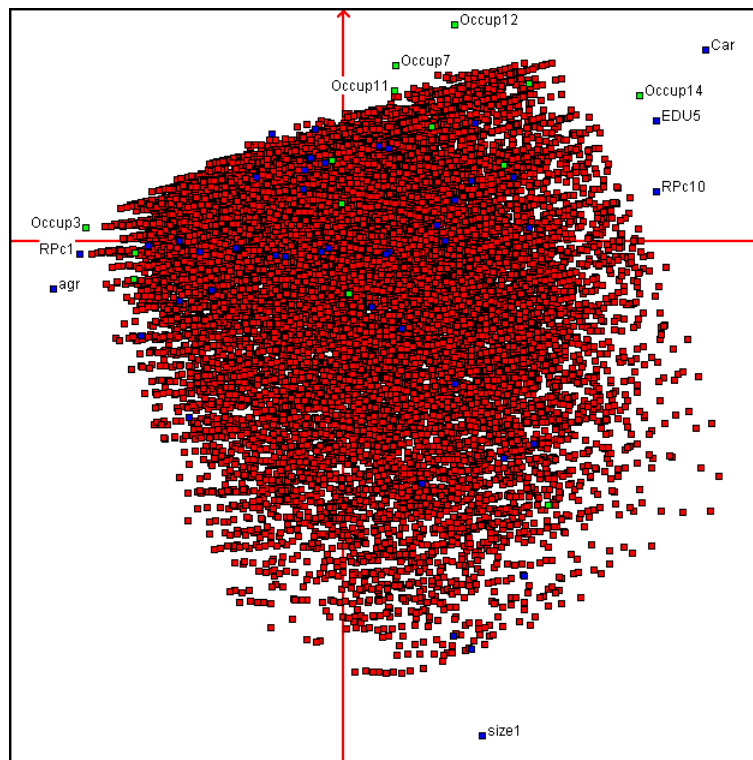


Silvio Griguolo



A Package for Exploratory Data Analysis
(Version 6.0 – February 2008)

User Guide



SILVIO GRIGUOLO ()*

ADDATI

A Package for Exploratory Data Analysis

User Guide

(Version 6.0 – February 2008)

University IUAV in Venice – Dept. of Planning

Chapter 1 – Installation and Generalities

1.1 – Settings

The most recent version of ADDATI for Windows (ADDAWIN) is always available, as a zipped file, from the page

<http://cidoc.iuav.it/addawin.html>

from which also some sample Datasets can be downloaded.

Just create an empty directory and extract to it the contents of the zipped file. *ADDATI is portable*: it does not modify the Window's Registry, and can be run from an USB flash disk.

If you are updating or replacing an existing version, simply overwrite the files to be updated.

The package has **a multilingual structure**. At present it can be used in English and Italian, but it would be very easy to add other languages. All the necessary files reside in the folder where the installation zipped file is exploded; since now, we will refer to it as the “*ADDATI folder*”.

***Note:** The installation language can be changed at any moment when running. This causes all the strings, help, etc. used in the package to immediately switch to the new language.*

Immediately after unzipping the installation file, it is convenient to create on the desktop a link to the management program ADDAWIN.EXE, and [adapt its settings](#) (see further on).

When it is launched, ADDAWIN (ADDATi for WINdows) loads from its initialisation file ADDATI.INI, included in the ADDATI folder, some values that control its default way of working. The programme's settings can be modified, fitting the user's needs. This can be done by editing the text file ADDATI.INI, choosing from the Menu the option *File→Edit/Show a text file*; or by displaying a specific dialog by selecting the option *File→Settings*.

1.1.1 – The initialization file ADDATI.INI

The contents of the INI file are similar, though not necessarily equal, to that shown below. Each line consists of **a keyword (never to be changed!)**, followed by a value used to initialise the programme, that can be adapted.

LANGUAGE	ENGLISH
HELPTYPE	WINHELP
MISSING_VALUE_CODE	-9999
INVALID_VALUE_CODE	-998
DEFAULT_N_CLASSES	6
DEFAULT_THRESHOLD_TYPE	EQUAL_FREQUENCY

Some short comments:

- The keyword **LANGUAGE** specifies the current language. At the time being, the possible values are ‘**ENGLISH**’ and ‘**ITALIANO**’.

