

January 2000

SPSS for Windows 8, 9 and 10

by Svend Juul

PART 1: GENERAL PART

1. Introduction	1
2. Notation in this Guide	1
3. Main structure of SPSS	2
4. In and out of SPSS	3
5. Windows in SPSS	4
6. Setting preferences	7
7. On-line help	10
8. Two ways to run SPSS	10
9. File types and file names	11
10. SPSS Syntax rules	12
11. Editing in the Syntax Window	13
12. Create SPSS system file	14
13. Analyse data in SPSS system file	16
14. Some common errors	17

PART 2: COMMANDS (see back cover)	18
--	----

PART 3: VARIOUS ITEMS	38
------------------------------------	----

20. More on missing values	38
1. String variables	39
2. Numbers: integers and non-integers	42
3. Dates, time, and Danish CPR numbers	43
4. Random samples, simulations	45
5. Exchange of data with other programs	46

Appendix 1: Exercises	47
------------------------------------	----

Appendix 2: On documentation and safety	53
--	----

Appendix 3: SPSS modules and manuals	56
---	----

Appendix 4: A few remarks on Windows 95/98	57
---	----

PART 2: COMMANDS	18
15. File commands	19
DATA LIST	19
SAVE OUTFILE	21
GET FILE	21
16. Data documentation	22
VARIABLE LABELS	22
VALUE LABELS	22
FORMATS	22
MISSING VALUES	23
COMMENT	23
DOCUMENT	23
17. Transformation commands	24
COMPUTE	24
IF	25
DO IF . . . END IF	26
RECODE and RECODE ... INTO	26
COUNT	27
DO REPEAT . . . END REPEAT	27
SELECT IF	27
N	28
SAMPLE	28
SPLIT FILE	28
WEIGHT	28
TEMPORARY	29
18. Procedure commands	30
EXECUTE	30
DISPLAY	30
DESCRIPTIVES	31
FREQUENCIES	31
CROSSTABS	32
MEANS	33
T-TEST	33
GRAPH	33
LIST	34
WRITE	34
PRINT	34
19. Advanced file commands	35
SORT CASES	35
AGGREGATE	35
ADD FILES	35
MATCH FILES	35
The LAG function	37

PART 1: GENERAL PART

1. Introduction

SPSS (Statistical Package for the Social Sciences) was primarily developed for processing data from questionnaires and interviews, etc. It is, however, useful for handling of data from other sources, and for analysis also in the health sciences.

This booklet is a short introduction to SPSS for Windows, version 10.0. The operating differences between version 8, 9 and 10 are minor, and you will have no trouble in using this guide with any of these versions.

The guide is for the beginner, but knowledge of fundamental Windows functions is necessary. It is intended for self-instruction, using exercises (Appendix 1). Only basic commands are described, and it is not intended to replace the manuals. You will find some examples of output, but during exercises you will get more experience about what kinds of output SPSS can create.

It is possible to perform most procedures without knowledge of the SPSS command language, using the menu facilities only. However, this is very impractical if you are more than an occasional SPSS user. The *User's Guide* gives virtually no information on command-syntax, and this booklet can be considered a supplement.

On the purchase of manuals, see Appendix 3.

For comments and advice mail to Svend Juul: sj@soci.au.dk

2. Notation in this guide

Windows menu choices are shown like this:

File ▶ Exit meaning:

- 1: select **File** from the menu bar using the mouse or [Alt]+[F]
- 2: select **Exit** from the File menu using the mouse or [X]

SPSS commands are written with this typeface:

CROSSTABS age BY sex.

Upper-case letters indicate SPSS keywords (**CROSSTABS** and **BY**), while lowercase letters indicate variable information (variable names **age** and **sex**).

When **you** enter commands in SPSS, you are free to use upper-case or lowercase letters.

Function keys etc. are shown as [F7], [Ctrl], etc. When two keys should be activated in sequence, it is shown by a space: [Home] [↑]. When two keys should be activated simultaneously, it is shown by a + : [Alt]+[F4]. In this case, keep [Alt] down while activating [F4].

3. Main structure of SPSS

SPSS was originally developed for handling questionnaire information, but it is useful for a much broader range of purposes. But now think of a questionnaire.

After collection of questionnaires, data are coded and entered in the computer. Data entry can take place by creation of an ASCII file (a simple text file with numbers) using a text editor (eg. EDIT or the SPSS syntax window). With an SPSS job (a set of SPSS commands) you next create an SPSS data set which besides data include descriptive information: variable names, text labels, and other specifications. An alternative way is to use the SPSS Data Editor (The Data Window), SPSS Data Entry (a stand-alone program) or Epi-Info for entering data directly into an SPSS data set.

Once an SPSS system file has been created it is quite easy to perform the analyses desired from the system file. The SPSS system file is a well documented data set, and at Department of Epidemiology and Social Medicine we use SPSS as the general system for storing and analysing data, even though some analyses must be performed by other software.

SPSS uses *rectangular* data sets, where each *case* is the information eg. from one questionnaire. Each case is described by the same *variables*:

	V A R I A B L E S				
	CASENO	SEX	AGE	CIVST	etc.
CASES	1	1	20	1	
	2	2	27	1	
	3	1	17	3	
	4	1	55	2	
	etc.				

Types of variables

Numeric variables

The information in a numeric variable can be any number, integer or decimal. I recommend to use numeric variables for any kind of information; data entry is faster, and handling of data is easier than with string variables. Almost all examples in this booklet use numeric variables.

String variables

The information in string variables is text strings. I recommend to avoid string variables, but often data from other sources (registers) are strings. Read more about string variables in section 21, if needed.

Date variables

Date variables are numeric variables with a special interpretation of the information. Read more about date variables in section 23, if needed.

4. In and out of SPSS

Double-click the SPSSWIN icon. You should find it in the Programs group, but you might prefer to have a shortcut at the desktop or the Start-button (see appendix 4 on Windows 95/98).

Leave SPSSWIN by clicking:

File ► Exit

or entering:

[Alt]+[F] [X]

If you have unsaved files (data, syntax, output), SPSS will ask if you want to save them. This seems nice, but can be dangerous:

DANGER!

As a general rule, **DO NOT** respond YES when asked if you want to save the contents of the Data Window. If you have made changes in the data set, eg. by a **SELECT IF** or a **RECODE** command, your original data will be overwritten, and that may be disastrous to you.

- If you did not make modifications to your data, you will not be asked this question.
- If you made modifications intended to be temporary, you should obviously not overwrite your original data. Respond NO.
- If you made modifications and want to save them you should do it in syntax:
 - give the modified data set a new name
 - save it explicitly with the **SAVE OUTFILE** command
 - also save the syntax file that created the modified data set.

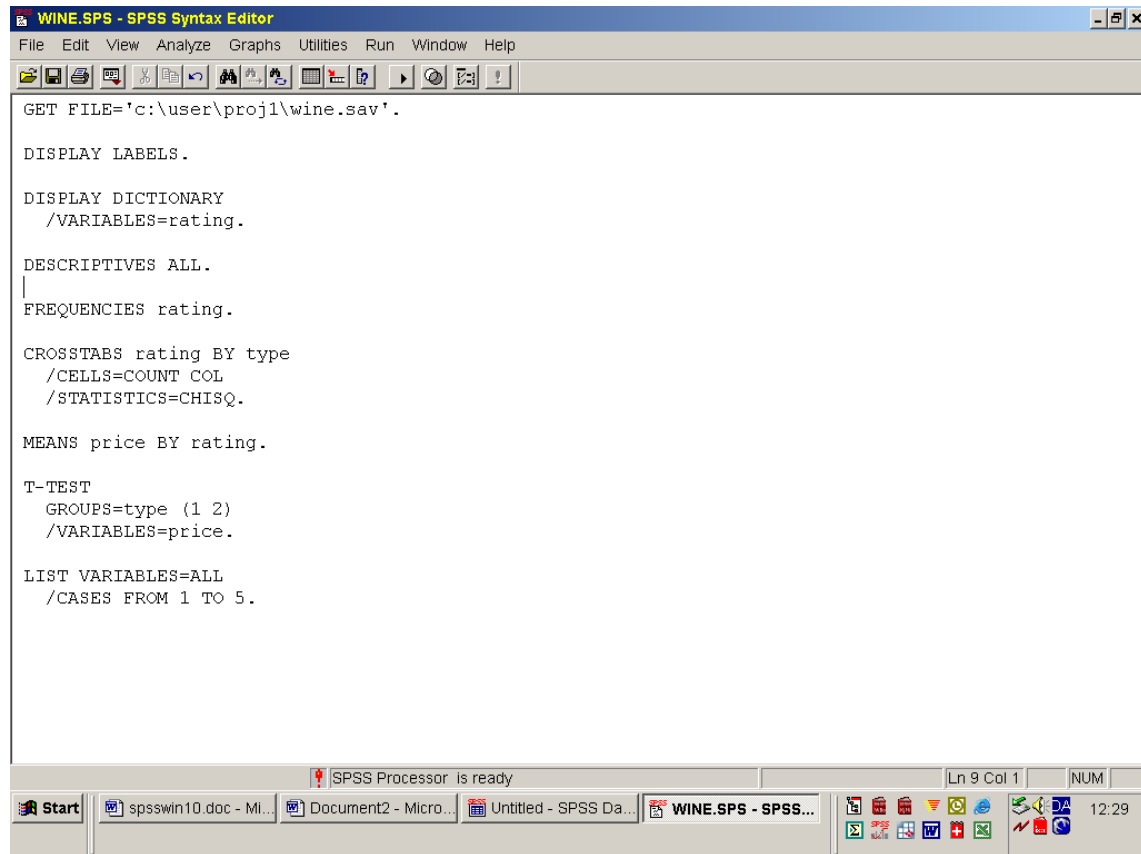
As a safeguard, do your analyses at a **copy** of your original data set (eg. in a working folder, C:\WORK), apart from the original data set.

Syntax files (**.sps**) and output files (**.spo**) can normally be saved at exit without data loss. It is the data set (**.sav**) you can damage if you are not careful.

5. Windows in SPSS

There are three main types of windows: The syntax window, the data window, and the viewer (output) window. The windows available can be seen at the Windows process line (bottom of screen). You may also select and open a window by Window in the Menu bar.

Syntax window



The syntax window is a text editor, where you can:

- write commands.
- include commands developed by the menu system. The *Paste* button includes the command in the designated syntax window.
- open an existing syntax file.
- edit commands, using the general Window editing rules (see inside back cover).
- highlight one or more commands, and execute them (see section 8).
- save all or part of the contents to a syntax file. A syntax file includes commands (instructions to SPSS). The default extension is **.sps**.

You can have several syntax windows. The *designated* syntax window receives *pasted* commands; it is marked by a ! prior to the file name. You can select the active syntax window to be designated by the [!] tool button.

You can select font for the Syntax Window by **View** ▶ **Font**.

I suggest a monospaced font, e.g. Courier New 9pt or Letter Gothic Bold 9pt.

Data window (data editor)

This window shows, in spreadsheet format, the contents of the working file.

	id	type	price	rating	var	var	var	var	var	var
1	1	2	41.95	2						
2	2	1	42.95	2						
3	3	1	42.95	1						
4	4	1	47.95	1						
5	5	2	.00	2						
6	6	1	52.95	1						
7	7	4	55.95	2						
8	8	1	60.95	1						
9	9	3	72.95	1						
10	10	2	76.95	1						
11	11	2	11.95	2						
12	12	2	27.95	3						
13	13	2	29.95	3						
14	14	3	37.95	3						
15	15	1	39.95	2						
16	16	3	41.95	3						
17	17	1	42.95	3						
18	18	2	43.95	3						
19	19	1	.00	3						
20	20	2	43.95	2						
21	21	3	45.95	2						
22	22	1	49.95	3						

The data window displays the contents of the working file. It can be used for:

- Declaration of new variables (select the Variable View tab – present from SPSS version 10). However, I recommend to create new variables in syntax (see example in section 12 B).
- Entering data. The data window is not a data entry system proper, but for data sets with few variables it is OK.
- Data corrections. However, I recommend to make corrections in syntax, for reasons of safety and documentation (see appendix 2).
- Printing of the data window contents (File ► Print). Should be avoided for large data sets to avoid waste of paper; I recommend the **LIST** command instead (see section 18).
- Saving data (File ► Save or File ► Save as...). However, I recommend to do this by syntax (the **SAVE OUTFILE** command) rather than by the menu facilities, for reasons of safety and documentation.

REPEATED WARNING: If you made modifications to your data, SPSS will offer to save the working file at exit. THE ANSWER SHOULD BE NO. See the warning in section 4.

