) value 999 (Table 3.5).

## Results

```
FREQUENCIES
 /VARIABLES POLITICS.
```

**Table 3.5** Recoding the Variable VOTES Into a New Variable POLITICS

| POLITICS | | | | | |
|---|---|---|---|---|---|
| | | **Frequency** | **Percent** | **Valid Percent** | **Cumulative Percent** |
| Valid | 1.00 | 529 | 26.8 | 26.8 | 26.8 |
| | 2.00 | 609 | 30.9 | 30.9 | 57.6 |
| | 3.00 | 506 | 25.6 | 25.6 | 83.3 |
| | 999.00 | 330 | 16.7 | 16.7 | 100.0 |
| | Total | 1974 | 100.0 | 100.0 | |

## 3.10 COMPUTE

### Function

The COMPUTE command is used to construct a variable using a formula. In Chapter 1 (Section 1.6), we used an example where age is calculated from the year of birth and the year of the interview. Another example is constructing "dummy variables"—that is, dichotomous variables with the values 0 and 1, commonly used in regression analysis (Section 4.10).

### Structure

The name of the variable that is to be created follows the COMPUTE command, which is then followed by an equal sign (=) and at last the formula. If the name of the variable already exists in the data set, the old values will be replaced, but in all other cases, a new variable will be created. (An introduction on arithmetic expressions and formulas to create variables can be found on p. 13 and in the appendix on p. 117). When the command is run, the (new) variable receives the new values. These values are displayed in the data window after the EXECUTE command (see p. 26). When a variable that is included in the formula contains many "system missings" or "missing values" (see p. 28), the result of the calculation will generally be a system missing as well. This is logical because the result of "missing + 4" is unknown, so SPSS turns this into missing. This means that the new variable will have many missings when it is the product of several variables with missing or invalid values. To overcome this drawback we use MEAN (see p. 118), whereby the means are based on a fixed minimum number of variables without missing or invalid values and without taking into account missing values on the other variables.

### Example

```
COMPUTE OWNHOME = (1995 – YBIRTH) – MOVEOUT.
```

MOVEOUT is a variable that indicates the age at which the respondent moved out of the parental house. The part of the formula in parentheses calculates the age of the respondent, as we have seen before. The result of the formula is the number of years that the respondent has not lived with his parents since moving. Respondents who still live with their parents will have a missing value on MOVEOUT and thus a system missing for OWN-HOME (Table 3.6).

### Results

```
FREQUENCIES
 VARIABLES OWNHOME
 /STATISTICS DEFAULT.
```

**Table 3.6** Mean, Standard Deviation, and Minimum and Maximum Value for the Variable OWNHOME

**Statistics**

| **OWNHOME** | | |
|---|---|---|
| N | Valid | 1831 |
| | Missing | 143 |
| Mean | | 22.0437 |
| Std. Deviation | | 12.60580 |
| Minimum | | .00 |
| Maximum | | 68.00 |

## 3.11 COUNT △

### Function

This command counts the number of times a certain value appears in a list of variables for each respondent. This results in a variable with a minimum value of 0 and a maximum value that equals the number of variables in the list. Generally, this command is used to count the number of missing or invalid scores a respondent has on a number of variables. It is also used to count the number of positive or negative responses that a respondent has given to a sample of questions.

### Structure

The results are stored in a new variable that needs to be given a name following the COUNT command. This is followed by an equal sign (=), a blank, and then the list of variables that need values counted, and the value(s) that need counting in parentheses. If the values that need counting are not the same for every variable, it is possible to indicate a second list of variables and values. Indicating the values that need to be counted is done in the same way as indicating values in the RECODE command (for details, see p. 32). Values should be separated by spaces and the words HIGHEST, LOWEST, THRU, MISSING, and SYSMIS can be used to count the highest value, the lowest value, an interval of values, a missing value, and a system-missing value, respectively.

### Example

```
COUNT MISSINGS = MARITALSTAT MOVEOUT (MISSING) WORKHOURS (999).
```

This creates a new variable called MISSINGS that contains the number of invalid or missing scores (MISSING and SYSMIS) on the indicated variables (see Section 3.7 on p. 28 for an explanation of the difference between MISSING and SYSMIS). The value 999 is counted as an invalid score for the variable WORKHOURS. Please note that the value 999 is not defined as a missing value and therefore is not recognized as such by SPSS. If you want to calculate the correct sample mean of the variable WORKHOURS, you will first have to turn the score 999 missing (invalid score); see p. 28–29 for an example (Table 3.7).

### Results

```
FREQUENCIES
 /VARIABLES MISSINGS.
```

**Table 3.7** Number of Respondents With 0 (1786), 1 (159), 2 (27), or 3 (2) “missings” on MARITALSTAT, MOVEOUT, and/or WORKHOURS

**MISSINGS**

| | | Frequency | Percent | Valid Percent | Cumulative Percent |
|---|---|---|---|---|---|
| Valid | .00 | 1786 | 90.5 | 90.5 | 90.5 |
| | 1.00 | 159 | 8.1 | 8.1 | 98.5 |
| | 2.00 | 27 | 1.4 | 1.4 | 99.9 |
| | 3.00 | 2 | .1 | .1 | 100.0 |
| | Total | 1974 | 100.0 | 100.0 | |

## 3.12 IF

### Function

The IF command is used to assign values to a (new) variable when certain conditions are met. Suppose you want to construct a nominal variable "family composition" including the following categories: "unmarried without children," "married without children," "unmarried with children," and "married with children." You can achieve this with two variables: (1) being married (yes/no) and (2) having children (yes/no). From the combinations no-no, yes-no, no-yes, and yes-yes, the four categories of the new variable "family composition" can be constructed using the IF command (also see http://www.ru.nl/mt/syntax/home under "IF command").

### Structure

A condition in parentheses follows the IF command. A brief explanation of these conditions can be found in the section on logical expressions (p. 13) and for a more extensive explanation, we refer you to the appendix (p. 121). The condition is followed by the variable name, a value, and an equal sign (=), and finally, the new value. The variable is constructed if it does not already exist, while all units of analysis (i.e., respondents) for which this condition does not apply will be assigned a "system missing" for this variable (see p. 28 for an explanation for "missing"). All other units will, of course, get the value that was indicated in the IF command.

### Example

```
COMPUTE DECREASE = 0.
IF (CHURCHNOW < CHURCH15) DECREASE = 1.
```

In this example, a variable is constructed that indicates a decrease in church attendance during one's life. First, the variable is set to 0 for all respondents, which, in this case, refers to respondents who do not attend church less nowadays than they did at 15 years of age. The IF command will assign the value 1 to all respondents who at the time of the interview attended church less than when they were 15 years old (Table 3.8).

### Results

```
FREQUENCIES
 /VARIABLES DECREASE.
```

**Table 3.8** Frequency Distribution of the Variable DECREASE

| DECREASE | | | | | |
|---|---|---|---|---|---|
| | | **Frequency** | **Percent** | **Valid Percent** | **Cumulative Percent** |
| Valid | .00 | 1183 | 59.9 | 59.9 | 59.9 |
| | 1.00 | 791 | 40.1 | 40.1 | 100.0 |
| | Total | 1974 | 100.0 | 100.0 | |

## 3.13 WRITE

### Function

The WRITE command is used to store data (as in Section 3.3; see p. 23). The difference, however, is that the SAVE command saves data in the SPSS format (.sav), which most programs cannot open. WRITE will save data as a universal readable text file. This file contains one or more lines for each respondent on which the values of the variables are lined up in a row. The command is mostly used to save data files to programs that can read these text (ASCII) files.

