

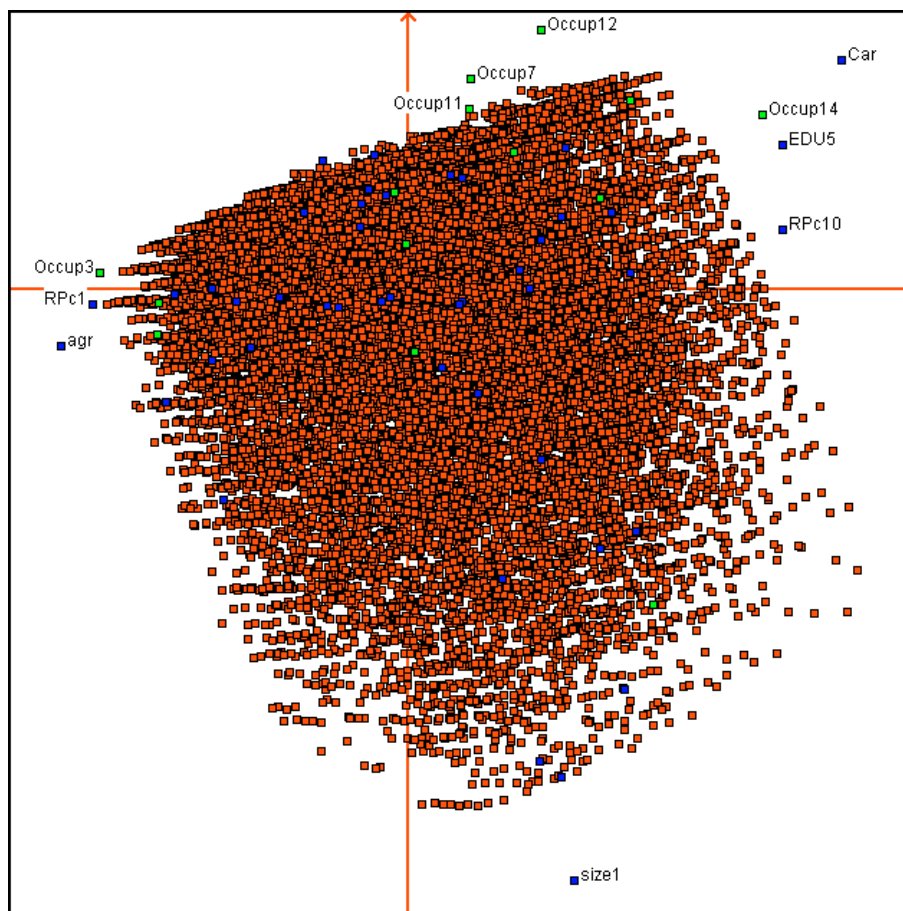
Silvio Griguolo



Version 5.2a – July 2003

A Package for Exploratory Data Analysis
(specially oriented at territorial data)

User Manual



University Institute of Architecture in Venice – Dept of Planning

S. GRIGUOLO

ADDATI

A package for Exploratory Data Analysis

User Manual

(version 5.2a – July 2003)

University Institute of Architecture in Venice – Dept of Planning

ADDATI, version 5.2a (July 2003) - Presentation

This is still a provisional mixed Windows/DOS version, but another step forward has been made towards a full Win32 package: the program FACPLAN, that displays the projections of statistical units and variables onto factorial planes, has been re-written as a Win32 application.

Before it was a DOS graphic program, as such the ADDATI utility most liable to create problems when running in a Win32 environment, with only-God-knows what type of video card.

The other programmes are mostly DOS 32-bit programs, computational workhorses with text output, compatible with all flavours of Win32, from 95 to XP. They are fast, they can fully access the available memory, so they should create no problem.

To install the package, unzip the file <ADDA52aINST.ZIP> to a temp folder, then run SETUP.

SETUP is a Win32 program that will install ADDATI in English or Italian, depending on your choice. The package is so structured as to support easily other languages (support for Spanish might be added in short).

Once installed, the configuration and the language can be changed at any time from within the general management routine ADDATI.EXE, which is also a Win32 program.

This version uses a different DOS Extender (WDOSX by Michael Tippach), declared to be compatible with the Win2000 Service Packs and with XP.

ADDATI should thus be perfectly compatible with 95/98/NT and 2000. The only residual problem with XP may arise from the visualisation of the graph of the objective function in the non-hierarchical classification programme NONGER.

Next step, very soon, will be to convert also it to Win32.

Some other minor features have been added, for example an option in the FILE menu that facilitates the conversion of tab-separated text files saved in EXCEL to the space separated form of data file required by ADDATI. The conversion of tabs to spaces is carried out in a controlled environment, that enables the user to eliminate redundant items and eventually draws to columns the output data file.

Documentation

While the help embedded in the DOS programs is up-to-date (it can generally be evoked by hitting the '?' key), the general description displayed by choosing Help/Doc from the menu is obsolete, and does not mirror the package's current features. At the moment, it is still there just because it offers an adequate description of the clustering technique implemented.

A PDF manual is available both in Italian and in English. However, the user should note that *the English version is still under development*. Based on partially pre-existing text, it has been prepared for “The Regional FIVIMS Training Course on Multivariate Analysis and ADDATI in Support of Capacity Building of National FIVIMS Analysts in Asia,” held by the Asia FIVIMS Trust Fund Project GCP/RAS/170/JPN on 29 April – 2 May 2003 in Bangkok, Thailand with support from the FAO Regional Office for Asia and the Pacific.

The English Manual will be revised and completed as soon as possible.

Attributions

- S. Griguolo conceived and designed the package and wrote most of the programmes.
- **RECODE** was conceived jointly by M.Mazzanti and S.Griguolo and mostly coded by M.Mazzanti
- A. Ciavarella is the author of the Win32 version of **FACPLAN** (projection onto factorial planes), with some contributions from S.Griguolo. M.Mazzanti (specially) and S.Griguolo were the authors of the previous DOS version of the program.
- This English User Manual is partly based on the one written in 1993 by Silvio Griguolo for the project FAO/GCPS/RAF/256/ITA (IGADD Early Warning and Food Information System for Food Security, Chief Technical Advisor Paolo Santacroce), to which also Annalisa Conte contributed. It is also partly based on the Italian Manual, written with M. Mazzanti’s contribution.

Venice, Italy, July 2003

Silvio Griguolo

The most recent version of ADDATI, as well as other software of the same author, mostly aimed at Remote Sensing, can be downloaded from the page <http://cidoc.iuav.it/~silvio/software.html>

The author can be contacted via e-mail at the address silvio@cidoc.iuav.it

CONTENTS

ADDATI, VERSION 5.2A (JULY 2003) - PRESENTATION.....	1
Documentation.....	1
Attributions	2
CHAPTER 1. – INSTALLATION AND GENERALITIES	1
1.1 – INSTALLING ADDATI	1
1.2 – THE MANAGEMENT PROGRAMME ADDATI.EXE	3
The FILE Menu.....	4
CHAPTER 2. – THE UTILITY MENU	1
The data file format.....	1
The Utility programmes	2
2-1 MERGCHAR	3
2-2 MERGFIELD	5
Possible execution errors.....	7
2-3 FIXFORM	10
2-4 MISSVAL	11
How MISSVAL works.....	11
Raw imputation.....	11
An example.....	14
The structure of the file MISSVAL.PAR	14
A sample MISSVAL.PAR file	15
2-5 SELECT	18
The ‘ / ’ operator.....	19
2-6 SHOWREC	21
2-7 RECODE	23
The recoding operations	25
Control keys used in RECODE	26
2-9 FACPLAN.....	27
2-9 INTEGRATE	29
CHAPTER 3 – THE USER INTERFACE	1
3-1 Some relevant features in ADDATI	1
3-2 The user interface (some general notes).....	2
Modifying the answers already entered	2
Loading a file of parameters.....	3
The case of multiple labels or indicators	3
Online help.....	3
The data entry format.....	4
CHAPTER 4 – FUNDAMENTALS	1
4-1 THE SCALES OF MEASURE.....	1
Quantitative (or continuous) variables	2
Categorical (or qualitative) variables	2
Ordinal categorical variables.....	2
Nominal categorical variables.....	3
4-2 STANDARDISATION OF CONTINUOUS VARIABLES	4
Example 1 - Indicators of central tendency and dispersion.....	6
Example 2 – Quantitative variables: normalisation	7
Example 3 – Measure of the level of association between quantitative variables	9
4-3 - TYPES OF TABLES.....	12
Descriptive tables	12
Contingency (or count) tables.....	13
4-4 THE GEOMETRICAL REPRESENTATION	13
The distance	14

<i>The centre of gravity of the cloud</i>	15
<i>The Inertia of the cloud</i>	15
<i>The interpretation of the relationships among the variables in R^n</i>	16
CHAPTER 5 – DISTRIBUTIONS AND CROSS-TABULATIONS	1
5-1 DISTRIB	2
<i>The structure of the file DISTRIB.PAR</i>	3
5-2 CROSSTAB	6
<i>An example, just to introduce some theoretical considerations</i>	8
<i>The table expected under the independence hypothesis</i>	10
<i>An indicator of the overall difference between the observed and expected tables</i>	10
<i>The file CROSS.PAR</i>	13
CHAPTER 6 – THE ANALYSIS MENU: FACTORIAL ANALYSES	1
6-1 TYPOLOG	1
<i>The input of parameters</i>	3
6-2 THE FACTORIAL ANALYSES: ACOMP AND ACORR	8
6-2.1 THE PRINCIPAL COMPONENTS ANALYSIS (ACOMP)	9
<i>An example</i>	11
<i>Entering the analysis control parameters</i>	11
<i>B. Parameters controlling the output</i>	17
<i>The table of contributions and their interpretation</i>	19
<i>Interpretation of the factors</i>	23
6-2.2 THE ANALYSIS OF CORRESPONDENCES (ACORR).....	25
<i>Entering the control parameters</i>	28
<i>The table of contributions and their interpretation</i>	29
CHAPTER 7 – THE ANALYSIS MENU: CLUSTERING	1
7-1 SOME NOTES ON NUMERIC CLASSIFICATION	1
<i>Hierarchical methods</i>	2
<i>Non-hierarchical methods</i>	3
<i>Some definitions</i>	3
<i>Diday's Clustering Method (dynamical clouds)</i>	4
7-2 THE CLUSTERING METHODS IN ADDATI	6
7-2.1 – <i>Some general notes</i>	6
<i>The non-hierarchical clustering method</i>	6
7-2.2 – <i>Non-hierarchical Clustering: the Method 1 (no longer implemented in ADDATI)</i>	6
<i>Method 1 – The exploratory stage</i>	6
<i>The Optimisation stage</i>	8
7-2.3 – <i>Non-hierarchical Clustering: the currently used Method 2</i>	8
7-3 THE PROGRAMME NONGER.....	11
<i>Exploratory phase</i>	11
<i>A. Questions enabling the program to read the data file</i>	11
<i>Block A1 (for a classification directly carried out on the descriptive variables)</i>	12
<i>Block A2 (Only for a clustering linked to a factorial analysis)</i>	17
<i>Block B. – Parameters for clustering</i>	17
<i>NONGER - Optimisation stage and description of the partitions</i>	19
<i>NONGER – The interpretation of the results</i>	23

Chapter 1. – Installation and Generalities

1.1 – Installing ADDATI

Unzip the installation file ADDA52aINST.ZIP in an empty folder. Five files are produced, three of which are necessary to install the package:

- **SETUP.EXE**, to be used to carry out the installation process;
- **ADDATI.ZIP**, the zipped file that contains the ADDATI modules. Do not unzip it yourself, it would not result in a working package. Let SETUP do the job.
- **UNZIP32.EXE**, a public domain unzipping utility

During the installation a version of the User Manual in the selected language (English or Italian) is copied to the subdirectory <DOC>, in the directory where ADDATI is installed. The Italian version installs also some examples unzipped from the file <EXMP_ITA.ZIP>.

When you run SETUP, the dialog shown in figure 1-1 appears. Its form is mostly common to that of the other dialogs found in ADDATI: the small window on the bottom of the dialog displays a contextual help when the mouse is passed over a control (a button, combo, edit box, etc.). Choose the language and click on OK.

***Note:** The installation language can be changed any moment when the package is run; this causes all the strings, help, etc. that are used in the package to immediately switch to the new language.*



Figure 1-1 Installation - The dialog for the choice of the language

Then the dialog of Figure 1-2 appears. The first edit box shows the name of the directory from which SETUP was launched, and is only for your information.

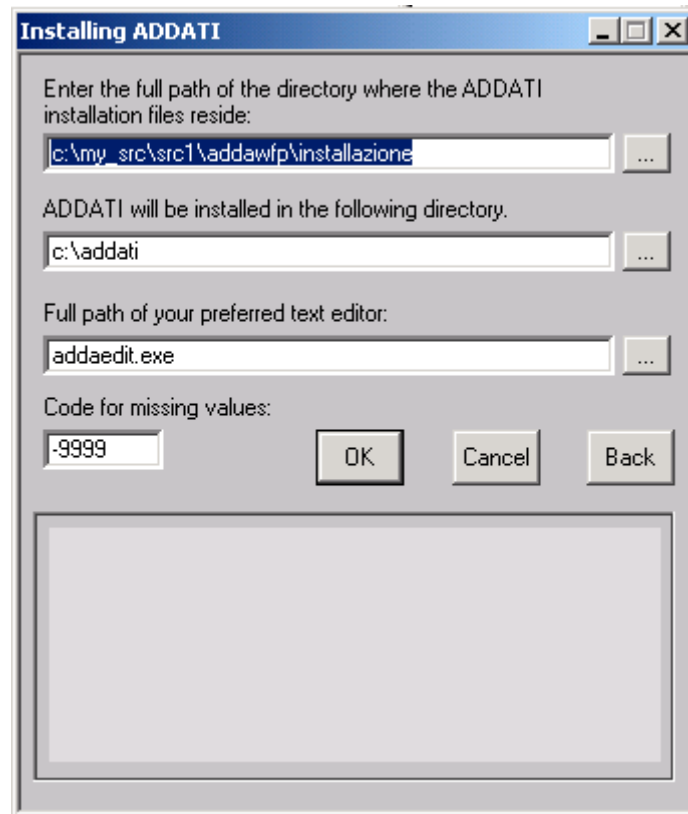


Figure 1-2 Installation - The settings dialog

The second one proposes you to install ADDATI in C:\ADDATI. Change the target folder as you like, but remember to use folder names following the 8.3 DOS format, otherwise some DOS programmes in ADDATI might encounter problems. So, please, no long directory names, nor including spaces.

The third box requests you to enter the path of an editor that ADDATI will run when it needs to edit some parameter files, or some text files. ADDATI proposes you “**addaedit.exe**”, its internal editor created exactly for that purpose, and with which it can communicate at best. You are strongly invited to accept it.

The fourth box sets the code used in ADDATI to indicate a missing value. You can accept it, and modify it later on when necessary.

It is important that ADDATI can recognise a missing value from a valid one. The code will be used in the utility routines like MERGFIELD or RECODE, when recoding variables or creating new ones, or computing distributions or cross-tabulations.

Remember The multivariate programmes, like ACOMP (Principal Components Analysis) or NONGER (Clustering), require that all the variables describing a statistical unit have valid values. They would therefore erroneously process a missing value as a valid one, supposing that missing values had already been previously detected and corrected by other utilities. The checking will be made more stringent in the Win32 version being prepared.

The values are stored in the initialisation file ADDATI.INI in the ADDATI folder. They can be modified any time either editing ADDATI.INI, that is a text file, or selecting the “**configure**” option from the FILE Menu in ADDATI.

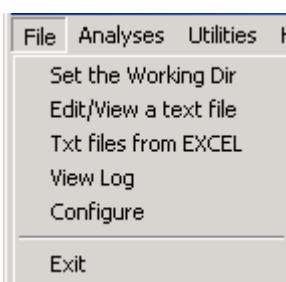
When the installation terminates you can remove the files you used and create on your desktop a shortcut to ADDATI.EXE, the management routine to be launched.

***Note:** The ADDATI icon is a little strange and has a curious history. Joking, I asked my daughter, who at that time was 10 years old: “Why don't you make me an icon for ADDATI?” “What does ADDATI do?.” she wanted to know. Well, what does ADDATI actually do?*

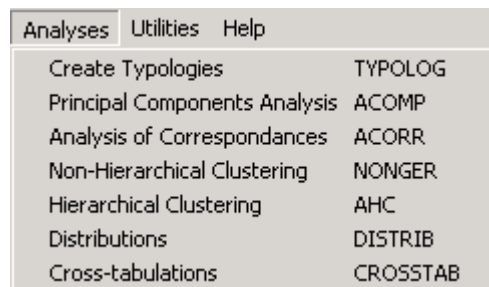
*A little uncertain, I tried to explain: “Well, it helps discovering things, answering some questions...”. In the evening she presented me an icon, done with the icon editor of the Watcom C Compiler. Maybe it is not particularly nice, but all those question marks capture quite well the exploratory purpose of the package...
Good work!*

1.2 – The Management programme ADDATI.EXE

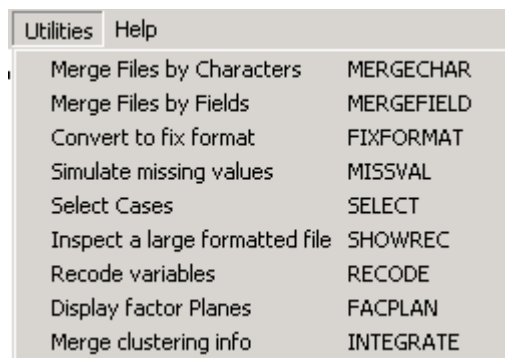
The ADDATI Menu has the three sub-menus (Help apart) shown in the Figure 1-3.



a) the FILE menu



b) the Analysis menu



c) the Utility menu

Figure 1-3 The three Menus in the management programme ADDATI.EXE

The FILE Menu will be briefly described here. Chapter 2 is devoted to The Utility Menu, and the following chapters will deal with the Analysis Menu.

1. Set the Working Dir

This is the first thing you have to do immediately after launching ADDATI. At start, the current directory is by default the one where ADDATI is installed, so better to move to the folder that contains the data you wish to process.

When you choose this item a dialog opens that allows you to change directory. The new working folder is confirmed in the client area of the main window. If now you run one of ADDATI DOS applications, the working folder is automatically inherited by the DOS routine.

If you run a DOS application, and a file you wish to open is not found, probably you have forgotten this step and the working directory is not the one you think. From within the DOS application you can hit F10 to open a new shell and assess which is the current working directory.

2. Edit/View a text file

This menu item displays a dialog aimed at choosing the text file to be opened (fig. 1-4).

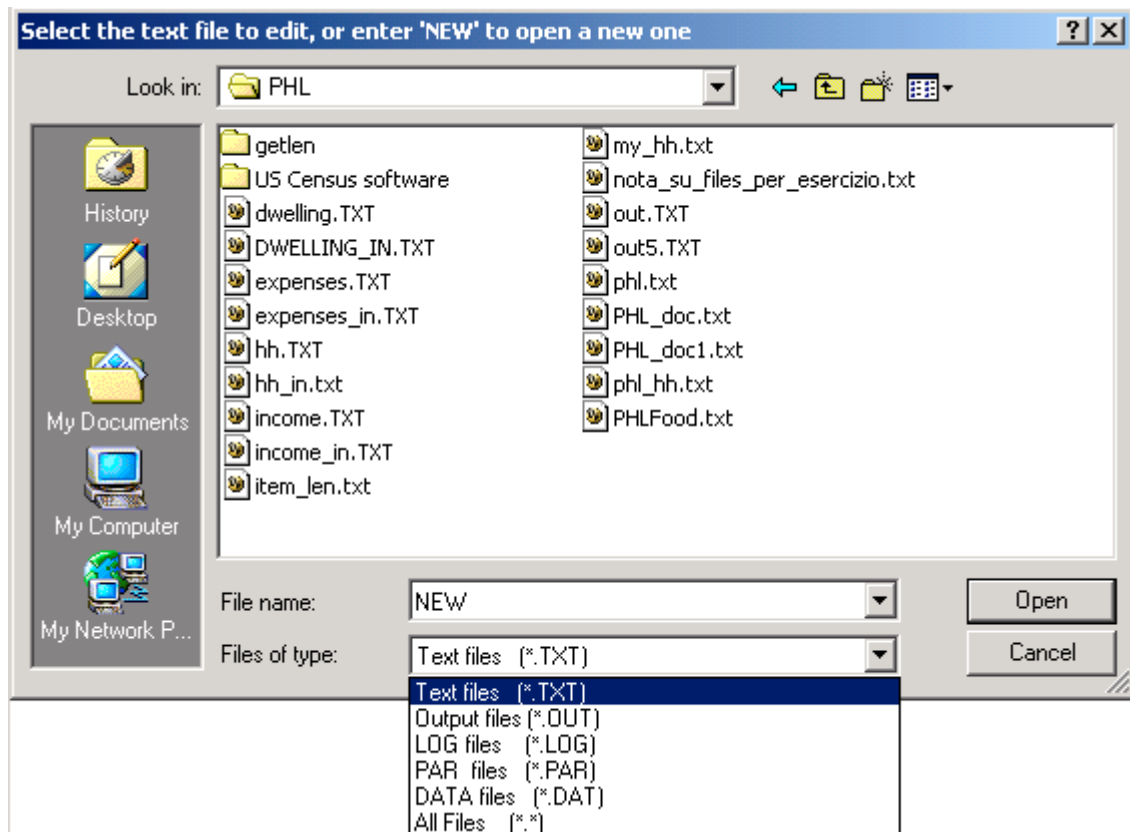


Figure 1-4 Selecting a text file to edit

The text files normally used by ADDATI have the extensions shown in the figure, under “Files of type:”: **TEXT** (generic Text files), **OUT** (output files produced by the Analysis programmes); **LOG** (ADDATI LOG files), **PAR** (parameter files, suitably filled by the user to control the execution of several ADDATI programmes); **DAT** (common extension for ADDATI data files). Select the extension to visualise all the files of the type you wish to open, then select the one you wish.

Enter “**NEW**” to open an empty file.

The ADDATI internal editor **ADDAEDIT** is launched to edit the file you selected (unless you indicated a different file on installation). **ADDAEDIT** has some characteristics that make its use advantageous: there is no limit in the file size, and the editor displays on the status bar the number of the line and of the column (character) where the cursor is located.

Besides, if the file being edited is a *data file* in which the values are separated by spaces, also the ordinal number of the field (or variable, see more on ‘fields’ further on) is displayed, where the cursor is located. Data file are automatically recognised as such if they have extension “DAT”. If this is not the case, the displaying of the field number can be forced by selecting “Show Field #” from the editor’s FILE menu.

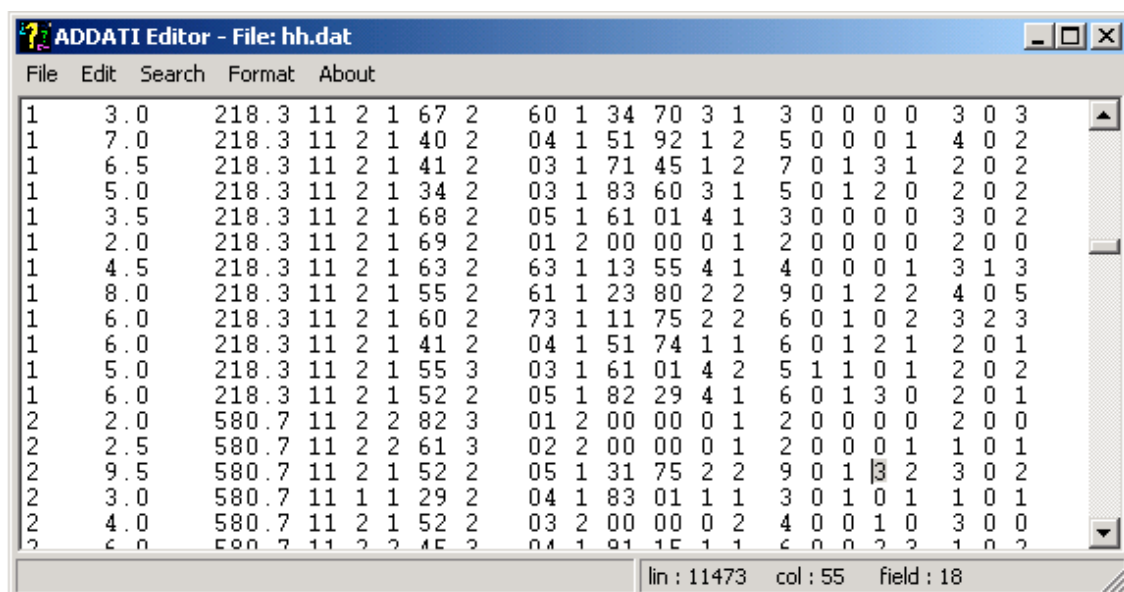


Figure 1-5 The ADDATI internal editor **ADDAEDIT**.

Figure 1-5 shows the editor window: the cursor, shown in grey, is located on record 11473 column 55, that corresponds to the variable 18. If another instance of the editor is opened to display the documentation (or *the dictionary*) accompanying the data file, the contents of the file can be inspected knowing what is what.

As many instances of the editor as necessary can be run at the same time. They are independent applications, so ADDATI can be simultaneously used with no restriction.

ADDAEDIT has also some other advantages, due to its capability of obtaining some limited information from ADDATI, that will be mentioned later on.

3. Text files from EXCEL

This option helps the user to import into ADDATI data from EXCEL.

ADDATI processes text files that must consist exclusively of data, so EXCEL files must be converted to text and all info other than data must be removed.

From EXCEL, the sheet containing the data must be saved as a **tab-separated text file**. This is one of the option found under “Save as” (see fig. 1-6). There is also the possibility to export using space as separator (the preceding option in fig. 1-6), which is indeed what ADDATI needs, but the problem with EXCEL is that files written choosing this option limit the record length to 240 characters (if I remember well), which is not sufficient in most cases. The way of exporting we suggest, using tabs as separators, does not have this inconvenience.

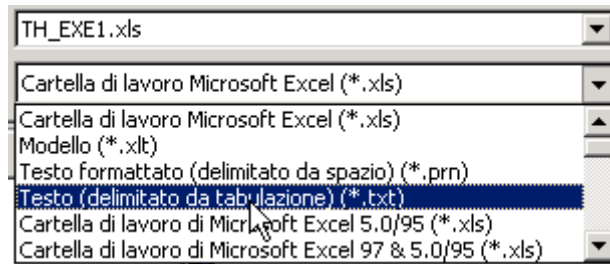


Figure 1-6 The option to be chosen to save EXCEL data as tab-separated text files

After exporting your data from EXCEL into a text file with extension .TXT (though this is not strictly necessary), click the “Text files from EXCEL” option in the ADDATI FILE menu to open the dialog of fig. 1-7. Enter in the upper edit box the name of the file saved by EXCEL, or browse to point to it. By default, ADDATI proposes as output the same filename, with the extension changed to .DAT, in which tabs are converted to spaces.

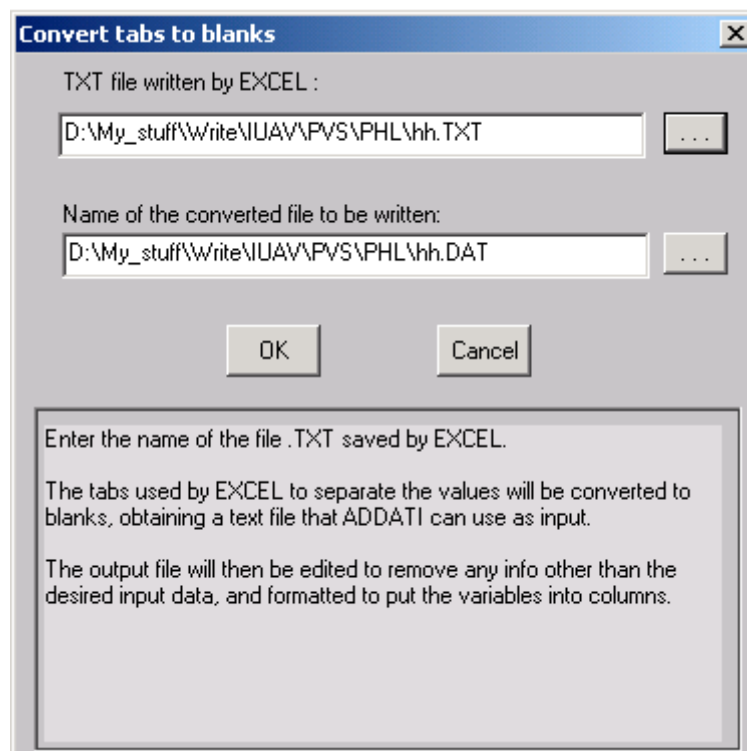
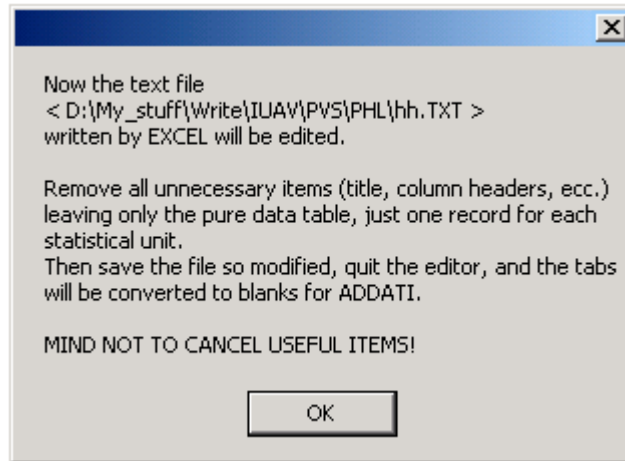


Figure 1-7 Importing data from EXCEL, the starting dialog

When you click on OK, the following message appears:



Click on OK: ADDATI loads the file written by EXCEL into the editor, to enable you to remove all documentation, labels, column heads, etc., **leaving only the data**. Save the file and close the editor: ADDATI will convert all tabs to spaces, inserting the code for missing values when necessary (i.e., when it encounters two consecutive tabs with no valid value in-between). It will then draw the variables to columns, to ease inspection. All errors encountered during this step (missing variables, records including more values than expected, etc.) are signalled.

Be careful, when you delete all that is superfluous from the file, not to remove also any essential part.

In particular, take the cursor to the beginning of a new line after the last record, to make sure that it terminated with an end-of-record, but make sure not to insert any space in the new line, otherwise it would be interpreted by ADDATI as an existing record with no field, resulting in an error.

4. View LOG

The editor is called to display the LOG where ADDATI saves copy of the messages written in the client space.

5. Configure

Used to change some configuration setting. The following dialog is displayed: the user can change the language, the path of the preferred editor, and the code that indicates missing values. Changes are immediately operational.

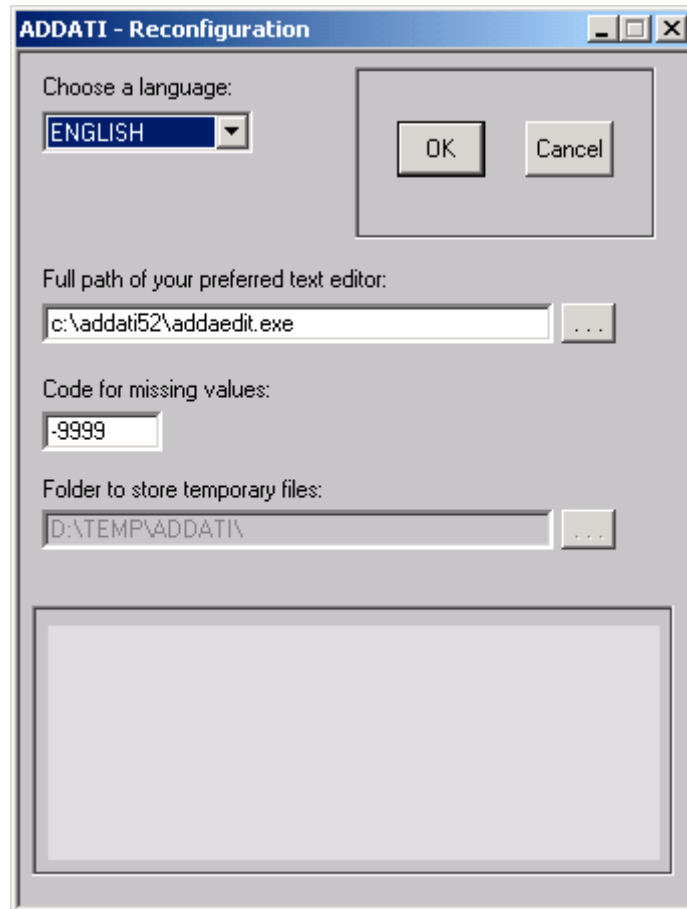


Figure 1-8 The re-configuration dialog

Chapter 2. – The Utility Menu

This menu includes some utilities of general interest, which can help the user to prepare the tables to be analysed, to display and interpret results, and more. All utilities (for merging, formatting, inspecting files and recoding variables) operate on **ASCII text files**. Therefore, if data are stored in binary form (e.g. by programs like EXCEL or DBASE) they must be "exported" as text files before being processed by ADDATI.

An facilitated sequence is offered (from **File/Text Files from EXCEL**) to import EXCEL files in text format, using blanks as separators.

***Note:** **MERGCHAR**, **MERGFIELD**, **FIXFORM** and **RECODE** are used in ADDATI to manipulate data files in order to prepare a table for a multivariate analysis, but their utility is not limited to this.*

The data file format

A **text** data file can be recorded in different **formats**. With this we refer to the way the values of the variables (or indicators) are written in each record. As a program must know the format in order to load the data correctly, the language convention we will follow in ADDATI must be clear.

A record consists of the sequence of the values taken by the variables for a statistical unit. The order of the variables must obviously be the same in all records. The data file should always be accompanied by a **documentation file**, usually a text file that specifies the place of each variable in the record, its meaning, its categories (if any), etc.

The variables can be stored in a record in one of the following modes:

1. **With no separation spaces:** a record appears somewhat like "01212341234...." and only the offset from the record beginning enables the user to recognise the meaning of the digits and to which variable they refer. Therefore, all records must have the same length (fixed length) and a given variable will occupy the same columns in all records. This compact format is used quite often when the number of the statistical units, and that of the variables, are very large. The purpose is to limit the file size. This is typically the case with Census data at the maximum detail level (households) or with large surveys.
2. **Separating them with spaces** (or commas, or semicolon) to make them immediately visible. We will call **field** a group of alphanumeric characters separated from other similar groups by one or more spaces.

A record structured in fields can have or not a fixed length. If it has a fixed length, and the fields are separated by spaces, each field (variable) occupies a given number of columns (or characters), the same in all records. This the variable's **field length**, generally different for the various fields. When this is the case, the variables appear like ordered columns, and the inspection of the file is much eased.

The table 2-1 shows the possible combinations. We will conventionally say that **the format is free** when the records are organised by fields (and their length is not necessarily constant). We will say that the file is **formatted** (or also in **fix format**) when its record have a *constant length* and each variables takes the same columns in each record.

The two definitions are not mutually exclusive: the *free format* refers to the structure per field, the *fix format* to the constant record length. A file in free format can *also* be formatted, and all its variables are then ordered in columns, separated by spaces; vice versa, a file *not organised by fields* , i.e. not in free format, must necessarily consist of records with fix length, otherwise it would be impossible to read it.

		SPACE AS SEPARATOR	
		Yes	No
RECORD	Fixed	Free format, formatted	Non-free format, formatted
LENGTH	Variable	Free format, non formatted	Not applicable

Table 2-1 Scheme of the conventional terminology about the format.

The Utility programmes

For sake of clarity and to deal with data more easily, different descriptions of the same geographical units (districts, meteorological stations, markets...) are often organised in several files, each describing a single homogeneous aspect. The units are stored in a fixed order: one file can contain socio-economic data, another the maize price series in a given year, and so on. Of course, all files must contain the same number of records (as many records as there are geographic units, always taken in the same order).

Therefore, it is sometimes necessary to create the table to be analysed by combining variables drawn from different files. That's what **MERGCHAR** e **MERGFIELD** are for. They operate respectively *by characters* (i.e. *on formatted files*) and *by fields* (i.e., *on free format files*).

MERGCHAR can insert some blanks between the selected characters, thus writing an output file in free format; **MERGFIELD** is also used to construct and save new variables.

FIXFORM operates on *unformatted files* (i.e., files in free format, but not formatted) and suitably inserts some blanks where necessary, drawing the variables into columns. The output file is still in free format (i.e., fields are blank-separated), but all variables are ordered in columns and all records have a fixed length

MISSVAL simulates plausible to replace missing values of quantitative variables.

SELECT allows the user to select and save to a new file only the units that have some particular features, specified through a logical condition on the values of some variables.

SHOWREC allows the user to inspect a *text formatted file* (typically, a data file) too large to load it with an editor. There is no limit to the file size.

RECODE is used to recode variables (e.g., convert them to another scale, or merge their categories, or similar operations). The file must have a fix format. **RECODE** offers four recoding operations, on both quantitative and qualitative variables.

FACPLAN, is a graphical program that, used after a Factor Analysis or a non-hierarchical Clustering following a Factor Analysis, displays the projections of variables and units onto the main factorial planes. In this version of ADDATI **FACPLAN** has been converted into a Win32 application.