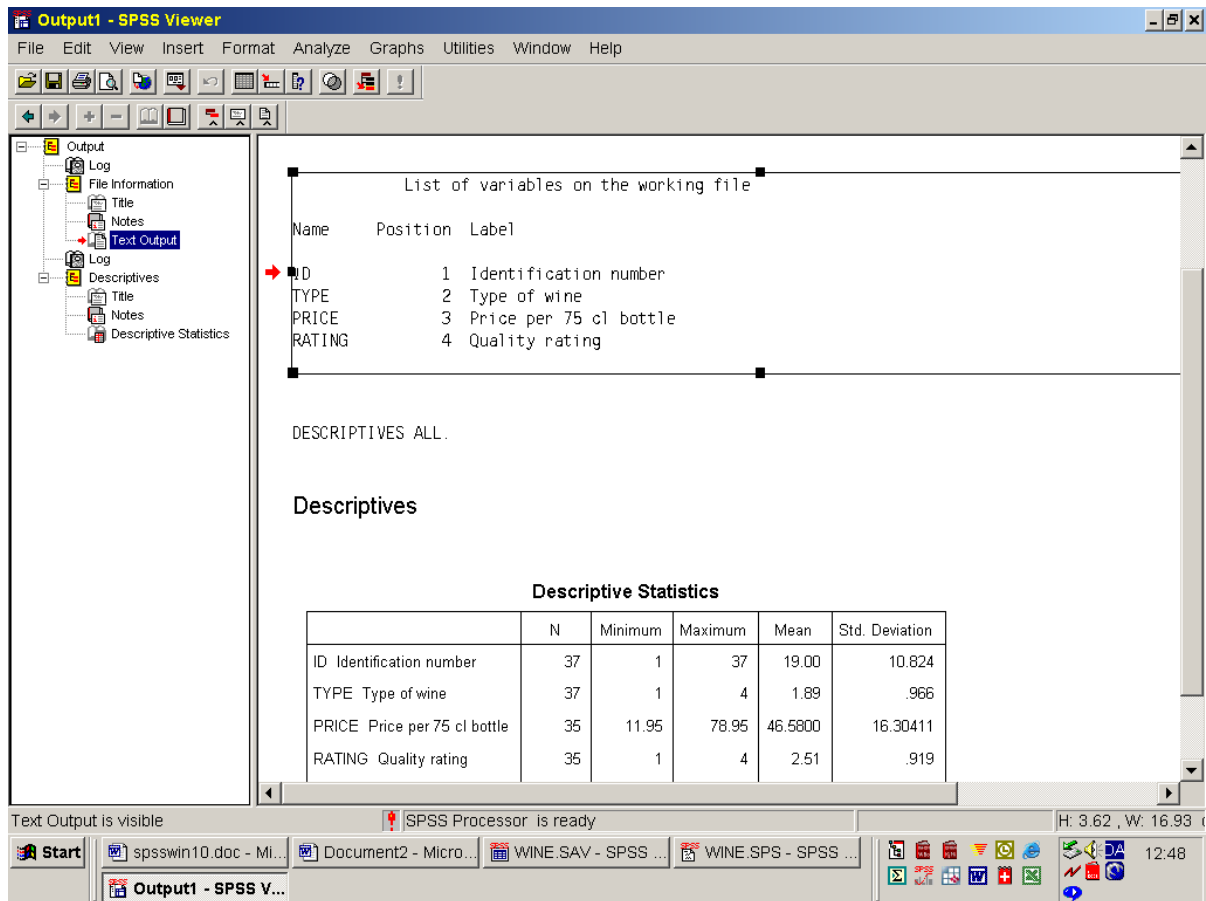
You can select default font for the Data window by View ► Font.

I suggest e.g. Arial 8pt.

Output window (Viewer)

The left window is the *Outline View*, displaying a list of the objects shown in the right window. There are three types of objects:

- Text output (syntax and output from some procedures)
- Pivot tables (output from some procedures)
- Graphs



The screenshot shows the SPSS Output window with the following content:

List of variables on the working file

Name	Position	Label
ID	1	Identification number
TYPE	2	Type of wine
PRICE	3	Price per 75 cl bottle
RATING	4	Quality rating

DESCRIPTIVES ALL.

Descriptives

Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation
ID Identification number	37	1	37	19.00	10.824
TYPE Type of wine	37	1	4	1.89	.966
PRICE Price per 75 cl bottle	35	11.95	78.95	46.5800	16.30411
RATING Quality rating	35	1	4	2.51	.919

In the output window you see three objects: text output (the selected text), a log text (syntax: **DESCRIPTIVES ALL**), and a pivot table (output from the **DESCRIPTIVES** command).

From the *Outline view* you may select an object or a group of objects; in the example the text output object was selected. A selected object may be:

- printed (File ► Print)
- copied to the *clipboard* ([Ctrl]+[C]), from where you may paste it to another window, e.g. the syntax window, or to another application, e.g. a word processing document.
- saved to a file (see below).

If you double-click an object you may edit it; you may eg. change cell sizes in a *pivot table*.

On editing pivot tables and other objects, see the *User's Guide*.

You may save part of or all of the output to a file (File ► Save or File ► Save as...). The default extension for SPSS output files is **.spo**.

6. Setting preferences

Here I give some recommendations on preferences that you should select before starting to work with SPSS.

Choose a default working folder

The installation default working folder (directory, mappe) is the Program folder (usually C:\SPSSWIN). **This is an extremely poor choice.** You should never mix your own documents and data files with program files; you might never find your own files again, you might accidentally delete them eg. when installing a new version of the program, or you might accidentally delete program files.

In appendix 4 I show how to create a desktop shortcut for Explorer (Stifinder). You should use the same method to create an SPSS desktop shortcut.

Make C:\DOKUMENTER the default SPSS working folder:

- *Right-click* the SPSS shortcut icon
- Properties ▶ Shortcut ▶ Start in ▶ **C:\dokumenter**
(Egenskaber ▶ Genvej ▶ Start i ▶)

Page setup

Default page size, margins etc. can be defined. In SPSS select the Output (Viewer) window and:

File ▶ Page Setup ▶ Options

I recommend to include printing date and time in the header. If you share printer with others you should also insert your own name in the header. The specifications will have effect for the current window, but if you press [Make Default] your header will always be printed.

Selecting font for the Syntax window

From the Syntax Window:

View ▶ Font.

I suggest e.g. Courier New 9pt or Letter Gothic Bold 9pt.

Selecting font for the Data window

From the Data window (Data editor):

View ▶ Font

I suggest e.g. Arial 8pt.

SPSS default options

In SPSS choose Edit ► Options. There are a large number of settings you may choose. The following is my recommendation for a start; the experienced user may make some other choices.

Tab	Item	Suggestion	Comments
General	Session journal	<input checked="" type="checkbox"/> Record syntax in journal (e.g. C:\WINDOWS\TEMP) <input type="checkbox"/> Overwrite <input checked="" type="checkbox"/> Append (beginners))	The journal file includes all syntax etc. and can be edited in a syntax window. Beginners should probably select <input checked="" type="checkbox"/> Append
	Special workspace memory limit	512 KB	
	Open Syntax window at Start-up	<input checked="" type="checkbox"/> Yes	Enables you to <i>paste</i> syntax from the procedure menus
	Measurement system	Centimetres	
	Variable lists	<input checked="" type="checkbox"/> Display names <input type="checkbox"/> File	Display variable names rather than labels in menus Display variables in data set order
	Recently used files list	9	Clicking on <u>F</u> ile shows the 9 most recently used files
	Output Type at start-up	<input checked="" type="checkbox"/> Viewer	I don't recommend the Draft viewer
	Output notification	<input checked="" type="checkbox"/> Raise viewer window <input checked="" type="checkbox"/> Scroll to new output <input type="checkbox"/> System beep	
Viewer (the output window)	Initial Output State	<input checked="" type="checkbox"/> Display commands in the log For all items (Log etc.): <i>Contents are initially:</i> <input checked="" type="checkbox"/> Shown	Gives IMPORTANT documenting information in output. You may want not to see the Notes, the other items are important.
	Title font	Arial 12pt Bold	
	Text output page size	<i>Width:</i> <input type="checkbox"/> Custom 100 <i>Length:</i> <input checked="" type="checkbox"/> Infinite	Some procedures give the good old text output. Width must match the text font: Courier New 9pt max. 100 Letter Gothic Bold 9pt max. 110 Infinite length saves a lot of paper
	Text output font	<input checked="" type="checkbox"/> Monospaced on Courier New 9pt or Letter Gothic Bold 9pt	Choose a small font to enable printing of wide tables. The font must be monospaced.
Draft Viewer			I don't recommend to use the draft viewer.
Output labels	Outline Labelling	Variables: Names Values: Values	
	Pivot table labelling	Variables: Names and Labels Values: Values and Labels	This gives the most informative output

Tab	Item	Suggestion	Comments
Charts	Fill Patterns and Line Styles	☉ Cycle through patterns	To avoid multi-coloured graphs
Interactive			I have no specific suggestions
Pivot tables	Table look	SPSS Doc (Corner)	You may prefer other table looks, but this one works
	Adjust column widths for:		
	Default editing mode	Edit only small tables in viewer	Larger tables can be edited in a separate window
Data	Transformation and merge options	☉ Calculate values before used	This runs fastest, but transformations wait until the first procedure. Beginners might prefer: ☉ Calculate values immediately
	Display format for new numeric variables	Width: 8 Decimals: 2	
	Century Range for 2-digit years	☉ Custom 1900	It is probably wise to prevent a year 2000 crisis by always using 4-digit years
Currency	Decimal separator	☉ Period	Also select Period in your Windows international settings (see below). Otherwise the result will be a bit confusing. In syntax you must <i>always</i> use Period.
Scripts		<ul style="list-style-type: none"> ✓ Enable autoscripting ✓ Crosstabs_Table_Crosstabulation_Create 	Creates a much nicer look for crosstables

On decimal periods and commas

The general Windows settings always override the SPSS preferences (the Currency tab). This means that you must choose between decimal period and comma in the Windows settings. The Windows settings are chosen from the Windows start button by:

[Start] ▶ Settings (Indstillinger) ▶ Control Panel ▶ International ▶ Numbers

The Windows settings have the following effects:

In output: Decimal commas/periods are shown as chosen in the Windows settings.

In syntax: No effect. You must always use decimal period; comma is interpreted as a delimiter regardless of settings.

ASCII data: In ASCII data read by the **DATA LIST** command, decimal signs must conform the Windows settings.

7. On-line Help

There are several ways to get on-line help:

Help ▶ Topics	Alphabetic index
Help ▶ Tutorial	A guided tour through SPSS
Help ▶ Syntax Guide	During installation (Custom) install the complete Syntax Guide.
[/--]	This syntax window tool button shows a syntax diagram for the command next to the cursor.

Many menu dialogues include a Help button which provides help specific to that dialogue.

8. Two ways to run SPSS

1. Mouse and menus

You use the mouse to select your choices from menus. The *User's Guide* describes the details. You can execute commands directly or *paste* them to the designated Syntax Window.

2. Write commands in the syntax window

Use the syntax window to write commands. Using mouse and menus, you can also *paste* the commands to the syntax window. This booklet describes the most common commands, and *SPSS Syntax Reference Guide* describes all commands available.

To execute one or more commands from the syntax window you must select (highlight) them. You can highlight commands in two ways:

- Move the cursor while holding the [Shift] key (my recommendation).
- Move the mouse while holding the left mouse button.

You can highlight the entire text in the syntax window by [Ctrl]+[A]

Execute selected commands by [Ctrl]+[R] or by clicking the Run-button [▶] in the tool-bar.

My recommendation

By far the fastest and safest method is to learn the fundamental commands and enter them in the syntax window (method 2). Requesting transformations (**COMPUTE**, **RECODE** etc.) and data documentation (labels etc.) with the menu system is *very* time-consuming.

For complex procedures I often use the menu facilities to create the command. I strongly recommend that you *paste* the command created to the syntax window, where you may edit it before execution.

If the syntax file is worth keeping, save it with a reasonable name before leaving SPSS. This is especially important for syntax files creating or modifying data (see section 9 on recommended file names).

9. File types and file names

SPSS uses the following standard extensions:

- .sav** SPSS system file (data with documentation)
- .sps** SPSS syntax file (one or more SPSS commands to be executed)
- .spo** Output files
- .jnl** Journal file

SPSS system files

An SPSS system file (**.sav**) includes data and data documentation (variable names, labels, etc.). It can be interpreted by the SPSS software only.

SPSS syntax files

A syntax file (**.sps**) includes one or more SPSS commands to be interpreted and executed by the SPSS software.

I suggest a special *prefix* (e.g. **gen.**) to identify syntax files that generate SPSS system files. Such syntax files include vital documentation, and they should not be lost in the crowd of less important syntax files. My recommendation is that e.g. the syntax file generating **vinol.sav** should be named **gen.vinol.sps**. You can easily identify these files by specifying **gen*.sps**.

Output files

Output files (**.spo**) include output as a result of statistical analyses etc. There are three types of 'objects' in the output:

- text output (in ANSI format, not ASCII, in case you want to copy it to a document)
- pivot tables
- charts

The journal file

The journal file (**spss.jnl**) includes commands etc. from the most recent SPSSWIN session. It can be used to reconstruct what happened, but in most instances you don't use it.