### Structure

The OUTFILE subcommand follows the WRITE command with the name of the file that is to be saved, a slash, the variables, and the format in which the file needs to be saved. Specifying the format is necessary because the values should now appear separated from each other by a (column) space or spaces. If left out, SPSS will produce a file wherein all the values of the variables are presented in a row without spaces, which is practically of no use. The variable format is indicated in parentheses prior to the variable name. This "description" starts with a letter indicating the type of variable. An "f" indicates a numerical variable, and an "a" indicates a string variable (e.g., the name of the place of residence). This is followed by a value that indicates the number of characters used by the variable. Always add one extra to the number of characters the variable needs, so that the variables are separated by a space. For a variable value with decimals, a period is placed after the figure indicating the number of characters, followed by the number of digits after the dot. The number −16.589, for example, is written as f8.3. There are eight positions necessary for this number: one space (to separate it from the former variable), the minus sign, two digits, a dot (separation sign for the decimals), and three decimals. The space can also be indicated by adding "1x" (see example). The file that is created with the WRITE command is only saved after the EXECUTE command or when a statistical procedure is executed (see p. 26).

## Example

```
WRITE OUTFILE "c:/data/example.txt"
 /IDNO (f4) SEX (f2) RELIGION (f5.2, 1x) SURVEY (a6).
EXECUTE.
```

In this example, the file "example.txt" is saved containing the variables in the command separated by spaces. The "(f4)" of IDNO indicates that four columns are used by this variable. The variable IDNO from the file "chapter3.sav" indeed covers up to four digits, and there is no need for a space to separate, because it is the first variable in the file. The variable SEX is separated from IDNO by a space, hence "f2" (1 space and 1 digit: 0 or 1 in this case). The variable RELIGION (a Likert scale) consists of four digits, two behind the dot, and needs an extra space to be separated from the variable SEX; therefore it is written as "f5.2." "1x" is added to ensure the space between RELIGION and SURVEY. The variable SURVEY indicates the specific version of the survey and consists of a string with a code (there were two versions of the survey, denoted as 1995A and 1995B). Because this variable uses six positions, it has to be written as "a6." It is not possible to write "a7" to create a space between RELIGION and SURVEY. You can open the file "example.txt" with a word processing program, like WordPad, Notepad, or Word. This command will create a file containing a table with four variables per respondent (see Table 3.9). Use the DATA LIST command to open the file in SPSS and check whether the command was properly executed (see Section 3.14).

**Table 3.9** "example.txt" Opened in Word

```
 1 1 1.20 1995BP

 2 2  .00 1995AP

 3 2 1.00 1995BP

 7 2  .80 1995B

10 2 1.25 1995AP

11 1  .00 1995BP

13 1 1.20 1995BP

15 2  .80 1995BP
```

*Note:* The first eight rows are shown.

## △ 3.14 DATA LIST

### Function

The GET command (see p. 21) cannot open a text file with respondent information listed without formatting codes (e.g., when the file is made with WRITE; see p. 38). This command will only open .sav files. The DATA LIST command, however, can open such text files, but it has to be clearly specified how the file is to be opened, as it only contains numbers, without any information regarding what the number means.

### Structure

The DATA LIST command is followed by the FILE subcommand, with the name of the file that is to be opened specified as a string. The file name is followed by a slash, and the variable names that are to be read from the file, as well as the format in which they are stored in the file. (See Section 3.13 (p. 38) for more information on the variable formats.) The syntax used for writing the data (see the WRITE command) can also be used to open the file. You only have to replace WRITE with DATA LIST (see the example below). The data will be displayed following the EXECUTE command or the execution of a statistical procedure (see p. 26) (Figure 3.5).

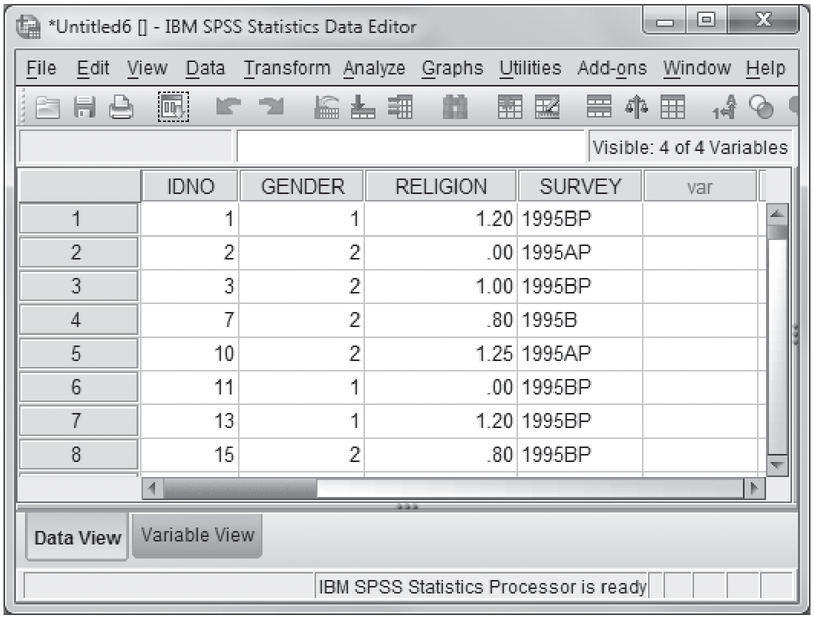
## Example

```
DATA LIST FILE "c:/data/example.txt"
 /IDNO (f4) SEX (f2) RELIGION (f5.2, 1x) SURVEY (a6).
EXECUTE.
```

*Note:* This example requires the use of the file saved using the WRITE command (see Section 3.13).

## Results

**Figure 3.5** The File "example.txt" Opened With DATA LIST (First Eight Rows)


*Untitled6 [] - IBM SPSS Statistics Data Editor
File Edit View Data Transform Analyze Graphs Utilities Add-ons Window Help
Visible: 4 of 4 Variables
Data View
Variable View
IBM SPSS Statistics Processor is ready

| | IDNO | GENDER | RELIGION | SURVEY | var |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1.20 | 1995BP | |
| 2 | 2 | 2 | .00 | 1995AP | |
| 3 | 3 | 2 | 1.00 | 1995BP | |
| 4 | 7 | 2 | .80 | 1995B | |
| 5 | 10 | 2 | 1.25 | 1995AP | |
| 6 | 11 | 1 | .00 | 1995BP | |
| 7 | 13 | 1 | 1.20 | 1995BP | |
| 8 | 15 | 2 | .80 | 1995BP | |

*Note:* The next section requires the use of the file "chapter3.sav"; please execute the GET command from Section 3.2 again.

## 3.15 DO IF/ELSE IF/ELSE/END IF

### Function

The DO IF command marks the beginning of a set of commands that are only executed if the condition behind the DO IF command is true. END IF marks the end of the set. The command resembles the IF command (p. 37), but instead of only assigning a value to a variable, a variety of data transformations can be executed here. This command is not meant to select respondents for analysis. (For more details, see sections 3.16, p. 44), and 3.17, p. 45.) The accompanying ELSE IF and ELSE commands specify transformations that will be executed when the condition behind DO IF is not true.

### Structure

The command *always* starts with DO IF, followed by a logical expression in parentheses (for an explanation on logical expressions see p. 13 and the appendix on p. 121). The DO IF command ends with a period. This is followed by the transformations to be executed for respondents who meet the condition in the logical expression. Last, DO IF is closed with END IF. It is also possible, however, to add more transformations by including ELSE IF followed by the expression in parentheses, before you use the END IF command. You can even use more than one ELSE IF command. The commands in ELSE IF are executed if the conditions before this command have not been true while the condition of this command is true. Finally, before closing the END IF command, you can add ELSE (without a condition). The commands in ELSE will be executed for all respondents with yet unmet conditions. You can use this construction for recoding or saving the data of separate groups with WRITE (p. 38)—for example, multisampling analysis in LISREL.

*Note:* The whole structure between DO IF and END IF has to be executed at once. If only partly executed, later commands may become part of the DO IF command. We recommend finishing the whole DO IF/END IF command with EXECUTE, so that you can be sure the data transformations are actually executed.