INTEGRATE is used as the final step in a clustering sequence started with the determination of typologies (program **TYPOLOG**). For each elementary unit, **INTEGRATE** writes to file the class to which it was assigned by the clustering procedure.

Here below, the Utility programs are described in the same order they have in the Menu.

2-1 MERGCHAR

Merging of files by CHARACTERS

Use	it can read, record by record, some desired columns of characters from up to nine input files , and save them to the output file in any order. Blanks can be inserted where it is necessary, or repeat the same column more than once.
Requir.	The input files must be formatted text files: they must all have the same number of records, each describing a statistical unit (administrative area, households, dwelling, etc.). The units must stay in the same order in all files.
Limits	The maximum record length is 5120 bytes .

MERGCHAR operated on **ASCII formatted files**: think of a formatted file as a rectangular table, whose rows are the records. A 'column' is formed by all the characters (like 'a', 'o', or digits, or a blank) that stay in a given position in all records. If a file is formatted, all its records consist of the same number of characters.

The various files to be merged are supposed to offer different descriptions of the same statistical units, descriptions that are archive separately for reasons of clarity, or to handle them more easily.

The user is prompted to enter the filenames from which the columns to be processed must be extracted. The name must include the path if the file does not reside in the working directory. The order of the input files is the order in which they will be read: it is important as it determines the alphabet letter that will be used to indicate the columns of each file.

***Note:** If the output file already exists, the user will be requested if he/she wishes to overwrite it, or provide a different name.*

The first record of every input file is read and a screen is displayed where the input files are listed, specifying their record length and the associated letter.

The user must enter in a scrollable window a string that identifies the columns to be copied from each file. Such string consists of alphanumeric groups separated by spaces. Each group, **that must have no internal blank**, commands the writing to the output file of one or more consecutive characters, or the insertion of a blank. An acceptable group has one of the following forms:

- **a character identifier** (like A3 or C17), consisting of one letter followed by an integer; the letter denotes the file from which the character is read (A for the first file, B for the second, and so on), while the integer specifies the position of the character in the record. Such group commands the writing of the corresponding character to output;
- **a sequence of characters** (drawn from the same file) expressed in compact form by means of the "/" operator. E.g., " c3/7 a1/20 " copies to output, in the order, the characters 3 through 7 from the third file and the first 20 characters of the first file.
- **the letter 'b'**, commanding the writing of a blank to output.

The program assumes that all the records in a file have the same number of characters (all records contain the same type of information for the different units, formatted in a similar way). Should it not be so, an error condition is raised when running and the user is warned about it.

An answer like **"b1/9 b a14 a15 a9 b16 b17"**

means that we want to copy to output, in the order, the following columns drawn from the two files: columns 1 through 9 from **file2**, then the 14th, the 15th and the 9th column of **file1**, then the 16th and 17th column of **file2**. Also, we want a space written to output in all records after the first nine characters.

***Note:** All input files must have the same number of records (i.e., they must contain information about the same units, in the same order), otherwise an error condition is remarked.*

2-2 MERGFIELD

Merging of files by FIELDS

Use	Like MERGCHAR , it allows the use to merge information drawn from different files, but it is field- and not character-oriented. It is also used to construct new variables, defining their values from those of variables included in the input files.
Files	Input files must be in free format .
Requir.	All the files to be merged must have the same number of records, describing the same statistical units. All the records in a file must consist of the same number of fields.
Limits	up to nine input files; max . 5120 characters or 600 fields per record (both in input and output files).
Warning	MERGFIELD recognises the missing value code (set in the configuration file ADDATI.INI). Impossible operations, or operations that involve missing values, result in a missing value code written to output. If the program behaves strangely , check if the current code for missing values is the expected one. if necessary, change it by reconfiguring the package from the File menu.

This is another utility for merging files, but it is oriented to **fields** and not to characters like MERGCHAR.

A **field** is a **group** of alphanumeric characters, delimited by blanks.

Example: *The record " 12 ax2 32.45 " consists of three fields.*

The files do not need to be formatted as in the case of MERGCHAR, where all records in a file must have the same length; instead, all records in a file **must include the same number of fields**. It is possible to write the same field more than once.

The files to be merged are supposed to contain different descriptions of the same statistical units; these descriptions are kept separate for sake of clearness and manageability. All input files must have the same number of records. Each record describes a unit (geographical areas, markets, households, dwellings, etc.). The statistical units must be in the same order in all files.

Statistical unit by statistical unit, MERGFLD allows the user:

- to copy fields at will from a maximum of nine input files, writing them in any order to the output file;
- to compute the values of any number of new variables, defined by the user on the basis of the variables contained in the input files, and to write them to output;
- to define up to five temporary variables whose values are used in computations and can optionally be written to output.

As in MERGCHAR, the user is prompted to enter the names of the files from which the fields to be combined originate, separating them with blanks. A path is accepted for files

not contained in the working directory. The order in which the filenames are entered is relevant, as it determines the associated labelling letter.

The first record of every input file is read and a screen is displayed, listing the input files, the number of fields in each of them and the associated letter.

The user enters into a scrollable window a string which specifies the fields to be copied and the new variables to be created.

The string consists of groups separated by blanks. Each group, **which must not include any blank**, commands the writing to output of one or several terms, or defines a temporary variable. A group is made up by combining the following tokens:

- **numbers** (1, -3.2, 300...), loaded as integers or floating-point values;
- **field identifiers** (like **A3** or **C17**), consisting of one letter followed by an integer; the letter denotes the file from which the field is loaded (**A** for the first file, **B** for the second, and so on), while the integer specifies the position of the field in the record;
- **the symbols v1 ... v5** which represent - when necessary - some temporary variables;
- **the operators** **+** **-** ***** **:** **^** ; they must stay between the numbers, the field identifiers or the temporary variables on which they act. Notice that **:** instead of **/** - which has a different meaning - is used to denote a division;
- **the sign '='**, which must be used **only to define a temporary variable**.
For example, $v2=(2*a1+3*a2):5$ defines a temporary variable $v2$ - to be later used for other computations - as the weighted average of the values of the first two fields in the first file ($a1$ and $a2$), with weights 2 and 3 respectively (5 is the total weight). Notice the use of brackets to specify precedence in computation. Previously-defined temporary variables can be used in the definition of other variables.
- **the function symbols "ln log exp sqrt"** , denoting in the order the **natural logarithm**, the **decimal logarithm**, the **exponential function** and the **square root**.

The groups entered by the user are interpreted in their order. A group can consist of:

- **one field identifier, or one number**. It commands the writing of that item to the output file. E.g., 'b7' commands the 7th field of the second file to be copied to output; '100' requests writing to output of the number 100.
- **a sequence of consecutive fields** expressed in compact form using ' / '. E.g., " c3/7 a1/20 " copies to output, in the order, the fields 3 through 7 of the third file and the first 20 fields of the first file.
- **the definition of a temporary variable**, whose value are saved for further computations on the same unit, but is generally not written to output.
- **a temporary variable's indicator**: e.g., 'v3' request the program to write to output the value of the temporary variable $v3$, previously computed.
- **a sequence of operations on numbers, field identifiers or temporary variables**. E.g., $\text{sqrt}((a1+a2):2)$ computes the square root of the arithmetic average of the first two fields in file 1 and writes the result to output.
- **a sequence of operations to be performed on a set of consecutive fields**; the results are written to output. E.g., suppose that an input file distributes the population of 200 municipalities in 10 age classes: the file will have 200 records (one for each

geographical unit), each containing 10 fields. The following string allows the user to convert the information from absolute values to percentages:

$$v1=a1+a2+a3+a4+a5+a6+a7+a8+a9+a10 \quad a1/10:v1*100$$

The first group defines the temporary variable **v1** - computed record by record, i.e. for each municipality - which stores the municipality's total population; the second writes to output the percentage values obtained by dividing **each** of the **a1...a10** fields by their total **v1** and multiplying the results by 100. The term "**a1/10**" is expanded to the ten fields it represents and the requested operations are performed on each field. Here the use of round brackets is not necessary, because ':' and '*' have the same priority and the evaluation takes place left to right, issuing the desired result. However, note that the group **(a1/10:v1)*100** is equivalent.

Remember: a sequence of fields like **a1/10** must always stay at the beginning of a group.

Round brackets can be used to increase priority. At the same level of parenthesis an expression is evaluated left to right according to the following precedence order: first the **ln, log, exp** and **sqrt functions** are computed, then the **raising to power** ^, then the * and : operations in their order and lastly the + and - operations.

***Note:** When the computation of new variables is requested, the program determines whether the concerned input fields are integer or real (i.e., whether they include any decimal digit). In general, the result is computed and saved as a floating-point value (i.e., with some decimal digits) whenever at least one of the terms used in the computation is real, or the ln log exp sqrt functions, a raising to power or a division are involved.*

When at least a floating-point result is to be computed, the user is prompted to enter the desired number of decimal digits. The answer must be a positive integer, or 0 to write only the integer part of the number.

All the floating-point values computed by the program are saved with the indicated number of decimal digits.

Possible execution errors

- For one of the input files the end-of-file is reached before the other files are finished: the program displays the name of the file that has less records than expected;
- Some records in a file have less fields than expected: 'null' is written to output for the missing field(s).

***Note:** The advice in this case is to submit the erroneous file to FIXFORM, that checks all records and saves the ordinal number of the defective ones. This problem almost never occurs for datafiles written by programs, but is quite common for files written by entering data manually from keyboard. FIXFORM offers a way to check them.*

- when an invalid operation is performed (a division by 0, the square root or the log of a negative number, etc.) an error message is written to a file named <ERRORS>. Inspect its contents after termination. In this case, the result written to output is not reliable and the input file should be corrected.

When an invalid operation occurs, the code for missing value, defined in the configuration file ADDATI.INI, is written to output. If possible, the execution should be

repeated after correcting the unacceptable input values. Remember that a table is being prepared, that must be submitted to a multivariate analysis: no missing value is admitted.

If the termination is regular, the output file is formatted before closing it: all its fields are drawn-up in columns (via an automatic call to FIXFORM; see below) so as to ease the inspection of their contents.

Example

<i>file1</i> (582 records)	<i>file2</i> (582 records)
MUN_01 32 25 14	MUN_01 25.12 36.02 15.00 0.05
MUN_02 13 114 13	MUN_02 124.35 181.23 27.65 0.24
.....

The two files to be merged contain different variables concerning 582 hypothetical municipalities. Each record consists of 4 fields in the first file, of 5 fields in the second. The first field is the municipality label.

The two files need not be formatted, but all the records in a file must have the same number of fields.

The command: "**a1 a3 a4 a3+a4 (a4:a3)*100 b2/4**" produces the following formatted output file

MUN_01	25	14	39	56.00	25.12	36.02	15.00
MUN_02	114	13	127	11.40	124.35	181.23	27.65

where the 4th field is computed by adding the 3rd and 4th field of the corresponding record of the first file; the 5th field is obtained by dividing the 4th by the 3rd field and multiplying by 100 (a request of two decimal digits is assumed).

Note: Notice the underscore '_' between "MUN" and "01": its purpose is to make the program accept "MUN_01" as one field: names cannot consist of two or more words separated by blanks, because they would be treated as separate fields, with undesired results.

Example: the string

"c1/5 a1^a3 a5 v1=b1+b2 v2=sqrt(v1) b1:v2 b2:v2 a1/10:v2"

uses fields read in from three input files.

Two temporary variables *v1* and *v2* are defined, respectively as the sum of the first two fields in the second file and as the square root of that sum. They are only used for intermediate computations and are not saved to output.

Record by record, the following fields are written to output:

- the first 5 fields of the 3rd file;
- the value obtained by raising to power the first field of the first file; the exponent is the value of the third field of the same file ($a1^{a3}$);
- the 5th field of the first file;
- the ratio between the first field in the second file and *v2*;
- the ratio between the second field in the second file and *v2*;
- the 10 ratios between the first ten fields of the first file and the intermediate variable *v2*.

Nineteen fields are thus saved to output. The last 12 are floating- point values and are recorded with the number of decimal digits chosen by the user.

2-3 FIXFORM

Converting a file to fixed format

Use	It converts a file from free to fix format
Files	It accepts as input a text file in free format.
Requir.	All records must have the same number of fields.
Limits	Max. 5120 characters and 600 fields per record.

Also this utility assumes fields (as defined above) as the basic units in a record. It converts free-format files to fixed format. When the format is free, the records have the same number of fields, but a variable length.

The conversion process suitably adds or removes blanks, giving all records the same format (the same length, with a given field occupying the same columns in all records). The input file is deleted and replaced by the new formatted file, whose contents can be viewed very easily (with the ADDATI internal editor, or using the **F3** key or the SHOWREC utility available in ADDATI itself) as all fields come in ordered columns.

The program prompts for the name of the file to be converted, and reads it twice. First, the overall greatest length is determined for each field; then, the file is rewritten using that length for that field in all records.

It is possible to give the same length to **all fields**, or to assign to each field the least fitting length (generally different from field to field), automatically determined, so as to produce a more compact output file.

2-4 MISSVAL

Simulation of missing values of quantitative variable

Use	MISSVAL is a programme that simulates missing values. Its use is limited to quantitative variables. The version is still experimental, and a good result is not always guaranteed for the reasons explained in Notes 1 and 2 below. Therefore, use it with caution.
Files	MISSVAL is controlled by the parameter file MISSVAL.PAR, just like DISTRIB and CROSSTAB (see Chapter 5). The parameter file is edited on user's request when the MISSVAL option is chosen from the menu.

How MISSVAL works

1. Missing values are skipped when computing the average and the variance of each variable; variables are skipped pairwise when computing the covariances, if the value of either is missing. This means that, instead of a fixed number of cases (all), there is a different number of valid cases for each variable, and for each pair of variables when covariances and correlations are computed.
2. Following the user's strategy defined in command file MISSVAL.PAR (see below), each missing value is replaced either by the variable's unconditional mean (**mean substitution**) or **by a plausible value**, computed from the correlations existing between the concerned variable and all other active variables, keeping into account the set of valid values associated with the one to be reconstructed in the concerned statistical unit (**raw imputation method**).

Raw imputation

For sake of simplicity, let us first consider only two variables x and y , with average \bar{x} and \bar{y} , standard deviations σ_x and σ_y and correlation $corr(x,y)$. If the value of x is unknown, but its correlation with y is not null, observing the value of y gives some information about the value of x . The distribution of x is altered: its **expected value** $ev(x)$ (i.e., the average of the conditional distribution) is generally shifted with respect to \bar{x} , and also the dispersion is changed. We can expect that:

- if $y = \bar{y}$ or the correlation is null, then $ev(x) = \bar{x}$;
- the conditioned dispersion of x depends upon the correlation: if $corr(x,y) = 0$, the standard deviation of the conditional distribution is σ_x as before; if $corr(x,y) = 1$ then the value of x is fully determined by that of y , and its dispersion is 0. Thus, the standard deviation of the conditioned distribution decreases somehow when the correlation increases. Anyway, we are not interested in it, only in the central value of the conditioned distribution.

As it is evident that the expected value of x for case i depends upon both the correlation between x and y and the value taken by y , it seems reasonable to assume that

$$z_i(x) = corr(x,y) * z_i(y) \quad (1)$$

where $z_i(x)$ and $z_i(y)$ are respectively the expected z-score of x and the observed z-score of y in i .

The (1) is consistent with the limiting conditions listed before: if y is one s.d. above average, and correlation is 1, then also x will be one s.d. above its average. If y is 0.5 s.d. below its average, and $\text{corr}(x,y) = 0.4$, we then assume that the distribution of x is so shifted that the most likely value is $0.5*0.4$ s.d. below \bar{x} , etc.

(I am not 100% certain that this is exact, the conjecture needs further investigation and the formula used in the software can be modified).

Anyway, the direction of the shifting is quite evident, and (1) gives a good approximation, certainly better than using \bar{x} (mean substitution).

Once we accept (1), we could dare to extend it to a case with p valid variables observed together with x as follows:

$$z_i(x) = \frac{1}{p} \sum_j \text{corr}(x, y_j) * z_i(y_j) \quad (2)$$

where $z_i(y_j)$ measures the deviation in i of the j -th known variable from its average, taking as unit of measure its s.d.; the index j runs from 1 through p , i.e. it includes all the simultaneously measured valid variables. (Actually, in MISSVAL only the best variables are used).

Of course, the result depends upon the set of explanatory variables chosen to predict the value of x , and it cannot be otherwise. Unfortunately, it is easy to see that one could alter the result by adding a lot of variables scarcely correlated with x : they would not contribute to the sum, due to their low correlations with x , but just because they are there, the division by p would mortify the predictive capability of a few variables highly correlated with x , that should instead be sufficient to reconstruct its value.

On the other side, just one single variable with correlation close to 1 should be sufficient to predict the value of x , and all others should be ignored. Therefore, the formula should give a higher importance to high correlations.

MISSVAL uses the absolute values of the correlations to weigh the most likely values individually generated by any variable taken into account, and expressed by (3). It goes as follows:

$$z_i(x) = \frac{\sum_j |\text{corr}(x, y_j)| (\text{corr}(x, y_j) * z_i(y_j))}{\sum_j |\text{corr}(x, y_j)|} \quad (3)$$

where for each statistical unit with a missing value to be reconstructed the index j runs over a user-defined number of variables: those that result the more highly correlated with the variable whose value is to be predicted. Expressing in (3) the z-scores in terms of the means and standard deviations, for case i we obtain

$$x_i = \bar{x} + \sigma_x \frac{\sum_j |\text{corr}(x, y_j)| (\text{corr}(x, y_j) * \frac{y_{i,j} - \bar{y}_j}{\sigma(y_j)})}{\sum_j |\text{corr}(x, y_j)|} \quad (4)$$

Suppose the analyst had decided to use p variables in the prediction. After computing means, standard deviations and correlations (skipping missing values), the data file is re-read and each time a missing value is encountered:

1. all variables with valid values for that unit are sorted in descending order according to the absolute value of their correlation with the variable to be simulated;
2. the first p are selected;
3. the formula (4) is used to compute a plausible value to replace the missing one.

Note *The procedure described is appropriate when missing values are Missing At Random (MAR). This is not always the case: it often occurs that the value of a particular variable has a higher probability to be missing for groups of similar units (households of similar social condition, geographically contiguous districts, etc.). In such case, the analyst should behave cautiously.*

*Anyway, remember that what is done is just a simulation, done to avoid to drop the whole case just because one variable is missing: **not the real value is obtained, only just a plausible one**, usually better than using the variable's mean, and far better than nothing.*

If the analysis concerns a set of geographic units, it can occur that the correlation of two variables, that is not that high over the universe, becomes significant when limited to a suitable subset (e.g., a set of contiguous units making up a homogeneous region). In this case, it can be worth to extract them (with SELECT), carry out the reconstruction separately, then re-assemble the data base. If a reliable pre-classification exists, this operation can be repeated separately for each class. In conclusion, the analyst should decide how to apply the simulation algorithm on the basis of his/her knowledge of the specific territorial context of the analysis.

Note *Beware of variables with too many missing values. They are unreliable and should be eliminated. If you retain them, and try to reconstruct the missing values, strange things might occur. For example, the table of the correlations saved by MISSVAL might display values higher than 1, or less than -1.*

*This depends on the fact that the mean and the standard deviation of the two variables concerned, as well as their correlation, **are not computed over the same statistical units**.*

*Mean and standard deviation concern a single variable: they are computed taken into account all the units for which the variable's value is valid. Instead, the correlation is computed taking into account the units for which **both variables** are valid, and this makes a lot of difference when at least one of the variables has a very high number of missing values, while the other has only few, or none. The computed statistics refers to very different sets of units, and the whole procedure becomes unreliable.*

*For example, in an exercise on China's administrative units 1732 values of a particular variable (out of 2114) were missing: its correlation with another variable, supposedly used for reconstruction, resulted to be as high as 1.240! Clearly, **values were not missing at random!***

***The conclusion** to be kept in mind is that the procedure works well when the missing-at-random hypothesis is fulfilled, and there are not too many missing values.*

Beware of extreme cases!.

An example

Below some records are shown, excerpted from a file that contained 148 records (the statistical units are the Census tracts of the historical Centre of Venice). The first field is the unit's id, the second is the weight (# of dwellings in the tract occupied by a household), then 11 variables follow, whose meaning is not relevant here.

In the first block of four lines the values of some variables have been replaced with -9999, the code that stands for a missing value. They are shown in bold red.

The second block shows the values generated by the reconstruction programme **MISSVAL**. As a maximum limit of three missing values per record had been set in **MISSVAL.PAR**, record #108, with four missing values, is excluded.

The third block shows the variables' observed values, while the last line gives the variables' means.

7	233	86.70	13.30	62.23	-9999	18.03	13.30	51.50	35.19	38.63	9.44	51.93
9	214	92.99	7.01	77.10	-9999	7.48	10.75	54.67	35.05	40.19	7.94	52.33
108	277	79.42	-9999	52.35	28.88	18.77	-9999	68.23	-9999	28.15	11.19	-9999
109	250	83.20	16.80	58.40	26.40	-9999	13.20	64.80	22.00	-9999	12.40	60.00
values simulated by MISSVAL												
7	233	86.70	13.30	62.23	22.198	18.03	13.30	51.50	35.19	38.63	9.44	51.93
9	214	92.99	7.01	77.10	18.477	7.48	10.75	54.67	35.05	40.19	7.94	52.33
109	250	83.20	16.80	58.40	26.40	15.442	13.20	64.80	22.00	26.480	12.40	60.00
actually observed values												
7	233	86.70	13.30	62.23	19.74	18.03	13.30	51.50	35.19	38.63	9.44	51.93
9	214	92.99	7.01	77.10	15.89	7.48	10.75	54.67	35.05	40.19	7.94	52.33
109	250	83.20	16.80	58.40	26.40	15.20	13.20	64.80	22.00	27.60	12.40	60.00
means												
		86.01	13.93	50.50	31.97	17.82	18.57	59.77	21.81	23.90	17.66	58.44

In all cases the simulated values appear to be a better choice than mean substitution.

The structure of the file MISSVAL.PAR

MISSVAL.PAR is self-illustrated. The file includes some comment lines that begin with a question mark "?", and others *active* lines that must be filled with the control parameters.

The comment lines explain how the others line must be filled to prepare the file for a particular analysis. **MISSVAL ignores the comment lines** and reads from the active ones the information necessary to execute.

At the beginning, it is quite easy to make some mistakes: the programme signals them with clear messages, to help correct them. So no hurry, and due attention when filling the parameter file.

A sample MISSVAL.PAR file

? This is the control file for **MISSVAL**.

? All records beginning with '?' are **comments**. They are ignored by the programme;
? their purpose is only to explain how to fill the records that do not begin with '?',
? that are actually loaded by the program.

? Each active line begins with a **keyword** (like **INPUT_DATA_FILENAME**, etc).

? **Do not modify the keywords**, only change the values (parameters) following them to
? fit them to the particular analysis to be carried out

? **1. Name of the input data file to be checked. Provide the full path if necessary.**

? **NOTICE!**

? The file is expected to be in ADDATI free format (blank-separated fields). If
? necessary, use the utilities provided by ADDATI (MERGCHAR, MERGFIELD and
? FIXFORMAT) to convert it.

INPUT_DATA_FILENAME venezia.dat

? **2. Name of the output data file to be written. Provide the full path if necessary.**

OUTPUT_DATA_FILENAME venezia1.dat

? **3. List the ordinal numbers (first is 1) of the continuous variables** to be actually
? checked and, in case, simulated. Use the space as separator, and as many
? '**ORD_#_OF_ITEMS_TO_CHECK**' lines as necessary.

? Be careful not to include here the variable to be used as weight, if any, identified by
? the value assigned to the '**WEIGHT**' keyword below.

? The variables indicated will be checked; missing values can be simulated.

? **WARNING!**

? All other fields not indicated for processing (categorical variables, alphanumeric
? labels, etc.) will be copied unchanged and will keep the same position in the
? output file.

ORD_#_OF_ITEMS_TO_CHECK 3 4 5 6 7 8 9 10 11 12 13

? **4. Weight assigned to statistical units**

? In the computation of variables' statistics a specific weight can be used for each case,
? or all cases can be assigned the same weight.

? If the statistical units are to be weighted, one of the variables included in the data
? file must be used to this purpose.

? **The weight must have a valid value for all units.**

? **MISSVAL** checks the weight values, but better to use **DISTRIB** before **MISSVAL**
? to make sure that a valid weight is provided for each unit, trap wrong values and
? correct them.

? Insert '0' after the keyword '**WEIGHT**' to indicate that the same weight is to be
? assigned to all units, otherwise enter the **1-based ordinal number** of the field to
? be assumed as weight.

? The value here below means that the second value in each record is the unit's weight.

WEIGHT 2

? **5. Variables' labels**

? Insert after the keyword '**LABELS**' a list of blank-separated alphanumeric labels to identify the variables in their order (do not include the weight, if any).

? Use as many '**LABELS**' statements as necessary. Max. 12 characters for each label.

? Here below some dummy labels are entered

LABELS var1 var2 var3 var4 var5 var6 var7 var8 var9 var10 var11

LABELS var12 var13

? **6. Maximum number of missing values per record**

? The higher the number of missing values encountered for a statistical unit, the less reliable the simulation.

? Provide the **highest tolerable number of missing values** for a unit: all cases with more than that will be excluded.

? (if this value is set to '0', all case with any number of missing values are excluded).

MAX_MISSING_VALUES 3

? **7. Setting a replacement strategy**

? If the preceding parameter is not set to 0, then you intend to replace missing values with other reasonable values. This can be done in several ways.

? The keyword **STRATEGY** can be followed by

- ? • '**MEAN**' to replace a missing value with the mean of that variable
- ? • '**CONDITIONAL**' to replace it with a value computed from the correlations between the concerned variable and the valid variables more correlated with it.
- ? If **MAX_MISSING_VALUES** is set to '0', the value of '**STRATEGY**' is ignored.

STRATEGY CONDITIONAL

? **8. Variables to be used to simulate missing values**

? If the value of '**STRATEGY**' is '**CONDITIONAL**', provide the ordinal numbers of the variables among which those to be used in the simulation will be chosen.

? Only the most reliable variables should be indicated.

? Provide a list of blank-separated ordinal numbers after the keyword. All of the variables must have been already indicated for processing in the preceding '**ORD_#_OF_ITEMS_TO_CHECK**' line(s), and none of them can coincide with the weight, if one has been specified.

? If '**STRATEGY**' is not '**CONDITIONAL**', the statement is ignored.

VARIABLES_TO_BE_USED 3 4 5 6 7 8 9 10

? **9. Select the best variables to be used**

? (used only if '**STRATEGY**' is '**CONDITIONAL**')
 ? Specify how many variables should be used in the simulation.
 ? Among the candidate variables already specified by the keyword
 ? '**VARIABLES_TO_BE_USED**', for each missing value those will be selected
 ? that have the highest correlation with the variable to be simulated.

SELECT_BEST **3**

? **Example** The following two lines:

? '**VARIABLES_TO_BE_USED** 1 2 3 4 5 10 11 12 13 14 15'
 ? '**SELECT_BEST** 3'

? mean that the variables with ordinal number 1 through 5 and 10 through 15 must be
 ? used to simulate others' variables missing values, choosing for each case the three
 ? of them that have the highest correlation with the variable whose value is
 ? being simulated.

? **10. Data reading format.**

? It's a string, included in double quotes, used to specify which variables, among all
 ? those included in the record, are to be processed, and which to be ignored.
 ? The format follows the usual ADDATI conventions (see Chp.3, or the online help
 ? available for the Multivariate Analysis routines).
 ? If a **free format** is used, the input records must have been prepared so that they
 ? include, in the order, **all and only the variables to be read**, separated by spaces.
 ? In such case, it is sufficient to enter **an asterisk** '*' instead of a full format.

FORMAT "*"

2-5 SELECT

A utility to select a particular set of statistical units

Use	The user can select from the input file only those statistical units for which some key-variables take some combinations of values defined by the user. Only the records representing the selected units are written to the output file.
Files	<p>The program reads:</p> <ul style="list-style-type: none">• an input file, that contains the records (one per statistical unit) on which the selection must operate;• a variable number of selection files (also called filter files), that include the information necessary to make the selection.
Requir.	<p>The input and the filter files must include information relative to the same units, considered in the same order.</p> <p>The input file, from which the records to be written are taken, can be in any format. Instead, the records in the filter files must be organised per fields, each corresponding to a variable, separated by blanks.</p>
Advice	<p>If the filter files do not consist in blank-separated fields, SELECT cannot load the values of the variables on which the selection is based, and cannot work. In this case, it is convenient to use MERGCHAR to copy to another auxiliary file only the variables used to operate the selection, separating them with blanks.</p> <p>The resulting file, in free format, can be used as filter file.</p>

Example: *If a region consists of 300 municipalities, all files must have 300 records exactly, and in each file the i-th record must refer to the i-th municipality. The reason for having several files to describe the same units resides in the advantage to keep separated different types of information: the series of the population per municipality year by year, variables describing the housing stock or socio-economic information can be stored in different files.*

Note: *The selection can also be carried out starting from information contained in the input file itself; i.e., it is legal to specify a filter file coincident with the input file.*

To operate the selection, the user must enter the **logical condition** (usually obtained combining a set of elementary conditions) that a unit must fulfil to be selected. This logical condition specifies the acceptable combination(s) of values of the selection variables.

The overall logical condition is formed by combining the following elementary items:

- **numbers** (1, -3.2, 300...) which are input as floating point values;
- **field identifiers**, that consist of one letter followed by an integer: the letter (starting from 'A' for the first file) denotes the selection file from which the field must be drawn; the integer is the ordinal number of the field in the file's records;
- the arithmetic operators " +, -, *, : " that must stay between the numbers or field identifiers on which they operate. Notice that ' : ' is used instead of ' / ' (whose meaning is different) to denote a ratio;

- **the relational operators** '<, >, =, >=, <=, <>' (the last symbol means 'not equal'); they must stay between the two terms on which they operate: numbers, or field identifiers or complex groups that must be pre-evaluated and include at least one arithmetic operator;
- **the logical operators** 'AND, OR, XOR, NOT'.

Round brackets are used to increase priority. At a given parenthesis level, the expression is evaluated left to right, according to the following precedence rules:

1. first the arithmetic operators are evaluated, then the relational operators, then the logical ones;
2. " * " and " : " are evaluated prior to " + " and " - ";
3. "NOT" is evaluated before the other logical operators.

Arithmetic operators operate on floating point quantities (numbers, field contents or results of previously performed arithmetic operations); they output a floating point value.

Relational operators compare floating point values (user-entered numbers, contents of fields specified by field identifiers, results of arithmetic operations); they return a truth value (TRUE if the condition is fulfilled, FALSE otherwise).

Logical operators act on two truth values and output a truth value as result. An exception is NOT', which acts on one term (conventionally following the operator).

Note: "AND" returns TRUE only when BOTH its input terms are TRUE;

"OR" returns TRUE when AT LEAST one of its input terms is TRUE;

"XOR" (eXclusive OR) returns TRUE when exactly one of its input terms is TRUE;

"NOT" returns TRUE when its input is FALSE, and vice versa..

The global condition entered by the user is converted to upper case (lower or upper case can be used indifferently, or they can be intermixed); its syntax is checked, as well as the acceptability of the field identifiers. Then, one record is read from the input file and from each of the selection files; the fields included in the condition are converted to floating point and assigned to the expressions where they act as terms. The global truth value is computed: if the result is TRUE, the record read from the input file is copied to output.

The operation is repeated for all the records of the input file.

The ' / ' operator

This operator allows the user to specify that a same condition must be fulfilled by a set of consecutive fields.

Example: instead of " A4 >.05 AND A5 >.05 AND A6 >.05 AND A7 >.05 " you can enter " **A4/7 >.05** "

This means that all the fields from the 4th through the 8th (included) in the first selection file must satisfy the condition. You can also enter "b3/6 <= a1", which means that all values from the 3rd to the 6th (inclusive) in the second file cannot be greater than the first value in the first file.

A condition entered using the ' / ' operator is expanded prior to all evaluation, and all the resulting logical conditions are assigned the same priority.

Remember The operator ' / ' must be included in the first term of a condition; it is illegal to include it in the second term or in both terms: "a4/8 > b3/7" is not acceptable.

Example: The condition: “ **a1/10 > .05 and (b3 < 5 or b3 > 10)** ” is worked out as follows.

The first condition, written in compact form, is expanded to

a1 > .05 and a2 > .05.... and a10 > .05

requiring all the first ten values in the first file to be greater than .05. The evaluation starts considering the expression in brackets: it is TRUE only when the variable contained in the third field of the second selection file is less than 5 or greater than 10. Imagine for example that the field contains the vegetation cycle's starting month: the condition in brackets is TRUE only for the districts whose cycle starts between November and April (inclusive). The overall condition specifies that besides this, a district must **also** have the first ten values in the first selection file **all** greater than .05.

The blanks can usually be omitted when this does not compromise the meaning. The condition above could also be written as:

a1/10>.05and(b3<5or b3>10)

and still be correctly interpreted. Notice however the blank between 'or' and 'b3': without it, 'orb3' would be assumed as a field identifier and rejected. For sake of clarity, better to use brackets and spaces when writing a condition.

Example: the condition “ **a3 <> b1 and a4:a5 = .5** ”

selects all the units for which the field 3 in file1 is different from field 1 in the second file, AND the ratio between fields 4 and 5 in the first file is exactly .5.

The formulations:

NOT (a3 = b1) AND a5 = 2 * a4 **or**

(a3 < b1 OR a3 > b1) AND a5 : a4 = 2

are other equivalent ways to express the same condition.

2-6 SHOWREC

Inspecting and modifying a large formatted data file by accessing it directly

Use	<p>The user can inspect a very large text file, too large to see it easily in a normal editor. After the addition to the package of the ADDATI Editor, that can handle very large files, this program can be considered obsolete. It is anyway still useful as it can move rapidly through large formatted text files without loading them in central memory. Therefore, its requirement on machine resources is very low.</p> <p>It works with files of any dimension, and it is possible to overwrite the current file's contents when little corrections are sufficient.</p>
Requir.	<p>It is absolutely required that the file have a fixed format. All its records must have, <i>at least logically</i>, the same length. The actual presence of an EOR (end-of-record) at the end of each record is not strictly necessary: it is sufficient that the record can be split in pieces of equal length, that can be considered as <i>logical records</i>. If it does not encounter EOR characters, SHOWREC prompts the user for a logical length and uses it to display the file contents suitably split.</p>
Limits	<p>No limit in the size of the file, that must have a fixed record length.</p>

This utility allows the user to inspect the contents of a text data file when its length or the excessive record size make it difficult or impossible to use an editor.

It is possible to browse very quickly through files of any length, overwriting them on the spot if a limited correction is needed.

For a regular use the file must have a fixed format, i.e. all records must have -at least logically- the same length. The actual presence of an EOR at the end of each record is not strictly necessary, as long as the file contains information coming in chunks of equal size. If it cannot find any end-of- record, **SHOWREC** prompts the user to enter a record logical length and displays the file contents split accordingly.

Up to 18 lines can appear on the screen at the same time. The cursor can be moved with the arrow keys (CTRL - ↑ ↓ → ←, PgUp and PgDn do it more quickly).

The **F5** and **F6** keys take the cursor respectively to the beginning and to the end of the current record. The cursor's position in the file (record and column) is displayed and continuously updated.

An area at the bottom of the screen shows the available commands.

Example "+1000" moves 1000 records forwards (or to the end of file);
 "-5000,10" moves 5000 records backwards (or to the top of file) and displays 10 records;
 "2000,12" moves to record #.2000 and displays 12 records;

The key **F1** is a status toggle key that lets you enter/exit the *Overwriting* condition.

You can write on the screen, so changing the displayed page; the changes are shown in a different colour and are saved on request.

Note: *SHOWREC is not an editor: only the displayed portion of the file is loaded in the core memory. This makes it very fast, but allows it only to overwrite the current file contents. Nothing can be inserted or appended, as the file cannot change its size.*

***Note:** **SHOWREC** can be very useful to correct errors in large data files (e.g., Census files at household level, or survey files). The best way to proceed is as follows:*

- *submit the file to **FIXFORM** if it is in free format, in order to convert all records to the same length;*
- *use one of the **ADDATI** routines that, albeit oriented to other objectives, control the correct coding of the variables and write to a file the errors encountered (e.g., **DISTRIB**, **TYPOL**, or **RECODE**);*
- *use this information to aim immediately, in **SHOWREC**, at the records to be corrected.*

2-7 RECODE

Conversion (recoding) of categorical or quantitative variables

Use	it recodes both quantitative and categorical variables according to various options. The output file has exactly the same format as the input file: the values resulting from the conversion replace the original ones, while the values of all the other variables are left unaltered.
Requir.	The input file must be a text file in fix format .
Limits	The worksheet permits the user to recode up to 15 variables at a time. The limit is actually only apparent, as a new work session can be started on the same file just after a conversion, without quitting the program. In such case, the file just written, that contains the recoded variables, is assumed as the input. The input records cannot be longer than 5120 characters.

RECODE is used in the preparation of the table for a factor analysis when any recoding is needed, but its utility goes beyond this.