The working file

The working file is a temporary file created when including data, eg. by a **DATA LIST** or **GET FILE** command. It is displayed in the Data Window (see section 5). It has no name, and is lost when leaving SPSS, unless you have saved it.

REPEATED WARNING:

When you close SPSS you are asked if you want to save the contents of the data window (the working file). THE ANSWER SHOULD BE NO. See the warning in section 4.

If you modified your working file, and want to save the modified data to disk, do it explicitly with a syntax file ending with the **SAVE OUTFILE** command.

10. SPSS syntax rules

Commands

A job is a sequence of **commands**. The following rules apply:

- Commands begin with a keyword (the name of the command), and are terminated by a . (period). A blank line also terminates a command.
- The maximum length of a command line is 80 characters. You can use continuation lines.
- Command lines begin in column 1. In continuation lines the first column should be left blank.
- A command can include one or more **subcommands**, these are delimited by / :
`CROSSTABS a BY b / CELLS=COUNT ROW / STATISTICS=CHISQ.`

Most SPSS keywords can be abbreviated, so the same command can be written:

```
CRO a BY b / CEL=COU ROW / STA=CHI.
```

But for readability I prefer this style:

```
CROSSTABS a BY b
  /CELLS = COUNT ROW
  /STATISTICS = CHISQ.
```

Variable names

Variable names are 1 to 8 characters, the first being a letter. The Danish Æ, Ø, Å, and most special characters are not permitted. Examples of valid variable names are:

```
sex quest7 a v_47 dia.ind
```

TO keyword

When defining new variables (eg. with `DATA LIST`, section 15), a series of numbered variables, eg. `q1 q2...q17`, can be indicated by the keyword `TO`:

```
DATA LIST FILE = 'c:\dokumenter\wine\vino.dat' LIST
  /q1 TO q17.
```

When reading from a working file, the keyword `TO` indicates a consecutive sequence of variables, in file order. If the variables `age height weight` are consecutive in the working file, they can be referred to by:

```
FREQUENCIES age TO weight.
```

ALL keyword

With the keyword `ALL` you indicate all variables in the working file:

```
DESCRIPTIVES ALL.
```

Scratch variables

The prefix `#` means a scratch variable; it lives until the next procedure, and it will not be saved with the 'normal' variables. Scratch variables are useful as intermediate variables in complex calculations:

```
COMPUTE #pi=3.1415926536.
```

11. Editing in the Syntax Window

The syntax window is a simple text editor using general Windows editing facilities.

You open an existing syntax file by: File ▶ Open ▶ Syntax

You create a new syntax window by: File ▶ New ▶ Syntax

Cursor movements

[Ctrl]+[Home]	Start of document
[Page Up]	One screen up
[↑]	One line up

[Home]	[Ctrl]+[←]	[←]
Start of line	One word left	One char left

[→]	[Ctrl]+[→]	[End]
One char right	One word right	End of line

[↓]	One line down
[Page Down]	One screen down
[Ctrl]+[End]	End of document

Other editing facilities

Action	Mouse/menu	Keyboard
Delete one character forward		[Delete]
Delete one character backward		[Backspace]
Highlight text block	Press left button while moving mouse	Press [Shift] while moving cursor
Highlight all		[Ctrl]+[A]
COPY highlighted text to <i>clipboard</i>	Edit ▶ Copy	[Ctrl]+[C] *
CUT (delete) highlighted text and copy to <i>clipboard</i>	Edit ▶ Cut	[Ctrl]+[X] *
PASTE <i>clipboard's</i> contents	Edit ▶ Paste	[Ctrl]+[V] *
CLEAR highlighted text	Edit ▶ Clear	[Delete]
Find	Edit ▶ Find	[Ctrl]+[F]
Replace	Edit ▶ Replace	[Ctrl]+[H]
Undo last correction	Edit ▶ Undo	[Ctrl]+[Z]

*) The same keys can be used in Windows Explorer (Stifinder) to move and copy files and folders (see appendix 4)

12. Create SPSS system file.

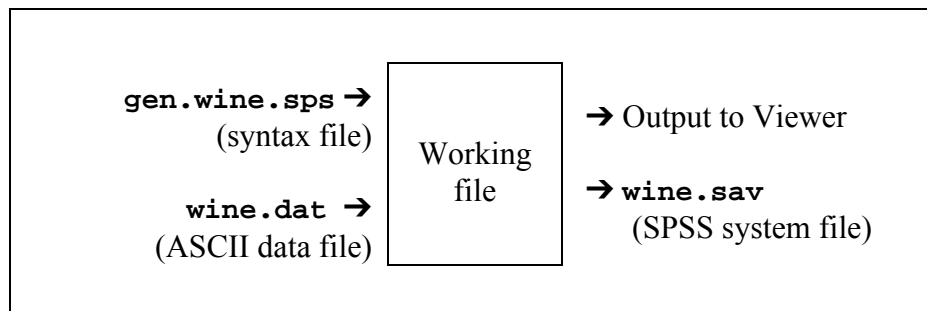
A. Read data from ASCII file

This example demonstrates creation of the SPSS system file **wine.sav** from the ASCII data file **wine.dat**. The syntax file **gen.wine.sps** is created in the syntax window, and executed by:

[Ctrl]+[A] (highlight all text)

[Ctrl]+[R] (run)

SPSS reads and creates the following files:



The syntax file **gen.wine.sps** could look like this:

The syntax file **gen.wine.sps**

```
DATA LIST FILE =  
'c:\dokumenter\p1\wine.dat'  
/id 1-3 type 4 price 5-10 (2)  
rating 11.
```

```
VARIABLE LABELS  
id 'Identification number'  
/type 'Type of wine'  
/price 'Price per 75 cl bottle'  
/rating 'Quality rating'.
```

```
VALUE LABELS  
type 1 'red' 2 'white' 3 'rosé'  
4 'undetermined'  
/price 0 'unknown'  
/rating 1 'poor' 2 'acceptable'  
3 'good' 4 'excellent'.
```

```
MISSING VALUES  
price (0).
```

```
SAVE OUTFILE =  
'c:\dokumenter\p1\wine.sav'.
```

Creates a working file. Data are read from an ASCII data file. Variables are given names, and for each variable the position in the data line is indicated. (see the **DATA LIST** command in section 15)

Defines a text label for selected variables, for documentation and improved readability of output.

Defines text labels for individual values of selected variables.

Defines a code for missing information, to be handled in a special way in calculations.

Copies the working file to disk.

The individual commands are explained in sections 15 and 16.

The information in the SPSS system file **wine.sav** can only be interpreted by the SPSS software. The system file consists of two parts: definitions (variable names, labels etc.) and data. You make definitions only once; they remain part of the system file.

B. Enter data in EpiData, and create an SPSS system file.

It is possible to define variables in the Data Window by clicking a lot with the mouse. You might try it, but I don't recommend it because documentation vanishes in the air, corrections are difficult, and it is time-consuming and inefficient. The Data Window is not a Data Entry system proper, and it is impractical for entering major data sets.

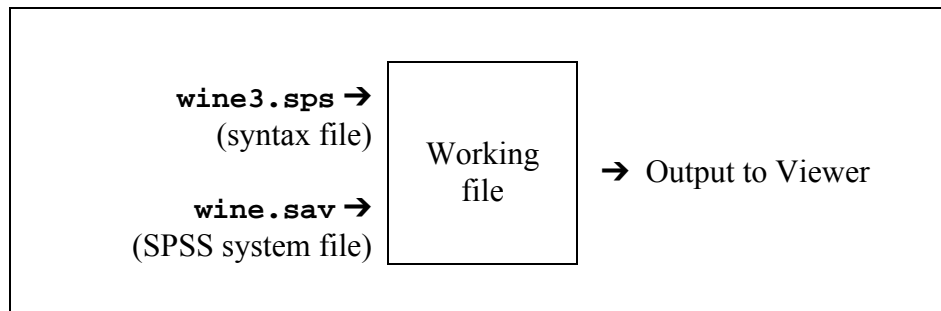
For entering data I recommend EpiData, available for free from www.epidata.dk. I wrote a short introduction in *Take good care of your data*, and from the EpiData site you can download more extensive documentation. EpiData can generate SPSS, Stata, SAS, EpiInfo, and Excel files.

With small data sets you can enter data directly in the syntax file (see **DATA LIST** command, section 15).

13. Analyse data in SPSS system file.

Once you have created a system file, production of tables etc. is rather simple. The example shows production of various tables from the system file; you have created the syntax file WINE3.SPS in the syntax window.

SPSS reads and creates the following files:



The syntax file **wine3.sps** could look like this:

<pre>GET FILE = 'c:\dokumenter\p1\wine1.sav'. DISPLAY LABELS. DESCRIPTIVES ALL. FREQUENCIES rating. CROSSTABS type BY rating.</pre>	<p>Copies SPSS data from disk to the working file.</p> <p>Performs four procedures, each creating a separate output. See section 18.</p>
--	--

14. Some common errors

SPSS gives an error message which is sometimes difficult to interpret. Quite often the error occurred *before* the line SPSS points to. The most frequent errors are due to missing or premature command terminators (the period).

<pre>FREQUENCIES rating.</pre>
Error # 105. Command name: FREQUENCIES This command is not valid before a working file has been defined.
No working file was defined. Most commands require that a working file has been defined, eg. by a GET FILE command.

<pre>GET FILE = 'c:\dokumenter\p1\vina.sav'.</pre>
Error # 31 in column 12. Text: C:\dokumenter\p1\VINA.SAV File not found.
No file with the name requested exists. You may have misspelled the name, or the file is in another folder.

<pre>GET FILE = 'c:\dokumenter\p1\wine.sav' FREQUENCIES rating.</pre>
Error # 5213 in column 1. Text: FREQUENCIES GET is expecting one of the keywords RENAME, KEEP, DROP, or MAP at this point, and the symbol is none of those. Either the keyword is misspelled, or there is a punctuation error.
You forgot to terminate the GET FILE command with a period. SPSS attempted in vain to interpret FREQUENCIES as part of the GET FILE command.

<pre>* I now want a frequency table FREQUENCIES rating.</pre>
You got no frequency table and no error message. Reason: You did not terminate the comment line with a period, and SPSS interpreted the next line as a continued comment.

<pre>GET FILE = 'c:\dokumenter\p1\wine.sav'. FREQUENCIES rating. /STATISTICS = MEAN STDDEV.</pre>
Error # 1. Command name: /STATISTICS The first word in the line is not recognized as an SPSS command.
You terminated the FREQUENCIES command prematurely with a period. SPSS attempted in vain to interpret /STATISTICS as the beginning of a new command.

For the errors shown you will get a message, but other errors are common, such as:

- making changes to data in the data window without documentation.
- unintentionally overwriting the original data at exit (see warning in section 4).
- permanently modifying a data set without saving the syntax file for documentation (see appendix 2 on documentation and safety).

PART 2: COMMANDS

Instructions to SPSS are given as commands. Commands can be created in several ways:

- You can enter the commands in the syntax window, following the general syntax rules (section 10) and the syntax specific to the command (this part of the booklet).
- Using the menu system you can create a command without knowing the specific syntax and *paste* the command to the designated Syntax Window. You now have opportunity to edit the command before execution.
- You can also execute the command directly from the menu system without editing. Also in this case a command is created; you can see it in the output window (if you have set preferences right, see section 6).

I recommend that you learn the syntax of the commands used most frequently and enter them directly in the syntax window (method a), while you use the menu system to create complex or unfamiliar commands and *paste* them to the syntax window before execution (method b).

Commands come in four families.

Section	Page	Family	Comment	Examples
15	19	File commands	Create, open or save system files	DATA LIST GET FILE SAVE OUTFILE
16	22	Documentation commands	Add supplementary information to the file	VARIABLE LABELS VALUE LABELS MISSING VALUES
17	24	Transformation commands	Create new variables or modify the value of existing variables	COMPUTE IF RECODE
18	30	Procedure commands	Create output, eg. tables and graphs	DESCRIPTIVES FREQUENCIES CROSSTABS

You cannot execute documentation, transformation, and procedure commands until a working file has been created with a file command.

15. File commands

File commands are used to read and write files. An SPSS job must always start with a file command that creates a working file (eg. **DATA LIST**, **GET FILE**)

DATA LIST

File ► Read Text Data

Defines variables for a new data set and reads an ASCII data file. You could have created **wine.dat** by using a text editor, eg. EDIT, NotePad, or an SPSS syntax window.

Fixed format data

In fixed-format data the information on each variable is in a fixed position in the data line:

wine.dat	Explanation (not included in data file)
0011 19951	ID=1, TYPE=1, PRICE= 19.95, RATING=1
0021 147004	ID=2, TYPE=1, PRICE=147.00, RATING=4
0032 49952	ID=3, TYPE=2, PRICE= 49.95, RATING=2

The command to read this fixed-format data file is:

```
DATA LIST FILE='c:\dokumenter\p1\wine.dat'  
  /id 1-3 type 4 price 5-10 (2) rating 11.
```

The command reads 4 variables from **wine.dat**; the variables are given names, and it is indicated from which positions the values should be read. **(2)** after **price** indicates that this variable should be read with two decimal digits (the decimal point needs not be included in the data file).