### Example

```
DO IF (EDUC < 3).
 WRITE OUTFILE "c:/data/low.txt"
 /VOTES (f3) SEX(f2).
ELSE IF (EDUC < 5).
 WRITE OUTFILE “c:/data/middle.txt”
 / VOTES (f3) SEX (f2).
ELSE.
 WRITE OUTFILE “c:/data/high.txt”
 / VOTES (f3) SEX (f2).
END IF.
EXECUTE.
```

The commands write the variables VOTES and SEX to three separate files depending on the respondent’s educational level. The command DO IF ensures that only respondents with a score lower than 3 on EDUC will be placed in the “low.txt” file. The ELSE IF command selects respondents with a score lower than 5 on EDUC. Because the respondents with a score lower than 3 are already placed in the “low.txt” file, only respondents with scores 3 and 4 will be placed in “middle.txt.” Respondents with a score of 5 and higher are the only ones left, and consequently are placed in the “high.txt” file, with the ELSE command. The block ends with END IF, and EXECUTE ensures that the transformations are actually made. Please note that the WRITE command is indented with two spaces. The advantage of this is that it is immediately clear what transformations are associated with each condition.

## Results

The three files can be opened again using DATA LIST (Section 3.14). The male/female distribution of the opened “middle.txt” file is displayed below (Table 3.10).

```
DATA LIST FILE “c:/Data/middle.txt”
 /VOTES (f3) SEX (f2).
FREQUENCIES SEX.
```

**Table 3.10** Male/Female Distribution in the File "middle.txt"

| SEX | | | | | |
|---|---|---|---|---|---|
| | | Frequency | Percent | Valid Percent | Cumulative Percent |
| Valid | 1 | 287 | 46.9 | 46.9 | 46.9 |
| | 2 | 325 | 53.1 | 53.1 | 100.0 |
| | Total | 612 | 100.0 | 100.0 | |

*Note:* The file "chapter3.sav" will be used in the next section; please execute the GET command from Section 3.2 again.

## △ 3.16 SELECT IF

### Function

SELECT IF is used to delete respondents *permanently* from the data file. All respondents (cases) who do not meet a certain condition are removed from the file. This could be useful to limit the data to a subset of respondents or to remove respondents with extreme scores (so-called outliers). However, the disadvantage of using this command is that if you save the new data file, the data of the excluded are deleted for good. For that reason, the TEMPORARY (Section 3.20, p. 49) and FILTER (Section 3.17, p. 45) commands can be used to temporarily exclude respondents instead.

### Structure

A logical expression (see p. 13 and p. 121) in parentheses follows SELECT IF. All respondents who do not meet that condition are excluded (you can make this visible in the data window after running an EXECUTE command!).

### Example

```
SELECT IF (SEX = 1).
EXECUTE.
```

The above command restricts future data operations and statistical analyses to all males only in the data file (Table 3.11).

## Results

```
FREQUENCIES
 /VARIABLES EDUC.
```

**Table 3.11** Frequency Distribution of EDUC After Selecting Men Only

| educ highest completed education | | | | |
|---|---|---|---|---|
| | **Frequency** | **Percent** | **Valid Percent** | **Cumulative Percent** |
| 1 Elementary school | 72 | 7.6 | 7.6 | 7.6 |
| 2 Middle school | 170 | 18.0 | 18.0 | 25.6 |
| 3 Junior high school | 83 | 8.8 | 8.8 | 34.4 |
| 4 Senior high school | 204 | 21.6 | 21.6 | 56.0 |
| 5 Vocational school | 50 | 5.3 | 5.3 | 61.3 |
| 6 College | 79 | 8.4 | 8.4 | 69.6 |
| 7 Bachelor degrees | 191 | 20.2 | 20.2 | 89.8 |
| 8 Master degrees | 96 | 10.2 | 10.2 | 100.0 |
| Total | 945 | 100.0 | 100.0 | |

*Note:* The file "chapter3.sav" will be used for the next section; execute the GET command from Section 3.2 again.

## 3.17 FILTER △

### Function

The FILTER command is used to select respondents temporarily. They are not excluded permanently from the data file, as they are with SELECT IF (p. 44). The respondents remain inactive but can be reactivated in future analyses. As this command is executed immediately, there is no need for the EXECUTE command.

### Structure

The BY operator follows the FILTER command to select a certain group and is then followed by the name of a variable. All respondents who score a 0, a missing, or an invalid (missing) value on this variable become inactive and are therefore excluded from further analyses. If the filter variable does not exist at this point, it must be created with COMPUTE (p. 34). All respondents are reactivated with the command FILTER OFF. If this command is omitted, all further analyses and transformations performed will be done with the selected respondents only!

### Example

```
* calculate MALE with value 0 for women and 1 for men.
COMPUTE MALE = (SEX = 1).
FILTER BY MALE.
FREQUENCIES
 /VARIABLES EDUC.
FILTER OFF.
```

The COMPUTE statement has created the variable MALE, with a value of 1 indicating men and a value of 0 indicating women (see the appendix on logical expressions on p. 121 for an explanation of these types of constructions). The data file has then been filtered on this variable, activating only the male respondents. The following command is only executed for men—the frequency distribution for educational level in this case. Finally, the filter is turned off again with FILTER OFF. The results of the example are equal to the results of the previous section (see Table 3.11).

## △ 3.18 SORT CASES

### Function

The SORT CASES command sorts respondents on one or more variables. This is especially useful because some commands (i.e., see sections 3.19 on p. 48, and 3.21 on p. 51) and certain programs (e.g., MLwiN) require the data to be sorted. Sometimes, it is also useful to have the respondents in a particular order to inspect the data yourself. As this command is executed immediately, there is no need for the EXECUTE command.

### Structure

SORT CASES is followed by the BY operator and then by one or more variable names. If only one variable is specified, the data are simply sorted on that variable. If multiple variables are specified, respondents who score the same value on the first variable will be sorted on the second variable, respondents who score the same value on the second variable will be sorted on the third variable, and so on. The respondents are sorted in ascending order by default: lowest scores appear first, while higher scores appear last. Respondents can be sorted in descending order by adding "(D)" after the name of the variable.

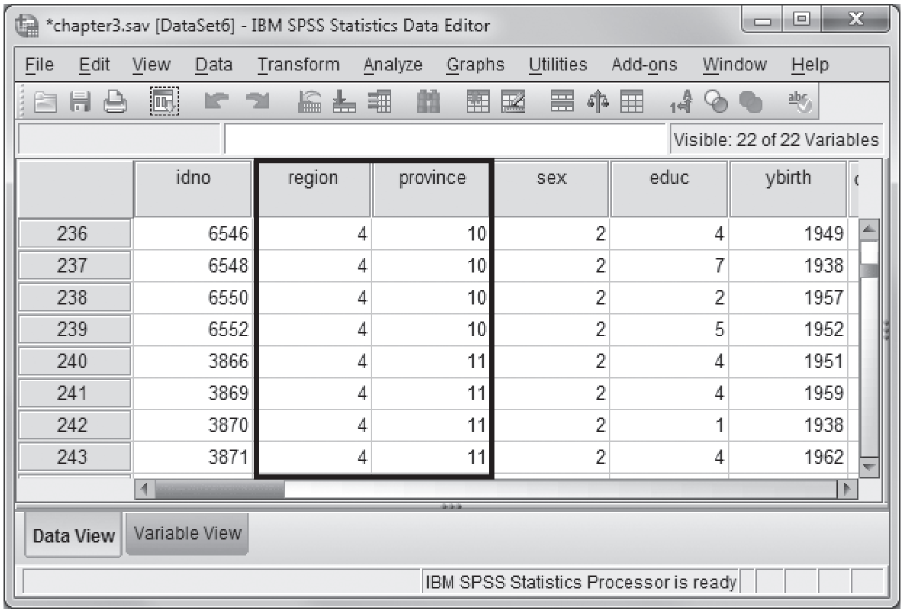
## Example

```
SORT CASES BY REGION (D) PROVINCE.
```

The command has sorted respondents by regions and provinces. The region codes are sorted in descending order and the province codes in ascending order (see Figure 3.6).