The user is prompted for the name of the input file, the one that contains the variables to be recoded. The file **must have a fixed format** (if necessary, run **FIXFORM** to convert it): a variable is recognised from the constant position it occupies in each record. The data input is *character-oriented, not field oriented*. This requires a particular care in the preparation of the file to be processed and its documentation.

The choice of processing the input file by characters has the advantage of dealing easily with missing values. When the record is organised by fields, a missing value is recognised only if it is replaced by a specific code. Instead, when the record has a fix format, blanks encountered where a variable is expected are easily interpreted as missing information.

Up to 15 variables can be recoded in a work session and are written to the output file **exactly in the same place as the original ones**, while the variables not affected by any recoding are copied unchanged. Therefore, the format of the output file is exactly the same as that of the input one.

When a recoding session is completed, it is possible to start a new session without exiting the program. The user can utilise the same input and output files, or change them. If the names are not changed, RECODE automatically assumes as input the output file written in the preceding session.

The program displays the worksheet shown in table 2-2. The user enters the information concerning the variables he wants to recode.

VARIABLES TO BE RECODED					THRESHOLDS OR CATEGORIES	
var. #	first Column	field. length	Operation type	# new classes		
01						01
02						02
03						03
04						04
05						05
06						06
07						07
08						08
09						09
10						10
11						11
12						12
13						13
14						14
15						15
IT'S THE VARIABLE'S STARTING COLUMN						

Table 2-2 The worksheet displayed by RECODE.

Imagine the worksheet on the screen divided in cells, each cell lying where a row and a column intersect. There are 15 rows (one for each variable to be recoded) and 5 columns. For each variable to be recoded, the following information must be entered in the cells of the corresponding row.

- Column 1:** contains **the variable's starting position** (in characters) from the beginning of each record. E.g., '12' means that the value of the variable is to be read in each record starting from the 12th character. The value entered in this field can range 1 to 5120. The program checks if this value is compatible with the record length and with the parameters already declared for other variables.
- Column 2:** contains **the variable's field length** (i.e., how many characters that variable occupies in all records). The value can range 1 to 10. The program checks if the value is consistent with the one entered in the FIRST COLUMN cell. Example: if the FIELD LENGTH for a variable is 4, while its FIRST COLUMN cell contains '12', the value of the variable will be obtained from the characters 12 to 15.
- Column 3:** it will contain a number in [1..4] representing the type of recoding operation to be carried out (see below);
- Column 4:** enter here the number of the classes (or categories) of the categorical variable resulting from the conversion. The number of the new classes can be in the range [1..15] for the recording operations 1 and 2, while for the operations 3 and 4 the user can request up to 25 classes.
- Column 5 :** this cell is to be used only for operations 1 and 2. The user will enter **the minimum value** of each class (for operation 1), or the list of the categories to be assigned to each new class, separated by blanks (for operation 2). In the former case, **values less than the minimum threshold will not be recoded**. In the latter case **a maximum of ten categories** can be assigned to each class, and **the categories not explicitly assigned will be automatically attributed to the last class**.

If more than 10 categories are to be assigned to the same class, the operation can be carried out in one step. At least a double step is necessary.

The value entered by the user is displayed in the active cell, that has a different colour. The value is confirmed with ↵ or moving to another cell with the arrow keys.

Note: Errors are detected and signalled at the beginning of the recoding phase, triggered with F1.

The recoding operations

Four types of recoding are available. They operate only on numeric values: all non-numeric input is assumed as an error.

After recoding, the new variables have values ranging from 1 to the user-defined number of classes.

While describing the **four conversion types**, we will indicate the type of variables on which each of them should "normally" operate.

- 1. (for quantitative or ordinal variables)** The variable's values are grouped according to thresholds provided by the user. The number of desired groups is entered in column 4, while the thresholds (as many as there are groups) are typed in column 5. For each group, you have to supply the variable's lowest value that results in an assignment to that group. When running, that group will be assigned all cases having the value of that variable greater than or equal to the supplied limit, but less than the lower limit of next group.

Example *the values of a quantitative variable must be splitted into five classes according to the following intervals: (-15, -5), (-4, 6), (7, 14), (15, 21), (over 21). The threshold values to be entered are:*

- *minimum for class 1: -15 (or any lower value);*
- *minimum for class 2: -4;*
- *minimum for class 3: 7;*
- *minimum for class 4: 15;*
- *minimum for class 5: 21;*

- 2. (only for non-ordinal categorical variables)** Aggregation of categories. You have to **enumerate** in column 5 the categories (max. 10) to be assigned to each resulting group (or class), using the space as separator.

Example: *the eight categories of a variable must be grouped into three classes as follows: categories 1 and 3 into class 1; categories 4, 6 and 7 into class 2 and the residual ones - 2, 5 and 8 - into the third clas. The user will enter:*

- *column 5, row 1: 1 3*
- *column 5, row 2: 4 6 7*
- *column 5, row 3: 2 5 8*

Note: Every category must be unequivocally assigned to one class. The last class is assigned not only the categories explicitly listed by the user, but also those for which no class has been indicated.

3. (for **quantitative variables**) The variable's values are grouped in **intervals of equal amplitude**, automatically determined by the program. The number of groups is entered in column 4. Column 5 is not used.
4. (for **quantitative variables**) The input values are grouped in intervals of different length so to achieve equifrequency, at least as far as possible. Each resulting group will include **approximately the same number of cases**. The categories of the resulting variable are **"well balanced"**: this is important in order to avoid any undesirable statistical dominance effect when that variable is included in a table to be processed via an Analysis of Correspondences.
The number of the groups is entered in column 4; column 5 is not used.

Control keys used in RECODE

- ESC** terminates and returns to the ADDATI menu
- F1** starts recoding
- F10** temporary exit to DOS; RECODE is resumed by typing "EXIT"
- ?** context-sensitive help
- ALT-C** clears the table on the screen, discarding all parameters already entered
- ALT-I** loads a new input file
- ALT-O** defines a new output file
- ALT-H** displays some notes on the program
- ALT-P** loads a parameter file

The **arrow keys** and **<ENTER>** allow the user to change cell or to confirm the value that has been entered.

Note: Beside saving the converted file, RECODE writes three more files:

- a report file in which details on the performed operations are listed. It is named "REC_????.DAT", where "????" is a progressive number automatically determined by the program, so as not to overwrite other reports in the same directory
- a file named "REC_????.ERR", which lists the errors encountered while reading the input data file. For each error, the following information is saved: the record's ordinal number, the variable and the type of error
- a file of parameters, containing all the parameters entered by the user during the last recoding session (file names included). It is automatically named "REC_????.PAR" by the program. By loading it, the user can repeat the same analysis, or change some parameters, without being obliged to enter all parameters from scratch.

Use **ALT-P** to re-load a previously saved file of parameters.

2-9 FACPLAN

Displaying the projection of units/variables onto factorial planes

Use	The projections are displayed using the information saved by ACORR , ACOMP and NONGER to specific with extension .FPL . The images obtained can be recorded to file, or printed.
Files	The input file must have a particular structure. The programme operates only on .FPL files written by ACORR , ACOMP or NONGER , or on files with similar structure.
Limits	The input file can include the values of up to six factorial co-ordinates.

In order to interpret the results of a factorial analysis – in particular, to assess the meaning of factors – the absolute and relative contributions of the variables must be examined. Yet, it is useful to inspect how the points representing units and/or variables are located on the main factorial planes, as this allows one to capture quickly the most important existing relationships.

FACPLAN is a Win32 application that displays the projections of variables and/or statistical units onto the most explanatory factorial planes.

On user's request, the factorial Analyses programmes (**ACOMP** and **ACORR**) save onto a file with extension **.FPL** the information needed by **FACPLAN**.

When run, **NONGER** (the non hierarchical Clustering programme) automatically re-reads what **ACOMP** / **ACORR** have written and modifies it, replacing the little squares that indicate the location of statistical units in the factorial planes with the number of their class. **NONGER** adds also the centres of the classes, and the output is saved to a file named NGnn.FPL, where 'nn' is the number of the classes in the partition.

Figure 2-1 shows a FACPLAN screen: it displays plain of factors 1-2 relative to an analysis on the 39615 households of the FIES2000 survey in the Philippines. The number of the units is written in grey near the '*active units*' checkbox: it can be seen clearly only if the picture is enlarged. The yellow circles show the location of the centres of the classes.

In this case the units are not displayed: they are too much. They are displayed automatically when FACPLAN starts only if they are less than 2000, but the user can always force their displaying by checking the checkbox. The units' labels are never displayed when the units are more than 2000: it would be too slow and confusing.

FACPLAN can:

- display any combination of active/supplementary variables/units;
- display the projections onto the plane spanned by any pair of factors among those saved on user's request (up to six);
- limit the view only to some classes indicated by the user (if the units are classified), in order to facilitate the interpretation of the characteristics of the classes with respect to the descriptive variables;
- display the labels identifying the variables or the units;
- allow the user to draw lines connecting some items displayed (typically, the categories of a categorical variable);
- ...do other things you will discover using the programme.

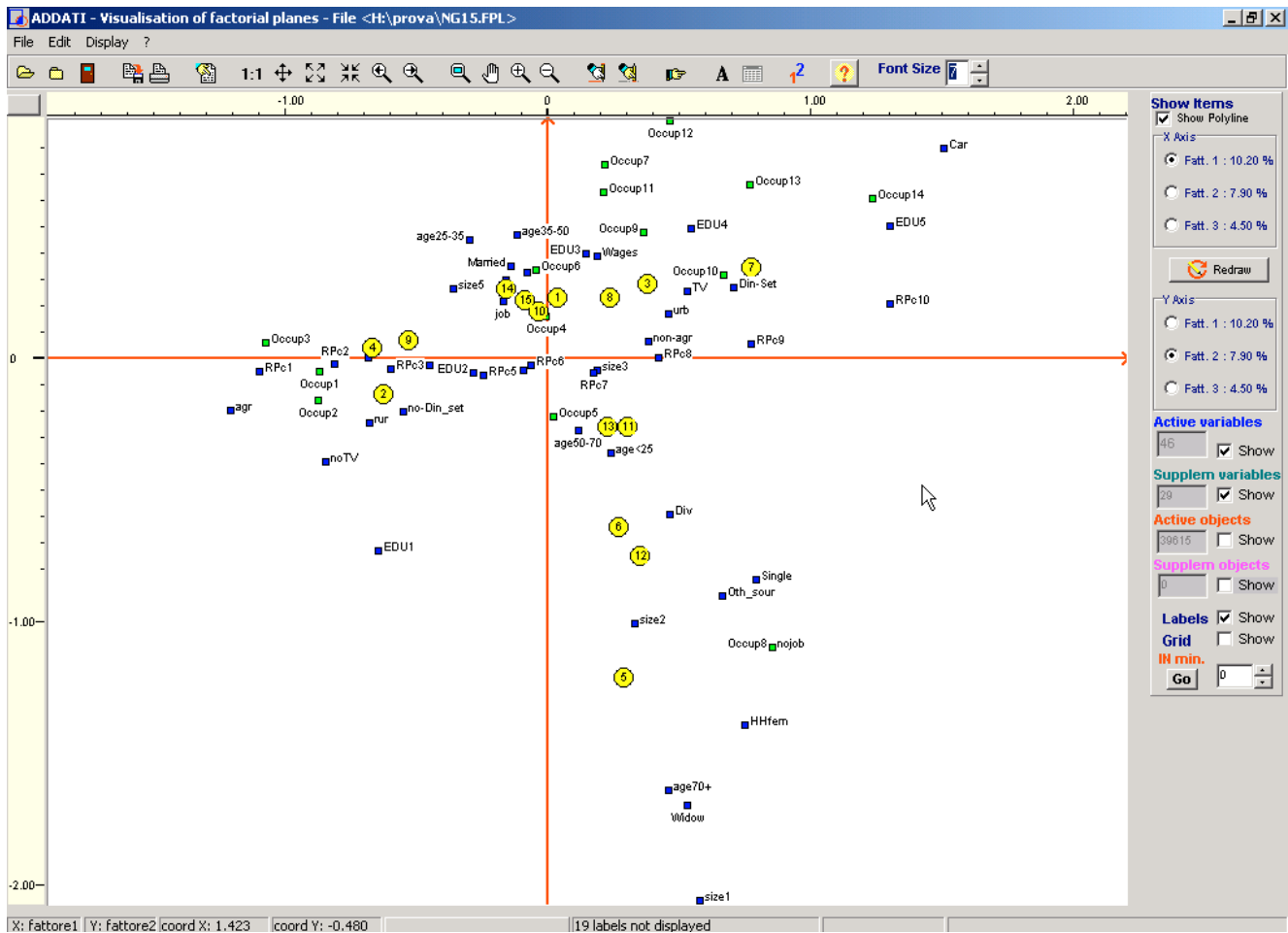


Figure 2-1 Sample of a FACPLAN screen.

2-9 INTEGRATE

Updating the original data file after clustering a set of elementary units (starting from TYPOLOG)

INTEGRATE is a very simple programme that puts together some info saved by **TYPOLÓG** and **NONGER** after a clustering sequence started with **TYPOLÓG**, i.e. aggregating a high number of elementary units (e.g., households) in typologies, on the basis of the values of the *active* variables.

- **TYPOLÓG** stores to a file named TYPCLAS the typology to which each elementary unit belongs;
- **NONGER** saves to a file named NGCLASnn.TXT, where ‘nn’ stands for the number of the classes, the class to which each typology has been assigned.

INTEGRATE reads these two files and writes to a file conventionally named CLASSI.CLS the assignment class of the elementary units, in their order. Using the utilities **MERGFIELD** or **MERGCHAR** the resulting column can then be added, as a new synthetic information, to the initial data file.

Chapter 3 – The User Interface

This menu (see table 3-1) includes some statistical analyses utilised to synthesise the most relevant aspects of the information conveyed by a large data table. A minor part of the information is ignored, but the most significant one is captured in easily interpretable forms. This enables the analyst to focus on the most relevant features of the case he is studying, without being confused or misled by comparatively unimportant facts, or by the huge amount of data.

Analyses	Utilities	Help
Create Typologies		TPOLOG
Principal Components Analysis		ACOMP
Analysis of Correspondances		ACORR
Non-Hierarchical Clustering		NONGER
Hierarchical Clustering		AHC
Distributions		DISTRIB
Cross-tabulations		CROSSTAB

Table 3-1 The Analysis Menu

It is mostly question of *multivariate techniques*, useful to analyse a large set of statistical units (geographic areas, households, firms, etc.) described by many variables. The purpose is to explore the relationships among the variables or the similarity among the units. The utility for Territorial Analysis is evident: Classification problems, Census Data Analysis, Survey data processing, have in ADDATI a tool purposefully conceived. Anyway, the package can be easily used also in other disciplines when problems of similar nature are to be faced.

The flow of information through programmes that work in a sequence and do the various steps of an analysis is automatic, carried out by writing and reading some work files with extension ".LV". Files with extension ".OUT" contain the results to be interpreted. It is convenient to create a new folder for each different analysis, to avoid that files concerning different analyses may interfere with one another.

The options offered in the Utility Menu help the user to extract the input variables necessary for a particular analysis and to handle them suitably, preparing the data table eventually submitted to the exploratory techniques included in the Analysis Menu.

3-1 Some relevant features in ADDATI

The Analysis and Utility programmes included in ADDATI are, with some exception, DOS programmes. Those carrying out multivariate analyses are 32-bit applications that run under a DOS Extender. This makes them very fast and capable to use, if necessary, the whole available core memory. They are written in C, a powerful high-level programming language that also allows a complete control of low-level aspects.

Clearly, to an analyst used to work with Windows the interface will appear rather rough, even if it works quite well. Anyway this will not be for long: a full Win32 version is being prepared.

For what concerns its interface, ADDATI has the following characteristics:

- the parameters required to control an analysis (number of variables or statistical units, labels for units and variables, et.) are entered in specific windows, and it is always possible to correct them afterwards. It is also possible to load a file of parameters saved during a previous analysis and modify it, to adapt it to the current analysis;
- the memory is used dynamically: exactly the necessary amount of memory is requested to the OS when needed, and immediately released when it is no longer necessary (this guarantees a rational use of the available core memory);
- the package is self-illustrated. It offers an online help: the key '?' displays one or more pages of contextual help;
- the input data can be both in *free or fixed format*: in the latter case the user must provide a format that follows some simple rules and is flexibly interpreted by the parsing routine. The format controls the way the data file is read. A FORTRAN format is generally accepted;
- the projections of variables/units onto the factorial plans are displayed by a new Win32 application, very powerful and friendly.

3-2 The user interface (some general notes)

Some aspects are common to all the Multivariate Analysis programmes (**TPOLOG**, **ACOMP**, **ACORR**, **NONGER** and **AHC**). We describe them shortly before dealing with the individual applications.

Modifying the answers already entered

When an analysis is started, the user is requested to enter the information necessary to execute (input filename, number of the statistical units, number of variables and their labels, etc.). The sequence of the questions asked by the programme depends on the answers supplied by the user.

The user can always her/his answers and modify some of them in case of error, or if she/he just changed her/his mind (e.g., on which variables are to be assumed as supplementary).

F1 to review the replies already entered, starting from the first one;

PgUp to review the answers starting from the last one

***Note:** the current place in the flow of questions is maintained. The answers previously entered can be seen with PgUp e PgDn.*

F2 brings back to the last question, exiting the correction phase.

The sequence of the questions may be altered if any of the answers already entered is modified. Answers to no longer necessary questions are discarded, and the memory used to store them is freed. A need of new questions might arise.

Each time the user hits F2 to quit the correction stage all parameters are checked for cross-consistency, and the necessary extra-questions are asked.

While the parameters are entered the key

'?' commands the displaying of contextual help, if any is available

F10 permits a temporary exit to DOS. Type "EXIT" to resume the programme.

(*The above option is obsolete* working in Windows: it is always possible to use ALT-ENTER to change the size of the DOS box, and perform any necessary side operation in Windows. F10 can still be used to verify which is the current working directory for the DOS programme being run).

F3 runs an internal programme that displays the contents of a text file. Also this option, used to inspect a datafile, or an output file, etc., is now obsolete: the ADDATI editor can be used to see the contents of any text file, with all the advantages listed in chapter 1.

Loading a file of parameters

All the analyses that load an external input data file (e.g., **TYPOLOG**, **ACORR**, **ACOMP**, **NONGER**, **AHC**) need some information that is usually entered via keyboard. This operation, quite fast for a skilful user with a good practice in the use of the package, can appear tedious and complicate to a new user, specially when, for example, the labels of many variables are to be listed.

It is certainly advisable to read carefully the available help screens to understand correctly the questions asked. Anyway, when all the parameters relative to a given analysis have been entered, they are stored for possible future use to a text file that has the same filename as the programme, and extension ".PAR" (ACOMP.PAR, NONGER.PAR, ecc.). This file can be loaded to repeat an analysis changing only some parameters: some ADDATI programmes ask a specific question to this purpose when they are started.

The case of multiple labels or indicators

The user is sometimes prompted to enter several indicators at a time: labels for statistical units or variables, the ordinal numbers of the supplementary variables (if any), etc. Remember that many ADDATI programmes accept a compact form: e.g., "**variable1/50**" is expanded to the 50 labels "variable01" through "variable50"; "**1/60**" is expanded to "1" through "60", etc.

You can also alternate individual labels and others in compact form, entering several alphanumeric groups separated by blanks. The program checks the number of the indicators and the acceptability of their format.

Online help

One or more pages of context-specific help is displayed when the question mark is hit "?", generally the key to get a help in ADDATI. Not all questions and not all programmes offer a help, but most do.

The help system is extensive and offers a good idea of what ADDATI does; it often includes also some methodological hints. At least as a beginner, keep on pressing '?' and read carefully. If no help is available you get a beep.

All the programmes included in the Analysis Menu that need to input data from an external file (i.e., not written by ADDATI itself in a chained analysis), prompt the user to enter a **format to read the input file**. This aims at informing the programme about the exact location in an input record of the various items to be loaded (unit's label, unit's weight, the variables to be read in), and those to be skipped.

Two answers are possible. The user can either type an **asterisk** '*' if the input file is in **free format**, (see chp. 2), or enter a valid **format**.

Free format

Each record in the input file must contain - **separated by blanks or commas** - exactly the items the program expects to load according to the preceding answers. **The order must be exactly as follows.**

- A label (a name) for the statistical unit to which the record refers (up to 12 characters) if you have declared that objects' labels will be read from file (the alternative is to enter them from keyboard).
- The weight to be assigned to that unit in the analysis; this applies only when the analysis uses a weight, and in the **WEIGHT** statement it was declared that each case has a specific weight (the alternative is to use the same weight for all cases). Weights must be integer or floating point valid values.
- All and only the variables to be loaded, in the same order you have declared them.

Obviously, a file containing exactly as many records as there are units, with each record having exactly the structure specified above, must have been purposefully prepared using some external software (e.g., EXCEL) or some of the utilities offered in ADDATI. This can be quick and easy if the user knows in advance exactly which variables will be included in the analysis, as *active* or *supplementary*. However if the user changes her/his mind for any reason, and decides to repeat the analysis excluding some variables, or including some new ones, a new file must be created.

It is then better to design a file that includes all the **potentially interesting variables** and use it as the input file for several analyses, specifying each time which variables must be loaded and which ignored. In order to do this a **format** is necessary.

The reading format syntax

It follows some very simple rules. It is a simplified version of the standard FORTRAN format, that the parsing routine is flexible enough to understand and accept in most cases. The following **types** are recognised (lower and upper cases are equivalent):

- **type 'A'** for an **alphanumeric label** (e.g., 'A6' 'a6' '6A' and '6a' all denote a 6 character label);
- **type 'P'** for the unit's **weight**, (e.g., both 'P7' and '7P' denote a variable with a 7 byte field length, to be used as the unit's weight);
- **type 'X'** to command a **skipping** (e.g., both '5X' and 'x5' command the skipping of 5 bytes from the current location in the input record);

- a **number** indicates the field length of a variable. For example, '4' stands for a variable with a 4-byte field length. It must be remembered that the file is formatted (see chapter 2), and therefore a **variable occupies exactly the same position** (a given amount of characters) in each record. This is what we mean here by variable's field length: it's the space in a record reserved for that variable. Part of it can contain blanks, if the actual value does not occupy all the space reserved for it. For several consecutive variables with the same field length, a compact form can be used as shown in the next example.

Example

*'5*4' stands for 5 consecutive variables, each 4 byte long; '4*10' denotes 4 consecutive variables, each of 10 bytes.*

*It is not necessary to specify if the values are integer or floating point: the programme interprets them correctly. Therefore, '5*6' is perfectly equivalent to the Fortran notations "5F6.3" or "5I6" (which are also accepted);*

- if a statistical unit is described by more than one record, "/" moves to the beginning of next record, "2/" or "/2" skip a record and start reading from the next one.

A **reading format** is made up by as many of these elementary groups as necessary, in any order, separated by **commas** or **blanks** and optionally enclosed - only for compatibility with Fortran - in round brackets. Of course, the format must be consistent with the parameters previously supplied (number of variables, record length, etc.). It is checked, and a diagnostic is issued in case of error or inconsistency.

Let us consider some examples.

Example 1 The format 'a6, p5, 2x, 5*6 5x 6*7'

commands the reading from each input record of a 6-character label and of a 5 byte value assumed as the unit's weight; 2 bytes are skipped and 5 consecutive 6-byte variables are read; 5 bytes are then skipped and six 7-byte variables are read in.

Obviously, this format must be consistent with the previous statements: as the format is immediately parsed, discrepancies, if any, are signalled and can be corrected.

The format above implies that label and weight be read from file. The variables to be loaded must be 11 altogether.

Example 2 The format '(3, 5X, 4, 4,/ 4A 10x, 6*3)'

means that a three-byte variable is read; five characters are skipped and two more variables, each of four bytes, are read. The rest of the record is skipped. From the beginning of the following record a four-byte label is loaded, ten bytes are skipped and six three-byte variables are loaded. No weight is read from file in this case.

Note that the unit's case label is not the first item loaded: this is compulsory if the format is free, otherwise the label can stay anywhere in the record, its place being determined unequivocally by the format.

The above examples are absolutely general: the same syntax holds for all the ADDATI programmes that require a format to be entered. Notice that **TPOLOG** does not use labels for the statistical units, and does not expect to load any; **ACORR** computes the units' weights internally from the data table, and does not prompt the user for them.

This is a user's manual, not a theoretical book on multivariate statistics. For a deeper understanding of how factor analyses and clustering techniques work, it is necessary to have recourse to some specialised textbooks. As ADDATI follows the findings of the French school of "*Analyse des données*" (Data Analysis), two good references in English are the following:

L Lebart, A Morineau, K Warwick, *Multivariate descriptive statistical analysis*, Wiley, 1984;

M Jambu, M O Lebeaux, *Cluster analysis and data analysis*, North Holland, 1983.

This chapter presents a simplified summary of the statistical ideas on which the multivariate procedures included in ADDATI are based, as well as of the terminology used throughout the package.

4-1 The scales of measure

When a multivariate analysis must be carried out, the preparation of the table is certainly the most important and delicate task.

It can be question of a **descriptive table**, with rows representing statistical units (say, administrative districts) described by some indicators (columns) directly observed or suitably constructed, qualitative or quantitative (say, the series of 10-day, or dekadal, NDVI values, or a set of variables concerning socio-economic or demographic characters). Generally speaking, the variables included in the table must represent in an appropriate and complete way all the units' properties that are essential for the purpose for which the table is being built. Nothing less - even though a compromise is sometimes necessary owing to data unavailability- but also nothing more, as the inclusion of an inappropriate variable can bias results in a probably undesired way. The choice of the variables must be based on substantial assumptions, on which a wide consensus is needed: only seldom is the perception of the problem the same for all the actors involved.

Other kind of tables can be dealt with by these methods: e.g., a table of choice alternatives, evaluated according to a set of criteria, with the objective of ranking the alternatives according to their global utility. The common feature is the **multidimensionality of the description**: the *statistical units* (or *objects*, each of them described by a row of the data table) are considered according to a plurality of aspects, among which an a-priori unknown network of relationships exists. The usual path of analysis explores this network and reduces the dimensionality of the phenomenon, dropping only a small part of information in an optimal way. Then, units are clustered according to their similarity, globally defined taking into account all elementary aspects.

The variables considered can be measured on different scales, and a path of analysis must be selected that fits the scale on which variables are measured. When variables to be included in an analysis are measured according to different scales, they must be submitted to a preliminary treatment ("**recoding**") in order to make them homogeneous, i.e. to express all of them according to a same scale (see program RECODE). This is generally not sufficient: in addition, for categorical variables the number of their categories and

their frequencies must be balanced with particular care, in order to avoid undesired effects of statistical dominance of a variable over the others in determining results.

	TYPE OF VARIABLES		
	QUANTITATIVE	QUALITATIVE	
		ORDINAL	NOMINAL
Do arithmetic operations on values make sense?	yes	no	no
Are values ordered?	yes	yes	no
Type of values	numeric	alphanumeric codes	alphanumeric codes

Table 4-1 The scales of measure for the variables.

The table 4-1 lists the types of scale. It is important to be able to recognise which of them is used for a variable, in order to set a methodologically correct analysis.

Quantitative (or continuous) variables

They assume numeric values, expressed in a given unit of measurement or sometimes a-dimensional; arithmetic operations make sense. Pro-capita income, NDVI values, mm. of rainfall, population, activity rate, the dwelling surface, etc. are examples of quantitative variables.

A variable takes a value for each statistical unit (individuals, households, administrative areas, meteorological stations, etc.): when processed, these values are often *weighed*. The weight expresses the relative importance of the concerned unit. Also the *mean* of the variable over all units and its *standard deviation*, defined further on, are generally computed and used in an analysis.

Categorical (or qualitative) variables

They can take a limited set of values, represented by codes. Numbers can be used as codes, but this is not compulsory: the meaning is non-numeric and no arithmetic operation makes sense. We can further split them into *ordinal* and *nominal* variables.

Ordinal categorical variables

They are usually obtained by recoding into qualitative form a continuous variable, in order to make it homogeneous to other qualitative variables in the same table. This operation causes the loss of a (usually small) amount of information, but does not change so much the behaviour of the statistical units and their similarity pattern. As an example, let us consider the activity rate of all the municipalities in a region: this value could take - at least in line of principle - any value between 0 and 100 and is therefore continuous. If it must be used in a table which includes also other socio-economic characters of the municipalities, *observed at the categorical scale*, the activity rate must be recoded: the 0-100 range must be subdivided in suitable sub-intervals (classes).

The table 4-2 shows an example. could be done: the interval 0-100 is split in four intervals of equal length, obtaining four classes of increasing activity rate, labelled with the numeric values 1 to 4. All the municipalities that fall in the same class get the same

label and their differences are lost after converting to the categorical scale. However, the major differences persist.

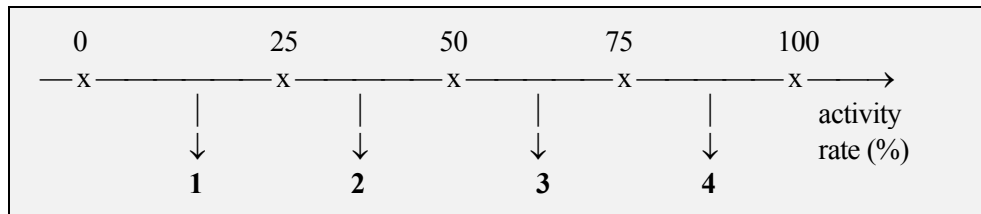


Table 4-2 Recoding of a quantitative variable into a four-category qualitative variable.

The above recoding operation can be inappropriate in many cases: for example, if actually the activity rate ranges from 17% to 45%, the third and fourth classes are empty. This prompts us to define our recoding rule **after determining** the minimal and maximal values for the variable concerned. If again we want four classes and equal length intervals, we get the recoding of table 4.3.

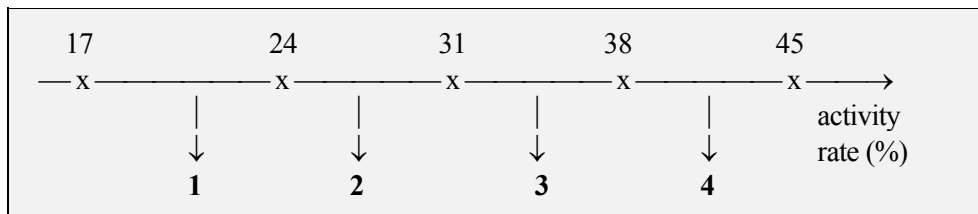


Table 4-3 The example above, with a 17-45% range.

Assume now that most municipalities have an activity rate between 30 and 40%: they will fall in the two central classes, while the two extreme ones would be almost empty. This is undesirable, as the behaviour of the system would be "flattened" (at least, for what concerns the description offered by this variable). A lot of information would be lost: namely, all difference actually existing amongst the municipalities belonging to classes 3 or 4.

It can be proved that **the loss of information is minimal when the thresholds limiting the interval are so determined as to achieve equi-frequency, i.e. when the resulting groups approximately consist of the same number of units**. This can be done automatically by using **RECODE** from the Utility Menu.

Even though numbers are used, labels 1...4 have here a non-numeric meaning: 4 is not the double of 2 and 'A' through 'D' would do as well. However, the underlying order is partially saved when a continuous variable is split: 2 does not mean the double of 1, but it indeed indicates a higher activity rate than 1. That is why the resulting qualitative variable is called **ordinal**.

Nominal categorical variables

They do not imply any underlying order; codes are simply labels representing different behaviours, with no order. The fact that numbers are used as codes is purely incidental. Variables with a "yes-no" meaning are nominal, but this is not the only type.

Think for example of assigning codes 1 through n to n types of crop: a variable defined as "district's prevalent crop" assumes a value 1...n for each district. It is a nominal variable with no underlying inherent order. As another example, consider a variable whose value

for each unit is the label of the class to which that unit has been assigned by a clustering procedure: it is also a nominal variable.

Codes used for categorical variables cannot be processed numerically; at least not directly. Operations like averaging or computing standard deviation are inappropriate for this scale. It is first necessary to go through a **conversion to binary form**: each categorical variable is split into as many new variables as it has categories, and each new variable takes value 1 for units that fall in the related category, 0 otherwise. As a case can fall only in one category, only one of the binary elementary variables obtained by recoding a categorical variable can be 1, and all the others are 0. This type of recoding is also known as **complete disjunctive recoding**.

The table 4-4 shows the values obtained by recoding in binary form the activity rate of table 4-3. As cases are grouped in four categories (classes), four binary variables are needed.

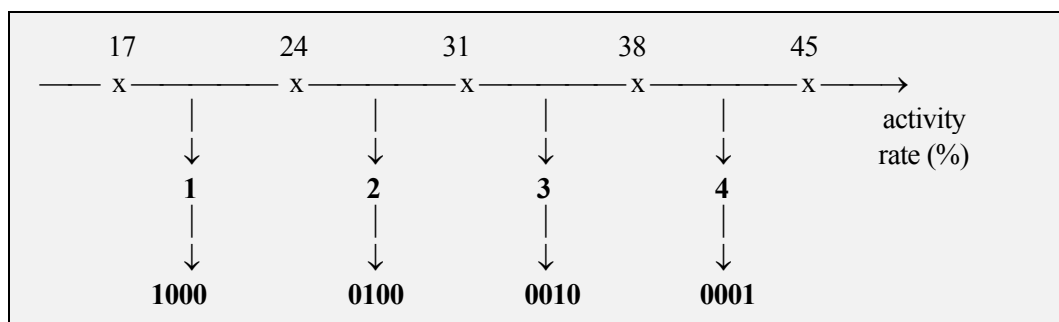


Table 4-4 The same example as above, followed by a conversion to binary form. Four binary variables are created and the one corresponding to the category taken assumes value 1.

Remember: A Principal Component Analysis (ACOMP) must be used exclusively on quantitative variables, while ACORR must be called for analysing contingency tables (see below), as well as tables obtained by recoding categorical variables in binary form..

4-2 Standardisation of continuous variables

It is worth to include a variable in an analysis only if it allows one to discriminate in an interesting way the behaviour of the various units. It must therefore generally take different values on different units (were it constant, it would be of no interest and should simply be dropped). It has already been remarked that for continuous variables arithmetic operations make sense. In particular, as the analysis focuses on the dispersion of the variable's values over the set of the statistical units, the very common concepts of **average** and **mean square deviation** are of great utility.

Let I be the set of the units, and n their number; let x_i denote the value taken by the variable x for unit i . We will mostly deal with units to which a **weight** is associated, representing their relative importance in that particular analysis: let m_i denote the weight of unit i . Weights are **normalised** by ADDATI, i.e. they are submitted to the following transformation:

$$m_i \leftarrow m_i / M$$

where $M = m_1 + \dots + m_n$ is the sum of all the weights. Once normalised, weights sum up to 1, and each of them measures in percentage terms the importance of the related unit.

The **weighted average** of variable x on I is given by the well-known formula

$$\bar{x} = \text{mean}(x) = m_1 * x_1 + m_2 * x_2 + \dots + m_n * x_n = \sum_i m_i * x_i \quad (4.1)$$

where the contribution of each unit to the average is affected by its weight (remember that weights are supposed to be normalised, i.e. they sum up to 1). As a particular case, when all weights are equal their normalised values are

$$m_1 = m_2 = \dots = m_n = 1/n$$

and the (4.1) becomes the well known **simple average**:

$$\bar{x} = (x_1 + x_2 + \dots + x_n)/n$$

The **average** is a measure of the variable's **central tendency**, and all values are scattered about it.

The variable's **variance** is a measure of its dispersion. It is defined as

$$\text{variance}(x) = \sigma^2(x) = m_1 * (x_1 - \bar{x})^2 + \dots + m_n * (x_n - \bar{x})^2 = \sum_i m_i * (x_i - \bar{x})^2$$

where $x_i - \bar{x}$ is the difference between the variable's value in the unit i and its average value. The variance adds up the squares of all these differences, each weighted with the importance of the unit where the difference appears.

The **standard deviation** of a variable is simply the square root of its variance:

$$\text{stdev}(x) = \sqrt{\sigma^2(x)}$$

As the values taken by a variable depend on the unit of measure assumed for it (and so do the differences with respect to the average, the variance and the standard deviation), a change of scale is recommended in order to bring all variables to the same relevance. With this purpose, the value x_{ij} taken by variable j for unit i is **standardised** (the term **normalised** is also used), i.e. transformed as follows:

$$x_{ij} \leftarrow \frac{x_{ij} - \bar{x}_j}{\text{stdev}(x_j)}$$

The variable's absolute values are converted into differences with respect to the average, and the scale is changed by dividing by the variable's standard deviation. This is a way to eliminate the effect of the unit of measure on the variable's absolute values. As a result of this transformation, the new variable obtained has average 0 and variance (and standard deviation) 1 by construction.

ADDATI's programmes **ACOMP** (Principal Component Analysis) and **NONGER** (non-hierarchical Clustering) perform in an automatic way a standardisation of the continuous variables in input, so that they are all assigned the same importance in the analysis.

Example 1 - Indicators of central tendency and dispersion

Let us assume three administrative units for which we specify the total population, the active population and the activity rate, as shown in the table 4-5.