If you have 10 consecutive variables with 2 digits each, the reading format can be specified by:

```
DATA LIST FILE='c:\dokumenter\p1\wine.dat'  
  /id 1-3 v1 TO v10 4-23.
```

WARNING: Contrary to other commands, **DATA LIST** does **not** use **/** as a delimiter between subcommands. Instead **/** is used for marking a line shift. In the following example **id** and **type** are read from line 1, and **price** and **rating** from line 2:

```
DATA LIST FILE='c:\dokumenter\p1\wine.dat'  
  /id 1-3 type 4  
  /price 1-6 (2) rating 7.
```

List format data

In list-format each data line includes information about one case. The variables do not occupy fixed positions in the data line, but a delimiter (blank or comma) separates the variables:

```
1 1 19.95 1  
2 1 147.00 4  
3 2 49.95 2
```

The command to read a list-format data file could look like this (it is recommended to specify the output format; see the **FORMAT** command):

```
DATA LIST FILE='c:\dokumenter\p1\wine.dat' LIST
  /id (F4) type (F1) price (F6.2) rating (F1).
```

Data might be separated with other characters than blank or comma. The following commands read semicolon-separated and tab-separated data.

```
DATA LIST FILE='c:\dokumenter\p1\wine.dat' LIST(";")
  /id (F4) type (F1) price (F6.2) rating (F1).
```

```
DATA LIST FILE='c:\dokumenter\p1\wine.dat' LIST(TAB)
  /id (F4) type (F1) price (F6.2) rating (F1).
```

Free format data

In free format data are read sequentially, without assigning any meaning to line shifts. The following may be the data for three cases:

1 1 19.95 1	2 1 147.00 4	3 2 49.95 2
-------------	--------------	-------------

The following syntax reads data of this structure:

```
DATA LIST FILE='c:\dokumenter\p1\wine.dat' FREE
  /id (F4) type (F1) price (F6.2) rating (F1).
```

Small data sets

With small data sets data can be entered directly in the syntax window (inline data). Do not specify a file name, but include data between **BEGIN DATA** and **END DATA**. This method is inconvenient for large data sets.

```
DATA LIST LIST
  /id (F4) type (F1) price (F6.2) rating (F1).
BEGIN DATA.
1 1 19.95 1
2 1 147.00 4
3 2 49.95 2
END DATA.
```

SAVE OUTFILE

File ▶ Save

Creates an SPSS system file on the disk from the working file:

```
SAVE OUTFILE='c:\dokumenter\p1\wine.sav'.
```

The **DROP** subcommand excludes variables from the system file.

```
SAVE OUTFILE='c:\dokumenter\p1\wine.sav'  
  /DROP = price.
```

The **KEEP** subcommand selects variables to be included in the system file:

```
SAVE OUTFILE='c:\dokumenter\p1\wine.sav'  
  /KEEP = id type rating.
```

GET FILE

File ▶ Open ▶ Data

Creates a working file from an SPSS system file.

```
GET FILE='c:\dokumenter\p1\wine.sav'.
```

The **KEEP** and **DROP** subcommands may be used to restrict the number of variables in the working file:

```
GET FILE='c:\dokumenter\p1\wine.sav'  
  /KEEP = type price rating.
```

16. Data documentation

Before you start, you should have made a complete **codebook** (see example in exercise 2). Create the codebook before coding and data entry. It is a necessary documentation for this process, and later when working with your data.

VARIABLE LABELS

(Data Window, Variable View)

To include explanatory texts of variables in a system file. Labels are automatically printed with tables etc. Contrary to variable names, any character can be used, including Æ, Ø, and Å.

VARIABLE LABELS

```
type 'Type of wine'  
/price 'Price per 75 cl bottle'  
/rating 'Quality rating'.
```

VALUE LABELS

(Data Window, Variable View)

To include explanatory texts for the individual values of a variable. A value label can have a maximum of 16 characters.

VALUE LABELS

```
type 1 'red' 2 'white' 3 'rosé' 4 'undetermined'  
/rating 1 'poor' 2 'acceptable' 3 'good' 4 'excellent'.
```

FORMATS

(Data Window, Variable View)

Used to determine formats for output. If a variable is read by **DATA LIST** (fixed format) the corresponding output format is in effect. **FORMATS** does not affect the internal values in a data set.

(**F2**) or (**F2.0**) causes the values to occupy 2 positions and no decimal digits, while (**F5.2**) means 5 positions (including decimal period), 2 of which are after the decimal point.

(**A10**) is the format for a 10 character string variable (on string variables, see section 21).

There are special output formats for dates; (**EDATE8**) corresponds to 22.12.98, and (**EDATE10**) to 22.12.1998 (on date variables, see section 23).

FORMATS

```
type rating (F1)  
/price (F7.2)  
/name (A30)  
/enddate (EDATE10).
```

MISSING VALUES

(Data Window, Variable View)

If you miss information on eg. the price of the wine, you must assign a special (unrealistic) code to the price. In the case of a price, 0 is a good choice. To make sure that cases with this code are not included in calculations of eg. the average price of wine samples, this code should be defined as a missing value:

```
MISSING VALUES
  price (0)
  /age height weight (999).
```

You can define 3 missing values per variable. Beyond this, there is a special **SYSMIS** value, which in output is shown by . (period). This value is created with missing data, and with undetermined results of calculations, eg. when a missing value is included in a calculation, or by division by 0.

More on Missing Values: See section 20.

COMMENT or *

You may include comments anywhere in a syntax file to explain (to yourself and others) the intent of e.g. transformations. This is very useful when making complex transformations. *Like other commands a comment must be terminated with a period*; if you forget the period, the next command will be considered part of the comment and is not executed.

```
* Calculate z: grouping according to sex and age .
***** .
(complex sequence of transformation commands follows).
```

You may also include a comment at the end of a line, using **/* :**

```
COMPUTE bmi=weight/(height**2). /* Calculate Body Mass Index.
```

DOCUMENT

You may include a text-block in an SPSS file to explain its contents. The document will be included in following versions of the data set.

```
DOCUMENT 4.7.1999. This file was combined from the prescription
data base and interview data. Information from
Landspatientregisteret will be included later.
```

To see any documents in a file:

```
DISPLAY DOCUMENTS.
```

17. Transformation commands

You can create new variables or change the value of existing variables by transformations. Transformations don't actually take place until a procedure is executed (a procedure makes the data to be read); if requested transformations have not yet been executed, you will see the text Transformations Pending on the bottom line.

The 'dummy' procedure **EXECUTE** does nothing but forcing the data to be read, and transformations to take place.

The **TEMPORARY** command makes the subsequent transformation commands temporary: they are in effect only through the first procedure.

COMPUTE

Transform ► Compute

Creates a new variable (or changes the value of an existing variable). If **weight** is a person's weight in kilograms, and **height** the height in metres, **bmi** (Body Mass Index) can be computed as $\text{weight}/\text{height}^2$:

```
COMPUTE bmi = weight/(height**2).
```

/ and ** are operators in the calculation. You can use the following operators, here written in order of precedence (operators with highest precedence are executed first, unless parentheses are used to indicate precedence):

- ** raise to power
- * multiplication
- / division
- + addition
- subtraction

To this come a number of functions; a full list is shown in the *Syntax Reference Guide*:

COMPUTE y=ABS(x).	absolute value of x. ABS(-7)=7.
COMPUTE y=SQRT(x).	square root
COMPUTE y=LN(x).	natural logarithm
COMPUTE y=LG10(x).	base 10 logarithm
COMPUTE y=EXP(x).	exponential: e ^x
COMPUTE y=TRUNC(x).	integer part. TRUNC(5.7)=5.
COMPUTE y=RND(x).	round to nearest integer. RND(5.7)=6
COMPUTE y=MOD(x,11).	remainder after division by 11
COMPUTE y=SUM(x1,x2,x3).	sum of 3 variables if at least one is non-missing
COMPUTE y=SUM.5(x1 TO x10).	sum of 10 variables if at least 5 are non-missing.
COMPUTE y=MEAN.2(x1,x2,x3).	mean of 3 variables if at least 2 are non-missing
COMPUTE y=LAG(x).	x from previous case
COMPUTE y=\$SYSMIS.	sets Y to sysmis.

The result of a calculation is set to **SYSMIS** (.) if one of the variables in the calculation has a missing value (exception: **SUM** and **MEAN**), or if the result is otherwise undetermined, eg. by division by 0.

IF

Transform ► Compute

For conditional transformations. The general form is:

IF (condition) transformation.

SPSS evaluates whether the condition for the current case is true or false; only if the condition is true, the calculation is performed, like **COMPUTE**:

IF (sex=1) fvs=fv/17.

In conditions you can use the following logical operators:

= or **EQ** Equal to
 <> or **NE** Not Equal to
 > or **GT** Greater Than
 >= or **GE** Greater than or Equal to
 < or **LT** Less Than
 <= or **LE** Less than or Equal to
AND or **&** Both conditions fulfilled
OR At least one condition fulfilled
NOT Condition false

You can combine conditions, and parentheses can be used to specify the precedence of evaluation:

IF ((sex=1)AND(age>=50))group=2.

IF (NOT((sex=1)AND(age>=50)))group=1.

but with complex conditions, the **DO IF...END IF** syntax (see next page) may be more transparent.

Missing values in conditions are tricky. If a variable in a condition is **SYSMIS** or missing, the condition is not fulfilled, and is evaluated as false. Below are illustrated circumstances when a condition is evaluated as true (T) and false (F). 99 was defined as missing value for **age**:

Condition	Code for AGE			
	40	55	99 (missing)	sysmis
IF (age>50)..	F	T	F	F
IF (NOT(age>50))	T	F	F	F
IF (VALUE(age)>50)..	F	T	T	F
IF (MISSING(age))..	F	F	T	T
IF (SYSMIS(age))..	F	F	F	T

DO IF ... END IF

You can decide that one or more transformations are conditional, taking place only if a condition is fulfilled. Conditions are specified as in the **IF** command. You can specify as many transformations as you want under the condition.

```
DO IF (id<1000).  
COMPUTE center=1.  
RECODE y (3=2).  
END IF.
```

The command enables to specify complex conditions, and the conditions can be nested. The various typefaces are for illustration purposes only.

```
DO IF (sex=1).  
DO IF (age<50).  
COMPUTE group=1.  
ELSE IF (age<70).  
COMPUTE group=2.  
ELSE.  
COMPUTE group=3.  
END IF.  
ELSE.  
COMPUTE group=4.  
END IF.
```

RECODE and RECODE...INTO

Transform ▶ Recode

You can change values of a variable, eg. by grouping a variable with many values:

```
RECODE  
  age (0 THRU 4=0)(5 THRU 14=1)(15 THRU 24=2)...(ELSE=SYSMIS)  
  /price (MISSING=50)  
  /rating1 rating2 (1 2 3=1)(4 5=2)(6 7 8=3).
```

You may use keywords **LO** and **HI** for the lowest and highest values of a variable:

```
RECODE age (LO THRU 34=1)(35 THRU 54=2)(55 THRU HI=3).
```

In the above examples, the variable **age** changes values. It is often desirable to keep the original variable, and create a new recoded variable, using the **INTO** keyword:

```
RECODE age (0 THRU 4=0)(5 THRU 9=5)...(ELSE=COPY) INTO agegrp.
```

By this command, **age** is kept unchanged, while **agegrp** is the recoded variable. (**ELSE=COPY**) has the effect that original values not specified in the list are transferred unchanged to the new variable; without this specification such values will be recoded to **SYSMIS**.

If you recode a variable which can take non-integer values, eg. because it was created by a calculation, you must be sure to include non-integer values in the groups desired:

```
RECODE age (LO THRU 34.999=1)(34.999 THRU 54.999=2)(ELSE=3).
```

WARNING: Missing values are recoded according to their numerical values. Even if 99 is defined a missing value, it is in the interval (90 THRU 100), and is recoded accordingly.

COUNT

Transform ► Count

Combines information from several variables by counting how many fulfill a specification:

```
COUNT vpos = v1 v2 v3 (1).
```

The new variable `vpos` is set to 0 if none of `v1 v2 v3` are 1, 1 if one of them is 1, etc.

DO REPEAT ... END REPEAT

This enables you to perform the same transformation for many variables with few commands:

```
DO REPEAT
  x = age1 age2 age3 age4 age5
  /y = agr1 to agr5.
COMPUTE y=10*TRUNC((x+.001)/10).
END REPEAT.
```

This sequence calculates 5 new variables (age in 10 year intervals) from 5 original variables (continuous age). The reason for adding a small number before truncation is that the internal representation of eg. 4 may be 3.9999999, which is truncated to 3, not 4.