## Results

**Figure 3.6** File Sorted for REGION (D) and PROVINCE (Rows 236 to 243 Are Shown)


*chapter3.sav [DataSet6] - IBM SPSS Statistics Data Editor
File Edit View Data Transform Analyze Graphs Utilities Add-ons Window Help
Visible: 22 of 22 Variables
Data View
Variable View
IBM SPSS Statistics Processor is ready

| | idno | region | province | sex | educ | ybirth |
|---|---|---|---|---|---|---|
| 236 | 6546 | 4 | 10 | 2 | 4 | 1949 |
| 237 | 6548 | 4 | 10 | 2 | 7 | 1938 |
| 238 | 6550 | 4 | 10 | 2 | 2 | 1957 |
| 239 | 6552 | 4 | 10 | 2 | 5 | 1952 |
| 240 | 3866 | 4 | 11 | 2 | 4 | 1951 |
| 241 | 3869 | 4 | 11 | 2 | 4 | 1959 |
| 242 | 3870 | 4 | 11 | 2 | 1 | 1938 |
| 243 | 3871 | 4 | 11 | 2 | 4 | 1962 |

## 3.19 SPLIT FILE

### Function

SPLIT FILE splits the respondents into groups and separately analyses them. When, for example, the samples of men and women are split, statistical analyses for men and women are done separately. Splitting a file is therefore mainly used when comparing groups, for example, with histograms or frequency distributions.

### Structure

First, the data have to be sorted on the variables that are used to split the file. You can use the SORT CASES command for this (see p. 46). After that the SPLIT FILE is followed by the operator BY and the name of the variable that is going to be split. It is also possible to split the file by multiple variables—the file must be sorted on all variables for this! The outcome of a split command is that statistical information for *every combination* of categories of those variables is displayed. If the variables have many categories (e.g., age), then this can lead to cluttered results. The split is turned off by the SPLIT FILE OFF command.

### Example

```
SORT CASES BY SEX.
SPLIT FILE BY SEX.
FREQUENCIES CHURCHMEMBER.
SPLIT FILE OFF.
```

The example is first sorted by SEX: first men (Value 1) followed by women (Value 2). Then, the file is split by SEX, and for both males and females, the frequencies for CHURCHMEMBER are displayed (see FREQUENCIES, p. 75). Finally, the split is turned off again (Table 3.12).

### Results

**Table 3.12** Separate Frequency Distributions of Church Membership for Men and Women

| Church membership | | | | | |
|---|---|---|---|---|---|
| **Sex** | | **Frequency** | **Percent** | **Valid Percent** | **Cumulative Percent** |
| 1 Male | 1 Yes | 364 | 38.5 | 38.5 | 38.5 |
| | 2 No | 558 | 59.0 | 59.0 | 97.6 |
| | 3 Don't know/ unsure | 20 | 2.1 | 2.1 | 99.7 |
| | 99 Not applicable | 3 | .3 | .3 | 100.0 |
| | Total | 945 | 100.0 | 100.0 | |
| 2 Female | 1 Yes | 454 | 44.1 | 44.1 | 44.1 |
| | 2 No | 550 | 53.4 | 53.4 | 97.6 |
| | 3 Don't know/ unsure | 20 | 1.9 | 1.9 | 99.5 |
| | 99 Not applicable | 5 | .5 | .5 | 100.0 |
| | Total | 1029 | 100.0 | 100.0 | |

## 3.20 TEMPORARY △

### Function

The TEMPORARY command is used to execute the subsequent command temporarily. This transformation is undone *after* a second command is executed. TEMPORARY can be used in combination with MISSING VALUES (p. 28), RECODE (p. 32), COMPUTE (p. 34), COUNT (p. 35), IF (p. 37), SELECT IF (p. 44), FILTER (p. 45), and SPLIT FILE (p. 48). This means that the changes these commands make in the file only apply to the following command. TEMPORARY is typically combined with SELECT IF to ensure that the selection is not permanent. TEMPORARY does not have subcommands.

## Example

```
TEMPORARY.
SELECT IF (REGION = 2).
FREQUENCIES SEX.
FREQUENCIES SEX.
```

Normally, the SELECT IF command deletes every respondent who does not meet the condition in the file. However, when it is combined with TEMPORARY, the selection is only applied to the subsequent command. The first FREQUENCIES command (see p. 75) is only executed for respondents living in Region 2 (the east). The second FREQUENCIES command, however, applies to all respondents. Thus, the results display a frequency distribution by SEX for respondents living in Region 2 (east) and a frequency distribution by SEX for all respondents (all regions) (Tables 3.13 and 3.14).

## Results

**Table 3.13** Frequency Distribution of Sex for Region 2

| Sex | | | | | |
|---|---|---|---|---|---|
| | | **Frequency** | **Percent** | **Valid Percent** | **Cumulative Percent** |
| Valid | 1 Male | 213 | 48.3 | 48.3 | 48.3 |
| | 2 Female | 228 | 51.7 | 51.7 | 100.0 |
| | Total | 441 | 100.0 | 100.0 | |

**Table 3.14** Frequency Distribution of Sex for All Regions

| Sex | | | | | |
|---|---|---|---|---|---|
| | | **Frequency** | **Percent** | **Valid Percent** | **Cumulative Percent** |
| Valid | 1 Male | 945 | 47.9 | 47.9 | 47.9 |
| | 2 Female | 1029 | 52.1 | 52.1 | 100.0 |
| | Total | 1974 | 100.0 | 100.0 | |

## 3.21 MATCH FILES △

### Function

MATCH FILES is typically used to connect variables and/or units from different data sets. There are two ways to do this. First, files can contain (a lot of) the same units of analysis (often respondents), but with different sets of variables. If this is the case, the files are set up side by side. The units of analysis are recognized by their identification numbers, and the variables from all data sets will be assigned to each unit (respondent). Second, it is possible that one or more of the data sets you want to match contain different units of analysis, that is, countries or regions. When, for instance, you want to match a file where respondents are the units of analysis with a file where regions are the units of analysis, the respondents need to have a region code. The files can then be matched on these region codes, so that the characteristics of the regions—such as unemployment rates or population statistics—can be linked to the respondents from that region.

### Structure

To successfully match the files, they have to be sorted (in ascending order; see Section 3.18, p. 46) by the variable(s) that identify the units of analysis. This variable (called the *key variable*) should be present in all files to be matched. When using data where respondents are the units of analysis, the key variable is usually the respondent identifier, which is a unique code. When a file contains information on regions, the region code is the key variable. After sorting the files, the MATCH FILES command can be used, followed by the FILE subcommand with the name of the first file (the file where the respondents are the units). If the file has not been opened yet, it can be indicated by a string value with the file name or just with an asterisk (*) if the current (active) file has to be used. Next, the second file has to be indicated. When both files contain the same units of analysis (e.g., respondents), the FILE subcommand can be used again, this time with the file name of the second file. However, the second file can also contain information on "higher" units of analysis. For instance, the units of the first file (respondents) can be clustered in units of the second file, such as regions or countries. If that is the case, the TABLE subcommand has to be used instead of FILE. The subcommands FILE and TABLE can be repeated, should you need to match more files. After specifying all the files, the BY subcommand is used, followed by the key variable. There could be more than one key variable: for instance, the variables “region” and “time,” if you have information for all regions on unemployment rates for every year in the 1980–2006 period. When more than one key variable is used, the files have to be sorted in ascending order by all these variables. Please note that the order in which the key variables are sorted and the order of the variables in the BY subcommand have to be the same!

## Example

```
SORT CASES BY PROVINCE.
MATCH FILES FILE *
 /TABLE "c:/data/provinces.sav"
 /BY PROVINCE.
EXECUTE.
```

In this example, the variable NOCHURCH (the percentage of non–church members in a province) is added from the file “provinces.sav” to the individual data from the opened file “chapter3.sav.” The location of the file “provinces.sav” is specified after the TABLE command. This file was already sorted by the variable PROVINCE. The active file is sorted by the SORT CASES command. The new, matched data are only visible after using the EXECUTE command (p. 26).