	Population pop_i	Active pop. att_i	Activity rate $t_i = pop_i/att_i$	diff.	Squared diff	weight m_i
Region 1	40.000	12.000	0.30	-.06	.0036	0.27
Region 2	100.000	40.000	0.40	+.04	.0016	0.667
Region 3	10.000	2.000	0.20	-.16	.0256	0.067
Total	150.000	54.000				1.0

Table 4-5 Sample data for three administrative units.

Is it correct to compute the activity rate of the overall system as follows

$$\bar{t} = \frac{0.30 + 0.40 + 0.20}{3} = 0.30 ?$$

That is, does it make sense to compute a simple average? It is necessary to consider that

- the three regions have a different population, therefore they give a different contribution to the computation of the statistics relative to the overall system;
- every region is a part of the overall system, and contributes to it in proportion with its population.

m_i = weight of Region i = $pop_i / \text{total population} = pop_i / (pop_1 + pop_2 + pop_3)$

The sum of the three weights so defined is one: $m_1 + m_2 + m_3 = 1$

The (**weighted**) **average** is an indicator of the central tendency of a distribution.

Weighted average: $\bar{t} = \sum_i m_i * t_i = m_1 * t_1 + m_2 * t_2 + m_3 * t_3 =$
 $= 0.27 * 0.30 + 0.667 * 0.40 + 0.067 * 0.20 = \mathbf{0.36}$

The weight must be taken into account also in the computation of all other statistics (e.g., the variance)

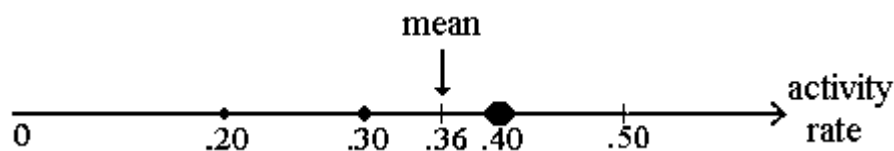


Figure 4-1 The activity rate values of the three regions represented on an axis. The size of each point is proportional to its population. The average falls in the centre of gravity of the three weighed points.

The variance of the activity rate for the three regions is

$$\text{Variance}(t) = \sigma^2(t) = \sum_i m_i (t_i - \bar{t})^2 = 0.27 * 0.0036 + 0.67 * 0.0016 + 0.067 * 0.0256 = 0.0072$$

$$\text{Standard deviation} = \sigma = \sqrt{0.0072} = .085.$$

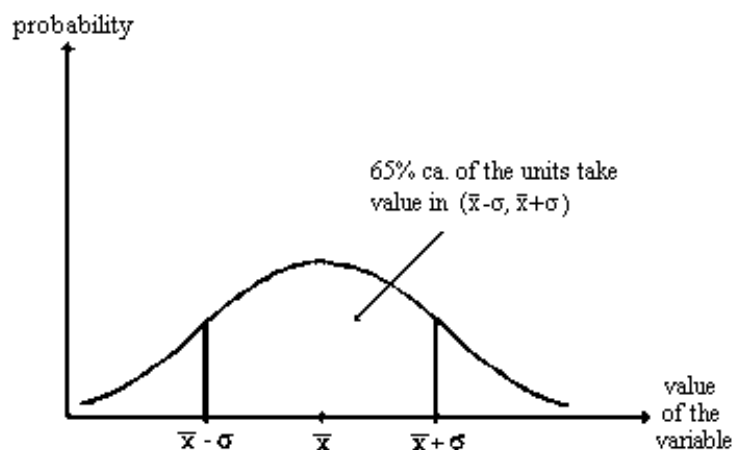


Figure 4-2 The *normal* (or *gaussian*) distribution and the meaning of σ .

The **standard deviation** σ (sigma) measures the distribution's dispersion. The figure shows the distribution of a normal variable and the meaning of the standard deviation.

Example 2 – Quantitative variables: normalisation

The table 4-6 shows the example of a town divided in five districts Q_i ($i = 1...5$), each with a population pop_i . As the total population is $pop_{tot} = 187,234$ individuals, a generic district i has a weight m_i , equal to

$$m_i = \frac{pop_i}{pop_{tot}}$$

	pop_i	m_i	$status_i$	$dipl_i$	$poor_i$	$crowd_i$
Q1	28125	.148	18.8	8.0	3.86	1.4
Q2	35853	.188	11.6	2.7	8.24	1.8
Q3	36169	.190	12.9	4.1	2.28	1.7
Q4	30329	.159	60.2	31.9	0.24	0.9
Q5	60028	.315	23.9	6.8	2.84	1.4
mean			24.52	9.69	3.49	1.45
σ			16.26	9.83	2.52	0.29

Table 4-6 Indicators of well-being in the five districts

The values of four available variables are considered jointly in order to **construct** an **indicator** of the districts **global well-being**.

status: percentage of entrepreneurs, professionals, managers and white collars over all the household heads;

high_ed: percentage of people with a (suitably defined) high educational level;

poor: percentage of poor (determined somehow) over the population;

crowd: the district's average **crowding index**, ratio between the district's population pop_i and the total number of occupied rooms.

It can be immediately observed that:

- Q4 appears to be the best district, with the highest level of *status* and *dipl* and the lowest levels of *poor* and *crowd*.
- It is difficult to appreciate directly the absolute values of the variables in the different districts. Instead, it makes more sense to compare them with the **overall mean**, in order to assess which districts assume values higher or lower than the average value characterising the overall system.
- The variables have different means and standard deviations, that are difficult to compare with one another.

If x_{ij} is the value (shown in the table) taken by the variable j in district i , let us compute the following transformation

$$y_{ij} \leftarrow x_{ij} - \bar{x}_j \quad (4.2)$$

where \bar{x}_j is the mean of variable j and y_{ij} measures the difference between the value of the variable in district i and its mean.

The (4.2) assigns to y_{ij} negative values in all districts for which x_{ij} is **below the mean** \bar{x}_j , positive values if x_{ij} is **above the mean**.

In particular, $y_{ij} = 0$ if x_{ij} is equal to the mean.

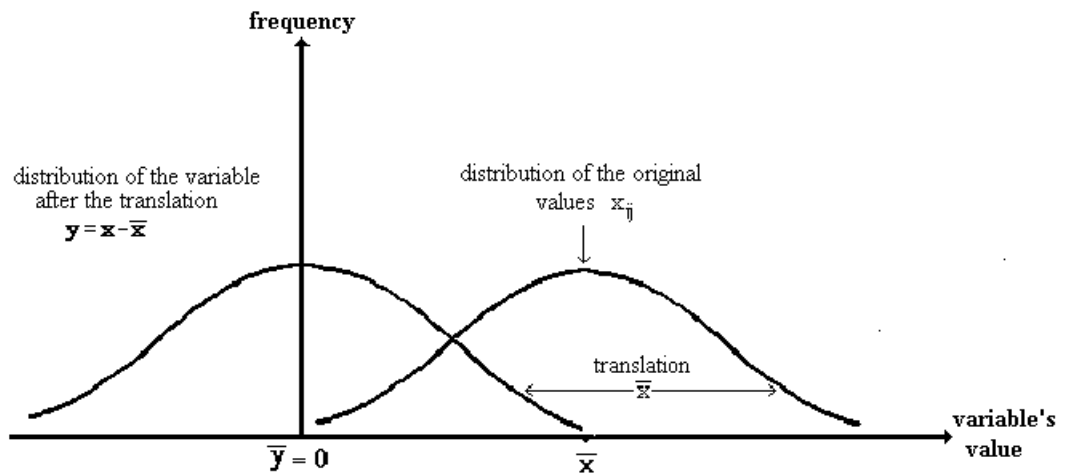


Figure 4-3 Distribution of the values of x before the translation, and distribution of the corresponding centred variable y .

The dispersion of the two variables is still the same ($\sigma(y) = \sigma(x)$), but the mean of y is zero (i.e., y is **centred** on the origin).

Now the sign of y tells us immediately whether the original value x was below or above the average.

How can we decide rapidly if a value of y (i.e., a difference with respect to the mean), is large or small? It would be necessary to compare it with the similar differences existing in the other districts, and this should be done variable by variable, as the various variables have different variances. Very long and difficult to carry out.

For example, the table 4.6 shows that the dispersion of *status* is much larger than that of *crowding*; as a consequence, a difference of 3.0 with respect to the mean is enormous for *crowding*, much less relevant for *status*.

In order to compare easily the differences of various variables with respect to their means ***it is convenient to reduce all of them to the same dispersion*** by dividing the difference y of each variable by its standard deviation σ . This affects more the differences with respect to the mean of variables with a higher dispersion (measured by σ). In other terms, the σ of each variable is used as the unit of measure for the differences.

It is easy to show that the mean of the variable z_j resulting from this operation is **0, and its variance is 1**. This fact is expressed with $z_j(0,1)$.

$$z_{ij} = \frac{y_{ij}}{\sigma_j} = \frac{x_{ij} - \bar{x}}{\sigma_j}$$

The variable z_j is said ***normalised*** or ***standardised***.

Normalised variables are immediately comparable, as they have the same mean (= 0) and the same standard deviation (= 1), whatever their original distribution.

The values z , that represent the differences from the average computed assuming σ as the unit of measure, are known as ***z-scores***.

	pop _i	p _i	status _i	high_ed _i	poor _i	crowd _i
Q1	28125	.148	-0.35	-0.17	0.15	-0.17
Q2	35853	.188	-0.79	-0.71	1.88	1.21
Q3	36169	.190	-0.71	-0.57	-0.48	0.86
Q4	30329	.159	2.19	2.26	-1.29	-1.90
Q5	60028	.315	-0.04	-0.29	-0.40	-0.17
mean			0	0	0	0
σ			1	1	1	1

Table 4-7 The values of table 4-6 converted to *z-scores*

Example 3 – Measure of the level of association between quantitative variables

The table 4-7 shows with great evidence the existence of a set of relationships among the variables:

- districts with *status* above the average (positive *z-scores*) tend to have also *high_ed* above the average, and *poor* and *crowding* below the average;
- districts with *status* below the average (negative *z-scores*) tend to have also *high_ed* below the average, and *poor* and *crowding* above the average;

There are some exceptions, but only for districts with *z-scores* next to zero, i.e. with behaviour close to the average.

Let us represent the two variables *status* and *crowding* on the two axes of a Cartesian plan: since every each is described by a pair of such values (see table 4-6), it is univocally represented by a point in the plan (*status*, *crowding*). Vice versa, the co-ordinates of every point of the plan can be assumed as the values of *status* and *crowding* for a possible district. Actually, since *status* and *crowding* cannot be negative, the location of points that make sense is limited to the first quadrant.

The figure 4-4a shows the location of the points representing our five districts: it is commonly known as a **cloud of points**, whose position on the plan is schematically given by the ellipsoid shown in the figure.

The term “*cloud of points*” is perhaps exaggerated when the points are so few, but in a real analysis they are usually much more, and the cloud is often quite dense.

The district-points are spread around the cloud’s **centre of gravity** **G**, whose co-ordinates are the average values of the variables. **G** represents the system’s central tendency, i.e. the **average behaviour** of the five districts taken as a whole.

The figure 4-4b represents the same cloud, but centred: the differences from the average have been used as co-ordinates instead of the observed values. In practice, a translation has taken the origin of the co-ordinate system to coincide with **G**. This operation has not affected the form of the cloud, or its dispersion, or the distances among the district-points, that remain the same in both cases.

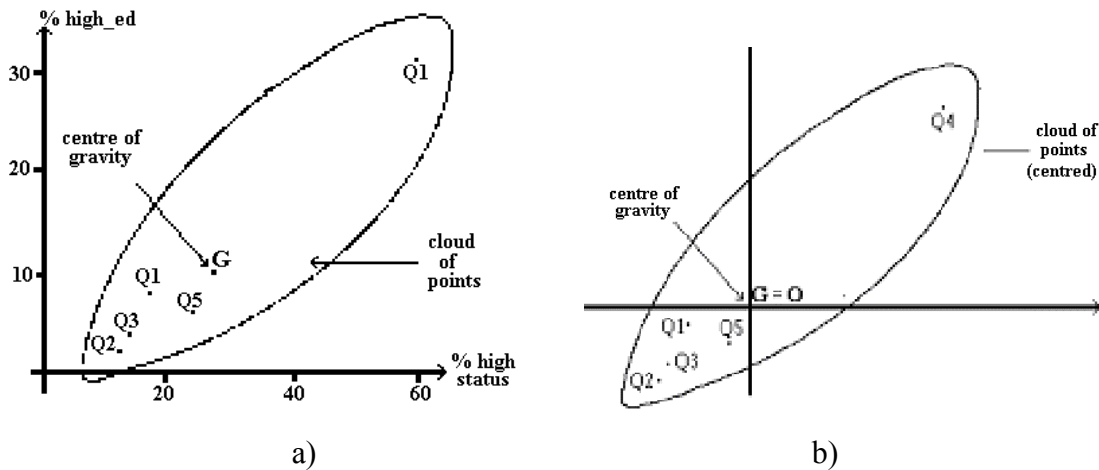


Figure 4-4 The cloud of points (4.4a), and the same cloud centred (4.4b)

Any pair of variables chosen out of the four observed ones could similarly be used.

The **covariance** of two quantitative variables x e y is a measure of the concordance in their way of varying:

$$\text{cov}(x, y) = \sum_i m_i (x_i - \bar{x})(y_i - \bar{y}) \quad (4.3)$$

Each unit i gives a *positive contribution* to the covariance when the differences of both variables from their respective average have in i the same sign (i.e., the two variables are in i both above or below their respective average, like *status* and *high_ed*). The contribution is *negative* when the signs of the differences are opposite (e.g., like *status* and *crowding*).

Notice that the contribution of each unit must be weighed with the unit’s weight m_i .

If two variables are **centred** (i.e., if $\bar{x} = 0$ and $\bar{y} = 0$), la (4.3) becomes

$$\text{cov}(x, y) = \sum_i m_i x_i y_i$$

The following particular cases can be found:

- high positive covariance: the two variables tend to take together values above or below the average, and the cloud has the form shown in fig. 4-5a.
- high negative covariance: the two variables tend to have differences from their respective average that have opposite signs (fig. 4-5b).
- covariance close to zero: there is no regular relationship between the differences of the two variables. Positive differences of one of them from its average tend to be randomly associated with differences both positive and negative of the other (fig. 4-5c).

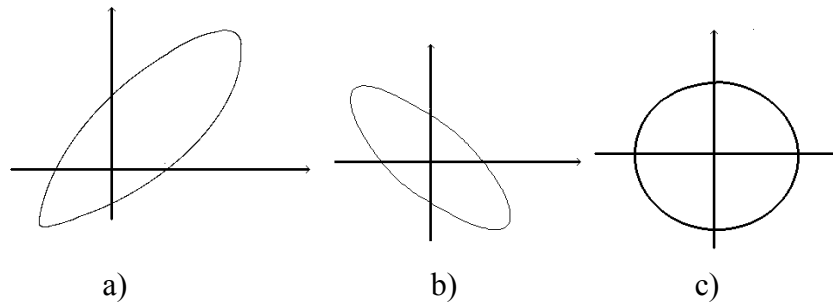


Figure 4-5 Schemes of different types of association between continuous variables

In particular:

- The covariance of a variable with itself coincides with its variance:

$$\text{cov}(x, x) = \sum_i m_i (x_i - \bar{x})(x_i - \bar{x}) = \sum_i m_i (x_i - \bar{x})^2 = \text{var}(x)$$

- The value of the covariance between two variables is not immediately interpretable. In fact, it depends on how the differences are dispersed and this is measured, for each variable, by its standard deviation σ . The differences (and therefore the covariance) can also change only because the unit of measure is changed, even though this does not affect at all the relationship between the variables.
- In order to cope with this, it is convenient to normalise both variables before computing their covariance. Doing so, the covariance is actually computed using the z-scores.

The value obtained does no longer depend on σ_x and σ_y , as the variance of both variables is 1 after normalising. It is known as the **correlation** between the variables.

$$\text{corr}(x, y) = \sum_i m_i \frac{(x_i - \bar{x})}{\sigma_x} \frac{(y_i - \bar{y})}{\sigma_y} = \frac{\text{cov}(x, y)}{\sigma_x \sigma_y}$$

It is easy to proof that $-1 \leq \text{corr} \leq 1$.

- If the correlation is close to 1 the two variables tend to take **together** values higher or respectively lower than the average.
- If the correlation is close to -1 the two variables tend to vary opposite to one another: when one of them takes values above its average, the other tends to take values below its average, and vice versa.
- If the correlation is close to zero the two variables are not significantly associated.

In the two first cases the information conveyed by one variable largely repeats the one provided by the other.

The table 4-8 shows the correlations (*1000) of the four variables of table 4-6.

	<i>Status</i>	<i>High_ed</i>	<i>Poor</i>	<i>Crowding</i>
<i>Status</i>	1000	984	-671	-948
<i>High_ed</i>	984	1000	-643	-915
<i>Poor</i>	-671	-643	1000	748
<i>Crowding</i>	-948	-915	748	1000

Table 4-8 The correlations (*1000) among the four variables.

- The matrix is symmetric: for each pair of variables (i, j) $corr(i, j) = corr(j, i)$. The correlation is *a property of the pair of variables*, measured over a given set of statistical units (the "context", consisting in our example in the five districts). *It does not depend on the order in which the variables or the units are considered*.
- The correlation of a variable with itself is obviously (shown as 1000 in the table 4-8).
- *Status* e *high_ed* have a strong positive correlation (= 0.984) and are both negatively correlated with *crowding* (correlation = -0.948 e -0.915 respectively). *Crowding* and *poor* are correlated positively, albeit less strongly (corr = 0.748).
- The strong correlation among *status*, *high_ed* and *crowding* shows that the information brought in by the first one is *almost repeated* by the others. The correlation of *poor* with the others is not so high: the information it conveys is at least partially original.

4-3 - Types of tables

Besides recognising the scale in which variables are expressed, it is also important to recognise the type of the table being analysed, as different statistical techniques are needed for different tables. We limit ourselves here to two types.

Descriptive tables

They consist of as many rows as there are statistical units (e.g., districts) and of as many columns as there are variables to be analysed. The input file can include more variables describing the same units, but not all of them may be pertinent to the particular problem; the decision about the selection is an important one. Variables refer to different aspects and may be measured on different scales; they must be reduced to a common scale before processing. A generic entry (or *cell*) of the table, at the intersection of row i and column j , represents the value assumed by the variable j in district i .

Remember: If the variables are all continuous, they are standardised and then a Principal Component Analysis (ACOMP) is run.

If they are categorical, they are all converted to binary form and the resulting table is submitted to an Analysis of Correspondences (ACORR).

Contingency (or count) tables

They are obtained by counting statistical units of the same type (individuals, families, acres...) according to the combination of the categories of two categorical variables (or more than two, if the resulting table has more than two dimensions).

Table 4-9 shows a contingency table cross-tabulating districts and crops. The count units are acres of cultivated land: each acre is added to the cell defined by the district to which the acre belongs, and the prevalent crop present in it.

The “*district*” is a nominal variable that has as many values as there are districts; each acre is assigned unequivocally to one and only one district.

	teff	mais	sorghum	other	
Area 1	40	60	50	100	250
Area 2	100	100	200	200	600
Area 3	80	120	100	200	500
	220	280	350	500	1350

Table 4-9 An example of contingency table.

The “*prevalent crop*” is also a nominal variable that , takes as many values as there are crops, each acre being assigned to the category of crop prevailing in it. A *row total* counts all the cultivated acres in a district, while a *column total* counts all the acres in the country with a given prevalent crop: these totals are known as the table's “*marginal values*”.

Of course, the table's *overall total* represents the acres of cultivated land in the whole country (irrespective of the district and the type of cultivation). *Any table for which it makes sense to consider row and column totals can be thought of as a contingency table.*

Contingency tables are analysed by means of an *Analysis of Correspondences*.

4-4 The geometrical representation

Let us consider a data table $X(n,p)$, in which the rows represent a set I of n units and the columns the values of p variables measured on those units. X is a descriptive table, and for sake of simplicity we will suppose that the variables are quantitative; however, the definitions and the concepts that will be given can be extended to any kind of table.

The behaviour of each row-unit (think again of a district) is represented by an array of p ordered real numbers (the corresponding values of the p variables). The p components of such an array can be thought as the co-ordinates of a point in a vectorial (geometrical) space \mathbf{R}^p with p dimensions; the unit can be identified with that point.

The set I of n units can be represented as a **cloud of n massive points** (remember that there is a weight attached to each unit). Every other point in \mathbf{R}^p can be thought of as a **virtual** unit (i.e., a combination of values of the p descriptive variables) that might be possibly encountered in another case, or another sample, or that can have a particular meaning for the given cloud, like its central point.

The Figure 4-6 gives an example in a simple case with two variables.

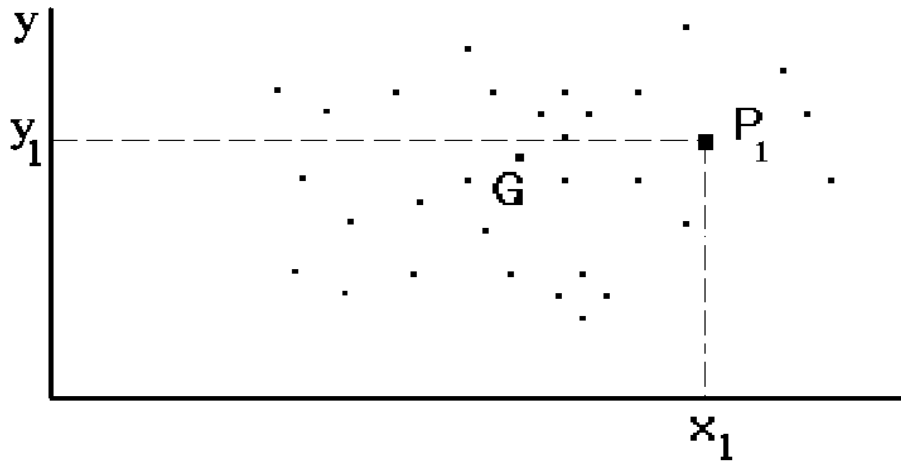


Figure 4.6 Geometrical representation of a set of units described by two variables. **G** is the cloud's centre of gravity.

In the figure each of the two orthogonal axes carries the values of a variable; each geometrical point identifies unequivocally an array of two values (i.e., a unit) and vice versa. If there are three variables, three orthogonal axes are needed to represent them and this is still easy to visualise. When there are more than three variables, our three-dimensional mind cannot visualise the picture; however, all the mathematics that can be conceived for handling the two- or three-dimensional case can be extended with no effort to the p -dimensional case.

We shall therefore consider the general case of a cloud of n object-points in \mathbb{R}^p , but you can think intuitively of the representation of a two-variable case without any loss of generality.

In a similar way, we can look at the table X by columns. Each column is an array of n numbers that represent the values assumed by a variable over the n units. It can be identified with a geometrical point in a n -dimensional space \mathbb{R}^n . In this case, we have a cloud of p variable-points in \mathbb{R}^n .

The table admits therefore two geometrical representations, respectively as a cloud of n unit-points in \mathbb{R}^p or of p variable-points in \mathbb{R}^n . As for the information contents, they are both perfectly equivalent to the numeric description given by the table X . It seems natural to focus on the cloud of n unit-points in \mathbb{R}^p for analysing the differences existing amongst the statistical units with respect to the descriptive variables, but also the other representation could be used, and actually is, when there is a good computational reason. The two spaces \mathbb{R}^p and \mathbb{R}^n are dual. It is generally convenient to study in \mathbb{R}^p the relationships among the units (e.g., two units globally similar with respect to the p variables are represented by points in \mathbb{R}^p close to one another, etc.) and focus on the cloud in \mathbb{R}^n to study the relationships amongst the variables (two uncorrelated variables are represented by points that lie in orthogonal directions with respect to the origin, while two highly correlated variables lie in directions that form a small angle, etc.).

The distance

We will assume as a ***global indicator*** of the dissimilarity between two statistical units the distance between the two points that represent them in \mathbb{R}^p : all variables contribute to its

determination. In the case of a quantitative descriptive table the *global dissimilarity* between units i and k is computed as

$$d^2(i,k) = (x_{i1}-x_{k1})^2 + \dots + (x_{ip}-x_{kp})^2$$

using the euclidean distance between point i and k in R^p (we will adopt a different metrics - i.e., *a different definition of distance* - in the case of a contingency table).

As the contributions to the distance coming from different variables are to be added up, they must be expressed in the same unit of measure, or be a-dimensional. It is also convenient to balance the contributions of different variables to the distance, so as to avoid the dominance of one or of a few of them, caused only by the unit of measure adopted. A way to obtain this is to standardise all the variables before analysing: this is performed automatically by the program. Once standardised (i.e., **centred** by subtracting from each value the variable's average, and **reduced** by dividing the values so obtained by the variable's standard deviation) The average of all variables is zero, and their variance is equal to 1.

The centre of gravity of the cloud

The centre of gravity of the cloud of unit-points in R^p is the point **G** whose co-ordinates are the average values of the p variables. It represents a virtual unit with the system's overall average behaviour. Obviously, if the variables are centred (i.e., the value of the average is 0 for all of them), the centre of the cloud coincides with the origin of the reference system (**G** \equiv **O**); the cloud is said to be **centred**.

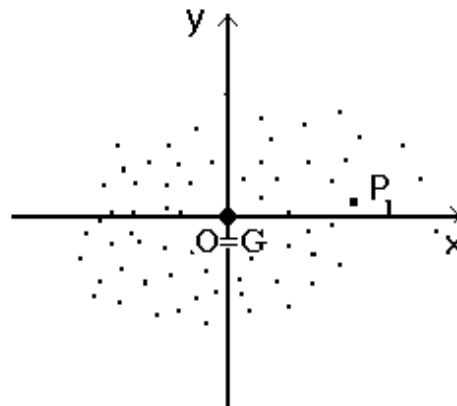


Figure 4-7 The cloud of figure 4-6 after centring

The analysis we want to perform concerns the differences existing in the behaviour of the n units, and to which variables these differences can be ascribed: from a geometrical point of view, we want to observe **how much** and **in which way** each unit differs from the average behaviour of the set I , represented by the cloud's centre **G** (or by the origin **O**, if the cloud is centred). It is reasonable to assume as an indicator of "how much" the **distance** of each unit-point from **G**, and to associate "in which way" with the direction of such elongation (i.e., to the variables which more contribute to that distance).

The Inertia of the cloud

Let us suppose the cloud to be centred. We call **Inertia of the unit i** with respect to the centre **G** \equiv **O** the product of the mass of i by the square of its distance from **O**:

$$\text{Inertia} (i) = m_i d^2(\mathbf{x}_i, \mathbf{O}) = \sum_j m_i x_{ij}^2$$

As a measure of the cloud's dispersion we assume the **total Inertia** $\text{In}_{\text{tot}}(I)$ of its points.

$$\text{In}_{\text{tot}}(I) = \sum_i m_i d^2(\mathbf{x}_i, \mathbf{O})$$

The Inertia of the cloud has a simple interpretation: it arises from the units' difference of behaviour, that is from the fact that the variables assume different values in different units, and have therefore a non-null variance in I . Were this not the case, the cloud would collapse onto its centre and its Inertia would be 0. It is easy to verify that **the total Inertia is equal to the sum of the variances of the p variables**

$$\text{In}_{\text{tot}}(I) = \sum_i m_i d^2(\mathbf{x}_i, \mathbf{O}) = \sum_i m_i (\sum_j x_{ij}^2) = \sum_j (\sum_i m_i x_{ij}^2) = \sum_j \text{var}(j)$$

If the p variables are standardised, the contribution of each of them to the Inertia is 1, and $\text{In}_{\text{tot}} = p$.

The interpretation of the relationships among the variables in R^n

In R^n every point can be interpreted as a variable, i.e. an array of n values (its coordinates) measured on the n units. If the variables are centred, it can be proved that:

- **the distance of a variable-point j from the origin is equal to the variance of that variable.** Therefore, if the variables are standardised all the points representing them lie on the surface of a hypersphere centred on the origin and of radius 1;
- the correlation between two centred variables j and k is equal to the cosine of the angle formed by the segments that join their representative points with the origin. Remember that the correlation of two variables is a measure of the strength of their association over the set I : it varies between +1 (perfect positive association) and -1 (perfect negative association). If the variables are standardised, two variables having correlation +1 are represented by coincident points; two variables having correlation -1 are represented by points opposite with respect to the origin.

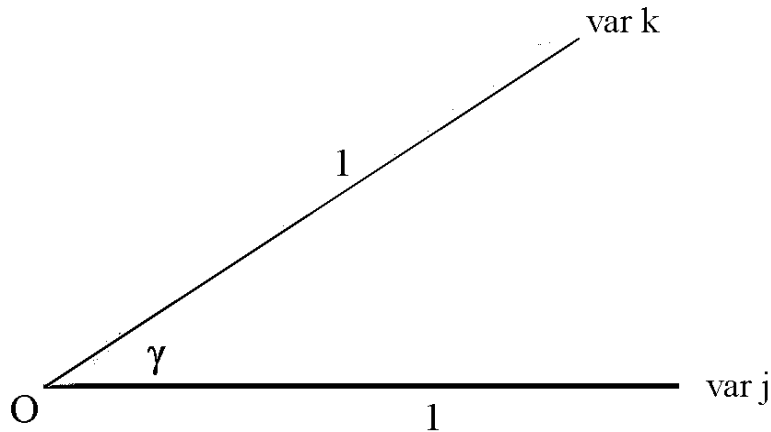


Figure 4-8 Representation of variable-points in R^n . It can be proved that $\cos \gamma = \text{corr}(\text{var } j, \text{var } k)$

i.e., the cosine of the angular distance between the two variable points j and k measures the correlation between the two variables.

The properties of the two clouds (statistical units, variables) are therefore different: if the variables are centred, the origin in R^p is the centre of the cloud of the units. They are scattered about it, the level of the dispersion being measured by the cloud's Inertia. If the cloud is projected onto any axis passing through the origin - not necessarily a co-ordinate axis - the resulting uni-dimensional cloud is also centred (see figure 4-9).

In the other space, as the angular distance between two points is related to the correlation of the variables they represent, the cloud is unevenly distributed around the origin: if all variables are highly positively correlated, the cloud lies on the same side of O , without any symmetry. This difference, that affects the interpretation of the analytical results in the two spaces, is a consequence of the different meaning of the rows and columns of the table and of the non-symmetry of the treatment to which the table is submitted (the average is computed by column and not by row, columns are standardised, etc.).

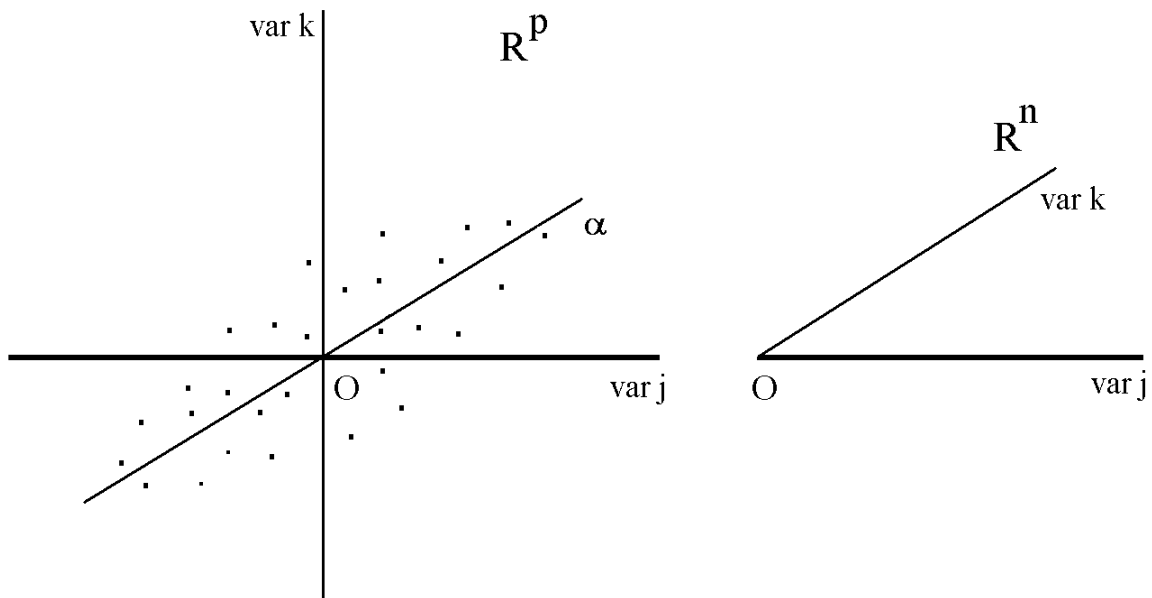


Figure 4-9 The cloud in the two spaces R^p and R^n . In R^p the cloud is centred and its projection onto any axis passing through O (e.g., the axis α) is also centred. The two variables j and k are highly correlated. This can be seen from the cloud's shape in R^p and from the small angle between the two variable-points in R^n .

Chapter 5 – Distributions and Cross-tabulations

Programmes DISTRIB and CROSSTAB load their control parameters (i.e., the statements that control the way they execute) from files DISTRIB.PAR and CROSS.PAR respectively. Templates of these files are provided in the ADDATI folder.

The user should copy the .PAR file to the working directory where the data to be processed reside, then use the ADDATI editor to modify its contents adapting it to the analysis to be carried out. The instructions that explain how to do it are included as comments in the PAR file itself.

When DISTRIB (or CROSSTAB) is selected in the Analysis Menu, ADDATI checks if a file named DISTRIB.PAR (or CROSS.PAR) already exists in the working directory. If it does, the file is opened (on user's request) in the editor and the user can modify it. If it does not exist, the prototype present in the ADDATI folder is copied to the working directory and then opened in the editor.

- **DISTRIB** works in batch mode: after the user has suitably filled the parameter file DISTRIB.PAR, the programme is run. It loads the PAR file, and if it finds any error it displays a diagnostic window and the PAR file is automatically edited, highlighting the wrong line (if a single line is responsible of the error). The user corrects the error, saves the file and DISTRIB is automatically re-run. When no error is found in the parameter file, the programme computes the requested distributions and saves them to an output text file named DISTRnn.OUT. 'nn' is a two-digit progressive number chosen by the programme to avoid overwriting files that already exist. The user can then inspect the output file using the "**Edit/View a text file**" option from the FILE menu.
- Also **CROSSTAB** reads a parameter file (CROSS.PAR), that specifies which is the input file, which variables are to be loaded, their features, etc. Once loaded the variables, the execution continues interactively: the user decides which variables to cross-tabulate, which segments of the overall population to consider (filtering on some of the variables), etc. The results are saved to a file named CROSSnn.OUT.

The *reading format* to be supplied to **DISTRIB** or **CROSSTAB** follows the same syntax as the other programmes in ADDATI (see chapter 3).

5-1 DISTRIB

Distributions of categorical and continuous variables.

Use	Computes the distribution of a set of categorical or continuous variables. For each categorical variable, the frequency of its categories is computed. For continuous variables the mean and the standard deviations are computed. In addition, the frequency of classes created either automatically, or using delimiting thresholds entered by the user, is computed.
Limits	Maximum input record length: 5120 characters.

The file DISTRIB.PAR, from which **DISTRIB** loads the parameter controlling the execution, **must reside in the working directory**.

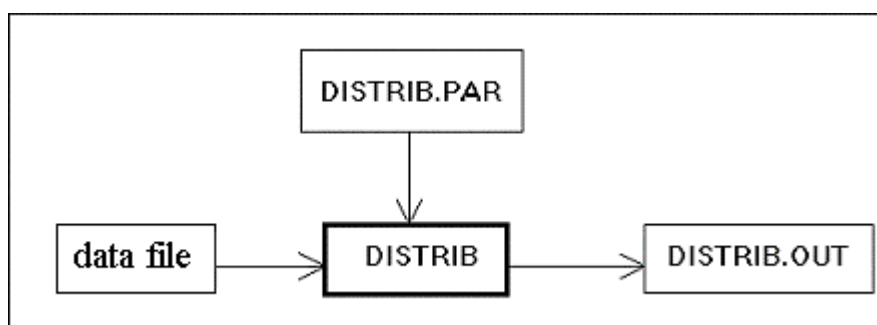


Figure 5-1 The files read and written by DISTRIB.

DISTRIB carries out the following operations:

- it computes the distribution of the statistical units over the categories of the qualitative variables loaded from the input data file;
- it computes the average, the standard deviation, the minimum and maximum value of the quantitative variables;
- it splits the variability range of each quantitative variable into a user-chosen number of segments of equal width, and determines how many statistical units fall in each of them, drawing the related histogram.

This last information is particularly useful in order to decide how to recode a **continuous variable** in a suitable number of classes, to later process it together with other variables that are categorical.

If the units are almost homogeneously distributed over the variable's range, it is sufficient, to prepare a recoding to be carried out with the programme RECODE include in the Utility Menu, to ask for a limited number of classes.

On the contrary, when the values of a variable are quite concentrated and few classes are requested, it may occur that most statistical units are included just in one of them, and the others are almost empty. The recoding operation would then result in a relevant loss of information. For this type of highly polarised variables it is convenient to request a very detailed distribution (i.e., a high number of classes) and decide only later, after inspecting the distribution obtained, which is the best way of recoding the variable.