- The keyword **HELPTYPE** sets the desired type of help. The possible values are ‘**WINHELP**’ (the help file has the usual extension .HLP) or ‘**HTMLHELP**’ (the file extension is .CHM). The various versions of Windows accept both formats, with the exception of Vista that has banned the .HLP format and only accepts .CHM help files. If Win Vista is your Operational System, you are bound to use the latter, unless you install WINHELP.
- **MISSING_VALUE_CODE** is followed by a code to be interpreted as ‘*missing value*’ for all those Datasets **that do not provide a specific one** in their Documentation file. The code for missing values must not coincide with any valid value, otherwise confusion arises.
- **INVALID_VALUE_CODE** is followed by a value the programme uses internally to mark the invalid values it encounters (out of coding values, unexpected or meaningless characters, etc.). Also this value has a special role, and cannot be assumed by the variables. Therefore, it must be modified if necessary.
- **DEFAULT_N_CLASSES** is followed by the default number of classes used when computing the distributions **of those variables for which no more appropriate value** is specified in the documentation file, or at runtime.
- **DEFAULT_THRESHOLD_TYPE** specifies how the classes of the distributions must be split: its possible values are ‘**EQUAL INTERVALS**’ (the variable’s range is split in intervals with the same amplitude), o ‘**EQUAL FREQUENCY**’ (intervals with approximately the same number of statistical units, or *quantiles*, automatically computed). This default option applies to all variables without specific and more appropriate values.
For any variable this distribution control parameter can be changed at runtime. The user can also freely provide a set of thresholds at will, if he has good reasons to do so, and in case after a preliminary exploration.

Remember

*Multivariate Analysis routines, like PCA (Principal Components Analysis) or NONGER (non-hierarchical Clustering), **do not accept missing values in the data tables submitted to them.** Units for which some values of variables to be used in the analysis are missing **are excluded.***

1.1.2 – Changing the Settings

The Menu option **File→Settings** displays the dialog of figure 1.1, that allows the users to modify the distributions default parameters, the language, the Help file format. The name of the ADDATI internal text editor is only shown for information, and cannot be changed. It was purposefully conceived for the package, and it offers too many advantages to think of changing it.

Remember that the new settings will be used for all those Datasets for which no specific ones are provided in the documentation file, or at runtime.

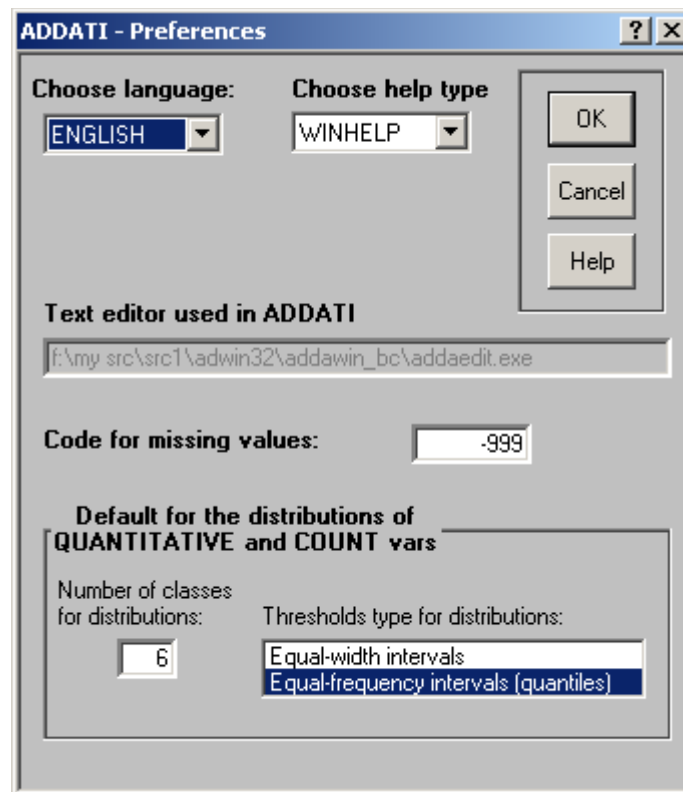


Figure 1.1 – The Settings Dialog

Note: The strange ADDATI icon 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...

1.2 – Some notes on how the programme works

The ADDATI Menu offers the three sub-menus shown here below.