## Result

```
SPLIT FILE BY PROVINCE.
FREQUENCIES NOCHURCH.
SPLIT FILE OFF.
```

To check whether the match was successful, the percentage of non–church members per province in the Netherlands is required (Table 3.15a).

**Table 3.15a** Percentage of Non–Church Members per Province

| Province Where the Respondents Were Interviewed | Frequency | Percent | Valid Percent | Cumulative Percent |
|---|---|---|---|---|
| 1 Groningen | 76.90 | 64 | 100.0 | 100.0 |
| 2 Friesland | 63.70 | 105 | 100.0 | 100.0 |
| 3 Drenthe | 62.10 | 48 | 100.0 | 100.0 |
| 4 Overijssel | 53.90 | 107 | 100.0 | 100.0 |
| 5 Gelderland | 59.20 | 295 | 100.0 | 100.0 |
| 6 Utrecht | 61.20 | 56 | 100.0 | 100.0 |
| 7 Noord-Holland | 77.60 | 283 | 100.0 | 100.0 |
| 8 Zuid-Holland | 63.60 | 464 | 100.0 | 100.0 |
| 9 Zeeland | 53.00 | 91 | 100.0 | 100.0 |
| 10 Noord-Brabant | 56.50 | 239 | 100.0 | 100.0 |
| 11 Limburg | 43.10 | 186 | 100.0 | 100.0 |
| 12 Flevoland | 45.20 | 36 | 100.0 | 100.0 |

## Optional Subcommands

The RENAME subcommand can be used in the command as well. This is especially useful when the key variable does not have the same name in all files. RENAME is used after the FILE or TABLE subcommand with the variable that needs to be renamed. One or more name changes follow the RENAME command. This includes the old variable name, an equal sign (=), and the new variable name, all within parentheses ().

As with the GET command (p. 21), variables can be excluded by the subcommands KEEP or DROP. These follow the BY subcommand. KEEP is followed by a list of variables that you want to include in the final file. The variables that follow the DROP command are those that you want to exclude from the final file.

## Extensive Example

```
MATCH FILES FILE="c:\data\chapter3.sav"
 /FILE "c:/data/chapter3match.sav"
 /RENAME (v0038= child)
 /FILE "c:/Data/chapter3match2.sav"
 /BY IDNO
 /DROP REGION PROVINCE.
```

In this example, three files are matched, each containing information on respondents. Two extra operations (RENAME and DROP) have been included here. First, the "chapter3.sav" file is opened, and second the "chapter3match.sav" file with the same respondents is opened. The latter file contains a variable V0038, which is renamed as CHILD. Please note that RENAME follows the command that opens the "chapter3match.sav" file, because this is what the RENAME command refers to. Next, the "chapter3match2.sav" file is opened, which contains information on the respondent's partner. The three files are matched, excluding two variables REGION and PROVINCE.

## Result

```
FREQUENCIES CHILD PARTNER.
FREQUENCIES CHILD REGION PROVINCE.
```

The above command displays the frequency distribution of the number of children (CHILD) and whether the respondent has a partner. The last line in the command box requests the frequency distribution of CHILD, REGION, and PROVINCE. A warning will follow in the output because the latter two variables no longer exist, indicating that the DROP operation was effective (Tables 3.15b and c).

**Table 3.15b** Frequency Distribution of CHILD

| child: number of children | | | | | |
|---|---|---|---|---|---|
| | | **Frequency** | **Percent** | **Valid Percent** | **Cumulative Percent** |
| Valid | 0 | 675 | 34.2 | 34.2 | 34.2 |
| | 1 | 232 | 11.8 | 11.8 | 45.9 |
| | 2 | 614 | 31.1 | 31.1 | 77.1 |
| | 3 | 303 | 15.3 | 15.3 | 92.4 |
| | 4 | 87 | 4.4 | 4.4 | 96.8 |
| | 5 | 33 | 1.7 | 1.7 | 98.5 |
| | 6 | 17 | .9 | .9 | 99.3 |
| | 7 | 4 | .2 | .2 | 99.5 |
| | 8 | 4 | .2 | .2 | 99.7 |
| | 9 | 5 | .3 | .3 | 100.0 |
| | Total | 1974 | 100.0 | 100.0 | |

**Table 3.15c** Frequency Distribution of PARTNER and a Warning for REGION and PROVINCE

| partner has got a partner | | | | | |
|---|---|---|---|---|---|
| | | **Frequency** | **Percent** | **Valid Percent** | **Cumulative Percent** |
| Valid | 1 Yes | 1449 | 73.4 | 73.4 | 73.4 |
| | 2 No | 525 | 26.6 | 26.6 | 100.0 |
| | Total | 1974 | 100.0 | 100.0 | |

**Warnings**

Text: REGION Command: FREQUENCIES
An undefined variable name, or a scratch or system variable was specified in a variable list that accepts only standard variables. Check spelling and verify the existence of this variable.
Execution of this command stops.

Text: PROVINCE Command: FREQUENCIES
An undefined variable name, or a scratch or system variable was specified in a variable list that accepts only standard variables. Check spelling and verify the existence of this variable.

## 3.22 ADD FILES

### Function

The ADD FILES command is used to group together unique respondents (cases) from different data files. The result is a new data file in which the original files are placed one after the other. As a consequence, the total number of cases in the new file equals the sum of the cases in all added files. The values from variables with the same name are placed under each other while the labels and definitions for the missing values are retrieved from the first file. Before files are merged this way, you have to make sure that a variable has exactly the same name in all data sets, as well as the same missing values and coding of categories, all of which may differ across the data sets, of course.

### Structure

ADD FILES is followed by the subcommand FILE, and the directory and name of the first file. You can specify the file name as a string or use an asterisk (*) to specify that the active file is to be used. After that you can add the additional FILE commands specifying the files that need to be added to the first file. The file names are specified as a string in this case as well.

### Example

```
ADD FILES FILE "c:\data\chapter3.sav"
 /FILE "c:/data/chapter3add.sav"
 /FILE "c:/data/chapter3add2.sav".
EXECUTE.
```

In this example, a total of 20 respondents from the files "chapter3add.sav" and "chapter3add2.sav" are added to the file "chapter3.sav," increasing the total number of respondents to 1994. The added files contain the same variables as in the original file, so there are no additional missing values. The EXECUTE command is necessary to visualize the new data in the data window (Table 3.16).

## Results

```
FREQUENCIES SEX.
```

**Table 3.16** Frequency Distribution of Sex After Adding 20 Respondents

| Sex | | | | | |
|---|---|---|---|---|---|
| | | **Frequency** | **Percent** | **Valid Percent** | **Cumulative Percent** |
| Valid | 1 Male | 955 | 47.9 | 47.9 | 47.9 |
| | 2 Female | 1039 | 52.1 | 52.1 | 100.0 |
| | Total | 1994 | 100.0 | 100.0 | |

## Optional Subcommands

With RENAME positioned after the FILE subcommand, you can specify that some variables need to be renamed first. For each variable, specify the variable name in parentheses, followed by an equal sign (=), and finally the new name. With KEEP and DROP, you can specify which variables are kept or dropped, respectively, in the new data file.

*Note:* The next section requires the "chapter3.sav" file again (so execute the GET command from Section 3.2 again).

## 3.23 DO REPEAT/END REPEAT

### Function

The DO REPEAT command is generally used for reducing the number of commands, for instance, when you have to compute a lot of new variables. It can also be applied to the recoding of many variables at the same time. It saves time and reduces possible errors when only using this command instead of all the repetitions. To illustrate, we will create dummy variables that are coded 0 and 1, and are often used in linear regression analysis.

## Structure

Following DO REPEAT, the name of the so-called stand-in variable must be specified. This particular variable only exists within the DO REPEAT command and represents all variables or numbers that follow this variable. After the stand-in variable, just type the equal sign (=) and then the list with variables (existing or nonexisting) or numbers. The names you use, of course, must follow the syntax rules (see Section 2.2).