The structure of the file DISTRIB.PAR

DISTRIB.PAR is self-illustrated. The file includes some comment lines that begin with a question mark "?", and others *active* lines that must be filled with the control parameters.

The comment lines explain how the others line must be filled to prepare the file for a particular analysis. **DISTRIB ignores the comment lines** and reads from the active ones the information necessary to execute.

At the beginning, it is quite easy to make some mistakes: the programme signals them with clear messages, to help correct them. So no hurry, and due attention when filling the parameter file.

A sample DISTRIB.PAR file

```
? This is the control file for DISTRIB.
? All records beginning with '?' are comments. They are ignored by the programme;
? their purpose is only to explain how to fill the records that do not begin with '?',
? that are actually loaded by the program.
? Each active line begins with a keyword (like CASES, DATA_FILE, TITLE, etc.).
? Do not modify the keywords, only change the values (parameters) following them to
? fit them to the particular analysis to be carried out.
?
? 1. Number of cases.
? This number specifies how many cases (records) must be loaded from the input file.
? To load them all, enter "ALL" instead of a number
CASES  ALL
?
? 2. Number of variables.
? This number specifies how many variables are to be actually loaded from the input file,
? and processed. It excludes those to be skipped when reading the input file (this is done
? by providing a suitable "reading format", see further on), and the weight (if any) to be
? assigned to a case.
VARIABLES 3
?
? 3. Weight to be assigned to each case.
? The WEIGHT keyword is followed by '0' to assign all cases the same weight,
? '1' otherwise.
? In this latter case, the weight must be the first field in the record if a free reading format
? is used (see below); it can be any field if its location is suitably specified in the format.
? Details about the way to enter a format can be found in the Manual, or in the online help
? available for multivariate programmes.
WEIGHT 0
?
? 5. TITLE of this analysis, in one line, delimited by double quotes.
? If you wish to omit a title, insert nothing after the "TITLE" keyword:
TITLE "Distribution of some Census variables"
?
? 6. Name of the data (input) file (provide full path)
DATA_FILE Census.dat
?
? 7. (and following) For each variable actually loaded, in the order they are read from
? the input file, enter the following information:
```

? **If the variable is categorical (qualitative)** (represented by integer consecutive codes,
 ? starting from 1):

? • **one record** headed by the keyword "**VARIABLE**", followed by the **name of the**
 ? **variable** (max. 35 character) and by the **number of its categories**. Use the space as
 ? separator.

? • **as many records as necessary**, each headed by the keyword "**CATEGORIES**". They
 ? must list the labels of all the variable's categories, in their order (max. 20 characters
 ? for each label). Split the labels in records as you like, using spaces to separate them.
 ? When all the categories have been listed, start a new '**VARIABLE**' line to specify
 ? another variable.

? **If the variable is quantitative (continuous):**

? • **one record**, headed by the keyword "**VARIABLE**" followed by the **variable's name**
 ? (max. 35 characters) and by '**0**', (**or nothing**) to mean that the variable is quantitative.

? • **one record** that controls the way the variable's distribution is described (as the
 ? variable is quantitative, it has no categories).
 ? There are two ways to do it: the user can request a certain number of classes, and
 ? let the programme split the variable's range, or the user can explicitly enter the
 ? thresholds that delimit the classes to be described.

? A record like "**CLASSES 6**"
 ? forces the programme, once the variable's minimum and maximum values have been
 ? determined, to split the variability range in intervals of **equal width** and count the
 ? number of statistical units in each of them.

? In alternative, a record like: "**THRESHOLDS s1 s2 s3 s4**"
 ? commands the programme to split the variability range using the threshold values
 ? supplied.

? In the example above, the four thresholds determine five classes, respectively consisting
 ? consisting of:

? • the units for which the value of the variable is <s1;
 ? • the units with values >=s1 and < s2;
 ? • the units with values >=s2 and < s3;
 ? • the units with values >=s3 and < s4;
 ? • the units for which the variable take values >=s4.

? Notice that each threshold value is assumed to belong to the upper class

? **WARNING**

? If the label of a variable or a category has some internal spaces, it must be included
 ? in double quotes to force the reading routines to interpret it correctly (see some example
 ? here below).

VARIABLE	LOCALITY	3
CATEGORIES	town village sparse	
VARIABLE	"HOUSEHOLD COMPONENTS"	0
THRESHOLDS	1 2 3 4 5 9	
VARIABLE	"Age of household head"	0
CLASSES	6	

? **8. The data reading format.**

? The **format** is a string, included in double quotes, used to specify which variables, among all those included in the record, are actually to be processed, and which to be ignored.

? It is also used to specify the place of the weight in the record, if a weight is expected.

? The **format syntax** follows the usual ADDATI conventions (see chp. 3, or the online help available for the multivariate analysis programmes).

? If a **free format** is specified, each input record is expected to contain exactly, in the order:

? • **the weight** of the statistical units to which the records refers (if "**WEIGHT** 1" was specified in the **WEIGHT** statement);

? • **all and only the variables** to be read, separated by spaces;

? In such case, it is sufficient to enter an asterisk "*" instead of a complete format.

? The following format sample statement means that for each record:

? • the first three characters must be skipped (3x);

? • the value of the first variable (*locality*) takes the two characters that follow;

? • the second variable (*household components*) stays in the three following characters;

? • two more characters are skipped;

? • the third variable (*HH age*) is eventually loaded from the three following characters.

FORMAT "(3x, 2, 41x, 3, 2x, 3)"

5-2 CROSSTAB

Cross-tabulations of categorical variables

Use	<p>Pairs of qualitative variables are cross-tabulated, and the frequency of each combination of categories of the two variables is computed.</p> <p>The contingency tables obtained by cross-tabulating are saved to file. For each cell, the percentage computed over the row and column totals are also computed. The value of the chi-square (χ^2) is computed to measure the strength of the association among the two variables.</p> <p>It is possible to limit the operation to some particular segments of statistical units, determined by filtering on the values of some variables (both quantitative and qualitative).</p> <p>Some general information necessary to execute (the datafile name, the names of the variables, their place in the record, etc.) are loaded from the parameter file <CROSS.PAR>, looked for in the working directory. This file must be filled by the user, who will adapt the template existing in the ADDATI installation folder. Thereafter, the execution continues interactively.</p>
Limits	The maximum input record length is 5120 bytes.
N.B.	It is always assumed that a variable with n categories takes the values 1...n.

CROSSTAB is quite simple to use. With the ADDATI editor the file **CROSS.PAR** is filled, that contains the control parameters. When **CROSSTAB** is run from the Analysis menu, the copy of **CROSS.PAR** (if any) existing in the working directory is opened on user's request. If the file is not found, the template residing in the ADDATI folder is copied.

Any error encountered in **CROSS.PAR** is detected and corrected, exactly like with **DISTRIB**. Then the program interactively prompts the user on the cross-tabulations to be computed (fig. 5-3). The name of the datafile, from which the variables are loaded, is specified in the file **CROSS.PAR**. Results are saved to a file named **CROSSnn.OUT**, that can be loaded into the editor **addaedit** to carry out the interpretation.

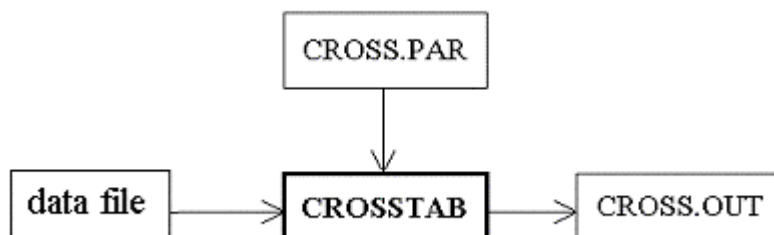


Figure 5-2 The files read and written by **CROSSTAB**.

In addition to computing the cross-tabulations, **CROSSTAB** also:

- checks if the variables have any value out of range. If some categorical variable has a value 0 (often indicating a missing value), or a value greater than n (because of data entering error), the record is excluded and its ordinal number is listed in **CROSSnn.OUT**. The user can then inspect the errors (using the editor or **SHOWREC**) and correct them.

- enables the user to select a **sub-population** (or **segment**), made up by all the statistical units that take particular combinations of values in some variables (**filter variables**): the computed cross-tabulations can then be limited to the segment. In order to select a segment, **four filter types** are available, that allow the user to choose:
 - all and only the units for which a filter variable takes a value **greater than** a given one: for example, “> 3” accepts all values greater than 3, excluding the others;
 - all and only the units for which a filter variable takes a value **less than** a given one: for example, “< 3” accepts all values less than 3, excluding the others;
 - all the units for which a filter variable takes exactly a value indicated: for example, “4” selects the units for which the value of the filter variable is 4;
 - all the units for which the filter variable takes a value **between** two values indicated;
- this window displays
the variables loaded

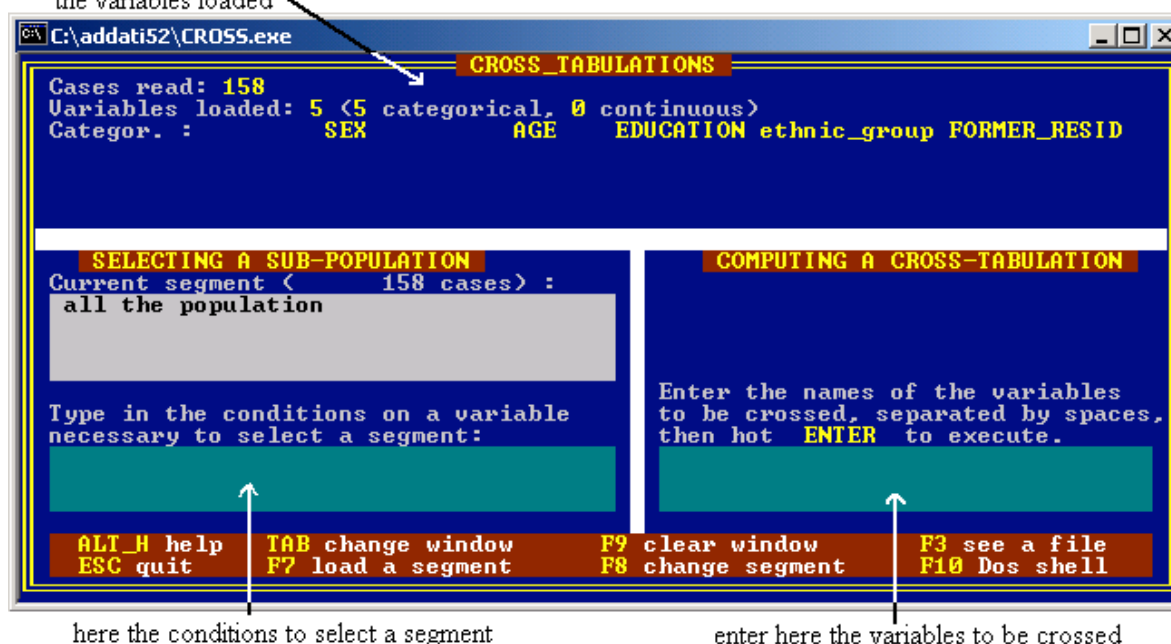


Figure 5-3 The **CROSSTAB** screen.

Also quantitative variables can be loaded, but *they can be only used for filtering*: being not categorical, they cannot be cross-tabulated (they could, if they were previously recoded into classes using **RECODE**). For example, it is possible to select all household more than 45 years old, or all dwellings with a surface less than 100 sqm, etc.

Up to four variables can be used for filtering, *then limiting the computed cross-tabulations to the segment obtained*.

For example, the focus can be in the relationship between *the tenure* and *the level of the dwelling facilities*, with the purpose to assess whether the dwelling conditions be comparable for owners and renters: it is sufficient to cross-tabulate *tenure with facilities*. Anyway, the relationship might depend on the *locality*, resulting different for central or sparse dwellings. To answer this question, dwellings located in the centre can be extracted, and *only for them* tenure and facilities are cross-tabulated; the same is done with sparse dwellings, and *the two tables obtained are then compared*.

Though stemming from a starting set of hypotheses, a research usually evolves along an operational path determined by *an accumulation of intermediate steps*, each orienting those that follow. It is therefore convenient not to cross-tabulate indiscriminately all the available variables, thus producing a high number of tables, difficult to read.

Instead, it is better to use one's intelligence to formulate, based on a initial set of objectives and hypotheses, an appropriate **cross-tabulation design**, aimed at inspecting the most relevant aspects, the most consistent associations among the variables considered, etc. Analysing partial results the research design can be updated, or integrated with the use of suitable filters.

The distribution of the analysed variables must be taken into account. After singling out groups of particularly associated variables, the natural follow-up is a multivariate analysis that simultaneously processes several variables, not only two at a time. More on this in chapters 6 and 7.

The natural analytical sequence is therefore **DISTRIB - CROSSTAB- Multivariate Analysis**.

An example, just to introduce some theoretical considerations

Table 5.1 shows the result of a cross-tabulation between the *household size* (qualitative variable with four categories) and the *HH age* (three categories), for all the household residing in the historical Centre of Venice, Italy, at the date of the 1981 Census.

Household size (HS1-4): 1, 3, 3, 4 or more persons;

Age of Household Head (HHA1-3) : <35 years, 35-55 years, >55 years.

	HHA1 < 35 years	HHA2 35-55 years	HHA3 > 55 years	total
HS1 (1 pers.)	1341	2852	9689	13882
HS2 (2 pers.)	1680	2924	3521	8125
HS3 (3 pers.)	1325	3693	1749	6767
HS4 (>3 pers.)	1066	2526	1628	5220
total	5412	11995	16587	33994

Table 5-1 All households in Venice, Historical Centre, Census 1981. Cross-tabulation between the household size and the HH age

The table 5-1 is a *contingency table* (or *table of frequencies*): it shows the frequency of each combination of categories of the two variables. For example,

- the cell (1,1) shows that there are 1341 households with only one person (obviously, the HH) less than 35 years old.
- the cell (4,3) shows that there are 1628 households of four persons or more, whose head is older than 55 years.

The last row and the last column are the table's **marginal distributions**. They give the distribution of the two variables, i.e. the frequencies of their categories.

The units included in a marginal value can be distributed in various ways over the cells lying in the corresponding row or column. The number of the cells whose value can be set arbitrarily, only subject to the constraints imposed by the row or column total, is the table's **degrees of freedom**. It is easy to see that a table with m rows and n columns has $(m-1)(n-1)$ degrees of freedom.

The contingency table can be converted into a **table of percentages** by dividing all its values by the total number of the statistical units (that are 33994 in our example).

	HHA1 < 35 years	HHA2 35-55 years	HHA3 > 55 years	f_i
HS1 (1 pers.)	0.039	0.084	0.285	0.408
HS2 (2 pers.)	0.049	0.086	0.104	0.239
HS3 (3 pers.)	0.039	0.109	0.051	0.199
HS4 (>3 pers.)	0.031	0.74	0.048	0.154
f_j	0.159	0.353	0.488	1.0

Table 5-2 The *observed* percentage frequencies.

Each cell shows the *percentage of households* in which the particular combination of categories that characterises the cell occurs.

Example *The households of one person and HH age <35 years are 3.9% of the total.*

Each marginal value gives the frequency of a category of one of the variables, *regardless of the value taken by the other*.

Example *The households consisting of one person, irrespective of the HH age, amount to 40.8% of all households; the households whose HH is less than 35 years old amount to 15.9%.*

We wish to assess if some particular categories of the two variables are *particularly associated*.

- If we observe that a household consists of one person only, *does this increase our expectation* that its HH be old?
- Does the observed fact that a household consists of more than three persons make *less likely* the chance that its HH be old?

The comparison is obviously with the frequencies *we would expect for the second variable, had we not observed the value of the first one*.

In other terms: does the knowledge of the value taken by one variable *provide some information* on the distribution of the other?

To answer this question, we must compute the cell values we would expect *if the two variables were independent*.

By '*independent*' we mean that the frequencies of the categories of one variable *remain significantly the same* whatever the value taken by the other. For example, if our two variables were independent the percentage of old HHs should be 48.8% and that of young HHs 15.9%, *whatever the number of persons* in the household, as these are the frequencies observed for the overall population of households. On the contrary, if this does not occur the variables are somehow associated: the higher their level of association, the more different the observed table and the one expected under the hypothesis of independence.

We must therefore compute the table expected if the variables were independent, and compare it with the observed one. To do this, we need to define a suitable *indicator of the overall distance between the two tables*.

The table expected under the independence hypothesis

Let us take as example the cell (1,1).

The frequency of one-person households is $f_{1.} = 0.408$ (i.e., 40.8% of all households). For how many of them will the HH be younger than 35 years? ***If the two variables were independent***, the probability of a young HH ***should be the same*** that characterises the whole population, i.e. $f_{.1} = 0.159$. That is, the one-person households with a young HH should amount to 15.9% of 40.8% of all the households. If f_{11}^e is the expected frequency in the first cell ('e' stands for '***expected***'), then

$$f_{11}^e = f_{1.} * f_{.1} = 0.408 * 0.159 = 0.068$$

which is significantly greater than the ***observed*** frequency 0.039 shown in the table 5-2.

Given the marginal distributions of the variables, the one-person households with a young HH, that should be 6.8% under the independence hypothesis, are actually only 3.9%. The two categories (1 person, <35 years) result to be less associated than expected, and the two variables do not seem to be independent. Then, if an association exists, how strong is it?

In general, the ***expected*** frequency in the cell (i,j) is $f_{ij}^e = f_{i.} * f_{.j}$, equal to the product of the two marginal frequencies. It must be compared with the empirically observed frequency f_{ij}^o .

	HHA1 < 35 years	HHA2 35-55 years	HHA3 > 55 years	$f_{i.}$
HS1 (1 pers.)	0.068	0.144	0.199	0.408
HS2 (2 pers.)	0.038	0.084	0.117	0.239
HS3 (3 pers.)	0.032	0.070	0.097	0.199
HS4 (>3 pers.)	0.024	0.054	0.075	0.154
$f_{.j}$	0.159	0.353	0.488	1.0

Table 5-3 Table (computed) of the ***expected*** frequencies $f_{ij}^e = f_{i.} * f_{.j}$

An indicator of the overall difference between the observed and expected tables

Such an indicator should

- take into account the differences in all the cells;
- account for the fact that a same absolute difference *is less significant* when the expected frequency is higher. Consider for example two cases, with expected and observed frequencies equal to 3% and 6% in the former, 50% and 53% in the latter. The absolute difference is 3% in both cases, but taken in percentage terms the former case is certainly more significant.

The value $\frac{(f_{ij}^o - f_{ij}^e)^2}{f_{ij}^e}$ measures *the contribution of a generic cell (i,j)* to the overall difference indicator between the two tables. This indicator, that is called **chi-square**, is the sum of the contributions coming from all individual cells, multiplied by the number N of the statistical units:

$$\chi^2 = N * \sum_{ij} \frac{(f_{ij}^o - f_{ij}^e)^2}{f_{ij}^e} \quad (\text{the sum is over all values of } i \text{ and } j).$$

The square enhances the importance of higher differences and makes all cell contributions positive, whatever the sign of the $f_{ij}^o - f_{ij}^e$ difference.

The chi-square is never negative; it is null only when **observed and expected tables exactly coincide**, i.e. when $f_{ij}^o = f_{ij}^e$ for each cell.

Given a contingency table, the value of its χ^2 (together with its degrees of freedom) allows to compute, by comparison with the χ^2 distribution table, the probability that the two cross-tabulated variables be independent.

Remember After computing a cross-tabulation, examine the probability of independence between the two variables. Only when the association is strong enough it is worth interpreting the table.

The table 5-4 shows the result of a cross-tabulation as **CROSSTAB** displays it. Each cell (i,j) contains three values:

- the **absolute number** x_{ij} of the statistical units in the cell;
- the **percentage over the row total** $x_{ij}/x_{i\cdot}$, where $x_{i\cdot}$ is the total of row i;
- the **percentage over the column total** $x_{ij}/x_{\cdot j}$, where $x_{\cdot j}$ is the total of column j;

The interpretation is done by comparing the percentage distributions in each row (or column) with the corresponding marginal distribution.

	HHA1 (< 35 years)	HHA2 (35-55 years)	HHA3 (> 55 years)	
HS1 (1 pers.)	1341	2852	9689	13882
	9.7	20.5	69.8	100.0
	24.8	23.8	56.4	40.8
HS2 (2 pers.)	1680	2924	3521	8125
	20.7	36.0	43.3	100.0
	31.0	24.4	21.2	23.9
HS3 (3 pers.)	1325	3693	1749	6767
	19.6	54.6	25.8	100.0
	24.5	30.8	10.5	19.9
HS4 (>3 pers.)	1066	2526	1628	5220
	20.4	48.4	31.2	100.0
	19.7	21.1	9.8	15.4
	5412	11995	16587	33994
	15.9	35.3	48.8	100.0
	100.0	100.0	100.0	100.0

Table 5-4 A cross-tabulation table issued by **CROSSTAB**. The first value in each cell represents the number of the statistical units assigned to it; the second is the percentage over the row total; the third is the percentage over the column total.

Note: *Sample and Universe – The meaning of the χ^2*

When dealing with a sample, the χ^2 help evaluating the probability that the difference between the observed table and the one expected under the independence hypothesis, be mere consequences of casual fluctuations affecting the sample choice.

*When dealing with the whole population (the **universe**), as is often the case, we should not talk of **probability the variables to be independent** (in the universe). In fact, as there is no sample, there is no uncertainty related to its extraction, and no inference to be derived: either the observed table coincides with the expected one (and the two variables are independent), or it does not. In the real world the two tables will almost never be exactly coincident. Anyway, the differences can be due to the particular and unexpected behaviour of few peculiar units: if they are very little, they can be considered not significant.*

What we want to understand is if the apparent association between the two variables is caused by some structural reasons, that are worth investigating, or it merely derives from the casual behaviour of a few units.

*We can think of our population as of a sample extracted from some hypothetical over-universe, in **which the two cross-tabulated variables maintain their marginal distributions**.*

*We can then have recourse to the χ^2 to decide if the global difference between our **observed table** (i.e., the cross-tabulation we computed in our universe, now seen as a sample) and the **expected** one (i.e., the table we would observed in our over-universe under the independence hypothesis) can be simply attributed to fluctuation and has presumably no structural meaning.*

Similar to DISTRIB.PAR, this file specifies the variables that **CROSSTAB** should load, their place in the record, the label of the categories, etc. Yet, differently from what **DISTRIB** does, the execution continues interactively: the user is prompted about the cross-tabulations to be done, with which filters, etc. Results can be inspected *without quitting* the program, thus deciding the best way to continue.

A sample CROSS.PAR file

```
? This is the control file for CROSSTAB.
? All records beginning with '?' are comments. They are ignored by the programme; their
? purpose is only to explain how to fill the records that do not begin with '?', that are those
? actually loaded by the program.
? Each active line begins with a keyword (like DATA_FILE, TITLE, etc.).
? Do not modify the keywords, only change the values (parameters) following them to
? fit them to the particular analysis to be carried out.
? 1. Name of the input data file (provide full path)
DATA_FILE      omoratte.dat
? 2. A TITLE for this analysis, in one line, delimited by double quotes. If you wish to omit
? the title, insert nothing after the "TITLE" keyword.
TITLE      "A cross-tabulation exercise on Omoratte data"
? 3. Number of the variables to be loaded
? Some variables must be read in, to be cross-tabulated or used as filters to delimit some
? population segment.
? Only categorical variables can be cross-tabulated, while those used as filters
? (i.e., to select a sub-population or segment) can also be quantitative.
? Remember!
? The programme expects that a categorical variable with n categories takes the integer
? values 1...n, while '0' or blank (if the file is fix-formatted) are interpreted as missing
? values.
? Enter the total number of the variables to be actually loaded:
N_VARIABLES      5
? 4. Labels of the variables and their categories
? As many blocks of lines must now be provided as there are variables, in the same order the
? variables have in the data file.
? For each variable please enter:
?   • one line headed by the keyword "VARIABLE" keyword, followed by the
?     variable's name (up to 12 caratteri) and, if the variable is categorical, by the
?     number of its categories.
?     If the variable is quantitative, enter "0" as the number of its categories, or
?     just leave empty the space after the variable's label.
```

? • **only if the variable is categorical**, one or more line, each headed by the keyword
 ? "**CATEGORIES**", followed by the labels (names) of the categories in due order.
 ? Use as many lines as necessary to provide all the category labels, then start a new
 ? VARIABLE line to start a block relative to another variable.
 ? **if the variable is quantitative**, only the "**VARIABLE**" line is expected.
 ? **WARNING**
 ? If the label of a variable or a category has some internal spaces, it must be included
 ? in double quotes to force the reading routines to interpret it correctly (see some example
 ? here below).
 ? **VARIABLE** SEX 2
 ? **CATEGORIES** male female
 ? **VARIABLE** AGE 3
 ? **CATEGORIES** <27 27-32 >32
 ? **VARIABLE** EDUCATION 3
 ? **CATEGORIES** illiterate primary higher
 ? **VARIABLE** "ETHNIC GROUP" 2
 ? **CATEGORIES** amhara other
 ? **VARIABLE** "FORMER RESIDENCE" 2
 ? **CATEGORIES** rural urban
 ? **5. Data reading format.**
 ? The use of the format has already been explained in section 5.1.
 ? The format specified below means that the five variables take two bytes each in the input
 ? records.
FORMAT (5*2)

Chapter 6 – The Analysis Menu: Factorial Analyses

6-1 TYPOLOG

Converting categorical variables to binary form and enumerating types

Use	<p>Used to convert to binary form a table of categorical variables (see section 4.1), preparing it to be submitted to an Analysis of Correspondences (ACORR).</p> <p>It is possible simply to convert unit by unit, or to aggregate the statistical units in typologies: all the units for which the variables chosen by the user as active take the same values are assigned to a typology.</p> <p>When the number of the elementary units is very high, their aggregation into typologies can significantly reduce the number of the cases (the typologies, no longer the initial units) submitted to ACORR.</p>
Limits	<p>The maximum length of an input record is 5120 characters.</p>
Advice	<p>There is no theoretical limit to the number of the elementary units, nor to the number of the typologies that can be built. Nevertheless, it must be kept in mind that a too high number of active variables – or variables with too many categories - can result in an excessive number of typologies, prohibitive for ACORR or NONGER.</p> <p>The beginner often indulges the temptation to use, as active or supplementary, all the available variables. This can result in computationally heavy analyses, whose results are often confused and difficult to interpret. It must be kept in mind that the number of the variables actually submitted to ACORR, and included in the profiles written by NONGER, is equal to the <i>sum of the categories</i> of the variables submitted to TYPOL. The purpose of an exploratory analysis is to extract the relevant information from a large data set, impossible to examine by direct inspection. Building tables much more huge than necessary should not frustrate this purpose: the analyst's skill reveals itself also in the construction of <i>essential tables</i>, often obtained through repeated trials.</p>

TYPOL is the first program to be run in an analysis when the variables are categorical (qualitative): it is followed by an Analysis of the Correspondences (**ACORR**) that processes the resulting table, then by the clustering sequence.

The standard input for an Analysis of Correspondences is a table obtained by drawing side by side one or several contingency tables, in which elementary units of a certain type (households, dwellings, firms, etc.) are counted.

When working with disaggregated data, however, a record often describes an elementary unit by means of *qualitative* variables. In such a case, the variables must be converted to

complete disjunctive form (a column for each category: the value is 1 if the unit takes that category, 0 otherwise) and the resulting binary table is submitted to a Correspondence Analysis (known in this case as "Multiple Correspondence Analysis").

Note: Here the term **binary** refers to the type of coding and means only that the table consists of values '0' and '1'. The file is a text file.

The transformation to binary form is needed because **ACORR** cannot deal directly with qualitative variables, for which codes have in general no numeric meaning. On the contrary, it can be proved that a table of zeros and ones, meaning respectively the absence and the presence of a given character, can be dealt with numerically.

The input variables can be **active** or **supplementary**: **TPOLOG** computes all the typologies (or *types*) that can be obtained by combining in any possible way the values of the **active** variables and assigns each individual unit to its type. The table to be dealt with by **ACORR** is saved to file: each record (or line of the table) represents a typology **weighted with the number of elementary units** assigned to it.

All units assigned to the same type take by definition the same values of all the active variables, while they can distribute differently over the categories of the supplementary variables (if any).

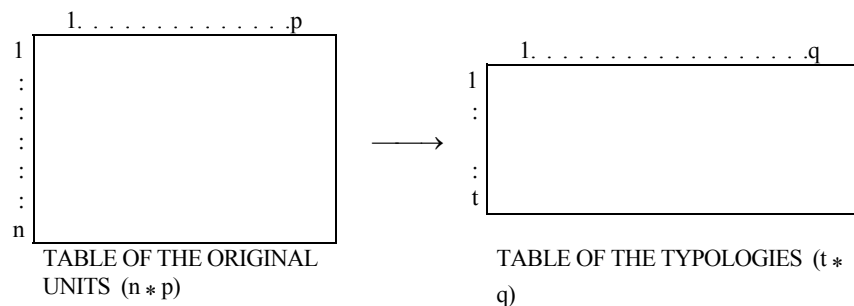


Figure 6-1 **TPOLOG** converts the initial table $n \times p$, which describes n statistical units by means of p categorical variables, into a table $t \times q$ that describes the t actually encountered typologies. The number q of columns is equal to the sum of the categories of the descriptive variables.

Figure 6-1 shows the scheme of the transformation. It can be proved that an Analysis of Correspondences performed on the typologies table gives the same results as one performed on the table of the elementary unit (in binary form). Anyway, the former offers a noteworthy computational advantage, especially in the classification stage.

All the variables in the data table input by **TPOLOG** must be **categorical**. If any of them is quantitative, it must be converted to categorical form (by means of the **RECODE** program in the utility menu) before running **TPOLOG** to convert it from categorical to binary form. For more details about **RECODE** see section 2-7.

TPOLOG can be used in two ways:

- only to convert a table of categorical variables to binary form: all records are read and converted one after another.
- to assign each unit to an appropriate type: a data reduction is achieved by submitting the types (and not the initial statistical units) to the clustering procedure that follows. The table of types, output by **TPOLOG** and passed over to **ACORR**, is in binary form.

This second way of using the program is by far the more interesting when it is a question of analysing a huge table, describing a large number of individual units (households, dwellings, firms, etc.): this is the appropriate case when elementary Census or Survey data are processed.

***Note:** TYPOLOG was used (on a PC!) to analyse the 237,917 household records of the Addis Ababa 1984 Census, issuing 749 types on the basis of the combinations of the variables chosen to describe the dwellings, and 3946 types on the basis of the households' socio-demographic features.*

The input variables can be **active** or **supplementary**: each combination of the values of the active variables defines a type.

***Example:** Suppose that three variables are selected as active, with 4, 3 and 4 categories in the order. The maximum number of types obtainable by combining their values is $4 \times 3 \times 4 = 48$. All units with the same values of the active variables **are equivalent** and are assigned to the same type (the other variables - the supplementary ones - can assume different values, but they are introduced only for descriptive purposes and do not contribute to the definition of types). The active variables represent the aspects of the description that are considered really important to discriminate the units' behaviour: their choice is **the most critical and delicate** assumption in an analysis.*

***Note:** Dealing with Census or Survey data, when some descriptive variables are categorical the whole description must be scaled down to the qualitative level. Then, before submitting them to a multivariate analysis, the conversion to binary form is necessary and useful. Anyway, when the elementary units are relatively few (up to some thousands, a number well tackled by clustering programs in modern fast Personal Computers), there is usually no significant data reduction in assigning them to types. A simple conversion to binary form, without determining typologies, is sufficient.*

The input of parameters

The following questions are asked by the program. Some of them can be skipped, as a consequence of the answers to the questions that precede.

The parameters for the analysis will be read:

- 1. from keyboard**
- 2. from file TYP.PAR, where the values concerning a previous analysis have been saved and can now be modified.**

***Note:** The use of files *.PAR has been described in detail in Chapter 3.*

Enter a comment (up to 2000 characters) to define the contents and purpose of the analysis, to be used to head the output file (just press ↵ if you want no title).

Name of the input file:

Enter the name of the file containing the input data table. If the file is not found, you will be prompted to correct its name. A complete path is allowed for a file not in the current directory.

Total number of CASES (elementary units):

Enter the number of the cases that you want to read from the input file. ("ALL"= read to the end of file). This question allows you to load and process only part of the units in the input file.

Total number of VARIABLES:

Provide the number of **all** the categorical variables that will be read and actually used in the analysis, both **active** and **supplementary**.

As the next question shows, in the input file the variables can be already coded in binary form (also called "complete disjunctive"), or they can be in categorical form. In the former case, the aim is to group units into types; in the latter, the user can also request a conversion to binary form, without producing any table of types.

Anyway, remember that the number to be entered here is that of the variables considered **in categorical form**: each variable counts as one, even though it is binary-coded and each of its categories actually corresponds to a column of 0-1's in the input table.

In the input files the variables are :

1. in **COMPLETE DISJUNCTIVE FORM**;
 2. in **CATEGORICAL FORM** (i.e., for each the code starts from 1 and proceeds by successive integers);
 3. In **FREE_CODED FORM** (i.e., the different categories are labelled by non-consecutive integers or alphanumeric codes of any kind).
-

Once again, we remark that the variables **must be categorical (qualitative)**.

Complete disjunctive (or binary) form

Each variable is split into as many binary variables (with value "0" or "1") as it has categories. For a given variable, each individual unit takes the value 1 for a category that it assumes, 0 for all the others.

Categorical form

Each variable takes only integer values. If it has n categories, the acceptable values range 1 to n.

Free-coded form

A category can be represented by any kind of code, numeric or alphanumeric. The user will be prompted to specify all the codes, so to allow the program to recognise them.

Remember: Mixed codes are not allowed.

All the variables must be in one of the three forms listed above. Even if only one variable (say sex) is free-coded (e.g. "M" and "F"), while all the others use consecutive integers starting from 1 (i.e., they are in **categorical form**), **free-coded form** must be declared. The codes must then be entered for **all** the variables (for a variable with n categories in categorical form you can enter "1/n").

The aim is :

- 1. to prepare a table to be analysed with ACORR**
- 2. only to carry out a conversion of data to complete disjunctive form, saving the resulting binary table to file.**

In the first case the table of the typologies, actually encountered in the input file and determined from the values of the active variables (declared later on), is recorded to a file conventionally named ACORINP.LV, which will be read in by **ACORR**. The file includes also some parameters passed over to **ACORR** (number of cases and variables, number of categories and their labels, etc.). Another file - named TYPCLAS - records for each unit the type to which it belongs. As the clustering procedure that follows will process the typologies, the file TYPCLAS, together with the information saved to file NGCLASS by the program **NONGER** (see section 7.3) enables the user to determine the class to which each elementary unit is attributed.

In the second case no types are determined, and therefore no distinction between active and supplementary variables is done. All variables are converted to binary form and the resulting table is written to a file named BINRECOD. Most of the following questions are unnecessary, and are not asked by the program.

Enter the number of the CATEGORIES of all the variables:

In the same order in which the variables are read from each input record, the number of their categories must be provided. Use commas or blanks as separators.

Constant reference is made to the **categorical** variables that describe the individual units. If a binary coding has been adopted, the total number of categories (i.e. the total number of values that are to be supplied here) will be equal to the total number of columns in the table.

Enter in order a label for every category of each variable:

The labels will be passed to **ACORR** via the work file ACORINP.LV. Each label has a label up to 12 characters long. It cannot include commas or blanks, which are used as separators. In order to avoid too "crowded" graphic projections, the labels should be as short as possible.

A compact form is accepted.

Example: *In order to label 4 income classes and 5 age classes, instead of:*

income1 income2 income3 income4 age1 age2 age3 age4 age5

you can write:

income1/4 age1/5

Note: *Possible errors (too few, too many or too long labels, or labels unacceptable for any reason), are detected and can be corrected.*

How many supplementary variables?

Enter the number of the variables to be dealt with as **supplementary** (if any; 0 = all variables are **active**).

Also here, reference is made to the **categorical** variables describing the units. A variable coded in binary form is nonetheless to be counted as **one**, independent of the number of its categories, even if these are actually the columns of the data table.

In the analysis with **ACORR** to follow, **active** variables contribute to the construction of factors, while the **supplementary** ones are used (and passed forward to the clustering routines) only for descriptive purposes. The total variance of the table, to be shared among the factors, is only the one contributed by the active variables.

In general, only part of the variance of a supplementary variable can be "explained" in the (factor) space spanned by the active variables.

Active and supplementary variables may be intermixed in the input record: in order to treat them differently, the user will be prompted to enter their ordinal numbers.

⇒ *The following question is asked only if there are supplementary variables:*

Enter the ORDINAL NUMBERS of the SUPPLEMENTARY variables:

The ordinal numbers of the supplementary variables are now required, in order to single them out from all the variables read from the data file. As usually, the reference is to the categorical form, even when the actual code is binary: each variable counts as one, no matter how many categories it has. A compact form is possible.

Example: *2/5 12,13, 20*

means that amongst all input variables the 2nd, 3rd, 4th, 5th, 12th, 13th and 20th are to be treated as supplementary. Obviously, in this case you must have declared exactly 7 supplementary variables and at least 20 input variables.