If you add `PRINT` to the `END REPEAT` statement the program displays the unfolded commands created (in the above example five `COMPUTE` commands):

```
END REPEAT PRINT.
```

SELECT IF

Data ► Select cases

This command reduces the working file to those cases that fulfill a condition, eg.:

```
SELECT IF (sex=1).
```

On syntax of conditions, see the `IF` command in section 17. `SELECT IF` is in effect for the rest of the session – unless you define a new working file, eg. with `GET FILE`. `SELECT IF` commands are cumulative: If you have several `SELECT IF` commands, you will keep only cases that fulfill all conditions specified.

A temporary selection, valid for the first procedure only, can be obtained by the `TEMPORARY` command:

```
TEMPORARY.
SELECT IF (sex=1).
FREQUENCIES age.
FREQUENCIES age.
```

The first frequency table is for `sex=1` only. The second frequency table will be for both sexes, since the selection was temporary (cancelled after the first frequencies procedure).

If you make temporary selections with the menu facilities, SPSS will create a strange and complex syntax, using the `FILTER` command. For transparency I strongly recommend to use `SELECT IF` instead. (Try it and be convinced).

N

Reduces the working file to the first n cases. The command is useful for test runs on large data sets. You select the first 20 cases with:

```
N 20.
```

SAMPLE

Data ▶ Select cases

You may reduce the working file to a 10% random sample by:

```
SAMPLE 0.1.
```

Read more on sampling in section 24.

SPLIT FILE

Data ▶ Split File

You can produce a set of parallel tables, eg. a frequency table for each sex:

```
SORT CASES BY sex.  
SPLIT FILE BY sex.  
FREQUENCIES age.  
SPLIT FILE OFF.  
FREQUENCIES age.
```

These commands produce a frequency table for each sex, and a joint table. Note that the working file must be sorted by the split variable before splitting.

WEIGHT

Data ▶ Weight Cases

WEIGHT weights cases differentially for analysis. You might want to correct for over- and undersampling in subgroups, or you might weight a sample up to population size. Another use is to enter tabular data, using the number of cases in each cell as the weighting variable. The following commands perform a full Mantel-Haenszel analysis:

```
DATA LIST LIST  
  /stratum exposure outcome n.  
BEGIN DATA.  
1 0 0 7  
1 0 1 14  
1 1 0 23  
1 1 1 19  
2 0 0 17  
2 0 1 12  
2 1 0 13  
2 1 1 29  
END DATA.  
WEIGHT BY n.  
CROSSTABS exposure BY outcome BY stratum  
  /STATISTICS= CHISQ RISK CMH.
```

You turn off weighting by:

```
WEIGHT OFF.
```

TEMPORARY

You can decide that transformations are temporary, having effect only up to and including the first procedure:

```
TEMPORARY.
```

```
RECODE age (0 THRU 4=0)(5 THRU 9=5)...(else=SYSMIS).
```

```
FREQUENCIES age.
```

```
FREQUENCIES age.
```

The first frequency table will show the recoded values of **age**. The second frequency table will show the original values, since the recoding was temporary.

18. Procedure commands

A procedure creates an output, eg. tables and statistical test results, while the transformation commands in section 17 do not create an output. Procedures make the data in the data set to be read, and transformations actually don't take place until a procedure is executed. Here are some elementary procedures.

EXECUTE

Transform ▶ Run Pending Transforms

Does nothing but read the data. This means that transformations already requested take place.

DISPLAY Utilities ▶ File info

Display shows information from the data definition part of a system file (labels, formats, missing values, etc.). No information is given about data values:

DISPLAY.

DISPLAY LABELS.

DISPLAY VARIABLES.

DISPLAY DICTIONARY.

DISPLAY DICTIONARY

/VARIABLES = v1 v2 v3.

DISPLAY LABELS.

ID	-	Identification number
TYPE	-	Type of wine
PRICE	-	Price per 75 cl bottle
RATING	-	Quality rating

DISPLAY DICTIONARY

/VARIABLES=rating.

RATING	Quality rating
	Print Format: F1
	Value Label
1	poor
2	acceptable
3	good
4	excellent

DISPLAY VARIABLES.

Name	Pos	Level	Print Fmt	Write Fmt	Missing Values
ID	1	Scale	F2.0	F2.0	
TYPE	2	Ordinal	F1.0	F1.0	
PRICE	3	Scale	F6.2	F6.2	0.00
RATING	4	Ordinal	F1.0	F1.0	9

CROSSTABS

Analyze ► Descriptive statistics ► Crosstabs

```
CROSSTABS a BY b
/c BY d
/CELLS = COUNT COL
/STATISTICS = CHISQ.
```

```
CROSSTABS rating BY type
/CELLS=COUNT COL
/STATISTICS=CHISQ.
```

			TYPE Type of wine				Total
			1 red	2 white	3 rosé	4 undetermined	
RATING Quality rating	1 poor	Count	4	1	1		6
		Column %	25.0%	9.1%	20.0%		17.1%
	2 acceptable	Count	2	5	1	1	9
		Column %	12.5%	45.5%	20.0%	33.3%	25.7%
	3 good	Count	8	4	3	1	16
		Column %	50.0%	36.4%	60.0%	33.3%	45.7%
	4 excellent	Count	2	1		1	4
		Column %	12.5%	9.1%		33.3%	11.4%
Total		Count	16	11	5	3	35
		Column %	100.0%	100.0%	100.0%	100.0%	100.0%

Chi-Square Tests	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	6.9131	9	.646
Continuity Correction			
Likelihood Ratio	7.530	9	.582
Linear-by-Linear Association	.242	1	.623
N of Valid Cases	35		

1. 14 cells (87.5%) have expected count less than 5. The minimum expected count is .34.

Pearson is the ordinary χ^2 test. Linear-by-Linear Association is a test for trend – which in this case is meaningless, since **type** is a nominal variable. Also note that χ^2 tests are invalid for this table: 14 out of 16 cells have an expected value < 5 .

The most important subcommands are:

```
/CELLS=  COUNT number of cases (default)
          ROW    horizontal percentages
          COLUMN vertical percentages

/STATISTICS=  CHISQ  $\chi^2$  test
              RISK  In 2x2 tables: Odds ratio, Relative risk, 95% CI
              CMH   Cornfield-Mantel-Haenszel analysis of a set of 2x2 tables. (available
                    from version 9.0).
```

You can create tables with up to 10 dimensions. The following command creates a 3-dimensional crosstable: a table for each value (stratum) of **country** with a full Mantel-Haenszel analysis:

```
CROSSTABS exposure BY outcome BY country
/STATISTICS=CHISQ RISK CMH.
```

MEANS

Analyze ► Compare means ► Means

Gives mean, Standard deviation, etc. for continuous variables, in subgroups:

```
MEANS age BY civst BY sex
  /STATISTICS=ANOVA LINEARITY.
```

You will get mean age, etc. for each subgroup (civst, sex). The subcommand tests hypotheses about association.

```
MEANS price BY rating.
```

RATING	Quality rating	Mean	N	Std. Deviation
1	poor	59.1167	6	13.6882
2	acceptable	45.2000	8	18.5376
3	good	44.3500	15	13.3673
4	excellent	46.2000	4	21.5619
Total		47.4652	33	16.0529

T-TEST

Analyze ► Compare means ► Independent samples T-test

Performs t-test for comparison of means between two groups. The following command compares between sexes two variables, height and weight.

```
T-TEST
  GROUPS=type (1 2)
  /VARIABLES=price.
```

	TYPE	Type of wine	N	Mean	Std. Deviation	Std. Error Mean
PRICE Price per 75 cl bottle	1	red	15	48.1500	12.6502	3.2662
	2	white	11	46.6818	24.6262	7.4251

		Levene's Test for Equality of Variances		t-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
									Lower	Upper
PRICE price per 75 cl bottle	Equal variances assumed	5.192	0.032	0.199	24	0.844	1.468	7.384	-13.772	16.709
	Equal variances not assumed			0.181	13.87	0.859	1.468	8.112	-15.944	18.881

Levene's test concerns the assumption of equal variances in the two subgroups. The variances (variance = SD^2) can not be considered equal, and the second test should be used.

GRAPH

Graphs ► . . .

A number of graphs can be produced. See the *User's Guide*. Also, a number of procedures produce graphs. A bivariate scatterplot can be produced by:

```
GRAPH
  /SCATTERPLOT(BIVAR) = weight WITH height.
```

LIST

You can have a list of the content of cases:

```
LIST.
LIST v1 TO v12
  /CASES FROM 1 TO 20.
```

The first command displays all variables for all cases. The second command displays the values of selected variables from the first 20 cases. Without a variable list all variables are displayed, and without the **CASES** subcommand, all cases are listed.

```
LIST
  /CASES FROM 1 TO 5.
```

ID	TYPE	PRICE	RATING
1	2	41.95	2
2	1	42.95	2
3	1	42.95	1
4	1	47.95	1
5	2	.00	2

WRITE

With **WRITE** data are written to an ASCII file.

```
WRITE OUTFILE = 'c:\dokumenter\p1\alfa.dat' TABLE
  /ALL.
EXECUTE.
```

Technically, **WRITE** is not a procedure, but a transformation, and it is not executed before the first procedure. Here, the 'dummy' procedure **EXECUTE** ensures that **WRITE** is executed. The **TABLE** option displays the format of the output file. There is opportunity to specify the output format explicitly (see *Syntax Reference Guide*).

PRINT

PRINT writes data to the output window.

```
PRINT /id age sex.
EXECUTE.
```

Technically, **PRINT** is not a procedure, but a transformation, and it is not executed before the first procedure. Here, the 'dummy' procedure **EXECUTE** ensures that **PRINT** is executed.

19. Advanced file commands

This section illustrates some of SPSS' capabilities for handling complex data structures.

SORT CASES

Data ▶ Sort cases

The sequence in the working file is changed:

```
SORT CASES BY type.  
SORT CASES BY type (A) price (D).
```

The first command sorts the file by **type** (ascending, default). The second command sorts primarily by **type** (ascending), secondarily by **price** (descending).

AGGREGATE

Data ▶ Aggregate

You create a new, aggregated file, which typically includes fewer cases than the original file:

```
AGGREGATE OUTFILE = 'alphaagg.sav'  
  /BREAK = sex civst  
  /number = N  
  /meanage = MEAN(age).
```

The command creates a new system file with as many cases as there are combinations of **sex** and **civst**. The file will contain the variables **sex civst number** (number of cases with that combination of sex, civst), **meanage** (mean age in the group).

For a number of aggregation functions (like **N** and **MEAN**), see the syntax manual.

ADD FILES

Data ▶ Merge files ▶ Add cases

Joins two or more system files, having the *same variables* but *different cases*.

```
ADD FILES  
  FILE = 'c:\dokumenter\p1\fila.sav'  
  /FILE = 'c:\dokumenter\p1\filb.sav'.  
SAVE OUTFILE = 'c:\dokumenter\p1\filab.sav'.
```

MATCH FILES

Data ▶ Merge files ▶ Add variables

If you want to combine the information from two files with information about the **same persons** but **different variables**, use **MATCH FILES**:

```
MATCH FILES  
  FILE = 'c:\dokumenter\p1\fila.sav'  
  /FILE = 'c:\dokumenter\p1\filb.sav'  
  /BY id.  
SAVE OUTFILE = 'c:\dokumenter\p1\filab.sav'.
```

The two files must be **sorted before matching** by the matching key (`id` in the example above), and the matching key must have the same name in both data sets. The other variable names would normally be different. Below, A and B symbolize the variable set in the two input files, while numbers represent the matching key. **SYSMIS** is shown by `.` (period):

FILA	FILB	FILAB
----	----	-----
001A	001B	001AB
002A		002A.
	003B	003.B
004A	004B	004AB

Diagnostics. The /MAP and /IN subcommands

The `/MAP` subcommand gives a useful list of variables from each source. The `/IN` subcommand is useful to check whether all cases in the two files were matched. In a perfect match the value of both IN-variables (`ina` and `inb`) is 1:

MATCH FILES

```
FILE = 'c:\dokumenter\p1\fila.sav' /IN=ina
/FILE = 'c:\dokumenter\p1\filb.sav' /IN=inb
/BY id
/MAP.
```

```
CROSSTABS ina BY inb.
```

Distributing information from a table file

If `filed.sav` gives information about doctors, and `filec.sav` about contacts (varying number per doctor), you can **distribute** information about doctors to the relevant contacts, by defining `filed.sav` as at table. The matching key (`doctorid`) must be in both data sets. Again, both files must be sorted beforehand by the matching key:

MATCH FILES

```
FILE = 'c:\dokumenter\p1\filec.sav'
/TABLE = 'c:\dokumenter\p1\filed.sav'
/BY doctorid.
```

```
SAVE OUTFILE = 'c:\dokumenter\p1\filedc.sav'.
```

This will lead to the following result:

FILD	FILC	FILDC
----	----	-----
001D	001C	001DC
	001C	001DC
002D		(nothing)
003D	003C	003DC
	004C	004.C

Aggregating and matching

If you want to **aggregate** information from contacts with each doctor and combine with information about doctors, it becomes more complicated. Be sure the files are sorted by the matching key. Create an aggregate file from the contact data set, with one case per doctor. Then match this file with the doctor data set.

```

GET FILE = 'c:\dokumenter\p1\filc.sav'.
AGGREGATE OUTFILE = * (* indicates the working file)
  /BREAK = doctorid
  /pctf 'Percentage female' = PGT(sex,1)
  /number 'Number of contacts' = N.
MATCH FILES
  FILE = *
  /FILE = 'c:\dokumenter\p1\filed.sav'
  /BY doctorid.
SAVE OUTFILE = 'c:\dokumenter\p1\filedc.sav'.