Existing variables that stand next to each other in the data matrix can be listed using TO. For instance, V1 TO V15 will include all 15 existing variables (provided no other variables lie in between). Similarly, we can generate a whole list of nonexisting variables. The command V1NEW TO V15NEW creates 15 new variables. The syntax operator TO can also be used to generate a series of numbers, such as "1 TO 20." With a forward slash (/), we can generate a second series of variables or numbers. Note that the series must be of equal length, so the first series "V1 TO V15" must be followed by "1 TO 15." Within the DO REPEAT structure, almost all commands to modify data can be used, but no commands that run statistical analyses. The structure is closed by the END REPEAT subcommand to signify that all enclosed commands end here. The results are visible in the data window following EXECUTE or a command that implies EXECUTE such as FREQUENCIES.

## Example

```
DO REPEAT DUMMY = ELEMENT MIDDLE JUNIOR SENIOR
 VOCATIONAL COLLEGE BACHELOR MASTER
 /NUMBER = 1 TO 8.
   IF (EDUC = NUMBER) DUMMY=1.
   IF (EDUC <> NUMBER) DUMMY=0.
END REPEAT.
```

In this example, eight dummy variables are created for all eight educational levels in the variable EDUC (this variable measures the highest attained educational level). For each level, a unique variable is created that has the value 1 for all respondents who attained that particular level and a value of 0 for all respondents who attained another level. We use two series in the DO REPEAT structure. The first is a series of eight new variables that are labeled (1) ELEMENT, (2) MIDDLE, (3) JUNIOR, (4) SENIOR, (5) VOCATIONAL, (6) COLLEGE, (7) BACHELOR, and (8) MASTER. Please note that we hold on to the rules for naming variables in SPSS (see Section 2.2). The second series also has eight elements, but this time it contains the numbers 1 to 8.

Within the DO REPEAT/END REPEAT structure, the two IF commands (see p. 37) are repeated (or "run") 8 times. In the first run, the variable "ELEMENT" is created together with "Number," which is set to 1. So the first IF command reads as follows: If (EDUC = 1) ELEMENT = 1. This means that all respondents with elementary school (coded 1 in EDUC) get the value 1 in the variable ELEMENT. The second IF command reads as follows: IF (EDUC <> 1) ELEMENT=0. So all respondents with scores other than 1 in EDUC score a 0 on ELEMENT. In the data set, we have four respondents with no valid score (so-called missings)—they will automatically receive a missing value as well in this structure. This makes sense because we do not know their highest attained educational level, and so none of the dummy variables apply here.

In a second run, the dummy variable changes to MIDDLE while the NUMBER variable now equals 2. So the IF commands are as follows: IF (EDUC=2) MIDDLE=1 and IF (EDUC <> 2) MIDDLE=0. So all respondents with middle school (code 2 in EDUC) receive a value of 1 in the variable MIDDLE, while all other respondents (save the four with a missing value) receive a 0 for this dummy variable. In the next six runs, the six remaining levels are treated likewise. When you have many variables that need recoding, this structure is most useful. Furthermore, it reduces the risk of making errors. However, as always, do not forget to run all necessary checks (for results, see Table 3.17).

## Results

*Note:* Only tables for EDUC and ELEMENT are shown.

```
FREQUENCIES
 /VARIABLES EDUC ELEMENT MIDDLE JUNIOR SENIOR
 VOCATIONAL COLLEGE BACHELOR MASTER.
```

**Table 3.17** Frequency Distribution for Education and "ELEMENT"

**educ highest completed education**

| | Frequency | Percent | Valid Percent | Cumulative Percent |
|---|---|---|---|---|
| 1 Elementary school | 215 | 10.9 | 10.9 | 10.9 |
| 2 Middle school | 338 | 17.1 | 17.2 | 28.1 |
| 3 Junior high school | 220 | 11.1 | 11.2 | 39.2 |
| 4 Senior high school | 392 | 19.9 | 19.9 | 59.1 |
| 5 Vocational school | 133 | 6.7 | 6.8 | 65.9 |
| 6 College | 140 | 7.1 | 7.1 | 73.0 |
| 7 Bachelor degrees | 388 | 19.7 | 19.7 | 92.7 |
| 8 Master degrees | 144 | 7.3 | 7.3 | 100.0 |
| Total | 1970 | 99.8 | 100.0 | |
| Missing system | 4 | .2 | | |
| Total | 1974 | 100.0 | | |

*Check:* The number of cases coded 1 in ELEMENT must be equal to 215, while the number of cases coded 0 in ELEMENT must equal 1,755 (1974 – 215 – 4).

**ELEMENT**

| | | Frequency | Percent | Valid Percent | Cumulative Percent |
|---|---|---|---|---|---|
| Valid | .00 | 1755 | 88.9 | 89.1 | 89.1 |
| | 1.00 | 215 | 10.9 | 10.9 | 100.0 |
| | Total | 1970 | 99.8 | 100.0 | |
| Missing system | | 4 | .2 | | |
| Total | | 1974 | 100.0 | | |

## △ 3.24 LOOP

### Function

The LOOP command resembles the DO REPEAT command (p. 57) with respect to the repetition of a number of commands that are executed for each line (each respondent) in a single run. The LOOP command is more flexible, however, which makes it possible to construct more complex data structures such as an "event history" file in which you can follow the life course of respondents.

## Structure

When a procedure needs to be executed a number of times, the LOOP command is followed by the name of the variable called the *index*. If the name of the index variable is preceded by a "#," the index variable is a temporary variable. The equal sign (=) follows the name, then the starting value, the word TO, and the end value. "LOOP #NUMBER = 1 TO 10" means that SPSS will run the LOOP 10 times for each respondent. During the run, the value of the variable #NUMBER increases by 1 for every run. If the variable has to increase by a value other than 1, for instance, 2 or even −1 (counting backward), the BY operator can be added + a number that indicates the value that the index variable has to increase (positive number) or decrease (negative number). The END LOOP command is placed at the end of the block to indicate the end of the loop. The EXECUTE command must be used to execute the LOOP.

## Example

In the example below, a loop is run 100 times, and for each time, the age (starting with age = 1) and the identification number (IDNO) is written in the outfile with XSAVE.[2] To check whether the file is correct, we use the GET FILE command (p. 21). In the example below, a life-course file is constructed in which each respondent gets a data line for each year of his or her life between 1 year and 100 years (Figure 3.7a).

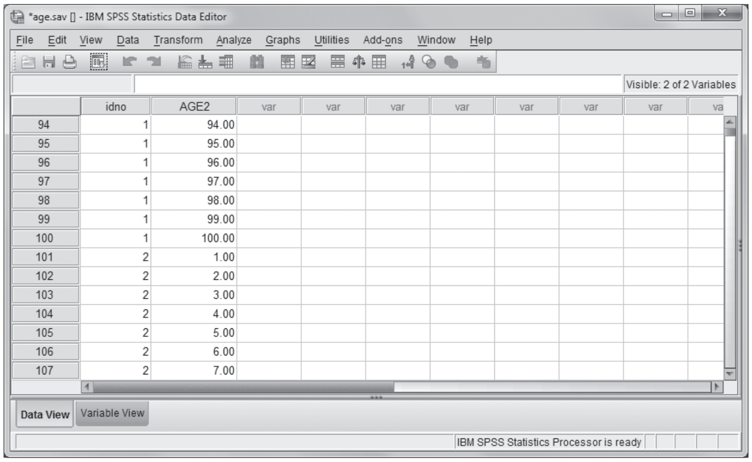
```
LOOP #NUMBER = 1 TO 100.
 COMPUTE AGE2 = #number.
 XSAVE OUTFILE = "c:/data/age.sav"
 /keep=IDNO AGE2.
END LOOP.
EXECUTE.
```

[2]XSAVE is similar to SAVE (see p. 23); however, with XSAVE data can be saved during an operation (such as LOOP). This is not possible with SAVE.