⇒ *If free codes are used at least for one variable, the free-coded form option is set. The user is prompted to enter the codes adopted for all variables, in the order they have been declared.*

The following question is asked as many times as there are variables:

Variable #.... Supply the codes for the categories:

The codes (alphanumeric or numeric) must be separated by **blanks** or **commas**.

Example: *The codes of a hypothetical variable "sex", with categories "M" and "F" (male and female) must be entered as "M F" or as " M, F «; for a three-category variable, with codes "0", "1" and "7", the user will enter " 0 1 7 " or "0,1,7".*

Every case is to be assigned :

1. the same weight
 2. a weight read from the data file.
-

As each case describes the behaviour of an elementary unit, the first option (equal weights) is more common. However, when data from a sample survey are analysed it is sometimes necessary to assign each case a different weight, depending on the adopted sampling strategy.

Weights are supplied as **real** or **integer** numbers, not necessarily normalised.

Note: *If the weights are loaded from the data file and a free format is used, the contents of the first field of each record are assumed as the case weight, if any.*

Enter a FORMAT for reading input data:

The format is necessary to specify the position of the weight (if any), and of the variables to be loaded from the input record.

Enter only **"*»** in case of **free format**: each record in the input file must then contain in order, **separated by blanks**:

- the unit's weight (if they are read from file)
- all and only the variables to be loaded.

The syntax of the format is described in detail in Chapter 3.

Note: **TYPOLG** does not load any label for the units; the names of the types are generated by the program itself and passed over to **ACORR**.

The files output by TYPOLG

The following files are saved by **TYPOLG** when the preparation of a table of types for **ACORR** is requested.

- **ACORINP.LV** contains the table of the types encountered on the basis of the user-specified active variables. Each record describes a type, and an alphanumeric label heads it. The first records in the file contain some general information passed over to **ACORR**.
- **TYPCLAS** consists of as many records as there are elementary units; each record contains the ordinal number of the type to which the unit belongs. Used together with **NGCLASS** (written by **NONGER** after clustering the types, and containing information on the class to which each type has been assigned), it allows the user to assign each elementary unit to a class, saving the information to file. This operation is carried out by the utility **<<INTEGRATE>>**, from the General Utility Menu
- **TIP.OUT** contains the values of the user-entered parameters and the list of the errors (if any) found during execution.

When the only purpose is to convert elementary cases to binary form, the converted table is saved to a file named **BINRECOD**.

6-2 The Factorial Analyses: ACOMP and ACORR

We will not go deep into the theory on which the two factorial analyses included in ADDATI - Principal Component and Correspondences - are based: this is certainly a very interesting and useful chapter of statistics, but it is beyond the limits of an operational manual. However, we recommend the user who wants to master and fruitfully exploit these powerful statistical tools as analytical instruments "per se" - and not only as a way to reduce the dimensionality of a description with a minimal loss of information - to have recourse to some specific textbook.

ACOMP and **ACORR** are quite similar: both take as input a data table – quite often a large one - and explore the relationships existing among its elements (rows and columns). The purpose is to simplify the representation by *constructing* a limited number of underlying new variables (the "*factors*") sufficient to save the most relevant aspects of the description with a minimal loss of details. This is done by rotating in an optimal way the reference system in the geometrical space in which the phenomenon is represented (see chapter 4: each row - as well as each column - of the table can be seen as a point in a suitably defined geometrical space).

The difference between the two analyses lies in the nature of the input table:

- a **quantitative descriptive table** in the case of **ACOMP**,
- a **table of contingency** or a **binary table** for **ACORR**.

Both tables require a preliminary transformation, different in the two cases, that is carried out by the program itself.

Below we will describe in detail the input of the parameters needed by **ACOMP**. **ACORR** asks almost the same questions, and we shall limit ourselves to a description of the few questions peculiar to it (see below "*The Analysis of Correspondences*").

6-2.1 The Principal Components Analysis (ACOMP)

Use	The program performs a Principal Components Analysis of a description table consisting of quantitative variables. The variables are standardised by the program, so that each of them has the same importance in the analysis.
Limits	<p>The table to be diagonalised is prepared while the records describing the statistical units are read from the input file. The computation time is roughly proportional to the number of the units, while the request of core memory is independent from them and increases with the number of the variables (i.e., with the number of columns), both active and supplementary.</p> <p>Anyway, the program is 32-bit Dos, which means that it can address all the central memory and it should be able to handle also tables consisting of hundreds of variables. The opportunity to reduce the number of the processed variables stems more from the need to obtain results interpretable with a minimum of clarity than from memory limitations.</p>
Advice	Here too, like with TYPOLÓG , the variables to be analysed must be carefully chosen, avoiding to invalidate the exploratory power of the method with carelessly selected variables. Again we remark that the analyst's skill manifests itself with the construction of <i>essential tables</i> .

The input data table describes a set of n statistical units (the rows) by means of p **quantitative** variables (the columns). First of all, the variables are standardised as explained in section 4.2.

Note: *The program performs the standardisation automatically. The values of each variable are shifted in such a way that its mean becomes 0: i.e., each value is converted into its difference with respect to the average. Besides this, all variables are so scaled that the variance of each of them becomes 1: each variable assumes the same importance in the subsequent analysis.*

Consider the figure 6-2. It shows a simplified case, with only two (standardised) variables. Each unit can be represented as a point in a two-dimensional space R^2 . The shape of the cloud is a stretched oval, as the variables are strongly correlated. This means that the value of one variable (e.g., y) can be inferred with a good approximation when the other is known, and vice versa. The second variable repeats part of the information already conveyed by the first one: only a little part of its information is really original.

Let us consider the bundle of all the straight lines passing through the origin O . The cloud of unit-points can be projected onto any such line: the resulting uni-dimensional cloud is dispersed about O , its dispersion being measured by its Inertia defined in Chapter 4.

In particular, owing to the fact that the original variables have been standardised, the cloud projects onto each of the two co-ordinate axes x and y with an inertia equal to 1, that exactly represents the value of their variance. On any other line through O , however, **the cloud generally projects with an inertia different from 1.**

An axis exists - indicated with α in the figure 6-2 - onto which the cloud projects with a maximum inertia (we could also say: maintaining at best the distances amongst its points).

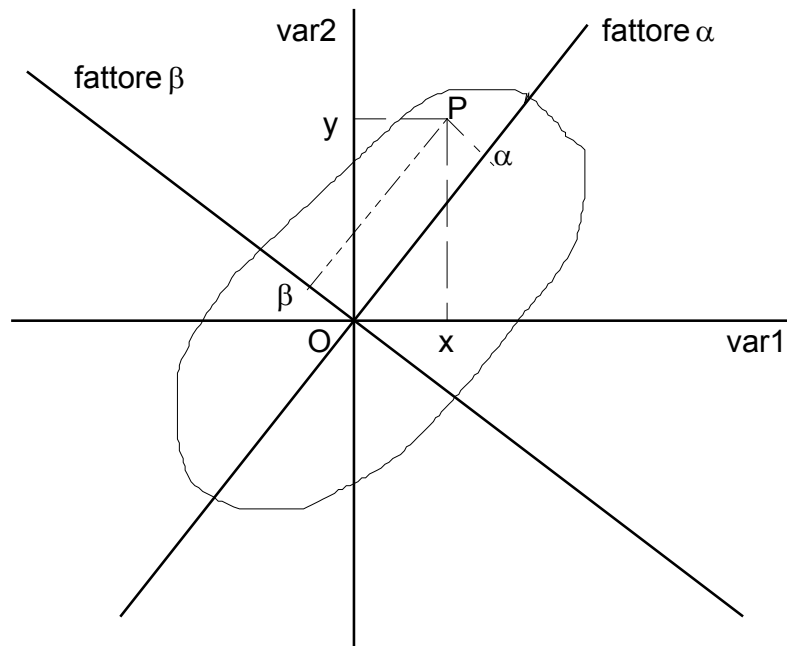


Figure 6-2 The generic point P is represented by the pair of co-ordinates (x,y) with respect to the original variables, by the pair (α, β) on the new reference system.

This is called **first factorial axis**, and the *signed* distance from **O** of the projection of each point is the point's first factorial co-ordinate, or first Principal Component.

The cloud projects onto the axis β perpendicular to α with a much smaller inertia. This completes the description if the dimensions are only two: it is easy to prove that the sum of the two inertias (on the axes α and β) is **exactly 2**, i.e. equal to the cloud's total inertia.

Every pair of orthogonal straight lines through **O** leads to a particular decomposition of the total Inertia. The advantage of the pair (α, β) with respect to (x, y) is that the small fraction of inertia "*explained*" by the axis β -can be easily ignored: this leads to an acceptable uni-dimensional simplification of the original bi-dimensional phenomenon.

These simple considerations can easily extend to the case of p variables: the set of units is represented by a cloud of n points in a p -dimensional space. The value of the total Inertia, after standardising, is p . It is always possible - and convenient, if at least some of the variables are sufficiently correlated - to determine an axis (called "*first principal axis*") onto which the cloud projects maintaining the maximum possible inertia. This amount of inertia is known as the **eigenvalue** associated with the axis. Then, a second axis is determined, orthogonal to the first, that retains the maximum possible fraction of the residual inertia, and so on, until the description is completed.

The set of the new axes can be assumed as a new reference system, alternative to the initial one. The phenomenon is the same, but our viewpoint, expressed by the axes, is changed, and this allows us to focus on the most relevant aspects expressed by the first factors. As the rate of explained inertia goes decreasing from the first to the last factor - i.e., factors are ordered according to decreasing values of the associated eigenvalues - ignoring the last factors leads to a reduction in the dimensionality of the description at the cost of a minimal loss of information.

An example

We will illustrate the operational use of **ACOMP** and the interpretation of results with a didactic example. The table processed describes some features of Kenya's 41 administrative districts. Of course, similar examples could be conceived for many other countries, but keep in mind that the variables depend on the objective of the analysis, and are generally not equivalent in different contexts. Statistical systems are designed differently in different countries, and collect variables defined differently. The method is portable, but no fixed recipe about data is possible.

The input file KENYA.DAT consists of 41 records (one per district). Each contains the values of the following variables measured in a district, separated by spaces:

- district's name (used to label the district in the printouts)
- population (used as weight - see below)
- fertility rate
- average wage (in the formal economy)
- amount of high potential land per capita
- activity rate (in the formal economy)
- cereal production per capita
- cattle per 1000 inhabitants
- goats and sheep per 1000 inhabitants
- cash crop production per capita

The first two fields contain the district's label and population, which is assumed as the district's weight throughout the analysis. The last seven variables - describing some economic features of the district - are *active* in the analysis while the fertility rate, not homogeneous with the others, is used as *supplementary*. This means that it will not contribute to determine the factors - that are computed on the basis of the active variables only - but its relationship with the active variables (and with the factors) will be investigated.

A table so conceived is aimed at exploring the economic features of the districts as well as the relationships existing amongst the variables.

The fertility rate does not make much sense as such, and is added here mainly to illustrate the use of supplementary variables, with little concern for the meaning; however, it will allow the user to single out the economic characters more correlated with a high or a low fertility rate.

Entering the analysis control parameters

When an analysis is run, the user is asked to enter the parameters necessary to control the execution. This is done by answering some specific.

A. Parameters that control the loading of data from the input file

The parameters for this analysis will be read :

- 1. from keyboard**
- 2. from the file ACOMP.PAR, where the values concerning a previous analysis have been saved and can now be modified**

The purpose of this question was already explained in detail in chp.3.

A title for this analysis :

The title can actually be a long comment (up to 2000 characters) that will head the printout. Hit '↵' for no title.

Name of the data file :

You have to enter the name of the file containing your input table. It must be organised as follows: each record must refer to a statistical unit and must contain the values of the descriptive (*quantitative*) variables in a fixed order. If some of them are not used in the analysis and are to be skipped when reading, the user must provide an appropriate reading format (see below).

Note *If the input file does not stay in the working directory, enter the complete path to trace it.*

📁 In our example, the answer is "KENYA.DAT".

Total number of units :

This is the **total** number of rows in the data table to be analysed (one record per statistical unit).

📁 In our example, the statistical units are the districts. They are 41.

Total number of the variables :

Enter the number of **all the quantitative variables** that will be read and actually analysed, both **active** and **supplementary**.

📁 In our example, the variables are 8 (7 active + 1 supplementary). Also the case label and weight are loaded, *but they are not to be counted as variables*.

Number of supplementary cases (or units, or rows in the table) :


(0 = no supplementary case)

WARNING! In the input file the supplementary records - if any - MUST FOLLOW the active ones.

Besides the **active** units, whose relationship with the variables will contribute to determine the factors, it is also possible to include in the analysis a set of so-called **supplementary** units, which have no part in constructing the factor space, but can be projected onto the factor planes and for which the **relative contributions** are computed (the **absolute contributions** are obviously nil, cfr. below). The aim is to draw more information from the way these supplementary objects are placed with respect to the active ones and the variables.

Example *If the active rows describe the behaviour of a group of districts in a given year, the supplementary rows may describe the same districts in a different year, thus permitting a qualitative visualisation and interpretation of the variations occurred.*

Note *Of course, active and supplementary objects must be described by the same variables (the columns of the table).*

 There are no supplementary cases in our example. The answer is '0'.

How many SUPPLEMENTARY variables?

Enter the number of the variables to be dealt with as SUPPLEMENTARY (if any; 0 = all variables are ACTIVE)

Active variables contribute to the construction of factors, while the **supplementary** ones are used only for descriptive purposes. The total variance of the table, to be shared amongst the factors, is only the one contributed by the active variables.

In general, only part of the variance of the supplementary variables can be "explained" in the (factor) space spanned by the active ones; in other words, in general a supplementary variable cannot be exactly expressed as a linear combination of the active ones. Active and supplementary variables may be intermixed in the input record: in order to deal with them differently, you will be later prompted to enter the ordinal numbers of the supplementary variables.

There is one supplementary variable (the fertility rate) in our example.

Do you want a printout of the input table?

(1 = yes; 2 = no)

You can print to ACOMP.OUT a copy of the input data table, if this can ease interpretation. It is better not to do this when the table is huge, or when there is no actual interest in checking input data.

The labels of the rows :


- 1. will be entered from keyboard**
 - 2. will be read from the input data file (if a FREE format is used, each label must lead the corresponding record)**
-

The question concerns the alphanumerical indicators that will label the units (rows) in ACOMP.OUT and in the projections onto factor planes (if requested). The maximum length of each label is 12 characters, and it cannot include commas or blanks (used as separators). For an efficient use of the available memory, and in order to avoid overcrowded graphic projections, it is advisable to choose the labels as short as possible.

Note *A compact form is allowed: e.g., the string "DISTRICT1/500" is expanded to DISTRICT1...DISTRICT500.*

If the labels for the units are read from the input file, the format for reading data - to be supplied later on - must specify their position in the record.

Remember: If a free format is used, each label must lead the corresponding record.

 In our example the name of each district heads the corresponding record in the input data file. Therefore, the answer is '2'.

Enter the labels for the variables :

A specific label must be provided for each variable, in the same order as they are read from the input file records. The labels are needed in order to distinguish the variables in the printout and in the factor planes.

Also these labels must not be longer than 12 characters, and cannot include commas or spaces. For an efficient use of memory, and in order to avoid overcrowded graphic projections, it is advisable to choose labels as short as possible (4-6 characters).

Use blanks or commas as separators.

A compact form is accepted: in our example we could simply enter

variab1/8

as we have 8 variables, but for sake of clarity it is far better to use labels easing an immediate identification of the meaning of the concerned variable. Here we could enter a string like the following:

fert_rate ave_wage hp_land act_rate cereals cattle goa_sh cash_crop

Any error (too many or too few labels, or unacceptable for any reason) is remarked and it is possible to correct it.

Remember: The order of the labels must reflect the position of the variables in the input file.

The following question is asked only when the unit labels are entered from keyboard. In our example the question is skipped, as labels are loaded from file.

Enter the labels for the objects (rows) :

Each label must not be longer than 12 characters, and cannot include commas or spaces. For an efficient use of memory, and in order to avoid overcrowded graphic projections, it is advisable to choose labels as short as possible (4-6 characters).

Use blanks or commas as separators. A compact form is accepted.

Example *To provide the labels for 150 active and 50 supplementary units without entering them one by one, you can type*

active1/150 suppl1/50

and the string would be automatically expanded to the 200 labels

active001 ... active150 suppl01 ... suppl50

Any error (too many or too few labels, or unacceptable for any reason) is be pointed out and it is possible to correct it.

Provide the *ordinal numbers* of the supplementary variables :


The ordinal numbers of the **supplementary** variables are now requested, to single them out from all the variables loaded from the data file.

A form of compact writing is allowed.

Example “ 2/5 12,13, 20 “

means that amongst all the variables that have been loaded the 2nd, 3rd, 4th, 5th, 12th, 13th and 20th are supplementary.

Obviously, in this case you should have declared exactly 7 supplementary variables and at least 20 variables globally.

 In our Kenya example there is one supplementary variable (the *fertility rate*), which is the first variable loaded. The answer is '1'.

Every case is to be assigned :


- 1. the same weight**
- 2. a weight read from the input data file**

For a table of quantitative variables the weight of each unit - i.e., its importance in the analysis - can be defined by the user.

For example, for a table of rates it is expedient to give back to each unit its absolute importance by attaching it an appropriate weight. Thus, weighted averages and variances are computed and correlations are correct; the resulting values are meaningful and represent the overall actual properties of the analysed system. If every row represents a geographical unit, described by variables calculated as rates over the population, it is convenient to use the population in each unit as the unit's weight. In this way, the variables' mean values correctly represent the average behaviour of the system.

The "equal weights" option is for use with elementary objects (households, dwellings, etc.). When different weights are used, the position of the weight in the input file is declared in the format. If the format is free, the weight is conventionally the second item in the record, if an alphanumerical label is present; the first otherwise.

Note *A weight must be supplied as a real or integer number, not necessarily normalised. It is not be counted among the variables.*

 In our example, the population of each district - read in from the input file - is assumed as weight. The answer is '2'.


Enter a FORMAT for reading data from the input file :

The format is necessary to specify the position of the unit's label (if any), of the weight (if any) and of the variables in the input record.

Type only "*" for **free format** reading: in such case, each record in the input file must contain in order, **separated by blanks** :

- the unit's label (if read from file)
- the unit's weight (if read from file)
- the variables

The format syntax is described in detail in paragraph 6.1.

 In our example the input file KENYA.DAT has been so prepared as to fulfil the **free format** requirements. We will therefore enter an asterisk "*".

At this point the data are loaded from the input file, the variables are standardised and their correlation matrix is computed and printed to the output file (conventionally named ACOMP.OUT). Starting from the correlation matrix the factorial axes and the associate eigenvalues are computed. Then, the following alternative is displayed:

Type :

- 1. to examine how the total inertia is shared amongst factors and decide how many Principal Components to print or save;**
- 2. to continue.**

With the first option, the results produced so far are displayed: the user can inspect the correlation matrix and the eigenvalues (see the tables 6-1 and 6-2), and decide how many factors he wants to have printed or saved to file.


The second option allows the user to continue and enter the number of factors to save.

If the choice is 1, the current contents of ACOMP.OUT is shown: it consists of a summary of the parameters supplied by the user, followed by the input data table (if requested) or by the weighted average value of each input variable. Then, the correlation matrix can be inspected: the table 6-1 shows it for our example concerning the 41 districts of Kenya. The matrix includes all the correlations among active and/or supplementary variables.

CORRELATIONS (*1000)

	ave_ wage	hp_l and	act_ rate	cere als	cat tle	goa _sh	cash_ crop	fert_ rate
ave_wage	1000							
hp_land	-429	1000						
act_rate	698	-263	1000					
cereals	-529	579	-157	1000				
cattle	-125	233	-257	100	1000			
goa_sh	-63	105	-240	-165	631	1000		
cash_crop	-310	116	162	323	-6	-96	1000	
fert_rate	-462	302	-193	625	-103	-330	264	1000

Table 6-1 The correlation matrix as it appears in ACOMP printout.

 The highest correlation is between "average wage" and "activity rate", mainly due to the urban districts of Nairobi and Mombasa; a high correlation also exists between "cattle" and "goat_sheep",

meaning that these two activities tend to be present in the same districts. The "fertility rate" appears to be positively correlated with the cereal production.

These considerations can in general be further developed, leading to a first identification of groups of particularly correlated variables.

Note *if the input variables make up a time series, they represent the successive values of a variable that varies in a continuous way. It can be expected that contiguous values tend to be strongly correlated. In such case the multidimensional cloud is usually highly stretched and a limited number of principal components can be determined, which summarise most of the information. The explanatory power of the first factors is generally high.*

```
TOTAL INERTIA    = 7.000000

|  |  |  | EXPLAIND | CUMULATD |
|  |  |  | INERTIA | INERTIA |
|  |  |  | (%) | (%) |
|----|-----|-----|-----|
| 1 | 2.5354424 | 36.221 | 36.221 | *****
| 2 | 1.7289826 | 24.700 | 60.920 | *****
| 3 | 1.0887727 | 15.554 | 76.474 | *****
| 4 | 0.8229769 | 11.757 | 88.231 | *****
| 5 | 0.4085767 | 5.837 | 94.068 | *****
| 6 | 0.2909169 | 4.156 | 98.224 | *****
| 7 | 0.1243322 | 1.776 | 100.000 | **
```

Table 6-2 The eigenvalues associated with the Principal Components.

The table 6-2 shows the eigenvalues for our Kenya example. They measure, in order, the explanatory power of the factors.

The value of the total inertia, distributed over the Principal Components, is 7, as there are 7 active variables. The cloud maintains an inertia equal to 2.54 when it is projected onto the first factorial axis: this corresponds to the 36.22 per cent of the total. The last column in the table shows the cumulated inertia explained by all the principal components considered so far: it can be seen that the first five factors summarise 94.07% of the total inertia. This means that on the 5-dimensional space spanned by the first 5 factors the cloud maintains 94.07 per cent of its inertia, while a residual 5.93% is lost.

This could be an acceptable simplification; yet, as the variables are only seven (and so are the factors) we will decide in this case to retain all of them for clustering.

B. Parameters controlling the output


The user decides how many factors fit his/her needs, then the factorial co-ordinates are computed and are both printed to output for interpretation and saved to file for clustering. The contents of the printed tables is described in some detail further on.

The following questions are asked by the program: they are aimed at controlling the program's output. The factor co-ordinates computed for each unit are written to ACOMP.OUT (to be printed and interpreted); to ACOMP.FPL (on request) for the FACPLAN program, that displays graphically the projections of the cloud on the factor planes; to COORRIG.LV or COORCOL.LV (on request) for the subsequent clustering phase.

It is always a good idea to ask for this printout, since the meaning of factors must be derived from the contributions of variables.

For each variable and each requested axis the following information will be printed:

- the factorial co-ordinate of the variable on the axis;
- the **relative contribution** (share of the variable's inertia explained by that factor);
- the **absolute contribution** (share of the factor's overall inertia coming from the variable);

 For our Kenya example we will request the contributions on 6 factors, sufficient to explain 98.22% of the overall variability (the explicative power of the last factor is negligible). We will use these contributions in order to interpret the meaning of factors.

For how many factors do you want to print the contributions for the objects?
Enter the requested number of factors.


It is profitable to have these contributions printed **only** when one intends to analyse results using also the contributions for the objects, as well as for the variables.

Skip it if there are too many units or when the behaviour of single units is uninteresting. You will avoid a waste of paper as the printout of contributions concerning 50 objects or variables requires one paper module.

***Note** This printout is generally useless when the input is a typological table produced by TYPOLOG.*

For each unit and each requested factorial axis the following information is printed :

- the factorial co-ordinate of the unit on the axis;
- the **relative contribution** (fraction of the unit's inertia explained by that factor);
- the **absolute contribution** (fraction of the factor's overall inertia coming from the unit);

 Kenya example: as units are only 41, it might be interesting to examine thoroughly the behaviour of some of them. Let us ask for the printout of the contributions for the first three factors (explaining 76.47 % of inertia).

VARIABLES (COLUMNS):
How many factorial co-ordinates do you want to save onto file?
(0 = none)

The factorial co-ordinates of the variables (table columns) must be recorded if you intend to cluster them. Their similarity of behaviour over the whole set of statistical units is shown up through aggregation. However, this is a kind of analysis which can be better performed by means of other statistical techniques.

 Kenya example: as we do not intend to cluster variables, the answer is '0'.

UNITS (ROWS)
How many factorial co-ordinates do you want to save onto file?

It is necessary to record the factorial co-ordinates before going on to a clustering of the objects with NONGER or AHC.

If the set of objects is very large (some hundreds or even thousands of units) it is better to limit the number of factors passed over to the clustering procedure, which is extremely time-consuming.

It is generally convenient to save a number of factors sufficient to explain 80% or 90% of the overall inertia.

- 📁 Kenya example: we have already decided (see above) to use all factors for clustering (yet the last, explaining only 1.78% of the inertia, could actually be ignored). The answer is '7'.

How many factors do you want to save for the graphic projections onto factor planes? (0 = no graphic projections)

Note *It is possible to see on the screen the projection of the cloud onto one or several factor planes, putting on/off at will variables or objects, both active and supplementary. Any part of the screen can be zoomed, and the graphic representation so obtained can be printed (see chapter 4).*

The projection onto a factor plane may ease interpretation when the plane accounts for a high fraction of the overall variance. One must, however, limit oneself to consider only those points that are well represented on that plane.

Even when the projections are used as a starting point, it is expedient to insist that the interpretation should always be based on the relative and absolute contributions.

- 📁 Kenya example: it is sufficient to save three factors in order to display the most interesting factor planes.

The table of contributions and their interpretation

The information detailed below is stored (on request) in the output file ACOMP.OUT, separately for active and supplementary units and variables.

-----*														
#	ACT	QLT	WEIG	INR	DIS	FAC	REL	ABS	FAC	REL	ABS	FAC	REL	ABS
	VAR					1	CON	CON	2	CON	CON	3	CON	CON
-----*														
1	ave_wage	939	1	143	1000	-850	723	285	-180	33	19	282	80	73
2	hp_land	1000	1	143	1000	722	521	206	136	18	11	136	19	17
3	act_rate	957	1	143	1000	-671	450	177	288	83	48	608	370	340
4	cereals	992	1	143	1000	692	479	189	487	237	137	171	29	27
5	cattle	997	1	143	1000	451	204	80	-671	450	261	411	169	155
6	goa_sh	999	1	143	1000	282	80	31	-816	666	385	289	84	77
7	cash_crop	991	1	143	1000	281	79	31	491	241	140	582	339	311
-----*														
-----*														
#	SUP	QLT	WEIG	INR	DIS	FAC	REL	ABS	FAC	REL	ABS	FAC	REL	ABS
	VAR					1	CON	CON	2	CON	CON	3	CON	CON
-----*														
8	fert_rate	487	1	143	1000	437	191	0	486	236	0	-77	6	0
-----*														

Table 6-3 The contributions for the variables on the first three factors.

The table 6-3 shows the information on the variables' contributions as it appears in ACOMP.OUT. The meaning is detailed in the table 6-4 below for the *hp_land* variable and the first two factors.

	#	ACT VAR	QLT	WEIG	INR	DIS	FAC 1	REL CON	ABS CON	FAC 2	REL CON	ABS CON
Variable's → ordinal number	5	hp_land	1000	1	142	1000	722	521	206	136	18	11
		↑ alphanumeric label					↑ information on factor # 1	↑	↑	↑ information on factor # 2	↑	↑

Table 6-4 The contributions for the variable *hp_land*.

QLT (quality of the representation): this is the fraction of the variable's inertia globally explained by **all** the factors for which information is printed (six factors were requested in this case).

It sums up the variable's relative contributions on the printed factors. The table shows that the first six factors explain 1000/1000 of the inertia of the variable *hp_land*.

INR (variable's total inertia): as all variables are standardised, they give the same contribution to the cloud's overall inertia which is exactly 7 (there are 7 active variables all having variance 1). INR is here expressed as a fraction of the total inertia: 143/1000, corresponding to 1/7.

In the case of a Correspondence Analysis, the variables (i.e., the columns of the table) have generally different values of INR, which depends on the point's weight and on its distance from the cloud centre, representing the overall average profile.

WEIG (**weight**) Importance of the variable in the analysis. As variables are standardised, WEIG has conventionally the same value for all.

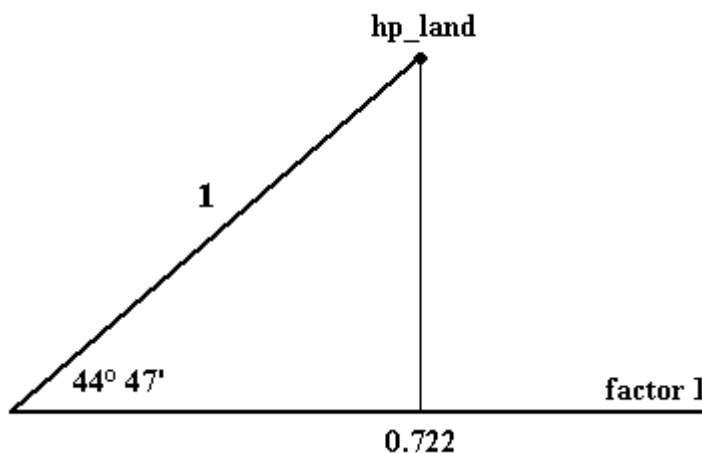


Figure 6-3 The co-ordinate of a variable-point is the correlation between the variable and the factor: $0.722 = \cos(44^\circ 47') = \text{correlation}(\text{hp_land}, \text{factor 1})$.

DIS is the square of the distance of the variable-point from the origin. **It measures the variable's variance.** Here it is 1 for all variables, as they are standardised. In the variable space, all points representing variables lie on the surface of a hyper-sphere with radius 1.

FAC1 is the co-ordinate of the variable-point on the first factor axis (it must be read as 0.722). As the distance of each variable-point from the origin is exactly 1, FAC1 is equal to the cosine of the angle formed by the segment joining the point with the origin and the first factor axis (see figure 6-3). It can be shown that this is a measure of the correlation

between the variable and the first factor (considered as a new variable constructed as the maximum-variance linear combination of all the original variables).

REL CON **relative contribution** (of the factor to the variable): this is the fraction (x 1000) of the variable's inertia explained by the factor.

Here the first factor explains 52.1 per cent of the variance (over districts) of the high-potential land rate. It can be easily shown that for a Principal Component Analysis the relative contribution is the square of the corresponding factor co-ordinate (FAC1); it is therefore equal to the square of the correlation between the variable and the factor.

ABS CON **absolute contribution** (of the variable to the factor's variance): this is the fraction (x 1000) of the factor's inertia contributed by the considered variable. Here the 20.6 per cent of the variance of the first factor is contributed by the variable *hp_land*.

The table 6-5 shows the information concerning the units as it appears in ACOMP.OUT.

#	ACT OBJ	QLT	WEIG	INR	DIS	FAC 1	REL CON	ABS CON	FAC 2	REL CON	ABS CON	FAC 3	REL CON	ABS CON
1	Busia	378	19	6	2177	-379	66	1	-120	7	0	-815	305	12
2	Bungoma	573	33	19	3999	1154	333	17	844	178	14	-500	63	8
3	Kakamega	684	67	12	1216	-175	25	1	281	65	3	-850	593	45
4	Siaya	750	31	7	1479	53	2	0	-215	31	1	-1030	717	30
5	Kisumu	295	31	5	1156	-373	120	2	-57	3	0	-446	172	6
6	S.Nyanza	218	53	16	2087	342	56	2	-148	11	1	-562	151	15
7	Kisii	748	57	17	2131	322	49	2	451	96	7	-1134	604	67
8	Turkana	610	9	36	27314	5	0	0	-4054	602	88	484	9	2
9	W.Pokot	552	10	6	3723	650	114	2	-1151	356	8	-555	83	3
10	T.Nzoia	789	17	40	16411	2710	448	49	2322	329	53	464	13	3
11	Nandi	960	20	47	16726	2977	530	68	1868	209	39	1926	222	67
12	U.Gishu	678	20	35	12653	2026	324	32	1957	303	43	807	51	12
13	El.Marakwet	439	10	5	3506	281	23	0	-1172	392	8	-293	24	1
14	Baringo	628	13	34	17968	1339	100	9	-3028	510	71	563	18	4
15	Samburu	743	5	48	1280	4143	257	34	-4960	368	71	2812	118	36
16	Kericho	903	41	64	10870	2434	545	97	628	36	9	1870	322	133
17	Nakuru	964	34	17	3403	663	129	6	1430	601	40	891	233	25
18	Laikipia	756	9	14	11139	2412	522	20	987	87	5	1277	146	13
19	Narok	915	14	34	17515	2096	251	24	-3130	559	78	1358	105	23
20	Kajiado	781	10	56	40181	794	16	2	-4829	580	131	2724	185	66
21	Nyandarua	520	15	23	10572	2251	479	30	655	41	4	-51	0	0
22	Nyeri	438	32	3	667	-289	126	1	457	313	4	-3	0	0
23	Kirinyaga	914	19	3	995	-212	45	0	191	37	0	-910	833	14
24	Murang'a	184	42	13	2124	-99	5	0	582	159	8	-207	20	2
25	Kiambu	393	45	60	9329	-373	15	2	1503	242	58	1125	136	52
26	Marsabit	804	6	40	44459	213	1	0	-5721	736	119	1719	66	17
27	Isiolo	569	3	11	27661	-265	3	0	-3614	472	21	1613	94	7
28	Meru	943	54	9	1124	-138	17	0	-105	10	0	-1015	916	51
29	Embu	963	17	3	1292	-731	413	4	-168	22	0	-826	528	11
30	Machakos	478	67	12	1285	-387	116	4	-480	179	9	-485	183	14
31	Kitui	749	30	16	3631	183	9	0	-1616	719	46	-271	20	2
32	Mandera	767	7	7	7129	-1866	488	9	-1332	249	7	-463	30	1
33	Wajir	600	9	5	4125	-689	115	2	-963	225	5	-1037	261	9
34	Garissa	622	8	8	6548	-726	81	2	-1879	539	17	-114	2	0
35	Tana_River	439	6	6	6604	194	6	0	-1673	424	10	249	9	0
36	Lamu	783	3	1	2869	-979	334	1	-80	2	0	-1132	447	3
37	Taita/Taveta	807	10	1	791	-378	181	1	-12	0	0	-704	626	4
38	Kilifi	964	28	8	1933	-702	255	5	49	1	0	-1170	708	35
39	Kwale	788	19	3	1263	-22	0	0	-876	608	8	-476	180	4
40	Mombasa	922	22	40	12597	-3304	866	96	470	18	3	689	38	10
41	Nairobi	960	54	212	27437	-4708	808	472	545	11	9	1966	141	192

Table 6-5 - The units' contributions on the first three factors.

The meaning is detailed in the table 6-6 below for the **Kericho** district and the first two factors.

	#	ACT OBJ	QLT	WEIG	INR	DIS	FAC 1	REL CON	ABS CON	FAC 2	REL CON	ABS CON
unit's ordinal. → number	5	Kericho	903	41	64	10870	-3305	834	17	628	36	9
		↑ unit's alpha- numeric label					↑ information on factor # 1	↑	↑	↑ information on factor # 2	↑	↑

Table 6-6 The contributions on the first two factors for the Kericho district.

The meaning is the following:

QLT (quality of the representation): this is the fraction of the unit's inertia globally explained by the factors for which information is printed (of which there are three in this case). It sums up the unit's relative contributions on the printed factors.

INR (**unit's total inertia**): this is the fraction (x 1000) of the total inertia that is contributed by the unit-point:

$$\text{INR} = (\text{unit's inertia}) / (\text{total inertia})$$

where the total inertia is the sum of the eigenvalues. The unit's inertia with respect to the origin (coincident with the centre of the cloud) is defined as the product of the point mass (WEIG) by the square of its distance from the origin (DIS).

WEIG **Weight** of the unit in the analysis (so scaled that the total of the weights sums up to 1000); it represents the unit's relative importance.

Here the Kericho district has a population (assumed as weight) that amounts to 41/1000 of the Kenya population, while its inertia is 64/1000 of the total inertia.

Generally speaking, the greater INR in comparison to WEIG, the more peculiar the unit's behaviour. In fact, if all unit-points were located at the same distance from the origin their inertia would be exactly proportional to their weight. In this case, as both INR and WEIG are scaled to 1000, the two quantities would have exactly the same value.

A value of INR greater than WEIG (as in the case of Kericho) means that the point has a distance from the origin greater than the average distance, and that its behaviour is quite particular (it must be remembered that the origin represents the system's average behaviour). A more careful inspection is needed in order to identify the profile components involved; i.e., in which aspects the district is peculiar. The results of the Principal Component Analysis could be used for this purpose, but it is far more simple to consider the profile of the class to which the district is assigned in the subsequent cluster analysis: if the variability within the class is neglected, the class profile can be assumed to represent the behaviour of all the units assigned to it.

It must be kept in mind that the distance between the unit-point and the origin is a measure of the global difference between the unit's behaviour and the overall average behaviour of the system (i.e., the set of all the units considered, or the concerned country). A Principal Component and a Correspondence Analysis always assume as reference the average behaviour of the whole system, and analyse in which way and how much the various units differ from it (and from one another). This admits a straightforward interpretation in most cases.