```

The file `filedc.sav` contains one case for each doctor, with the variables from `filed.sav`, and the variables `pctf` (percentage females) and `number` (number of contacts).

The LAG function

The `LAG` function reads a value from the previous case:

```
COMPUTE y=LAG(x).
```

If you have records for one or more hospital admissions for a number of persons you can use the lag function to find the number of admissions for each patient, and the first and the last admission. `id` is the patient identifier and `admdate` the date of admission:

```

SORT CASES BY id admdate.
COMPUTE admno=1.
IF(id=LAG(id))admno=LAG(admno)+1.
SORT CASES BY id (A) admno (D).
COMPUTE admtot=admno.
IF (id=LAG(id))admtot=LAG(admtot).

```

The first three lines calculate the admission number (`admno`). Next, cases are sorted in reverse (descending) date order, meaning that for the 'first' case for each patient `admno` is the total number of admissions (`admtot`); this is transferred to the other cases for that patient. Now, first admissions can be identified by `admno=1`, and last admissions by `admno=admtot`:

ID	ADMDATE	ADMNO	ADMTOT
001	12.04.1994	1	3
001	01.10.1995	2	3
001	03.12.1995	3	3
002	31.01.1993	1	1
003	...		

SELECT IF and the LAG function

When combining `SELECT IF` and the `LAG` function the latter must be executed before a selection affected by the `LAG` calculation, otherwise you may invalidate the intended action. The solution is to include the dummy procedure `EXECUTE` before the `SELECT IF` command:

```

IF (cpr = LAG(cpr))xx=1.
EXECUTE.
SELECT IF (xx=1).

```

PART 3: VARIOUS ITEMS

20. More on Missing Values

Numerical variables can have missing values, ie. values not included in calculations. Missing values have no meaning for string variables, even a blank string is a valid string value.

User-defined missing values

See the **MISSING VALUES** command, section 16.

System-missing value

The system-missing value (**SYSMIS**) is shown as . (period) in the output. This value is created:

- if nothing is entered in a cell in the Data Window
- as the result of illegal calculations, eg. division by 0 or the logarithm of a negative number.
- as the result of calculations that include one or more missing values.
- if the value is otherwise not defined.

Missing values in calculations

If one or more values are missing in the example below, the result will be **SYSMIS**.

```
COMPUTE xx=a1+a2+a3+a4+a5.
```

This command gives a valid result if at least two of the variables a1-a5 are non-missing:

```
COMPUTE xx=SUM.2(a1 a2 a3 a4 a5).
```

Missing values in conditions

If a variable in a condition is missing (user-missing or **SYSMIS**), the condition is not fulfilled, ie. evaluated as false (see the **IF** command, section 17). To check for a missing value use:

```
IF (MISSING(age))y=1. (user-missing or SYSMIS)
```

```
IF (SYSMIS(age))y=1. (SYSMIS only)
```

RECODE and missing values

Surprisingly, user-defined missing value are **not** excluded from recoding. If 99 is defined missing for **age**, the following command will set cases coded 99 (missing) to 2:

```
RECODE age (LO THRU 49.5=1)(49.5 THRU HI=2) INTO agegr.
```

You may recode a value to **SYSMIS**:

```
RECODE age (-1=SYSMIS).
```

or missing to a value:

```
RECODE age(SYSMIS=999). (recodes SYSMIS)
```

```
RECODE age(MISSING=999). (recodes all missing values)
```

To set a variable to **SYSMIS**, use the **\$SYSMIS** system variable:

```
COMPUTE x=$SYSMIS.
```

21. String variables

Throughout this text I have demonstrated the use of numeric variables, but SPSS also handles string (text) variables. In almost all circumstances it is easier to handle numeric variables than string variables, but you might receive data sets with string variables from other sources.

String variables can include all characters, also numbers; however numbers are not interpreted by their numeric value, but just as a sequence of characters. String variables are at nominal level, and they are not available for calculations or specification of ranges. The relational operators `>` and `<` (see the `IF` command, section 17) have no meaning with string variables, but `=` and `<>` do have meaning.

In syntax string values must be included in single or double quotes:

```
IF (nation='Danish')z=3.
```

Note that `'Danish'`, `'danish'`, and `'DANISH'` are different string values.

DATA LIST and string variables.

In the `DATA LIST` command the default variable type is numeric; string variables are defined by `(A)` after the variable name and location. The following command reads `id` as a 4 digit numeric variable and `diag` as a 5 character string variable. If you attempt to read data including non-numeric information with a numeric specification you will receive a warning, and the result will be `SYSMIS`.

```
DATA LIST FILE='c:\dokumenter\...\list1.dat'  
  /id 1-4 diag 5-9 (A).
```

Variables may be declared prior to entering data in the data window (see section 12 B):

```
DATA LIST  
  /id (F4.0) diag (A5).  
BEGIN DATA.  
END DATA.
```

Declaration of string variables

While new numeric variables need no declaration before assignment of values, new string variables must be declared first (`DATA LIST` is itself a declaration, and no prior declaration is needed):

```
STRING nation (A10).  
IF (id>=1000)nation='Danish'.
```

Conversion between string and numeric variables

Number strings to numbers

If a CPR number is recorded in `cprstr` (type string), no calculations can be performed. Conversion to a numeric variable `cprnum` can be obtained by:

```
COMPUTE cprnum=NUMBER(cprstr,F10.0).
```

This reads a number from the string, with the format specified.

Non-number strings to numbers

If a string variable is coded as eg. 'M' and 'F', conversion to a numeric variable can be performed by:

```
RECODE sexstr ('M'=1)('F'=2) INTO sexnum.
```

Or you may use **AUTORECODE** which automatically replaces string values with consecutive integers, using the string values as value labels. The **PRINT** subcommand displays the relationship between the string and numeric coding.

```
AUTORECODE sexstr  
  /INTO sexnum  
  /PRINT.
```

Numbers to strings

A number variable can be written to a string variable using the **STRING** function. The format specification defines the write format:

```
STRING cprstr (A10).  
COMPUTE cprstr=STRING(cprnum,F10.0).
```

String manipulations

You may isolate part of a string variable by the **SUBSTR** function. The parameters of the **SUBSTR** function are: Name of source variable; start position; length. In the following **str1** will be the first three characters of **svar**.

```
STRING str1 (a3).  
COMPUTE str1=SUBSTR(svar,1,3).
```

The **CONCAT** function joins (concatenates) two or more strings:

```
STRING svar (A5).  
COMPUTE svar=CONCAT(str1,str2).
```

The **UPCASE** function converts lower case to upper case characters. The **LOWER** function converts upper case to lower case characters. Imagine that ICD-10 codes had been entered in an inconsistent way, the same diagnosis sometimes entered as **E10.1**, sometimes as **e10.1**. These are two different string values, and you want them to be the same (**E10.1**):

```
COMPUTE scode=UPCASE(scode).
```

Handling ICD-10 codes

In the ICD-10 classification of diseases all codes are a combination of letters and numbers (e.g. E10.1 for insulin demanding diabetes with ketoacidosis). This is probably convenient for the person coding diagnoses (an extremely important consideration). However, for the data handling it is quite inconvenient.

My suggestion is to split the 5-character ICD-10 string variables (**scode**) into a one-character string variable (**scode1**) and a 4-digit numeric variable (**ncode2**).

```

STRING scode1 (A1)
  /scode2 (A4).
COMPUTE scode1=SUBSTR(scode,1,1).
COMPUTE scode2=SUBSTR(scode,2,4).
COMPUTE ncode2=NUMBER(scode2,F4.1).

```

What did we obtain? Two variables: a string variable with 26 values (A to Z) and a numeric variable (0.0-99.9). Diabetes (E10.0-E14.9) can now be identified by:

```

COMPUTE diab=0.
DO IF (scode1='E').
IF (ncode2>=10 AND ncode2<15)diab=1.
END IF.

```

Without the splitting, each of the 50 string codes for diabetes should have been specified.

If you received data in ASCII format, the same result can be obtained by letting the **DATA LIST** command read the same data twice as different types:

```

DATA LIST FILE='c:\dokumenter\...\list1.dat'
  /id 1-4 scode 5-9 (A) scode1 5 (A) ncode2 6-9.

```

22. Numbers: integers and non-integers

Numbers created by transformations

The following problem is not specific to SPSS: The sum or product of two integers (Danish: heltal) is an integer. However, the internal representation of the result of the multiplication 5×5 might be not 25, but 24.99999999... (Defining a **FORMAT** of **F3.0** does not affect the internal value, only the output format). In most cases this leads to no serious problems, but in this example the following command would exclude rather than include $x=5 \times 5$.

```
SELECT IF (x >= 25).
```

and this command might be safer:

```
SELECT IF (x > 24.999).
```

Also with **RECODE**, the result may be wrong. To avoid errors with transformed variables, you may define a group like this:

```
RECODE x ... (24.999 THRU 29.999=25) ... INTO xx.
```

If you know that a result must be an integer (e.g. the product of two integers) you may prevent problems by rounding the result to the nearest integer:

```
COMPUTE x=RND(a*b).
```

Problems when importing data from other programs

When importing data from other programs, imprecisions may arise. An example:

Data from a Paradox[®] data base on hospital admissions were translated with DBMS/COPY[®] to an SPSS data file. Patients were identified by the numeric variable **cpr**, and the admission number (**admno**) for each patient was identified by:

```
SORT CASES BY cpr admdate.
```

```
COMPUTE admno=1.
```

```
IF (cpr=LAG(cpr)) admno=LAG(admno)+1.
```

Fortunately it was detected that something went wrong: The same person apparently could have more than one first admission. The explanation was that during translation inaccuracies occurred, and the CPR number 0605401449 could be represented as 605401449.0000...01 or as 605401448.9999...99, meaning that the lagged comparisons did not work. The solution was to round **cpr** to the nearest integer before sorting with:

```
COMPUTE cpr=RND(cpr).
```

To test whether all values of a variable are integers, calculate the remainder after division by 1 and print a frequency table. If the frequency table includes one value only: 0.00000..., all values of **cpr** are integers:

```
COMPUTE test=MOD(cpr,1).
```

```
FORMATS test (F20.16).
```

```
FREQUENCIES test.
```

23. Dates, time, and Danish CPR numbers

Date variables

Date variables are numeric variables, the internal value is the number of seconds since 14 Oct 1582 (start of the Gregorian calendar). In output they can be displayed with different formats; below I show the **EDATE** (European date) formats. On other date formats see *Syntax Reference Guide*, Universals.

Reading date variables

In **DATA LIST** the **EDATE** format reads a date of the format *dd.mm.yy* (eg. 06.05.02) or *dd.mm.yyyy* (eg. 06.05.2002), dependent on whether you specify 8 or 10 digits:

```
DATA LIST FILE='c:\...\alfa.dat'  
  /bdate 1-10 (EDATE) odate 11-20 (EDATE).
```

If an ASCII file includes date information in a non-date format, (eg. 060502), you should read the date information as three separate variables (day, month, year) to enable further calculations:

```
DATA LIST FILE='c:\...\alfa.dat'  
  /bday 1-2 bmon 3-4 byear 5-6.
```

Output formats

EDATE8 displays dates with the format *dd.mm.yy* (06.05.02). **EDATE10** displays *dd.mm.yyyy* (06.05.2002). The above command read two dates (a birth date and a date of operation). Since each variable occupied 10 digits, the output format was automatically set to **EDATE10**.

The corresponding **FORMAT** command is:

```
FORMATS bdate odate (EDATE10).
```

Calculations with dates

Internally, date values are seconds. You can calculate a time interval in years (here an age):

```
COMPUTE opage=(odate-bdate)/(86400*365.25).
```

(1 day=86400 seconds; 1 year=365.25 days).