## Results

```
GET FILE = "c:/data/age.sav".
EXECUTE.
```

**Figure 3.7a** The Last 7 Years of Respondent 1 and the First 7 Years of Respondent 2 From "age.sav"



Visible: 2 of 2 Variables

| | idno | AGE2 | var | var | var | var | var | var | var | va |
|---|---|---|---|---|---|---|---|---|---|---|
| 94 | 1 | 94.00 | | | | | | | | |
| 95 | 1 | 95.00 | | | | | | | | |
| 96 | 1 | 96.00 | | | | | | | | |
| 97 | 1 | 97.00 | | | | | | | | |
| 98 | 1 | 98.00 | | | | | | | | |
| 99 | 1 | 99.00 | | | | | | | | |
| 100 | 1 | 100.00 | | | | | | | | |
| 101 | 2 | 1.00 | | | | | | | | |
| 102 | 2 | 2.00 | | | | | | | | |
| 103 | 2 | 3.00 | | | | | | | | |
| 104 | 2 | 4.00 | | | | | | | | |
| 105 | 2 | 5.00 | | | | | | | | |
| 106 | 2 | 6.00 | | | | | | | | |
| 107 | 2 | 7.00 | | | | | | | | |

IBM SPSS Statistics Processor is ready

## Extensive Example

With respect to the life-course file, we generally follow all respondents from a certain age until the age at the time of the interview. To this end, a DO IF/END IF command (p. 42) is placed in the LOOP command. It is also possible to include variables that contain information about respondents at a certain age with the IF command (p. 37).

*Note:* First open the "chapter3.sav" file before running the syntax below!

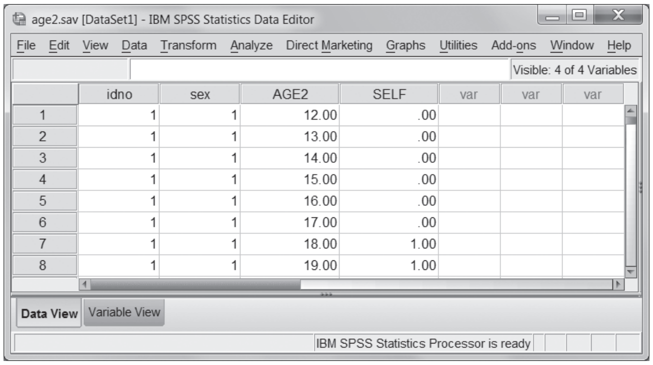
```
LOOP #NUMBER = 1 TO 100.
 COMPUTE AGE2 = #NUMBER.
 DO IF (AGE2 > 11) and (AGE2<= (1995 − YBIRTH)).
 COMPUTE SELF = 0.
 IF (MOVEOUT <= AGE2) SELF=1.
 XSAVE OUTFILE = "c:/data/age2.sav"
  /keep=IDNO SEX AGE2 SELF.
 END IF.
END LOOP.
EXECUTE.
```

A record is written for every year of a respondent's life into "age2.sav." In this file, age starts at 12 and ends with the respondent's age in 1995 (see the DO IF command and note the positioning of the END IF command!). Furthermore, the variable SELF is calculated to indicate whether the respondent was living at the parental home at a certain age (SELF=0) or left the parental home (SELF=1). This variable is written to the "age2.sav" file, together with the variables IDNO, SEX, and AGE (see Figure 3.7b).

## Results

```
GET FILE = "c:/data/age2.sav".
EXECUTE.
```

**Figure 3.7b** Life Course of Respondent 1 Between the Age of 12 and 19 Years (Left Parental Home Since the Age of 18); Data From "age2.sav"


age2.sav [DataSet1] - IBM SPSS Statistics Data Editor
File Edit View Data Transform Analyze Direct Marketing Graphs Utilities Add-ons Window Help
Visible: 4 of 4 Variables
Data View
Variable View
IBM SPSS Statistics Processor is ready

| | idno | sex | AGE2 | SELF | var | var | var |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 12.00 | .00 | | | |
| 2 | 1 | 1 | 13.00 | .00 | | | |
| 3 | 1 | 1 | 14.00 | .00 | | | |
| 4 | 1 | 1 | 15.00 | .00 | | | |
| 5 | 1 | 1 | 16.00 | .00 | | | |
| 6 | 1 | 1 | 17.00 | .00 | | | |
| 7 | 1 | 1 | 18.00 | 1.00 | | | |
| 8 | 1 | 1 | 19.00 | 1.00 | | | |

*Note:* The "chapter3.sav" file has to be opened for the next section (so execute the GET command from Section 3.2 again).

## 3.25 WEIGHT

### Function

The command WEIGHT can be used in two ways: (1) to eliminate under- or overrepresentation in a sample or (2) to analyze a data file where one data record refers to several respondents simultaneously. Such files can save a lot of time compared with putting all units in individual records (see the second example in “chapter3weight.sav”).

### Structure

WEIGHT is followed by the subcommand BY and the name of the weight variable (also called weight factor). End the syntax file with WEIGHT OFF to undo the weighting (Table 3.18).

### Results

```
* First example: weighting to correct ratio.
* In the population the male-female ratio is 49.5/50.5.
* In chapter3.sav that ratio is 47.9 / 52.1.
IF SEX EQ 1 SEXRATIO=49.5/47.9.
IF SEX EQ 2 SEXRATIO=50.5/52.1.
WEIGHT BY SEXRATIO.
FREQUENCIES
 /VARIABLES SEX.
WEIGHT OFF.
```

**Table 3.18** The Male-Female Ratio After Weighting

| Sex | | | | | |
|---|---|---|---|---|---|
| | | **Frequency** | **Percent** | **Valid Percent** | **Cumulative Percent** |
| Valid | 1 Male | 977 | 49.5 | 49.5 | 49.5 |
| | 2 Female | 997 | 50.5 | 50.5 | 100.0 |
| | Total | 1974 | 100.0 | 100.0 | |

```
* Second example: analysis on a data file where a line represents more
than one respondent.
GET FILE "c:/Data/chapter3weight.sav".
WEIGHT BY NUMBER.
FREQUENCIES
 /VARIABLES SEX.
WEIGHT OFF.
```

Please take a look at the data structure of "chapter3weight.sav" and see Table 3.14 for checks on the results after weighting.

*Note:* The "chapter3.sav" file has to be opened for the next section (so execute the GET command from Section 3.2 again).

## 3.26 SAMPLE

### Function

SAMPLE is used to draw a random sample from a data file. This is useful when the data file is too large for analysis. This command is also used as a form of sensitivity analysis to check the stability of the results.

### Structure

Following SAMPLE, a proportion is given (e.g., .25) or two numbers separated by FROM, where the first number specifies the exact number of cases required in the subsample, and the second number is normally equal to the sample size (e.g., 25 from 100) (Table 3.19).

### Results of a 10% Sample

*Note:* Outcomes will differ for each draw.

```
SAMPLE .10.
FREQUENCIES
 /VARIABLES SEX.
```

**Table 3.19** Random Sample of Approximately 10% From "chapter3.sav"

| Sex | | | | | |
|---|---|---|---|---|---|
| | | Frequency | Percent | Valid Percent | Cumulative Percent |
| Valid | 1 Male | 87 | 44.8 | 44.8 | 44.8 |
| | 2 Female | 107 | 55.2 | 55.2 | 100.0 |
| | Total | 194 | 100.0 | 100.0 | |