DIS is the **square of the distance** of the unit-point from the origin. The greater DIS, the more the unit's profile globally differs from the average behaviour, represented by the cloud centre.

FAC1 is the **co-ordinate** of the geographical unit on the first factorial axis (for Kericho, it is to be read as 2.434).

- REL CON** **relative contribution** (of the factor to the unit): this is the fraction (x 1000) of the unit's inertia explained by the factor. In the case of Kericho, the first factor is sufficient to explain 54.5 per cent of the district's inertia.
- ABS CON** **absolute contribution** (of the unit to the factor's variance): this is the fraction (x 1000) of the factor's inertia contributed by the unit. Here, 9.7 per cent of the first factor's variance is contributed by the Kericho district.

Interpretation of the factors

The meaning of the factors, considered as new variables, can be derived by their correlation with the initial variables, i.e. by considering which are the variables that more contribute to them.

Factor 1

Let us consider the table 6.6: the highest absolute contributes to the factor come from *ave_wage* and *act_rate* (which lie on the negative side of the axis) and from *hp_land* and *cereals* which lie on the positive side. This represents the most relevant set of relationships existing in the correlation matrix. *Ave_wage* and *act_rate* are positively correlated with one another, and so are *hp_land* and *cereals*; the two variables in the first group are negatively correlated with those in the second group. Thus, the first factor captures the main variability present in the system (at least, according to the set of variables we chose), namely the one between rural districts - with low values of activity rate and average wage associated with values of *hp_land* and cereal production above the average - and urban districts, characterised by opposite features.

An inspection of the table 6-5 shows which districts are more involved in this opposition: it is sufficient to single out those receiving a strong contribution (REL CON) from the first factor. They are Mombasa, Nairobi and, to a less extent, Mandera on one side and T.Nzoia, Nandi, Kericho, Laikipia, Nyandarua and Embu on the other. The other districts do not seem involved: their behaviour cannot be reduced to the above opposition and seems to depend on other relationships existing among the variables.

***Note** The first factor captures and summarises a relation among the four mentioned variables in which the above mentioned districts are involved. Some other district might have a high value of *act_rate*, but not associated with a high value of *ave_wage* and with a low level of *cereals* production and *hp_land*.*

Factor 2

It summarises the positive association existing between the presence of **goats_and_sheep** and **cattle** in some districts (those showing in table 6-5 a high REL CON value for factor 2, associated with a negative value of the FAC2 co-ordinate: Turkana, Baringo, Narok, Kajiado, Marsabit, Kitui, Garissa, Kwale and others to a minor extent). From table 6-3, a negative correlation appears also to exist between the presence of cattle/goat_sheep and that of cereals/cash_crop (characterising a few districts; e.g., Nakuru) showing on the second factor a high REL CON value associated with a positive FAC2 co-ordinate.

The interpretation could continue for the other factors, that become anyway less and less relevant.

The files output by ACOMP

- **ACOMP.OUT** is the file to be printed, containing the information to be interpreted.
- **COORRIG.LV** (written on user's request) contains the factor co-ordinates (together with some more information) needed by NONGER to cluster the units.
- **COORCOL.LV** (written on user's request) contains the factor co-ordinates (together with some more information) needed by NONGER to cluster the variables.
- **ACOMP.FPL** (written on user's request) contains the information passed over to FACPLAN to display the projections onto factor planes.

6-2.2 The Analysis of Correspondences (ACORR)

Use, limits and advices

What was said for **ACOMP** holds also here.

The standard input table for a Correspondence Analysis is a **contingency table**, i.e. a table obtained by cross-tabulation of two categorical variables.

If the two variables have n and p categories respectively, the resulting contingency table has n rows and p columns; the table's generic cell (i,j) counts the units that take simultaneously the i -th category of the first variable **and** the j -th category of the second one.

Some **only apparently** different tables can be actually thought of as contingency tables, and dealt with via a Correspondence Analysis:

- tables obtained by putting side-by-side several contingency tables that count the same units;
- binary tables obtained from qualitative descriptive tables by converting the qualitative (categorical) variables to (binary) complete disjunctive form (see section. 4.1).

The latter case is less intuitive, but quite interesting in practice.

Example: *Think of a table where the n rows represent n statistical units, described by p categorical variables (some of which may have been obtained by using **RECODE** to convert to categorical form some variables directly observed as quantitative. When the p variables are converted to binary form, each of them produces a table of zeros and ones, having as many columns as there are variable's categories. This way, p side-by-side binary tables are obtained, each of which can be seen as a contingency table cross-tabulating the variable "unit" with one of the descriptive variables: for each row (unit), a cell corresponding to a category not assumed by that unit contains 0 (i.e., it includes no unit), and a cell corresponding to a category assumed contains 1 (i.e., it counts **exactly one unit**). The structure is quite banal, but it can be thought of as a multiple contingency table. In this case, as the rows represent a set of units, their mutual similarity structure can be analysed via **ACORR**, followed in case by a classification.*

In a contingency table rows and columns have similar role and are dealt with symmetrically in **ACORR**. The purpose of the method is to analyse the similarity among rows (with respect to columns), the similarity among columns (with respect to rows) and the relationships existing between rows and columns.

The analytical treatment can focus on rows as well as on columns, owing to the table's symmetry. The table 6.10 shows a small example: a system consists of three geographical units, each described by the amount of cultivated land per type of crop (only three crops are considered explicitly, and the fourth category summarises all the others). **ACORR** would convert the initial table 6-7a) to that of row profiles 6-7b) or of column profiles 6-7c). Actually only either of these tables need to be analysed and the results concerning the other are derived through simple transformations.

	teff	mais	sorghum	other	
Area 1	40	60	50	100	250
Area 2	100	100	200	200	600
Area 3	80	120	100	200	500
	220	280	350	500	1350

a)

	teff	mais	sorghum	other	
Area 1	.16	.24	.20	.40	250
Area 2	.17	.17	.33	.33	600
Area 3	.16	.24	.20	.40	500
	.16	.21	.26	.37	1350

b)

	teff	mais	sorghum	other	
Area 1	.18	.21	.14	.20	250
Area 2	.45	.36	.57	.40	600
Area 3	.3680	.43	.29	.40	500
	220	280	350	500	1350

c)

Table 6-7

a) An example of a contingency table: each cell counts the area (in thousands of acres) per administrative area and type of crop.
The totals (row and column *marginal values*) are also shown.

b) The *row profiles* computed from the table **a)**. Each row shows, for the concerned area, the percentage of the cultivated land dedicated to each type of crop; the last row gives the same information for the whole system and is assumed as the "normal" (or average) behaviour.

The relative specialisation of an area is determined by comparing its profile with the overall profile. In **ACORR**, the weight of each area is proportional to its marginal (i.e., to the total cultivated land in the area) and is computed by the program itself.

c) The *column profiles* computed from the table **a)**. Each column shows, for the concerned crop, how the cultivated land is distributed in percentage on geographical areas. The last column gives the same information for the whole system and is assumed as the "normal" (average) crop distribution on geographical units.

The relative concentration of a crop is determined by comparing its distribution with the overall profile. In **ACORR**, the weight of each column is proportional to its marginal (i.e., to the total cultivated land for the concerned crop) and is computed by the program itself.

Owing to the symmetrical role of the two variables, the analysis can focus indifferently on the rows or the columns. The table 6-7 shows a little didactic example: a system consists of three geographic units, each described by the area dedicated to some types of crops (only three types are considered explicitly, while the fourth counts all residual crops). **ACORR** converts the initial table 6-7a) into the table of *row profiles* 6-7b) or in that of

column profiles 6-7c). It is sufficient to analyse only one of them, and the program automatically determines which is computationally the more convenient. The results relative to the other table are then derived through some simple transformations.

According to the table 6-7b, as the descriptive variables are four, each area is represented by a point in a four-dimensional space, whose co-ordinates are the components of the profile. The point is assigned a weight proportional to the cultivated land in the area (the row's margin). All together, there are three weighted profile patterns in a four-dimensional space.

Symmetrically, according to the table 6-7c each crop can be represented as a point in a three-dimensional space (there are three geographical units), whose co-ordinates are the components of the corresponding column-profile. The crop-point is assigned a weight proportional to the overall area where that crop is cultivated. In this case we have four weighted profile patterns in a three-dimensional space.

Consider the table 6-7b. The profiles of the first and the third units are identical; the reason is that the two corresponding lines in table a) are proportional. The two units have a different amount of cultivated land (and therefore a different relevance in the analysis) but an identical percentage distribution of that land amongst crop types. The two corresponding unit-points are coincident in the representation space. Were all the lines in the a) table proportional, all the unit-points would be coincident: there would be no cloud, only a point, and no variability to be analysed. On the contrary, when the units have different behaviours the corresponding points are scattered about the cloud centre, which represents the system's average behaviour (i.e., the overall crop mix, given by the marginal row in the table 6-7b).

Similar considerations can be developed for the table 6-7c.

The distance between two profile-points in RP is computed according to a modification of the usual Euclidean formula, and is known as "**chi-square distance**". For the definition the user is referred to a multivariate statistics textbook.

ACORR processes the table of profiles in a way very similar to what was already explained for **ACOMP**. Eigenvalues, factorial co-ordinates and contributions are determined, on which the interpretation will be based. The following differences must be remarked:

- in **ACORR**, differently than in **ACOMP**, rows and columns play a totally symmetrical role. The tables of row and column contributions, saved to file ACORR.OUT, are interpreted in exactly the same way. As in **ACORR** no standardisation of variables is performed, no correlation table is printed and we prefer to speak of a stronger or weaker association of two given lines (two rows or two columns) with respect to the lines of the other set.
- in **ACORR** the first eigenvalue (called "trivial" or "banal") is always 1: it is of no interest, as it is a mere consequence of the transformation to which the original table a) has been submitted to compute the profile table b); therefore, it is ignored. All other eigenvalues (the meaningful ones) lie between 1 and 0.
- The number of non-null eigenvalues is generally less than what could be expected from the size of the data table. The dimensionality related to the number of the columns is only apparent (and redundant): in every line of each single contingency table the values of the cells add up to the same total, i.e., to the number of the counted units. Therefore, the columns are not independent, and this reduces the actual dimensionality of the feature space.

- The total Inertia of the cloud can be computed in a very simple way from the number of the categorical variables and that of their categories:

$$Inertia = \frac{n. of categories}{n. of variables} - 1$$

Thus, for example, if the table to be processed consists of five side-by-side contingency tables (i.e., if there are five initial categorical variables), with 4, 4, 3, 4 4 categories respectively (19 categories altogether), the total Inertia of the cloud is $2.8 = (19/5) - 1$.

- If the data table was written by **TPOLOG**, the explanatory power of the first factors is less than what could be expected in **ACOMP** (or even in **ACORR**, when processing a real contingency table). This is an effect of the conversion to complete disjunctive form, that has increased the number of the columns of the table passed on to **ACORR**, introducing some fictitious inertia. Even if the fraction of inertia loaded on the first factors seems to be low, their importance when interpreting results is still relevant.

Entering the control parameters

ACORR and **ACOMP** need almost the same parameters to be specified in order to drive an analysis, and the two programs ask the user almost the same questions. Therefore, the reader is referred to the full description already given for **ACOMP**. We limit ourselves here to illustrate the only question specifically concerning **ACORR**.

Working after program TPOLOG?

1. **no - independent analysis**
2. **yes - the input is a typological table produced by TPOLOG**

An **Analysis of the Correspondences** usually takes as input a table obtained by setting side by side one or several contingency tables, which count elementary units of the same type (households, dwellings, individuals, etc.).

However, while working with urban or regional data (especially data drawn from Census or surveys) tables are often dealt with where each line describes an elementary unit (e.g., a firm or a household), by means of some **qualitative** variables. In such case, variables must be converted to **binary** (or **complete disjunctive**) form: a column for each category, with value 1 if that category is assumed by the unit concerned, 0 otherwise. The resulting binary table can be submitted to a correspondence analysis, known in this case as a **Multiple Correspondence Analysis**.

The program **TPOLOG** reads the file of elementary data, recognises all the units that are identical with respect to the values of the **active** variables, aggregates them in suitably weighted types (or "typologies") and writes a binary table used as input by **ACORR**.

The purpose of the question is to inform **ACORR** about the nature of its input file. If **ACORR** does not work on **TPOLOG**'s output, all the necessary information (how many units and variables, their labels, etc.) must be entered from keyboard.

The correct answer is "2" if you intend to analyse a typological table saved by **TPOLOG** to a file conventionally named ACORINP.LV. This means that you must have run **TPOLOG** to create the table of types on the basis of the active (categorical) variables.

Note If **TYPOLOG** was used only to convert to binary form a table of qualitative variables, without determining types, the correct answer is "1". The file **BINRECOD** written by **TYPOLOG** in this case **does not contain** any information about the size of the table, label for categories, etc., as **ACORINP.LV** does. You have to enter "BINRECOD" as the name of the input data file, and supply from keyboard all the required parameters.

The table of contributions and their interpretation

The information detailed below is stored (on request) in the output file **ACORR.OUT**, separately for active and supplementary rows and columns. The interpretation of results is the same for both rows and columns.

QLT	(quality of the representation) : fraction of the point's inertia globally explained by the factors on which information is printed. It sums up the point's relative contributions on the printed factors.
INR	(point's total inertia) : fraction (*1000) of the total inertia that is contributed by the point: $INR = (\text{point's inertia}) / (\text{total inertia})$ where the total inertia is the sum of the eigenvalues. The point's inertia with respect to the origin (coincident with the centre of the cloud) is defined as the product of the point's mass (WEIG) by the square of its (chi-square) distance from the origin (DIS).
WEIG	<p>Weight of the point in the analysis (so scaled that the total of the weights of all the points in a set - rows or columns - sums up to 1000); it represents the point's relative importance.</p> <p>Generally speaking, the greater INR in comparison to WEIG, the more peculiar the point's behaviour. In fact, if all points were located at the same distance from the origin their inertia would be exactly proportional to their weight. In this case, as both INR and WEIG are scaled to 1000, the two quantities would have exactly the same value.</p> <p>A value of INR greater than WEIG means that the point has a distance from the origin greater than the average distance, and that its behaviour is quite particular (it must be remembered that the origin represents the system's average behaviour). A more careful inspection is needed in order to identify the profile components involved; i.e., in which aspects the point is peculiar. It must be kept in mind that the distance between a point and the origin is a measure of the global difference between the point's behaviour and the overall average behaviour of the system (e.g., if each row represents a district, the cloud's centre of gravity represents the country's average behaviour).</p> <p>A Correspondence Analysis always assumes as reference the average behaviour of the whole system, and analyses in which way and how much the various units differ from it (and from one another). This admits a straightforward interpretation in most cases.</p>
DIS	is the square of the distance of the point from the origin. The greater DIS , the more the point's profile globally differs from the system's average behaviour, represented by the cloud centre.
FAC1	is the co-ordinate of the concerned point on the first factorial axis.
REL CON	relative contribution (of the factor to the point): this is the fraction (x 1000) of the point's inertia explained by the factor.
ABS CON	absolute contribution (of the point to the factor's variance): this is the fraction (x 1000) of the factor's inertia contributed by the considered point.

The files written by ACORR

- **ACORR.OUT** is the output file, to be printed and interpreted.
- **COORRIG.LV** (written on user's request) contains the factor co-ordinates (together with some more information) needed by NONGER to cluster the units.
- **COORCOL.LV** (written on user's request) contains the factor co-ordinates (together with some more information) needed by NONGER to cluster the variables.
- **ACORR.FPL** (written on user's request) contains the information passed over to FACPLAN to display the projections onto factor planes.

Chapter 7 – The Analysis Menu: Clustering

7-1 Some notes on numeric classification

The purpose of a numeric classification is to group sufficiently similar statistical units into a limited number of groups (also called **classes** or **clusters**). The *similarity* between two units can be directly observed (e.g. in a survey, by asking specific questions) or it can be computed on the basis of a set of observed variables that offer a suitable description of the concerned units.

Consider for example the provinces in a country, described by the series of their per capita income over some years. Which districts have similar behaviour? There is no *absolute* answer: the result depends on the method used, and includes some elements of subjectivity. For example, we could perform all possible pairwise comparisons of districts and rank pairs in order of decreasing perceived similarity.

The similarity depends upon the variables considered; i.e., it is relative to the particular description. Two provinces can have very similar demographic structure, but they can be very different for what concerns the educational or occupational levels.

The similarity level of two units can be defined in several ways.

Consistently with the geometrical representation adopted so far, according to which each statistical unit is considered as a point in a space that has as many dimensions as there are active variables, we will adopt also for clustering the same notion of distance already introduced for the factorial analyses: an Euclidean distance (after standardisation) for quantitative variables (treated with **ACOMP**); a **chi-square distance** in the case of qualitative descriptions (dealt with by **ACORR**). The **distance** is a **complex indicator** that takes into account contributions coming from all the variables. We conventionally assume it as an indicator of **dissimilarity**, and consider two units more similar than two others when their representative points are closer to one another in R^p than the representative points of the other two units. This seems a good assumption, on which there can be consensus.

Even if we agree about the definition of similarity, some other operational problems arise:

- how can we measure the optimality of a partition, and how can we compare partitions with the same number of classes and decide which is the best?
- how many classes should we construct? How can we be sure that this number fits the structure of the set to be clustered?
- which clustering algorithm should we adopt?

We can identify two large groups of clustering methods, known respectively as **hierarchical** and **non-hierarchical**. Both methods work iteratively: they repeat a given sequence of operations that depend on the selected algorithm, until a final configuration is reached. Both have advantages and drawbacks.

The **Ascending** (aggregative) **Hierarchical Methods** perform iteratively the following operations on a set of n elementary units or groups already formed:

- compute the similarity for each pair of items;
- merge the two most similar items, thus reducing their number to $n-1$.

At starting there are as many groups as there are elementary units, each consisting of exactly one unit. At the end of the process, after $n-1$ aggregation steps, all the units are grouped in one large cluster. An acceptable solution, with the elementary units regrouped in a number of clusters small enough to guarantee synthesis and large enough to save a consistent fraction of the information, lies somewhere between these two extremes.

The aggregation process is graphically represented by means of an aggregation tree.

The units to be aggregated are shown on the left, at the base of the tree (figure 7-1). From left to right the units are progressively aggregated, at a distance proportional to their dissimilarity. A partition can be obtained by cutting the tree vertically at some intermediate level. Moving to the right the number of the resultant classes becomes smaller and smaller, but also smaller becomes their internal homogeneity.

A compromise criterion is needed to decide how to cut the tree conveniently.

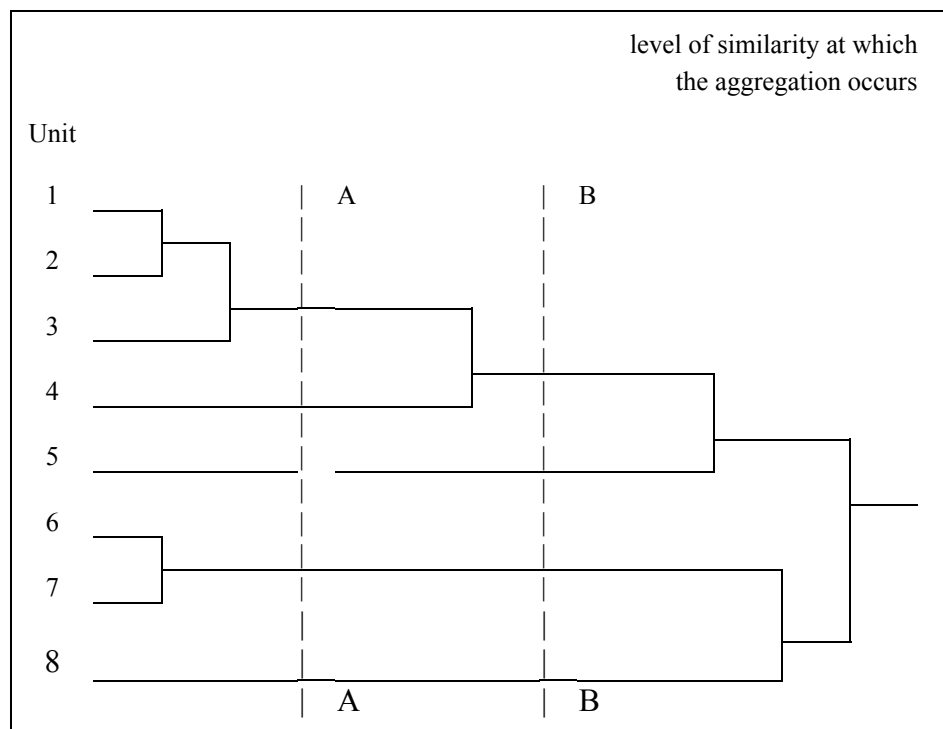


Figure 7-1 A hierarchical aggregation tree. Eight units are clustered, and the number of the resulting groups decreases moving to the right of the tree, corresponding to an increasing dissimilarity. The section AA produces a partition in 5 clusters; the section BB produces 4 clusters

A hierarchical procedure is convenient when the number of units does not exceed some tens. If there are more (let us say, over a hundred) the aggregation tree becomes difficult to read. Beyond this, if n is the number of groups currently existing, $n(n-1)/2$ similarity values must be computed at each step: the time needed grows quadratically with n .

Another severe drawback of these methods is the irreversibility of the choice made at each step: when two units are joined, this is forever. Anyway, the algorithm selects the pair to be joined only on the basis of local considerations, with no global concern: it is like a chess player that steadily chooses the move which gives the best immediate result, with no consideration for what can occur in next moves, i.e., with no concern for strategy. In the case of hierarchical methods, the overall aggregation path might sometimes evolve in a more satisfactory way if an aggregation other than the locally optimal one were selected at the current step. As a consequence of this, a partition obtained by cutting the tree at any intermediate level is usually far from optimal.

Non-hierarchical methods

An initial partition with the user-requested number of classes is determined somehow. Its quality is then improved iteratively, by moving some element from one to another class when this increases the value of the **objective function**, which is a suitable measure of the partition optimality.

The process continues until a final configuration is reached that cannot be further improved through local re-assignment of units. The partition obtained is *a local optimum*: this means that small changes in the allocation of the units to the groups cannot improve it. However, some better partitions with the same number of classes might exist, unreachable from the current one via a small change.

The partition eventually obtained depends on the starting configuration assumed and on the number of the requested groups.

Some definitions

When we introduced the representation of a set of units as points in a multidimensional space we assumed the Inertia, defined in section 4.4, to which all units contribute, as a measure of the overall variability of the data table (or of its information contents). We will here act consistently with those concepts.

Let $In_{tot} = \sum_i m_i * d_i^2$ be the total Inertia of the cloud with respect to its overall centre, and let us consider a generic partition of the cloud in k clusters. Each unit belongs to one and only one group. Let G_j represent the centre of the j -th cluster: its co-ordinates are the average values of the p variables, computed keeping into account only the units belonging to the class.

The generic class j of the partition has an **Internal (or intra-class) Inertia** defined as

$$In_{int}(j) = \sum_{i \in I_j} m_i * d^2(i, G_j)$$

where the sum extends only to the units belonging to the j -th class and the distances are computed from the class centre G_j .

The **internal inertia** of a class is a measure of the dispersion of its elements about the class centre. A good partition should consist of groups as homogeneous as possible, i.e. with a low internal inertia.

The **Internal Inertia of the partition** (also called Inertia Within Classes) is the sum of the internal inertia of all the classes. Its value should be as low as possible, which means that all the classes should be as homogeneous as possible. The average characteristics of the units included in a class are represented by the co-ordinates of the class' centre G_j .

The purpose of clustering is to offer a simplified view of a phenomenon, in which all units belonging to the same class are identified with the class' centre, neglecting as irrelevant the differences existing amongst them.

The initial cloud is thus reduced to a new cloud formed by the k centres of class, spread about the cloud's global centre. Its inertia is the **External Inertia** of the partition (or Inertia Between Classes):

$$In_{ext} = \sum_j M(j) * d^2(G_j, G)$$

where $M(j)$ is the mass of the j -th class (equal to the sum of the masses of the units belonging to it) and $d^2(G_j, G)$ is the square of the distance between G_j and the overall centre G .

Let us suppose that the cloud has already been somehow split into k clusters, whose centres are G_1, G_2, \dots, G_k : the well known Huyghens' theorem proves that the Total Inertia can be decomposed as follows:

$$In_{tot} = In_{ext} + In_{int}$$

where In_{tot} represents the cloud's Total Inertia, In_{int} and In_{ext} are the Internal and External Inertia defined above.

The **objective function** assumed for non-hierarchical clustering in ADDATI is

$$\max (In_{ext} / In_{tot}) \quad \text{equivalent to} \quad \min (In_{int} / In_{tot})$$

corresponding to a set of clusters which are globally as compact as possible. The value of the objective function varies between 0 and 1 (the highest the best, if the number of the clusters is the same).

In a hierarchical aggregation there are at start as many clusters as there are units. For this situation, $In_{ext} = In_{tot}$ and $In_{int} = 0$.

When the units are progressively aggregated the Internal Inertia goes increasing, while the External Inertia decreases exactly of the same amount. When at the end of the process all the units are regrouped in one class, $In_{int} = In_{tot}$ e $In_{ext} = 0$. At each step, those two units are joined for which the unavoidable increase of the Internal Inertia is minimal.

Diday's Clustering Method (dynamical clouds)

The clustering strategy used in ADDATI is quite complex, and will be illustrated in some steps. An important component is the non-hierarchical clustering method proposed by Erwin Diday in 1971.

Diday's method requests the user to enter the number of the groups that should be constructed (tentatively, the number of groups he would like eventually to obtain) and to provide in one way or another an equivalent number of points $\{S_1, S_2, \dots, S_k\}$ in the feature space, to be assumed as initial centres (or **seeds**) for the aggregation procedure.

The method repeats iteratively the two steps shown in the figure 7-2. The distance from all k seeds is computed for each unit, and the unit is assigned to the class associated with the closest seed. A provisional partition with k classes is generated: each unit belongs to one and only one class.

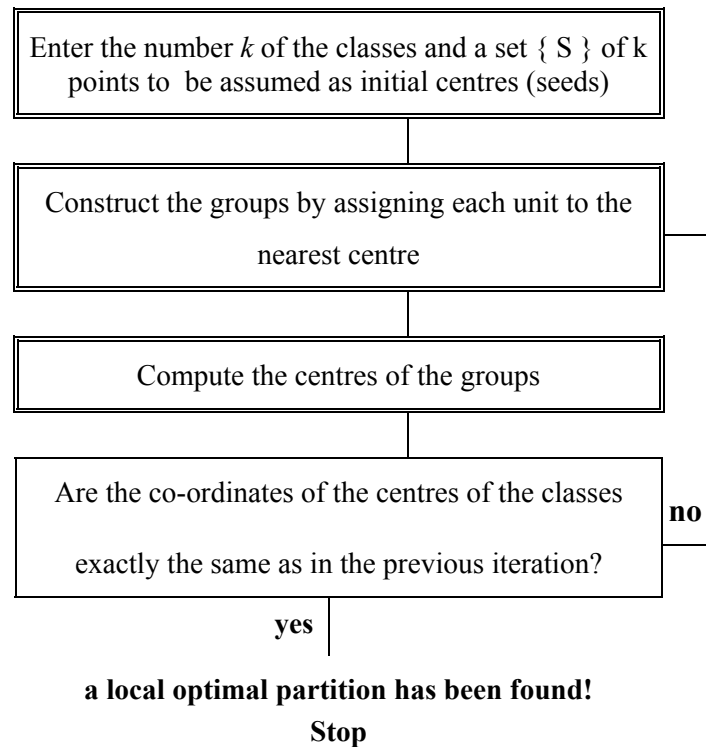


Figure 7-2 Scheme of Diday's non-hierarchical clustering algorithm.

The centres of these classes are computed and replace the initial centres; the assignment procedure is then repeated, the centres are recomputed, and so on. At each iteration some units change class, until a stable configuration is reached.

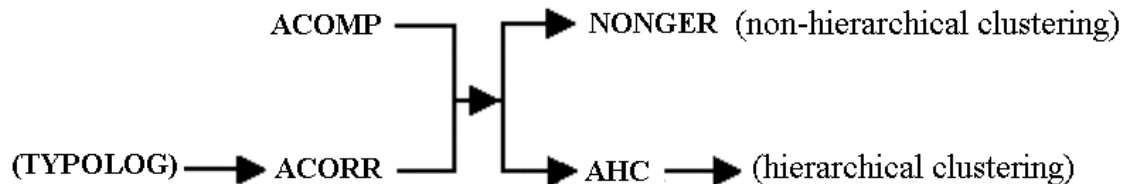
It can be proved that with each iteration *the Internal Inertia of the partition cannot increase* (it actually decreases, otherwise a minimum is reached and the procedure stops). This means that the groups become more and more compact.

The final partition corresponds to a minimum of the Internal Inertia \mathbf{In}_{int} or, because of Huyghens' theorem, to a maximum of the External Inertia \mathbf{In}_{ext} . It is only a *local optimum*: this means that the partition cannot be improved by changing the assignment of a few units, but might be improved by a more radical re-attribution. We are never certain to have found the global optimum, i.e. *the best* of all the possible partitions with that number of classes: owing to the size of the problem, such certainty would generally require an enormous computing time.

Once the number of the groups has been chosen, the final partition depends only on the set of the initial seeds $\{S_1, \dots, S_k\}$, as the algorithm is totally deterministic.

7-2 The clustering methods in ADDATI

7-2.1 – Some general notes



ADDATI includes both a hierarchical and a non-hierarchical analysis path. Both of them are based on the theory briefly illustrated in the preceding sections, and are respectively implemented in the programs **NONGER** and **AHC** (Ascending Hierarchical Clustering).

The non-hierarchical clustering method

Diday's algorithm, that iteratively re-assigns the units to the groups in order to achieve an optimal partition, requires from the user an initial decision about the number of the groups. When the number indicated by the user mirrors poorly the structure of the set to be segmented, the quality of the produced partition is generally unsatisfactory.

Besides the obtained partition, that represents a local optimum, not the global one, depends on the choice of the *seeds* (the initial centres) around which the units are aggregated according to a criterion of minimum distance. Various strategies can be conceived for the choice of the seeds, but in general if they are modified the result changes.

In order to cope at least partly with such problems, ADDATI implements a classification strategy that uses in an integrated way both hierarchical and non-hierarchical procedures. The currently implemented analytical path represents the evolutionary result of a different strategy used in past, indicated here as *Method 1*. Here we will limit ourselves to describe with some detail the two methods (the one used in the preceding versions of the package and that currently used) and their relationships.

Instead, we think that the Ascending Hierarchical Classification, implemented in the program **AHC**, is rather simple and direct and does not need a detailed explanation.

The input can be a table of quantitative description, a contingency table or a table of factorial co-ordinates saved by **ACORR** or **ACOMP**.

7-2.2 – Non-hierarchical Clustering: the Method 1 (no longer implemented in ADDATI)

In order to issue a satisfactory partition this method proceeded in two steps: an **exploratory phase** yielded information about the most suitable number of groups and suggests a good choice of the initial seeds; a successive **optimisation phase** produced the final optimal partition

Method 1 – The exploratory stage

Instead of one, several partitions (say, some tens) are constructed with exploratory purposes. In line of principle the requested number of groups (the same in all partitions) is

the same that one would like to obtain in the final optimal partition. The initial centres are usually randomly chosen (though some alternatives are possible).

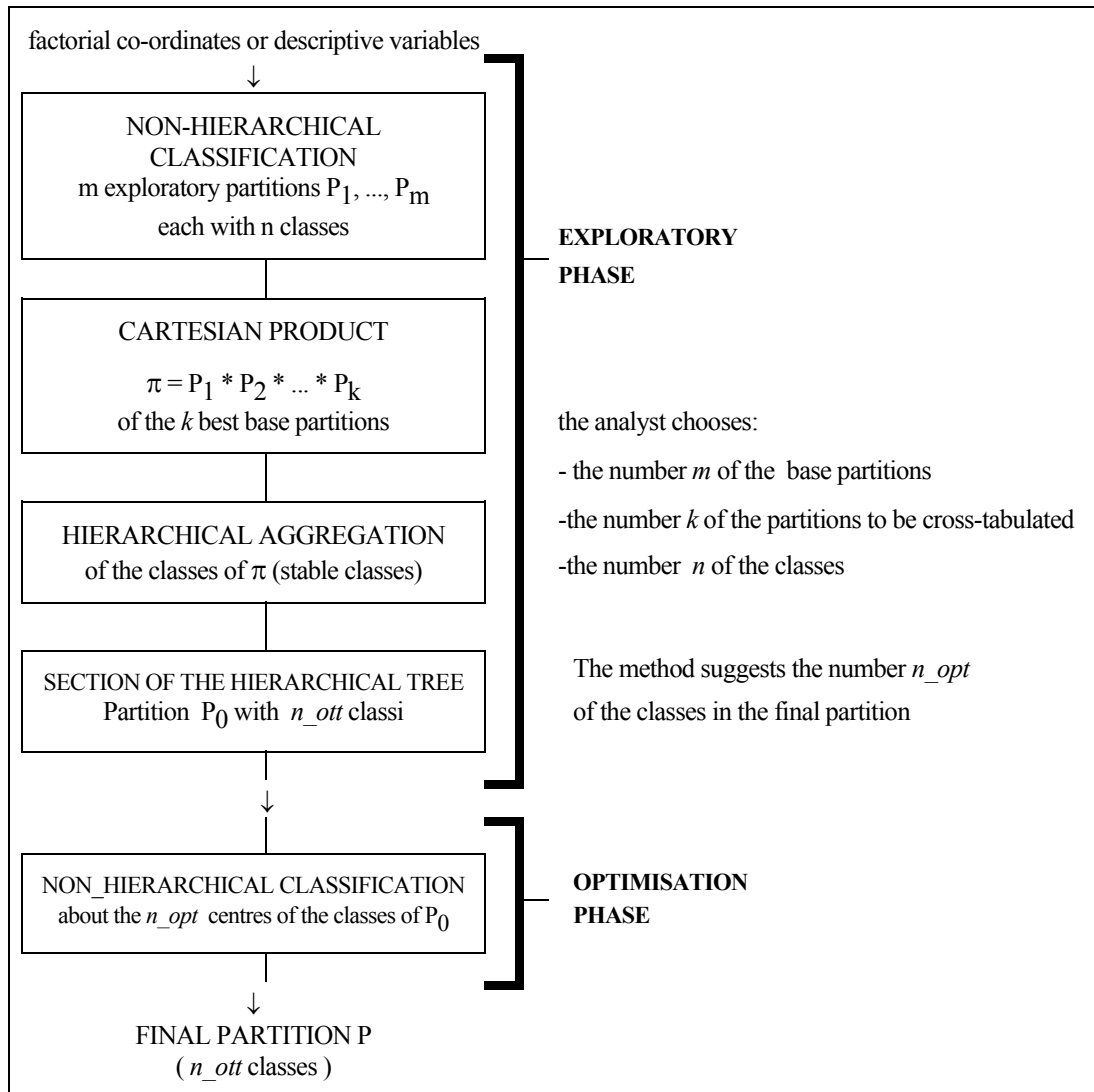


Figure 7-3 Scheme of the mixed clustering strategy previously implemented in ADDATI (Method 1).

The two or three partitions with the highest values of the objective-function (minimal internal inertia of the groups, i.e. maximal homogeneity within the groups), which are the best from the statistical point of view, are cross-tabulated. The number of groups in the **product-partition** is a-priori unknown. By construction, the units in the same group (i.e., belonging to the same cell of the rectangular table resulting from the cross-tabulation) were clustered together in all the cross-tabulated basic partitions. Therefore, we can have a reasonable confidence on their similarity. For this reason, the groups of the product-partition are called **stable classes** or **strong forms**. Even though they are often too numerous for the purpose of the research, they offer a detailed and often exhaustive description of the most important behaviours emerging in the context of that analysis.

After describing the stable classes (listing the units assigned to each of them and computing the average profile of each class, i.e. the average value of all the variables in the class), a routine is called that clusters them hierarchically according to their similarity:

as their number is usually not so high (some tens or so), the use of a hierarchical method is acceptable.

At this point the hierarchical tree was displayed on the monitor, and the user could decide the number of classes in the final partition. To this purpose, ADDATI offered a routine that could compute and save the description of every partition that could be obtained by cutting the tree so as to get a convenient number of classes (at least hypothetically). The analyst could then compare the features of the **candidate partitions** (the value of the objective-function and the meaning of the classes in each of them).

The exploratory phase allowed one to focus on a satisfactory partition (albeit not optimal): the number of its classes – in general not coincident with the one proposed at the beginning of the procedure - represented a useful **emerging information**.

The Optimisation stage

The partition selected after the exploratory phase, whose details were saved to a temporary file, was in general not even a local optimum. It was often possible to improve it by re-assigning suitably to the groups the units close to the borders of the classes. Changes were sometimes minor, sometimes more relevant: it was question of how neatly defined was the structure of the set to be clustered, and of how well the method had worked during the exploratory phase.

7-2.3 – Non-hierarchical Clustering: the currently used Method 2

An intensive use of the package in his research work has convinced the author of the following facts.