The **DATE.DMY** function creates a date variable (seconds since 14 Oct 1582):

```
COMPUTE bdate=DATE.DMY(bday,bmon,byear).  
FORMATS bdate (EDATE10).
```

1 July 1985 will be displayed as 01.07.1985.

On CPR numbers: extracting key information

Sometimes you get date information as a CPR number. You can read the CPR number as one variable and the date information from the same columns in the data file:

```
DATA LIST FILE='c:\...\alfa.dat'  
  /cprnum 1-10 bday 1-2 bmon 3-4 byear 5-6 control 7-10.
```

It is also possible to extract the date and sex information from a CPR number read as one variable. `cprnum` is a numeric variable; `cprstr` is the corresponding string variable:

```
STRING cprstr (A10).
COMPUTE cprstr=STRING(cprnum,F10).
COMPUTE bday=NUMBER(SUBSTR(cprstr,1,2),F2).
COMPUTE bmon=NUMBER(SUBSTR(cprstr,3,2),F2).
COMPUTE byear=NUMBER(SUBSTR(cprstr,5,2),F2).
COMPUTE control=NUMBER(SUBSTR(cprstr,7,4),F4).
```

The information on sex can be extracted from the control variable, the `MOD` function calculating the remainder after division by 2 (male=1, female=0):

```
COMPUTE sex=MOD(control,2).
```

Validation of CPR numbers

The modulus 11 test checks the validity of CPR numbers. To check a CPR number, multiply the digits by 4,3,2,7,6,5,4,3,2,1, and sum these products. The result should be divisible by 11.

In order to perform the test, each digit must be a separate variable:

```
DATA LIST FILE='c:\...\alfa.dat'
  /cprnum 1-10 c1 TO c10 1-10.
```

– or the digits can be extracted from the string variable `cprstr`:

```
STRING cprstr (A10).
COMPUTE cprstr=STRING(cprnum,F10).
COMPUTE test=0.
DO REPEAT #i=1 to 10
  /#x=4,3,2,7,6,5,4,3,2,1.
COMPUTE #c=NUMBER(SUBSTR(cprstr,#i,1),F1).
RECODE #c(missing=0).          (1st character in cprstr may be blank, meaning 0).
COMPUTE test=test + #x*#c.
END REPEAT.
```

Now perform the test and display invalid CPR numbers by:

```
COMPUTE test=MOD(test,11).
SELECT IF (test>0).
LIST cprnum test.
```

A year 2000 crisis?

Hardly, but I recommend always to record years with 4 digits.

In CPR numbers the 7th digit includes information on the century of birth:

Pos. 7	Pos. 5-6 (year of birth)		
	00-36	37-57	58-99
0-3	19xx	19xx	19xx
4, 9	20xx	19xx	19xx
5-8	20xx	not used	18xx

Source: <http://www.cpr.dk>

24. Random samples, simulations

Random number functions

SPSS can create 'pseudo-random' numbers:

```
COMPUTE y=UNIFORM(x).      Uniformly distributed in the interval 0-x (each value has
                           the same probability).
COMPUTE y=NORMAL(x).      Normal distribution, mean=0, SD=x.
COMPUTE y=10+NORMAL(2).   Normal distribution, mean=10, SD=2.
```

A number of other random variable functions are available (see *Syntax Reference Guide*, Universals).

If you run the same syntax twice, it will yield *different* numbers. If you need to reproduce a series of random numbers, initialize the seed (a large integer used for the initial calculations):

```
SET SEED = 7654321.
```

Random samples and randomization

You may use the **SAMPLE** transformation to select a random sample of your data set:

```
SAMPLE 0.1.      Selects an approximately 10 per cent random sample.
```

You may also assign a random number to each case, and use that for selecting cases:

```
COMPUTE y=UNIFORM(1).
COMPUTE treat=1.
IF (y>0.5) treat=2.
```

Now the cases are assigned randomly to two treatments.

Creating artificial data sets

You may use **INPUT PROGRAM** to create a working file with 'artificial' data, eg. for simulation purposes. The following sequence defines a file with 10,000 cases and one variable (**i**). Next it is used to study the behaviour of the difference (**dif**) between two measurements (**x1 x2**), given information about components of variance (**sdtotal sdwithin sdbetw**).

```
INPUT PROGRAM.
LOOP i=1 TO 10000.
END CASE.
END LOOP.
END FILE.
END INPUT PROGRAM.
EXECUTE.

COMPUTE sdtotal=20.
COMPUTE sdwithin=10.
COMPUTE sdbetw=SQRT(sdtotal**2-sdwithin**2).
COMPUTE x0=50+NORMAL(sdbetw).
COMPUTE x1=x0+NORMAL(sdwithin).
COMPUTE x2=x0+NORMAL(sdwithin).
COMPUTE dif=x2-x1.
```

25. Exchange of data with other programs

Possibilities vary somewhat between SPSS versions. Use the menus:

File ▶ Save as... and File ▶ Open ▶ Data

and pick the appropriate file type.

If you need to exchange data with SPSS on other platforms (e.g UNIX), create a file in portable format (**.por**) which is common to all SPSS versions.

You may read and write e.g. Excel (**.xls**) and dBase (**.dbf**) files. SPSS versions prior to 10 read only Excel version 4.0 files.

SPSS writes Excel files version 4.0. The syntax is:

```
SAVE TRANSLATE OUTFILE='c:\...\...\xls'  
  /TYPE=XLS  
  /KEEP=v1 v3 v4 v7  
  /FIELDNAMES.
```

The **/FIELDNAMES** subcommand instructs SPSS to write variable names to the first row in the Excel worksheet.

DBMS/COPY and Stat/Transfer

These versatile programs translate between a large number of statistical packages.

Writing and reading ASCII files

Any spreadsheet or analysis program can write and read ASCII files. Exchange of information via ASCII files is not very practical: any data documentation (variable names, labels, missing values, etc.) is lost and must be defined again.

Precautions

Translation between programs may go wrong. Always check if the translation worked as intended, by comparing the contents of the source and the target file. Especially missing value definitions sometimes go wrong. Also take care with date variables. An example:

SigmaPlot imports Excel files, and SPSS data can thus be transferred to SigmaPlot via Excel. SPSS **SYSMIS** is translated to #NULL! in Excel, and SigmaPlot translates #NULL! to 1 – a quite likely valid value.

Appendix 1. Exercises

The purpose of these exercises is to learn SPSS by doing.

READ THIS BEFORE YOU START.

You should start by setting preferences (see section 6).

Next, copy the files needed for exercises to your hard-disk.

I strongly recommend that you enter commands in the syntax window by writing them (occasionally by pasting and editing them) before execution (see section 8). The reasons for this recommendation are:

- You will soon learn that it is much faster to write commands in the syntax window than to zap around in the menus. It is *easy* to learn the fundamental commands and to recall them.
- Using the command language you have a nice tool to plan what to do. Intuitive computing has its merits, but if you are going to produce results of any importance, planning is a good idea.
- The syntax file documents what you did, while it can be impossible to reproduce a series of clicks with the mouse

I also recommend that you save the syntax file for each question (**2g.sps** being the syntax file for question 2g). Once you have saved a syntax file, delete it from the syntax window, to avoid confusion. This means that you for most questions should create a syntax file, starting with a **GET FILE** command.

For exercise 2g the syntax file (`c:\dokumenter\spsskurs\2g.sps`) should look like this:

```
get file='c:\dokumenter\spsskurs\rygel.sav'.
frequencies tobacco.
list variables=cigaret cheroot pipe tobacco
/cases from 1 to 50.
```

Doing this, you will for each question have a good documentation of what you did. Including the **GET FILE** command means that you will be certain what data set actually was analysed.

Some jobs create new system files. This syntax file *must* be saved for documentation (note the recommendation on file names, section 9).

Delete unnecessary text in your output window before printing. In some cases you might want to save the output (**1a.spo** being the output from question 1a).

In the exercise questions I sometimes give a hint about the procedure to be used (eg. **DISPLAY**). Lookup the command syntax in this booklet.

Exercise 1

Objective: To get used to running SPSS jobs and to be acquainted with various procedures and their output. The exercise uses the SPSS system file **beer.sav**.

The meaning of variable names etc. is:

Variable	Meaning	Codes
ID	Brand of beer	
RATING	Rating	1 excellent 2 good 3 not good
COUNTRY	Country of origin	
COST	Price, \$ per bottle	0 missing
CALORIES	Kcal / litre	999 missing
SODIUM	Sodium g/l	99 missing
ALCOHOL	Alcohol vol per cent	99 missing

- a) Look at file contents in the data window.
- b) Create a list of variables in **beer.sav**, including labels etc. (**DISPLAY**). When you have succeeded, print the output.
- c) Create an overview of minimum and maximum values for all variables in **beer.sav** (**DESCRIPTIVES**). Print it.
- d) Create frequency tables for all variables in **beer.sav** (**FREQUENCIES**).
- e) Examine the relationship between price and rating (**MEANS**).
- f) Describe the distribution of **rating** in different price groups (**CROSSTABS**). **cost** must first be grouped in eg. 3 groups (**RECODE**). It is a good idea, when you have recoded, to control the correctness (**LIST**).
- g) Make a graphical description (**GRAPH /SCATTERPLOT**) of the relation between price and alcohol content.
- h) Examine other relationships which you might find interesting.

Exercise 2

Objective: Learn to create an SPSS system file from an ASCII data file. Further experience with output.

Your input is the ASCII data file **ryge.dat**; it is concerned with smoking. The format of **ryge.dat** is shown in the *Codebook* below:

Variable	Meaning	Values	Digits	Position
ID	ID number	1-250	3	1-3
SEX	Sex	1 male 2 female 9 no information	1	4
AGE	Age in years	0-98 99 no information	2	5-6
WEIGHT	Weight in kg	40-150 999 no information	3	7-9
HEIGHT	Height in cm	100-250	3	10-12
SMOKER	Smoker?	0 no 1 current smoker 2 former smoker 9 no information	1	13
CIGARET	Cigarettes/day	0-98 99 no information	2	14-15
CHEROOT	Cigars or cheroots per day	0-98 99 no information	2	16-17
PIPE	Packs of pipe tobacco per week	0-8 9 no information	1	18

- a) See **ryge.dat** on your screen by opening it in e.g. NotePad or a word processor. Is **ryge.dat** an ASCII file?
- b) Create the system file **ryge.sav** from **ryge.dat** (see an example in section 12). You should define Variable labels, Value labels, and Missing values. You should name the syntax-file **gen.ryge.sps** (see section 9 on recommended file names).
- c) See **ryge.sav** on your screen by opening it in e.g. NotePad. Is it an ASCII file? **(NB! Don't print from the data window; you waste a lot of paper).**
- d) Do the same exercises with **ryge.sav** as in exercise 1, question c and d. However, don't create a frequency table for **id**. Print the tables; you need them for the next questions.
- e) Examine graphically the relation between height and weight (**GRAPH /SCATTERPLOT**). Do the same for women only (**SELECT IF**).

- f) Create a new variable, **agegrp**, which is a reasonable grouping of **age** (**RECODE**). Create a new variable, **tobacco**: tobacco use in grams per day (1 cigarette = 1 g, 1 cigar/cheroot = 2 g, 1 pack of pipe tobacco = 40 g) (**COMPUTE**). Define labels for **agegrp** and **tobacco** (**VARIABLE LABELS**). Create a new system file, **ryge1.sav**, including the two new variables (**SAVE OUTFILE**).

What name did you give the syntax file creating **ryge1.sav**? Did you save it?

- g) From **ryge1.sav**: see a frequency table for **tobacco**. Compare with the frequency tables for cigarettes etc. (from question 2d) and decide if the result makes sense. Also, use **LIST** for the first 50 cases to see if calculations have been made as intended. If wrong, redo exercise 2f.
- h) **agegrp** could have been made with **COMPUTE**, using the **TRUNC** function. Try to do that. (Don't feel sorry if you can't find out).
- i) Describe the joint age and sex distribution of the study population (**CROSSTABS**).
- j) Create a new variable, **bmi** (Body Mass Index) = $\text{weight}/\text{height}^2$ (weight in kg, height in m). See a frequency table for **bmi**. See the average **bmi** by sex and age groups (**MEANS**). Test if the **bmi** distribution is different for men and women (**T-TEST**).
- k) Group **bmi** in 3 groups (**RECODE**). See the grouped **bmi** distribution by age and sex (**CROSSTABS**).
- l) **kbmi** (invented just for the sake of this exercise) is a corrected Body Mass Index. For women **kbmi=bmi**. For men, **kbmi** is 90% of **bmi**. Examine the relationship between **kbmi** and **age** (**GRAPH, MEANS, CROSSTABS**). When reasonable, group **age** and **kbmi**.
- m) Make a list of all men, showing the variables **id age weight height bmi kbmi** (**SELECT IF, LIST**). For the new variables, you should beforehand define number of decimals etc. in the output (**FORMAT**).
- n) Create an ASCII data file **rygm.dat** (**WRITE**) with the same content as listed in question 2m. See it on the screen. (An ASCII data file can be used as input to other software).

Exercise 3

Objective: Experience with the whole process of data collection, preparation for data entry, data entry, analysis, and documentation.