## 3.27 SPSS Syntax: An Overview

```
*** 3.2: opens the data file for third chapter.
GET FILE "c:/Data/chapter3.sav".
* Macintosh users: GET FILE "/data/chapter3.sav".

* opens the data file for third chapter, with DROP and RENAME.
GET FILE "c:/Data/chapter3.sav"
 /DROP LIST FUTURE
 /RENAME GOD2 = BELIEVE_NOW GOD1 = BELIEVE_15.
EXECUTE.

*** 3.3 saves the file.
SAVE OUTFILE "c:/Data/chapter3new.sav".
GET FILE "c:/Data/chapter3new.sav".

*** 3.3 saves file with KEEP and RENAME.
SAVE OUTFILE "c:/Data/chapter3short.sav"
 /KEEP SEX EDUC YBIRTH CHURCHMEMBER VOTES
 /RENAME VOTES = POLITICS.
GET FILE "c:/Data/chapter3short.sav".

* PLEASE NOTE: reopen data file for third chapter.
GET FILE "c:/Data/chapter3.sav".

*** 3.5 adds informative variable labels.
VARIABLE LABELS LEAVECHURCH "age at which the respondent left
church" YBIRTH "year of birth of the respondent".
FREQUENCIES
 VARIABLES=LEAVECHURCH YBIRTH.
```

```
*** 3.6 adds informative names for categories.
VALUE LABELS SEX 1 “Male” 2 “Female”
 /REGION 1 “North” 2 “East” 3 “West” 4 “South”.
FREQUENCIES
 VARIABLES=SEX REGION.

*** 3.7 specifies missing values.
MISSING VALUES VOTES (14 15 16) CHURCHMEMBER
LEAVECHURCH (99).
*** undoes missing values.
MISSING VALUES HEAVEN ().
FREQUENCIES
 VARIABLES=LEAVECHURCH CHURCHMEMBER VOTES HEAVEN.

*** 3.8 displays variables.
DISPLAY DICTIONARY
 /VARIABLES SEX MARITALSTAT EDUC.

*** 3.9: recodes a variable.
RECODE VOTES (1 5 12 = 1) (2 4 9 10 = 2) (3 6 THRU 8 11 = 3)
 (ELSE = 999) INTO POLITICS.
FREQUENCIES
 VARIABLES=POLITICS.

*** 3.10: computes new variable.
COMPUTE OWNHOME = (1995 − YBIRTH) − MOVEOUT.
FREQUENCIES
 VARIABLES=OWNHOME
 /STATISTICS DEFAULT.

***3.11: counts scores.
COUNT MISSINGS = MARITALSTAT MOVEOUT (MISSING)
WORKHOURS (999).
FREQUENCIES
 VARIABLES=MISSINGS.

***3.12: constructs a new variable by a condition.
COMPUTE DECREASE = 0.
IF (CHURCHNOW < CHURCH15) DECREASE = 1.
FREQUENCIES VARIABLES=DECREASE.

***3.13: saves to a different type of data file.
WRITE OUTFILE “c:/Data/example.txt”
```

```
 /IDNO (f4) SEX (f2) RELIGION (f5.2, 1x) SURVEY (a6).
EXECUTE.

***3.14: reads txt type data file.
DATA LIST FILE "c:/Data/example.txt"
 /IDNO (f4) SEX (f2) RELIGION (f5.2, 1x) SURVEY (a6).
EXECUTE.

* PLEASE NOTE: open data file for third chapter again.
GET FILE "c:/Data/chapter3.sav".

***3.15: executes commands under condition.
DO IF (EDUC < 3).
 WRITE OUTFILE "c:/Data/low.txt"
 /VOTES (f3) SEX (f2).
ELSE IF (EDUC < 5).
 WRITE OUTFILE "c:/Data/middle.txt"
 /VOTES (f3) SEX (f2).
ELSE.
 WRITE OUTFILE "c:/Data/high.txt"
 /VOTES (f3) SEX (f2).
END IF.
EXECUTE.
DATA LIST FILE "c:/Data/middle.txt"
 /VOTES (f3) SEX (f2).
FREQUENCIES SEX.

*PLEASE NOTE: open data file for third chapter again.
GET FILE "c:/Data/chapter3.sav".

***3.16: (permanently) selecting cases.
SELECT IF (SEX = 1).
FREQUENCIES
 /VARIABLES EDUC.

*PLEASE NOTE: opening file for third chapter again.
GET FILE "c:/Data/chapter3.sav".

***3.17: (temporarily) selecting cases.
COMPUTE MALE = (SEX = 1).
FILTER BY MALE.
FREQUENCIES
 /VARIABLES EDUC.
FILTER OFF.
```

```
***3.18: sorts the data file.
SORT CASES BY REGION (D) PROVINCE.

***3.19: splits the data file.
SORT CASES BY SEX.
SPLIT FILE BY SEX.
FREQUENCIES CHURCHMEMBER.
SPLIT FILE OFF.

***3.20: temporarily executes selection and one analysis.
TEMPORARY.
SELECT IF (REGION = 2).
FREQUENCIES SEX.
FREQUENCIES SEX.

***3.21: matches files, example with file/table.
SORT CASES BY PROVINCE.
MATCH FILES FILE *
 /TABLE "c:/Data/provinces.sav"
 /BY PROVINCE.
EXECUTE.
SPLIT FILE BY PROVINCE.
FREQUENCIES NOCHURCH.
SPLIT FILE OFF.

* extensive example with file/file.
MATCH FILES FILE="c:\data\chapter3.sav"
 /FILE "c:/Data/chapter3match.sav"
 /RENAME (v0038= child)
 /FILE "c:/Data/chapter3match2.sav"
 /BY IDNO
 /DROP REGION PROVINCE.
EXECUTE.
FREQUENCIES CHILD PARTNER.
FREQUENCIES CHILD REGION PROVINCE.

*** 3.22: adding respondents.
ADD FILES FILE= "c:\data\chapter3.sav"
 /FILE "c:/Data/chapter3add.sav"
 /FILE "c:/Data/chapter3add2.sav".
EXECUTE.
FREQUENCIES SEX.
```

```
*PLEASE NOTE: open file for third chapter again.
GET FILE "c:/Data/chapter3.sav".

*** 3.23: makes dummy variables the easy way.
DO REPEAT DUMMY = ELEMENT MIDDLE JUNIOR SENIOR
                                  VOCATIONAL COLLEGE BACHELOR MASTER
 /NUMBER = 1 TO 8.
IF (EDUC = NUMBER) DUMMY=1.
IF (EDUC <> NUMBER) DUMMY=0.
END REPEAT.
FREQUENCIES VARIABLES EDUC ELEMENT MIDDLE JUNIOR SENIOR
 VOCATIONAL COLLEGE BACHELOR MASTER.

***3.24: writes cases more than once.
LOOP #NUMBER = 1 TO 100.
 COMPUTE AGE2 = #NUMBER.
 XSAVE OUTFILE = "c:/data/age.sav" /keep=IDNO AGE2.
END LOOP.
EXECUTE.
GET FILE = "c:/data/age.sav".
EXECUTE.

*PLEASE NOTE: open data file for third chapter again.
GET FILE "c:/Data/chapter3.sav".

LOOP #NUMBER = 1 TO 100.
 COMPUTE AGE2 = #NUMBER.
 DO IF (AGE2 > 11) and (AGE2 <= (1995 − YBIRTH)).
 COMPUTE SELF = 0.
 IF (MOVEOUT <= AGE2) SELF=1.
 XSAVE OUTFILE = "c:/data/age2.sav"
 /keep=IDNO SEX AGE2 SELF.
 END IF.
END LOOP.
EXECUTE.
GET FILE = "c:/data/age2.sav".
EXECUTE.

*PLEASE NOTE: opening file for third chapter again.
GET FILE "c:/Data/chapter3.sav".
```

```
***3.25 weights to population.
* First example: weighting to correct proportions.
* The male/female ratio in the population is 49.5/50.5.
* In chapter3.sav this ratio is 47.9 / 52.1.
IF SEX EQ 1 SEXRATIO =49.5/47.9.
IF SEX EQ 2 SEXRATIO =50.5/52.1.
WEIGHT BY SEXRATIO.
FREQUENCIES
 /VARIABLES SEX.
WEIGHT OFF.

* Second example: analysis on data file in which a data row represents
more than one observation (respondent).
GET FILE "c:/Data/chapter3weight.sav".
WEIGHT BY NUMBER.
FREQUENCIES
 /VARIABLES SEX.
WEIGHT OFF.

*PLEASE NOTE: open file for third chapter again.
GET FILE "c:/Data/chapter3.sav".

*** 3.26 Random sampling.
SAMPLE .10.
FREQUENCIES /VARIABLES SEX.
```