- The quite complex sequence of operations in which Method 1 consists *actually succeeds in mitigating the dependence of the resulting optimal partition from the choice of the initial seeds*. However, the result is still biased by the number of classes initially requested: the program is forced to use that number when computing all the exploratory partitions, even though it does not fit well the set's structure. This may have a negative impact on the results of the overall procedure.
- The partitions obtained by suitably cutting the hierarchical tree that represents the sequential aggregation of the stable classes can be far from optimal. It is true that the partition eventually selected is later optimised running **NONGER** once more, but the tree from which it is extracted does not represent a sequence of optimal partitions, and therefore it does not allow one to determine with certainty how many classes it is convenient to retain.

The strategy illustrated in the figure 7-4 is at least partially different. Again, a rather high number of exploratory partitions is computed, still with a number of classes tentatively requested by the user, and the best performing partitions are crossed-tabulated to create the stable classes.

The product-partition issued, that almost always consists of a too high number of classes when confronted with the purpose of research, is assumed as the starting configuration for optimising. It has been constructed in such a way that its groups should represent in some detail the different behaviours emerging in the set to be clustered.

At this point, two routines are called iteratively: the first optimises the current partition and saves to a temporary file an essential description, sufficient to reconstruct it easily;

the second decreases by one the number of the groups by merging the two most similar ones. This continues until all groups have been merged.

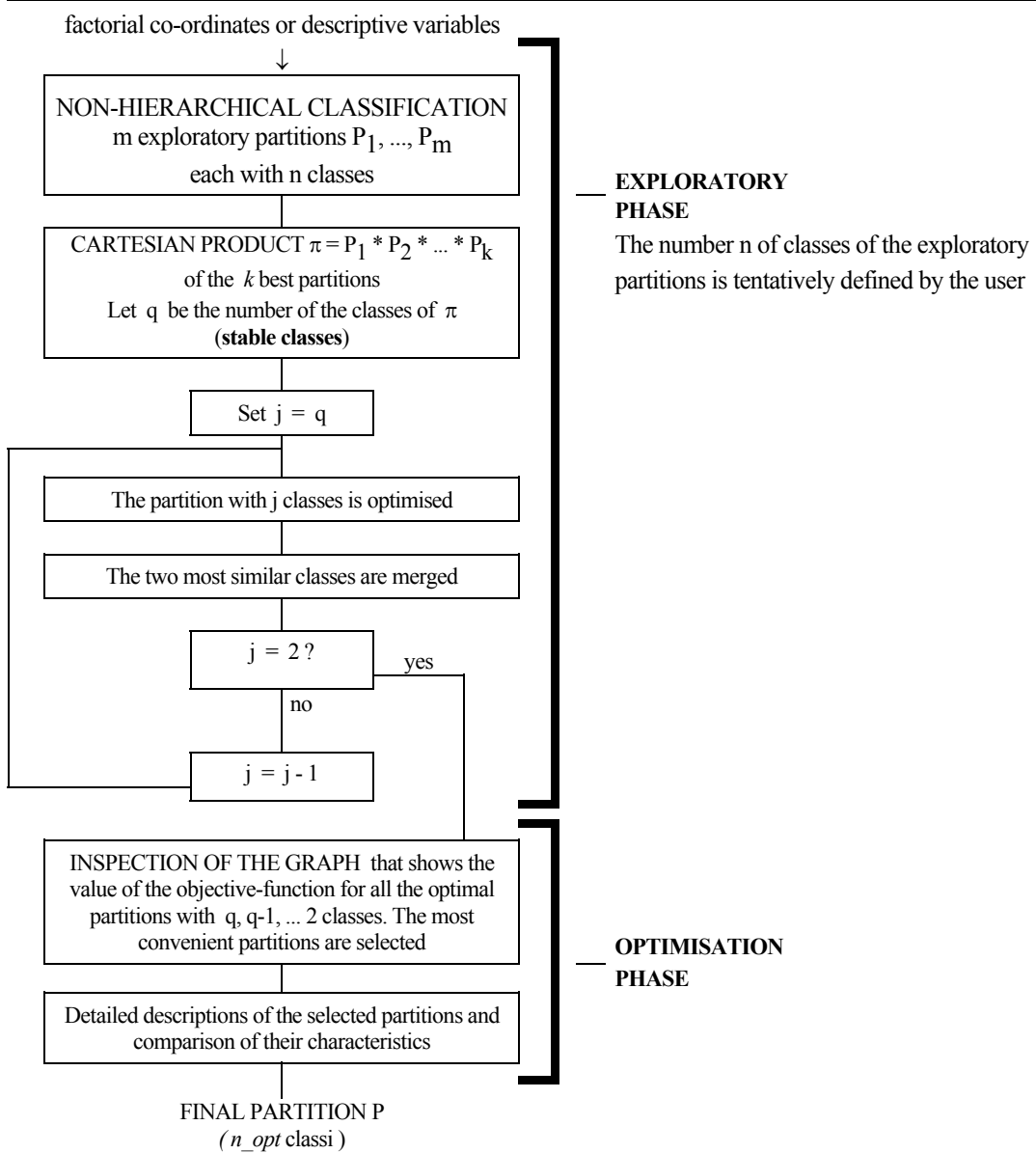


Figure 7-4 Scheme of the clustering strategy implemented in ADDATI (Method 2).

A graph is automatically displayed on the screen, that plots the value of the objective function vs. the number of classes in the partition (remember that in this case **all partitions are optimal**). It is so possible to focus on the most promising, i.e. those for which the loss of Inertia is large when the number of groups is further decreased by one unit. On user's request, the selected partitions are completely described, and their comparison leads to the final selection.

In Method 2 the overall sequence is carried out by **NONGER**; and this is operationally more convenient with respect to the preceding method, that needed the consecutive use of three programs, (**NONGER**, exploratory phase - **PROFIL** - **NONGER** optimisation phase). However, this difference is only formal and could have been easily overcome, in that also the method 1 could be compacted in one single program.

The substantial difference is that in Method 2 the initial arbitrary choice of the number of classes of the exploratory partitions is used *only to construct the cross-tabulated partition*, i.e. the starting configuration for next step, in which a sequence of partitions with a progressively decreasing number of groups are optimised.

The number of the classes in the partition finally selected - which is chosen after inspecting the graph that displays the values of the objective function and after comparing the features of the candidate partitions if there are more than one - should really represent an intrinsic property of the set to be clustered.

Of course, we can still not be sure to have attained the globally optimal partition with a given number of classes (the so called **optimum optimorum**) and probably we have not. Both methods are heuristic, and yield a partition of good quality, not the absolutely best one. But this is well known for combinatorial reasons, and we have to accept it.

The first method might also produce by chance - for a given number of classes - a final partition with a higher value of the objective function than the other method. You can never say. But if the purpose is to have an idea on the number of clusters that better fits the set's internal structure, then Method 2 - that produces **a sequence of optimal partitions** - appears more reliable.

7-3 The programme NONGER

Use	It computes an optimal partition of the set of units to be clustered, using a non-hierarchical method.
Limits	In order to speed up the computation the data table is stored in central memory. NONGER is a 32-bit DOS program, running under a DOS Extender, capable to use all the available core memory.
Advice	In the exploratory phase better to request the cross-tabulation of no more than three base partitions (see later on). For what is possible, do not exaggerate also with the number of the classes you request.

Exploratory phase

Here the questions asked by the program are briefly reviewed and explained. We will include between horizontal lines the questions that depend from the answers already supplied.

We can divide the questions in two blocks:

- questions necessary to load data correctly;
- questions asked by the program after loading the data and aimed at getting the parameters that control how the clustering is carried out.

A. Questions enabling the program to read the data file

Is this classification linked to a Factorial Analysis?

1. no – autonomous analysis

2. yes – the input is a table of factorial co-ordinates written by ACORR o ACOMP

The purpose is to inform the program about the structure of the input data file, and its contents. If it is a file of co-ordinates written by a preceding factor analysis, it includes information that should otherwise be entered from keyboard.

The profiles of the groups are always expressed in terms of the initial variables.

Autonomous analysis

The statistical units (the rows of the analysed table) are directly clustered on the basis of the values included in the initial description file. **It is not possible to cluster the variables** (i.e., the table's columns).

In this case the user must supplied all the necessary information on the analysis to be performed (number of units and variables, their labels, etc.), by answering to specific questions asked by the program.

Execution linked to a Factorial Analysis

The similarity between units is computed from their factorial co-ordinates, loaded from file. Ignoring the last factors, that usually have a low explanatory power, speeds up the computation.

Most parameters are read in from a work file.

In this case it is also possible to cluster the variables according to their similarity over the statistical units (i.e., according to their correlation).

The answer to this question causes a bifurcation in the sequence of questions that follow. If the clustering is based on data issued by a factor analysis, skip the questions in block *A1* (*appropriate for an autonomous analysis*) and go straight to the block *A2*.

Block A1 (for a classification directly carried out on the descriptive variables)

The parameters for this analysis will be

- 1. entered from keyboard**
- 2. read from file NG.PAR, where the values relative to a previous analysis have been saved and can now be suitably adapted.**

Each time an autonomous analysis is carried out (i.e., an analysis that assumes as input a set of directly observed descriptive variables, without going through a factorial analysis), the sequence of the answers entered by the user is saved to a file named **NG.PAR** and can be later retrieved at will. This avoids the tedious operation of re-entering parameters when for any reason an analysis is repeated or slightly modified. The user can review or modify the parameters with **F1** or **PgUp**. Each time a change is made, the consistency among the new values and all the other is checked and new questions are asked if necessary.

The user can quit the modification phase with **F2**.

The parameters loaded from the file NG.PAR include all the answers to the questions in block A1.

Name of the input data file:

Each record (or block of records) in the file must refer to a statistical unit and contain the value of the variables that describe the unit, optionally preceded by the unit's label and/or its weight.

The file can also reside in another folder: in this case, the complete path necessary to trace it must be provided.

Total number of the units to be loaded from the data file :

The answer must also include the supplementary units, if any (see later on). Each unit corresponds to a constant number of records (usually one) in the input file. Exceeding records will be ignored.

Total number of the variables :

This is the number of all the variables - **both active and supplementary** - that are to be read from the data file. If the file contains some extra-variables, the format (see below)

will specify which are to be loaded. **The unit's label and weight, if read from file, must not be counted as variables.**

Number of the supplementary units (rows) :

The supplementary records - if any - **must follow** the active ones in the input file. Enter "0" if no supplementary unit is to be loaded.

Apart from the **active units** (rows of the table), on whose similarity the partitions will be computed, it is also possible to introduce into the analysis a set of so-called **supplementary units**, that have no part in constructing the partitions, but which are assigned to the most similar group. The purpose is to draw extra information by examining how these units are located with respect to the active ones.

For example, if the active rows describe the behaviour of a group of districts in a given year, the supplementary rows can describe the same districts at a different time, thus permitting a qualitative interpretation of the changes occurred.

Remember: Active and supplementary objects must be described by the same variables (the columns of the table).

How many SUPPLEMENTARY variables?

The number of the variables to be dealt with as supplementary must be entered, if any (0 = all variables are **active**). Active and supplementary variables can be intermixed in the input record: in order to single them out and deal with them suitably, the user will be later prompted to enter the ordinal number of the supplementary ones.

The **supplementary variables** (if any) describe the units, but have no part in constructing the classes. For this, only the global similarity of the units with respect to the **active** variables is taken into account.

The alphanumeric labels for the units

1. will be entered from keyboard

2. will be loaded from the data file (if a free format is used, each the label must lead its record).

The program uses an alphanumeric label (up to 12 characters long; must not include commas or blanks) to identify each unit. This question is aimed at letting the programme know if the labels will be supplied from keyboard, or will be loaded from the input data file. In the latter case, the format will specify the position of the label in the record. If a **free format** (see later on) is used, each label must lead its record.

Enter the labels for the variables ::

Each variable (active or supplementary) is identified by an alphanumeric label. Each label must not be longer than 12 characters, and cannot include commas or blanks.

The labels must be entered from keyboard in the same order the variables are read from file, separated by commas or blanks. A compact form is allowed.

Example: *To assign some generic labels to 4 income and 5 age classes, instead of*
income1 income2 income3 income4 age1 age2 age3 age4 age5

the answer

income1/4 age1/5

is accepted and expanded.

Any error (too many or too few labels, or labels unacceptable for any reason) is rejected and can be corrected.

⇒ *The following question is asked only if before it was specified that the units' labels are entered from keyboard:*

Provide the labels for the statistical units (objects) :

They follow the same rules as the variables' labels. Also in this case a compact form is allowed.

Example: *In order to label 150 active and 50 supplementary units without listing their names one by one, the user can enter:*

active1/150 supplem1/50

and the string will be automatically expanded into the 200 labels

active001....active150 supplem01....supplem50

⇒ *The following question is asked only if before it was declared that some variables are to be treated a supplementary :*

Enter the ORDINAL NUMBERS that identify the SUPPLEMENTARY amongst all the variables loaded :

The ordinal numbers of the supplementary variables are necessary to recognise them among all the variables read from the data file. The sequence of the ordinal numbers refers only to ***the variables actually loaded***; other variables included in the input file but skipped (be means of a suitable reading format) do not count. Here too a compact form is allowed.

Example: *the string "2/5 12, 13, 20 "means that amongst all the variables that have been read, the 2nd, 3rd, 4th, 5th, 12th, 13th and the 20th are supplementary. Obviously, in this case the user must have declared exactly 7 supplementary variables and at least 20 variables globally..*

The table submitted to the analysis is:

1. a table of quantitative measures

2. a table obtained by aligning side-by-side one or more contingency tables.

The program needs to know the type of table in order to use the appropriate metrics when computing the distance (assumed as a measure of dissimilarity) between the units.

For a descriptive table of **quantitative variables**, the (Euclidean) distance is computed by means of the usual pythagorical formula after standardising the variables. For contingency tables (see ACORR); a chi-square metrics is used.

This information is also necessary to compute the average profiles of the resulting classes: when the variables are **quantitative** a class' profile gives the average value of each variable in the class, to be compared with the overall average value. In the case of contingency tables, the profile gives the frequency of each variable (or category) in the class.

⇒ *The following question is asked only for a table of measures (quantitative variables) :*

Every statistical unit is to be assigned::

- 1. the same weight**
- 2. a weight read from the data file**

It is well known that each unit can be represented as a massive point in a suitable geometrical space. Therefore, each unit must be attached a suitable weight (or mass).

For a contingency table (see ACORR) there is no need to provide weights exogeneously: they are computed endogeneously from the input data by the programme itself. For a table of quantitative variables, the weight of each unit can be supplied exogeneously.

***Note:** For a table of rates it is expedient to give each unit its absolute importance by assigning it an appropriate weight..*

Weighted averages and variances are then computed. If weights are used, the correlations between variables are correct; the resulting values are meaningful and represent actual properties of the analysed system. The centre of gravity of each group represent the group's actual average behaviour.

***Example:** If each unit is a district, described by variables computed as rates over the district's population, it is advisable to use the population in each district as its weight.*

The place of the weight in the input record is specified in the **reading format**. If a **free format** is used, the weight is assumed to stay immediately **after** the unit's label (if any) and **before** the variables.

The "*equal weights*" option is for use with elementary objects (individuals, dwellings, etc.). Otherwise, weights are to be supplied as **real** or **integer** values, not necessarily normalised (the program will take care of this). **When declaring the number of the variables, the weight must not be counted.**

⇒ *The following question is asked only for contingency tables.*

How many side-by-side contingency tables are there in the table to be analysed?

A **positive integer** is expected (there is at least one contingency table). Every contingency table represents the way in which **the same elementary units** are distributed over the categories of a qualitative variable.

***Example:** Think for instance of a table, whose rows represent the districts in a region and whose columns show, for each district, how the population is split in 4 age classes and 4 educational levels. In such case, the counted units are the inhabitants of the region and the table is made up by cross-tabulating first the district and the age class, then the*

district and the educational level, for all the inhabitants of the region. The two tables obtained are then set side-by-side.

All the contingency sub-tables **must count the same elementary units**, and must therefore add up to the same total. If this is not the case the clustering is carried out correctly, but the class profiles are generally not appropriate. In this case **it is convenient to modify** the input table by varying **proportionally** the values in each sub-table so as to bring all of them to the same total.

Example: *If one sub-table counts persons, and another counts dwellings, the two totals are generally different. In order to compute the profiles correctly it is necessary to assume one of them as reference - say, the population - and to relate to it the other sub-table by multiplying all the figures in a row by an appropriate value. If a district has 20,000 inhabitants distributed according to their educational level and 7,000 dwellings distributed according to their size, the number of the dwellings in each size class must be multiplied by 20,000/7,000. This way, the row total becomes 20,000, while the percentage distribution is unchanged.*

Note: *If the input data table recodes in disjunctive form a set of categorical variables, the number to be entered here is the number of these variables, each of which has been expanded into a contingency table.*

Supply a FORMAT for reading the data from the input file:

A format is required in order to detail the place and the length of the unit's label, the weight (if any), and the variables previously specified, skipping those not to be loaded.

The chapter 3 explains in detail how a format works and its syntax. Here please just note that a **free format** is declared by entering an asterisk "*".

This means that the input file has been prepared so that each record contains, in this exact order and **separated by spaces**, the following items:

- the unit's label (if read from file)
- the unit's weight (only for a table of measures, and only if it has been previously declared that each unit is assigned a weight read from file)
- the variables

Remember: If this is not the order, or if the items to be loaded are not separated by blanks, a format must be supplied.

At this point, the input data file is opened and read. The parameters are saved to file NG.PAR and can be reloaded if necessary.

Block A2 (Only for a clustering linked to a factorial analysis)

In this case almost all parameters are read from the work file written by **ACORR** or **ACOMP**. Only two questions are asked before loading the data:

Enter a comment (up to two pages) to be used as a title for the analysis. It will head the output file, where results are saved.

If no title is desired, simply hit ↵.

The user wishes to cluster :

- 1. the rows of the table submitted to the factor analysis**
- 2. the columns of the same table**

The information is necessary to enable the programme to open the appropriate file that contains the factorial co-ordinates (COORRIG.LV for the units, COORCOL.LV for the variables) and to compute the profiles of the output classes in case the units are clustered. No class profile is output if the clustering concerns the variables.

This is all for the case **A2**. The data file (COORRIG.LV or COORCOL.LV) is now opened and read.

Now a second set of questions follows, concerning the parameters to control the clustering procedure.

Block B. – Parameters for clustering

How many exploratory partitions?

Indicatively, around ten. A higher number of *base* (or *exploratory*) partitions is more likely to produce some good quality partitions to be cross-tabulated, but requires a proportionally longer computing time (anyway, this is no longer a problem with the modern computers). Memory occupation is not affected.

Obviously, if the set to be clustered is little numerous (a few tens) it is unnecessary to compute many partitions; when the units are some hundreds or thousands, it is useful to increase the number of the partitions, hoping to find a good local optimum.

How many exploratory partitions (the best ones) are to be crossed in order to produce the *stable classes*?

Amongst the base partitions, computed for exploratory purposes, the best ones (i.e. those with the highest value of the objective-function, their number being decided by the analyst) will be cross-tabulated in order to determine the most homogeneous groups emerging from the analysis (the so-called *stable classes* or *strong forms*).

The choice about how many partitions to be cross-tabulated depends upon the level of detail desired for the stable classes. For instance, if some seven-class partitions are requested, cross-tabulating two of them could produce (at least theoretically) up to 49 stable classes (i.e., 49 combinations of the two classes to which a unit has been assigned in the two partitions).

However, the more strongly-structured the set, the more consistent the two partitions will be, and the number of stable classes will be reduced accordingly. They would be seven if the two cross-tabulated partitions were identical.

In order to avoid an excessive fragmentation of the *product-partition*, which would make the interpretation harder, it is usually convenient to cross-tab no more than 2 or 3 base partitions. However, this is a choice based on the user's experience, and also dependent on the number of classes requested for each partition.

How many classes in each partition?

The user is prompted to specify the number of classes for each partition to be computed. This number is tentative, and should represent the ideal number of groups that the user would like to obtain finally.

The number of the classes of the final partition will be eventually decided after a careful inspection of the diagram that shows how the objective function decreases when the number of the classes is reduced.

Options for the choice of the starting clustering centres:

- 1. repeatable random choice**
 - 2. non repeatable random choice**
 - 3. initial clustering centres entered from keyboard**
-

A random choice of the starting seeds is advisable (the programme itself takes care of that), but the user can choose any of the options indicated.

Repeatable random choice

If n units are to be clustered, for each partition as many values between 1 and n are generated as there are classes, and the units with those ordinal numbers are assumed as the initial aggregation centres. The random generation starts from a **fixed seed** (a value that determines the sequence of the numbers drawn randomly). If the analysis is repeated, the same clustering centres are determined and the same sequence of partitions is produced.

Non repeatable random choice

In this case the "seed" varies, as it is determined from the computer's internal clock. Repeating the analysis will therefore generate different aggregation centres and will in general produce different results.

Clustering centres supplied from keyboard

Sometimes it may be useful to try to drive the grouping procedure by assuming some exogeneously chosen units as clustering centres. This can be done when there is some reason to think that this can lead to a neater and better usable partition.

If this is the choice, the user is prompted to enter - for each exploratory partition to be computed- the ordinal numbers of the units to be assumed as aggregation starting centres.

⇒ *The following question appears only if an analytical path has been followed that includes ACORR. In all other cases the information is read from the work file.*

Of how many side-by-side contingency tables does the table to be analysed consist?

The meaning of the question has already been illustrated in the case of an analysis directly carried out from the descriptive variables.

⇒ *The following question is asked only if there are some supplementary units.*

The analysis includes some supplementary units. You can decide :

- 1. to cluster only the ACTIVE units**
- 2. to base each partition only on the similarity relationships between *active units*, and then assign each supplementary unit to the most similar class. In this case the profiles of the classes will take into account only the ACTIVE UNITS included in each of them.**

At this point the requested exploratory partitions are computed. The best of them, in the number requested by the analyst, are then cross-tabulated to determine the *stable classes* (or *strong forms*).

NONGER - Optimisation stage and description of the partitions

Let q be the number of the *stable classes* generated in the exploratory phase: **NONGER** merges the two closest groups and tries to optimise the partition obtained by suitably re-allocating some units. An optimal partition with $q-1$ groups is thus produced. The same operation is iterated on this latter partition, yielding another partition with $q-2$ classes, and so on until eventually an optimal partition with only two classes is determined.

The partitions obtained, with a number of classes ranging from q to 2, are *local* (or *relative*) *optima*.

At this point **NONGER** calls an ADDATI internal utility to display the graph that plots how the value of the objective function for the sequence of optimal partitions decreases when the number of the classes is decreased by merging. The figure 7-5 refers to the Kenya example, in which 10 *stable classes* are determined.

By inspecting this graph the user can focus on one or more promising partitions, with a number of classes within the range he/she would like to obtain and a value of the objective function sufficiently high. Two possible partitions - with seven and four classes respectively - are pointed out in figure 7-5.

To choose, it is necessary to consider the **trade-off** between the level of synthesis achievable (few classes are always more convenient) and the value of the objective function, that represents the rate of information maintained. The number of the clusters should be reduced as much as possible, but the value of the o.f. must not decrease too much. It is advisable to merge further if the o.f. level of the resulting partition (the next on the right along the graph) is still satisfactorily high. Otherwise, the price to be paid in terms of information loss may result unaffordable.

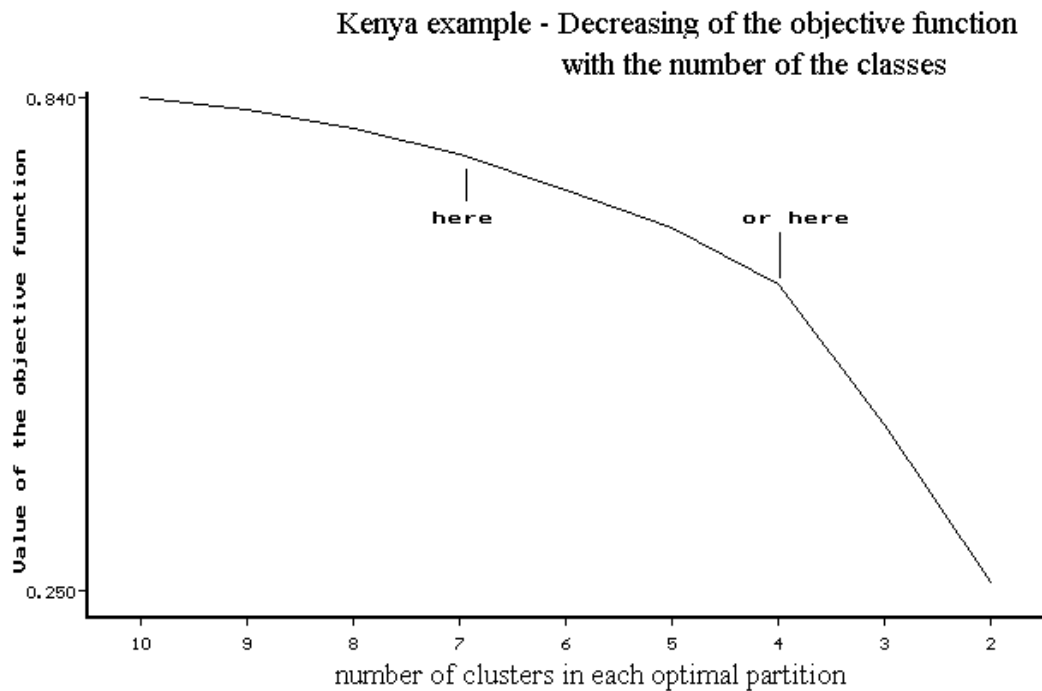


Figure 7-5 The graph of the values of the o.f. when the number of the clusters is progressively decreased by iteratively merging and optimising.

It is therefore convenient to select a partition such that the value of the objective function decreases significantly by merging further (i.e., the slope of the curve changes quite abruptly).

The decision should take into account the three following aspects.

- The number of clusters that better fit the analyst's objectives should orient him/her towards the part of the curve to be more carefully considered.
- The decreasing of the value of the o.f. with each merging step should lead to focus on few *candidate partitions*, to be described in detail.
- The final decision should be taken after examining carefully the features of the candidate partitions – specially their class profiles – choosing the one that better mirrors the objectives of the analysis.

After choosing the candidate partitions to be further investigated, the user closes the graph with the 'EXIT' option in the 'FILE' Menu, and **NONGER** resumes control. The user is prompted to enter the number of classes of the optimal partition to be described. This request is repeated again and again, until all the partitions are described that are necessary to reach the final choice by comparing their characteristics.

Once the groups have been determined, and before writing their profiles to the output file **NG.OUT**, a message like the one in table 7-1 (for quantitative variables) or 7-2 (for qualitative variables) is displayed.

$xm(j,i)$ be the **average value** of variable j in class i
 $xg(j)$ be the **overall average value** of the same variable j .

The variable j is relevant for cluster i when the difference $xm(j,i) - xg(j)$ is far from 0. In order to evaluate its significance, the difference is compared with the variable's standard deviation $\sigma(j)$. The value of the ratio

$$R = [xm(j,i) - xmg(j)] / \sigma(j)$$

is then compared with four *suitably chosen thresholds* s_1, \dots, s_4 whose current values are shown here below. The value of the aid string is then determined according to the following scheme:

--- - ~ ++ ++++ value
 --->|<--->|<--->|<--->|<--->|< of the
 -1.00 -0.20 0.20 1.00 ratio R

Table 7-1 The default thresholds proposed to help interpreting the profiles in the case of quantitative variables.

The + and - signs are used to facilitate the inspection of the class profiles. They are to be interpreted as follows.

The reference is to the **ratio** between the frequency of each variable in the cluster and its overall frequency. With the current threshold values such ratio is represented as follows:

$\begin{array}{ccccccc} \text{---} & & \text{--} & & \sim & & ++ & & ++++ \\ \text{---} > | < \text{---} & \text{---} > | < \text{---} & \text{---} > | < \text{---} & \text{---} > | < \text{---} \\ -1.00 & & -0.20 & & 0.20 & & 1.00 & & \end{array}$

Table 7-2 The default thresholds proposed to help interpreting the profiles in the case of one or more contingency tables.

The class profiles must be compared with the overall profile in order to assess which variables characterise particularly the various classes. The aids are aimed at facilitating the interpretation by quickly focusing the user's attention on the most important variables in each cluster's profile (because their value is significantly higher or lower than the corresponding value in the overall profile). To this purpose, one of the following alphanumeric strings is written under each profile component:

"_", " _", " ~~~~~", "++", "++++"

The string actually written is determined by the **ratio** between the concerned component and the corresponding component of the overall profile. The ratio is **computed for each component** of all class profiles, and compared with a set of thresholds values. The aid string is chosen according to the scheme shown in the two tables 7-1 and 7-2. The thresholds shown there are the programme's default; fitting in most cases.

Note: *If the table submitted to the analysis consists of quantitative variables, the default thresholds are automatically computed by the programme and depend, for each variable, from its standard deviation. While the classes are determined from the factorial co-ordinates or – if the original data are directly processed without going through **ACOMP** – from the standardised variables, the profiles are computed using the original variables and the initial unit of measures. This makes the interpretation easier.*

Profile components

The profile components have a different meaning in the two cases.

- **Quantitative variables** (tables of measure): for each group, a profile component is the average value of a variable in the class; the corresponding component of the overall profile is the overall average of that variable.
- **Contingency tables** (including the analyses on qualitative descriptions starting with **TPOLOG**): a profile component represents *the frequency* in the class of a category of a qualitative variable. The corresponding component of the overall profile is the overall frequency of that variable.

As a general guideline, for quantitative variables the way of determining the aids to interpretation seems suitable: their values should not be modified by the user. For contingency tables it may sometimes happen that the different units (and therefore also the clusters) are very little differentiated and that the default thresholds do not seem appropriate (e.g., in almost no group do the variables' frequencies differ from the corresponding values of the overall profile by more than 20 per cent). The user can then change the thresholds, but notice that in this case they are the same for all the variables and do not depend upon each variable's particular variability, as in the case with quantitative variables. A compromise is therefore often necessary.

On the screen that shows the default thresholds, also the following question appears:

Type :

- 1. to keep these threshold values**
- 2. to change them at will**

In case of change, the user is prompted to enter four new threshold values, **real** and in **ascending order**, separated by blanks.

The values are checked for consistency. If they are accepted, the aid strings are changed accordingly. After writing the profiles to the output file, the following question appears.

The profiles have been described. Now you can :

- 1. continue**
- 2. inspect the profiles and decide whether to change the thresholds**
- 3. change the threshold values**

The user can inspect the output file and decide whether the current thresholds do fit, i.e. whether they represent conveniently the differences existing amongst the profiles of the classes. If this is not the case, the user can change the thresholds and the part of the output file that contains the description of the profiles is overwritten. This process can be repeated until a satisfactory description is achieved.

✉ We shall illustrate NONGER's output with reference to the Kenya example developed for ACOMP (section 6-2). Suppose to have clustered the districts on the basis of seven factorial co-ordinates (that explain 100 per cent of inertia), and enter the following values of the clustering parameters in the exploratory step:

- exploratory partitions to compute: 10
- no. of best exploratory partitions to be cross-tabulated: 2
- no. of classes in each exploratory partition: 7
- repeatable random choice of the initial aggregation centres

In our Kenya analysis the exploratory phase produces 10 stable classes, and the graph of figure 7-5 is displayed. The user can request the description of any optimal partition determined, ranging from 10 to 2 classes.

The results are saved to file **NG.OUT**.

After a summary of the parameters enter by the user to control the analysis, the output file reports the number of iterations that were necessary to achieve convergence for each partition computed, and the final value of the objective function.

For each partition whose description has been requested, the following information is provided:

- a table like table 7-3 that summarises the number of the groups in the partition, the number of units in each group and its weight. Remember that the weight was assumed proportional to the district's population.

10 STABLE CLASSES IN THE CROSS-TABULATED PARTITION											
CLASS	1	2	3	4	5	6	7	8	9	10	TOT
UNITS	10	7	6	6	3	2	2	2	2	1	148
WEIGHT (%)	31.7	19.5	12.8	11.8	2.9	1.3	6.1	7.6	1.9	4.5	100.0

Table 7-3 Kenya example : the stable classes obtained in the exploratory stage.

- a detailed description of the classes (the table 7-4 shows an example)

* CLASSE 6 *

UNITS : 2 WEIGHT: 1.26 %
UNITS ASSIGNED TO THE CLASS :
Kajiado Isiolo
UNIT CLOSEST TO THE CLASS CENTRE (d2 = 0.4111) : Kajiado
UNIT FARTHEST AWAY FROM THE CLASS CENTRE (d2 = 4.8285) : Isiolo
CLASS RADIUS : 1.18697
DISTANCE OF THE CLASS CENTRE FROM THE OVERALL CENTRE : 5.99533

Table 7-4 An example of the information printed for each class

The units included in each class are listed, together with the values of the following indicators:

- **the class' radius**, which is an indicator of class compactness.

Let $In_{int}(j)$ be the internal inertia of the j -th class, obtained by adding up the inertia of all the units belonging to the class, computed with respect to the class centre G_j . This value can be written as

$$In_{int}(j) = \sum_i m_i * d^2(i, G_j) = M_j * d_j^2$$

where M_j is the weight of the class and d_j its average radius, i.e. the distance from G_j at which the class' mass should be distributed in order to have an Inertia equal to $In_{int}(j)$. It can be derived that

$$d_j = [In_{int}(j) / M_j]^{1/2}$$

- **the distance of the class centre from the overall centre** of the cloud, as an indicator of the peculiarity of the class' features: the greater this distance, the more different the average features of the class from the overall mean characters represented by the overall centre

After describing the clusters, some more general parameters characterising the partition as a whole are provided:

- **the total Internal Inertia In_{int}** ;
- **the total external Inertia In_{ext}** ;
- **the overall Inertia**, equal to the sum of the eigenvalues associated with the factorial co-ordinates taken into account when clustering. For our Kenya example this value is 8.0, as we had eight normalised variables, and all the Principal Components had been passed on to the clustering procedure.
- the value of the **objective function**, that measures the quality of the partition and is equal to the ratio between the external and the total Inertia. Its maximum is 1, achieved only when there are as many classes as there are units.

Then the class profiles follow (see table 7-5 as an example).

The descriptions of all the requested partitions are written sequentially to file **NG.OUT**. For each of them, the information on the class to which the statistical units have been assigned is stored to a **text file** named '**NGCLASnn.TXT**', where 'nn' stands for the number of the classes. The format of this file is compatible with ARC/VIEW, that can be used to draw a map representing the classification when geographical units (districts, Census tracts, regions and similar) are clustered.

CLASS	NUM	fert_ rate	ave_ wage	hp_land	act_ rate	cereals	cattle	goa_sh	cash_ crop
1	23	6.804 ~~	69.490 ~~	0.033 ~~	4.092 --	0.138 ~~	0.051 ~~	0.040 ~~	0.150 --
2	6	7.618 ++	59.532 --	0.075 ++	9.385 ~~	0.516 ++	0.049 ~~	0.025 ~~	0.609 ~~
3	4	6.321 --	81.480 ++	0.056 ~~	3.832 --	0.210 ~~	0.389 ++++	0.237 ++	0.000 --
4	3	5.705 --	71.839 ~~	0.041 ~~	3.517 --	0.000 --	0.117 ~~	0.412 ++++	0.000 --
5	2	7.470 ++	40.957 ----	0.066 ~~	9.362 ~~	0.451 ++	0.143 ++	0.078 ~~	2.152 ++++
6	2	5.861 --	126.930 ++++	0.003 --	38.011 ++++	0.000 --	0.001 --	0.000 ~~	0.193 --
7	1	6.920 ~~	67.470 ~~	0.019 ~~	11.090 ++	0.100 --	0.013 ~~	0.017 ~~	2.460 ++++
OVERALL PROFILE	41	6.835	71.217	0.038	7.963	0.191	0.063	0.053	0.428

Table 7-5 Kenya example: the profiles of the optimal partition with 7 classes.

The files written by NONGER

NONGER writes the following files.

- **NG.OUT** lists the units assigned to each class, describes the profiles of the classes and offers some other information useful to the interpretation.
- **NGCLASnn.TXT**, where ‘nn’ is the number of the clusters of the partition to which the file refers (there is therefore a file for each partition described).

These files have as many records as there are clustered units, plus a heading one. Thanks to their format, they can be directly loaded, as text files, by **ARC/VIEW**. Each record contains the identifier of the geographical unit **both as a string and as a number**, followed by the number of the class to which the unit is assigned. The information enables the user to extend the original data archive or – in case geographical units have been clustered – to visualise a map of the classification.

- **NG.FPL** is a file written by **NONGER** for **FACPLAN** only when the analysis path includes **ACORR** or **ACOMP**, that must have saved the file necessary to visualise the projections. If this file exists, **NONGER** reads its contents, adds some information and saves it as **NGnn.FPL**, where ‘nn’ is the number of the clusters. When **FACPLAN** displays its contents, the location of each unit is no longer represented with a small square, replaced by the number that represents the class to which each unit belongs.

Also the centres of the clusters are shown. This is another way of visualising how the classes (visible as sub-clouds of numbers ‘1’, ‘2’, etc.) are located with respect to the variables.