Imagine a survey among 15 persons. The questionnaire looked like this:

Questionnaire number:
Sex: <input type="checkbox"/> Male <input type="checkbox"/> Female
Which year were you born?
At what level did you leave school?
Before finishing 9th grade 1
After 9th grade..... 2
After 10th grade..... 3
After high school (gymnasium) 4
Other 5
Do you have a vocational education? (write)

The information in the 15 questionnaires was:

Questionnaire	Sex	Year of birth	School education	Vocational education
1	M	1940	4	Physician
2	F	1963	3	Office clerk
3	F	1936	1	None
4	F	1943	4	Architect
5	M	1950	2	Mason
6	M	1947	3	Carpenter
7	F	1964	4	Nurse
8	F	1961	4	Social worker
9	M	1957		
10	M		5	Sailor
11	M	1932	1	None
12	F	1939	2	Tailor
13	F	1947	2	Shop assistant
14	M	1951	3	None
15	F	1957	4	Law school

- a) Write a codebook (using pencil and paper or a word processor) for the study, as shown in exercise 2. Give numerical codes to sex. Group vocational education by your own choice and give numerical codes.
- b) In EpiData prepare for entering data by creating the dataset definition file **educ0.qes** and the data entry form **educ0.rec**. See more on EpiData in section 12 B and in *Take good care of your data*, appendix 7.
- c) From EpiData print the data documentation for **educ0.rec**. Compare with your codebook.
- d) Enter data in EpiData and save the EpiData file **educ1.rec** and the SPSS file **educ1.sav**.
- e) In SPSS create the variables **age** (age by 31 December 1988) and **agegrp** (age in groups 0-4, 5-14, 15-24, . . . , 65+). Save the file with the new variables as **educ2.sav**.
- f) Print the key tables for your data (**DESCRIPTIVES**, **FREQUENCIES**).
- g) Create whatever tables you find interesting.

This exercise reflects the typical sequence of an investigation. At the end of the exercise you should have the following vital documents and files:

Written documents	Syntax files	Data files	
Codebook	educ0.qes	educ0.rec	Empty EpiData file
Questionnaires		educ1.rec educ1.sav	EpiData file with data SPSS data set
	gen.educ2.sps	educ2.sav	File with added variables

The codebook, the questionnaires, the syntax files, and the data files should be stored in a safe place (safety, documentation, accountability).

Appendix 2. On documentation and safety.

When keeping financial accounts you should be able to document all expenses by identification of the original vouchers. This is accomplished by giving each voucher a unique number, and by enabling you – and the auditor (revisor) – to go back from the final balance sheet to each voucher (the audit trail).

When working with data you should be able to document each piece of information by identification of the original document (eg. questionnaire). This means that an ID (case identifier) must be included both in the original documents and the data set. All modifications to the data set must be documented (syntax files), and each analysis must be documented (syntax files).

One purpose of this is to enable external audit (revision), but the main purpose is to protect yourself against mistakes, errors, and loss of information.

Data documentation procedures must be included all the time when working with data; otherwise it can be impossible – or at least very time-consuming – to reconstruct what happened.

Source data: Questionnaire, hospital records, etc.
Codebook: Describes rules for coding of source data (see example in exercise 2)
If reading data from ASCII data file Format (location of information) is described in the codebook Syntax file creating first generation of SPSS system file: <pre>DATA LIST FILE= 'c:\dokumenter\...\alfa.dat' /.... VARIABLE LABELS... VALUE LABELS... MISSING VALUES... SAVE OUTFILE='c:\dokumenter\...\alfa.sav'.</pre> This syntax file could have the name gen.alfa.sps (see section 9 on recommended filenames).
Error checks: <pre>DESCRIPTIVES ALL. to see minimum and maximum values for all variables FREQUENCIES v1 v7. to see more if needed for selected variables CROSSTABS v1 BY v7. to check impossible combinations (e.g. pregnant males) LIST. to identify cases with suspected errors: TEMPORARY. SELECT IF (sex > 2). LIST id sex.</pre>

Corrections:

It is easy to change values in the data window, but it is dangerous, and documentation is lacking. I strongly recommend to make corrections in syntax:

```
GET FILE='c:\dokumenter\...\alfa.sav'.
IF (id = 2473)sex=2.
IF (id = 2715)bday=17.
SAVE OUTFILE='c:\dokumenter\...\alfa1.sav'.
```

This syntax file could have the name **gen.alfa1.sps**.

Creating next generation of a data set:

```
GET FILE = 'c:\dokumenter\...\alfa1.sav'.
(transformations)
VARIABLE LABELS...
VALUE LABELS...
MISSING VALUES...
SAVE OUTFILE='c:\dokumenter\...\alfa2.sav'.
```

This syntax file could have the name **gen.alfa2.sps**.

Analyses:

You should be able to document the analyses leading to the published tables (syntax files). For this reason (and to be sure to analyse the data set intended) include a **GET FILE** command before the analysis:

```
GET FILE='c:\dokumenter\...\alfa2.sav'.
SELECT IF (sex = 1).
CROSSTABS agr BY item7.
```

If this created the information for table 7 you might save it as **tab7.sps**.

Remove external identification:

The data protection authorities (Registertilsynet) require that you remove external identification from your analysis file as soon as possible. The syntax file **gen.alfakey.sps**:

```
GET FILE='c:\dokumenter\...\alfa2.sav'.
SORT CASES BY id.
SAVE OUTFILE='a:\alfakey.sav'
/KEEP=id cpr.
SAVE OUTFILE='c:\dokumenter\...\alfa3.sav'
/DROP=cpr.
```

The key file (**alfakey.sav**) linking the internal identification (**id**) with the external identification (**cpr**) should be stored separately (ie. not on the same computer as the information). Here I used a diskette, but beware: diskettes are not very stable, so make an extra backup copy.

If you later need to include **cpr**:

```
MATCH FILES FILE='a:\alfakey.sav'
/FILE='c:\dokumenter\...\alfa3.sav'
/BY id.
```

ANOTHER NOTE ON FILE NAMES

Data input	Syntax file	Result
ALFA.DAT (ASCII data file)	GEN.ALFA.SPS <pre>DATA LIST FILE = 'c:\dokumenter\..\alfa.dat' / (variable list) . VARIABLE LABELS... VALUE LABELS... MISSING VALUES... SAVE OUTFILE = 'c:\dokumenter\..\alfa.sav'.</pre>	ALFA.SAV (1st generation SPSS data set)
ALFA.SAV	GEN.ALFA1.SPS <pre>GET FILE = 'c:\dokumenter\..\alfa.sav'. (transformations; create new variables) VARIABLE LABELS... VALUE LABELS... MISSING VALUES... SAVE OUTFILE = 'c:\dokumenter\..\alfa1.sav'.</pre>	ALFA1.SAV (2nd generation SPSS data set)
ALFA1.SAV	TAB1.SPS <pre>GET FILE = 'c:\dokumenter\..\alfa1.sav'. CROSSTABS agegrp sex BY treat. MEANS age BY treat BY sex.</pre>	Analyses for table 1
ALFA1.SAV	TAB2.SPS <pre>GET FILE = 'c:\dokumenter\..\alfa1.sav'. (another analysis)</pre>	Analyses for table 2

Syntax files worth keeping forever:

- Syntax files generating new versions of the data set (**gen.alfa.sps**, **gen.alfa1.sps**). The purpose of the prefix (**gen.**) is to enable you to identify them easily and safely. These syntax files *must* include both the name of the input data file (**DATA LIST** or **GET FILE**) and the output data file (**SAVE OUTFILE**).
- Syntax files generating information for your final publication. Give them names like **tab1.sps**, **tab2.sps**. These syntax files *must* include the name of the input data file (**GET FILE**) to avoid ambiguity on which data set was actually used.

Syntax files probably not worth keeping (forever):

Interim analyses not resulting in information for the final publication.

Appendix 3. SPSS modules and manuals

Module	The module includes:	Manual with US\$ price	Comments
Base	All data handling and transformation procedures. Descriptive statistics, Analysis of variance, linear regression etc.	SPSS Base 10.0 User's Guide Package US\$ 49	The manual is good in describing operations via the menu system while syntax information is virtually absent.
		SPSS Base 10.0 Syntax Reference Guide US\$ 49	A systematic description of the complete syntax. During installation from CD-ROM you may download the manual in PDF format on your computer
		SPSS Base 10.0 Applications Guide US\$ 49	Nice introduction to a variety of statistical analyses
Advanced Models	General linear models, survival analysis including Kaplan-Meier and Cox regression	SPSS 10.0 Advanced Models US\$ 49	Needed eg. for survival analysis
Regression Models	Binomial and multinomial logistic regression, nonlinear regression.	SPSS 10.0 Regression Models US\$ 49	Needed eg. for logistic regression
Tables	Complex tabulations for presentation	SPSS Tables 8.0 US\$ 41	Few users need Tables.
Missing value analysis	Examine missing value patterns. Tools for substitution of missing values	SPSS Missing Value Analysis 7.5 US\$ 38	Substituting missing values should be done with care – or not at all.
Trends	Time series and forecasting analysis	SPSS Trends 10.0 US\$ 29	Hardly used in health research
Conjoint	Conjoint analysis	SPSS Conjoint 8.0 US\$ 20	Hardly used in health research
Categories	Correspondence analysis	SPSS Categories 10.0 US\$ 39	Hardly used in health research

The primary manual is: *SPSS Base 10.0 User's Guide*. It describes how to use the menu facilities in SPSS for Windows.

The command language (common to a number of SPSS platforms) is described in *SPSS Base 10.0 Syntax Reference Guide*. This guide is also included in the installation CD-ROM.

If you have manuals version 8 or 9 you probably don't need to replace them by version 10.

Manuals can be purchased from:

Polyteknisk Boghandel, Anker Engelundsvej 1, 2800 Lyngby, Tel. 4588 1488.

Appendix 4. A few remarks on Windows

It is rather unsafe to use any program without mastering the fundamental structure and facilities in Windows. There are nice and cheap booklets for sale in many kiosks.

My main comments and recommendations apply to handling of the folder (directory, bibliotek, mappe) structure. There are several ways to move and copy files; I only show one technique.

Create a smart folder structure

Don't mix your own data and documents with program files; this is risky and will inevitably lead to confusion.

Create a main folder for all of your own files (data, syntax files, text documents), eg. C:\DOKUMENTER, with all of your own files in subfolders under your main folder.

Example of folder structure.

C:\	C:\ is the root folder
<pre> Programs EpiData SPSS Stata WordPerfect WinZip Games Solitaire Doom Windows </pre>	<p>Program folders should include programs only, <i>never</i> data nor documents created by yourself.</p>
<pre> Dokumenter Personal CV Secrets (encrypted) Project 1 Protocol Administration Data Safe Manuscripts Project 2 Protocol Administration Data Safe Manuscripts </pre>	<p>C:\Dokumenter is your own main folder.</p> <p>All of your own data and documents should be placed in subfolders under your main folder. Organize the folders by subject, not by file type.</p> <p>This structure:</p> <ul style="list-style-type: none"> - Makes it easy for you to locate your own files. - Facilitates the selection of files to be backed up (C:\Dokumenter and its subfolders).

This structure has several advantages:

- a. You avoid mixing own files with program files
- b. You can select your main folder (C:\DOKUMENTER) as the default root folder for all of your own folders (see below, and section 6: setting preferences), so that when opening or saving files, you see only your own folders, not the program folders.
- c. It is much easier to set up a practical backup procedure.

Use Windows Explorer (Stifinder)

I recommend to use Explorer rather than My Computer, and to put a shortcut at your desktop:

- *Right*-click the [Start] button and select Open
- Open the Programs folder
- Select the Explorer shortcut icon and copy it to the clipboard by [Ctrl]+[C]
- Click anywhere on the desktop and paste the icon by [Ctrl]+[V]

Make your main folder default when opening Explorer

- *Right*-click the Explorer shortcut icon
- Properties ▶ Shortcut ▶ Path ▶
(Egenskaber ▶ Genvej ▶ Sti ▶)
`C:\WINDOWS\EXPLORER.EXE /n, /e, C:\dokumenter`

Make Explorer display file name extensions

For reasons not understood by me, Microsoft decided not to display file name extensions by default. This is inconvenient (you can not distinguish the syntax file **alpha.sps** from the data file **alpha.sav**), and you should set Explorer to display file name extensions.

- Open Explorer
- View ▶ Options
(Vis ▶ Indstillinger)
- Uncheck: "Hide MS-DOS file extensions"
("Undlad at vise MS-DOS filtyper")

How to create a new folder

The example is to create the folder PROJECT3 under C:\DOKUMENTER

- Double-click the Explorer (Stifinder) icon at the desktop
- Click C:\DOKUMENTER (root folder for own files)
- Files ▶ New ▶ Folder
(Filer ▶ Ny ▶ Mappe)
- Rename 'New Folder' (Ny Mappe) to '**project3**'

How to rename a folder or file

- In Explorer, *right*-click the folder or file and select Rename
- Write the name desired and press [Enter]

How to copy a file or a folder to another folder or to a diskette

- In Explorer, highlight the source file or folder; press [Ctrl]+[C] (copy to clipboard)
- Move to the target folder (or A:); press [Ctrl]+[V] (paste from clipboard)

How to move a file or a folder to another folder

- In Explorer, highlight the source file or folder icon; press [Ctrl]+[X] (copy to clipboard and delete source file)
- Move to the target folder; press [Ctrl]+[V] (paste from clipboard)