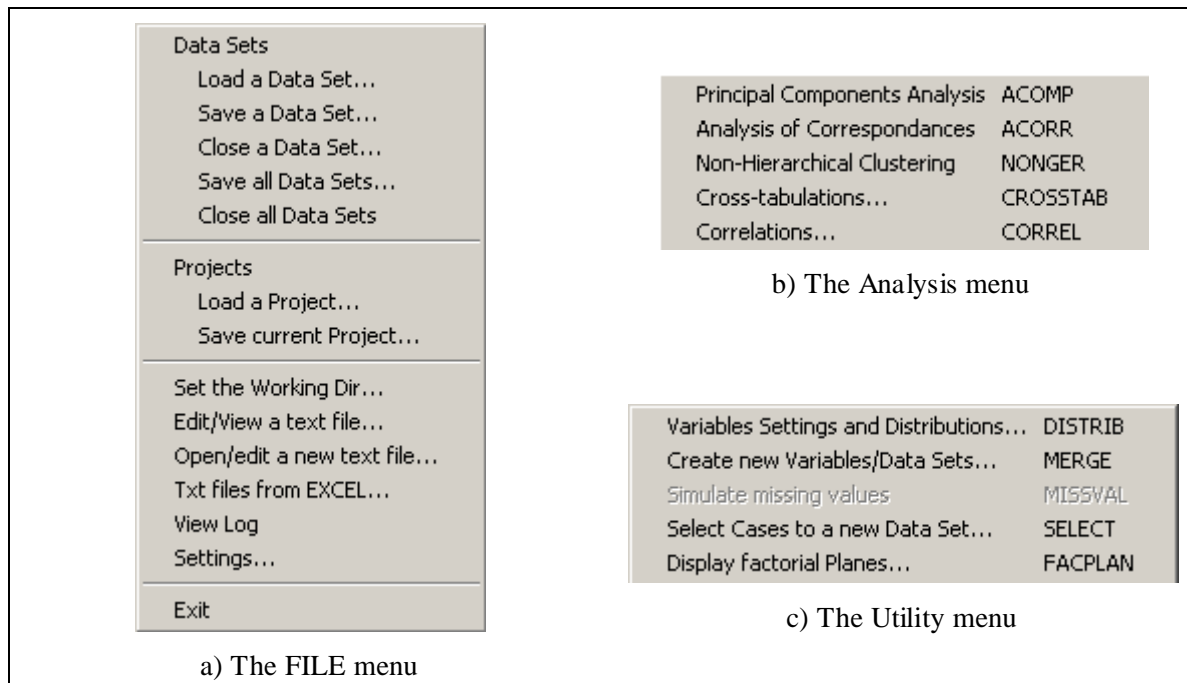


Figure 1.2 – ADDATI's three menus

Some items are disabled until at least one Dataset is loaded.

1.2.1 – The Datasets

A Dataset consists of

- a file of data;
- a documentation file that describes its contents;
- a **label** (a **name**) that identified it amongst the other Datasets loaded.

Any number of Datasets can be loaded at the same time. However, remember that operations that use more than one DS (like the creation of a new DS, obtained by copying or recoding existing variables, or creating some new ones according to various options) can obviously involve only DS that describe the same statistical units.

Data and documentation files must be in text format. Their structure is described in detail in chapter 3.

Data files

Each record consists of a set of *variables* (or *indicators*), that describe the statistical units (persons, households, geographic units, etc). The default (but not compulsory) extensions for these files are **.DAT** or **.CSV** (Comma-Separated Values).

In a record values can be separated by spaces, commas or semicolons, known as **field delimiters**. There can also be no separator, and in this case each variable must occupy exactly the same columns in every record, in order to make it possible to distinguish it from the others.

The format without delimiter is accepted only in input. It can be found in very huge files, that record data at the maximum level of disaggregation (e.g., when each record corresponds to one household questionnaire), often distributed by National Census Bureaus or collecting the data from large Surveys.

Documentation files

Data file contents are described by the documentation file associated with it. By default, ADDATI expects it to have the same filename as the data file it describes, and extension '**.TXT**', but the user can point to any file whose contents is appropriate. It can be prepared using any text editor, like Windows Notepad or the ADDATI internal Editor, and following the rules [specified later on](#). A limited help to its creation is foreseen, but not yet implemented.

New Datasets created or modified in ADDATI are saved together with their documentation file, produced automatically.

1.2.2 – The Help on line

Every dialog displays a HELP button, that offers a general descriptions of the dialog's purpose. For a *specific help* on a dialog's control (buttons, edit boxes, etc.) drag onto it the '?' from the top-right corner of the window, or press **F1**.

1.2.3 – Some information always available

Almost all dialogs include some controls - like those in figure 1.3 - that enable a quick exploration of any loaded Dataset.



Figure 1.3 – All dialogs enable the user to explore the open Datasets.

One can visualize the data, their documentation, and some elementary statistics for all the variable of the selected Dataset. This is sometimes quite useful to orient the analyst on how to fill the dialog.

The requested information is displayed in ADDATI's internal editor.

1.2.4 – The internal Editor ADDAEDIT

ADDATI uses its internal editor (ADDAEDIT.EXE, located in the ADDATI folder) for all operations on text files. The editor was written to this purpose, and is well integrated with the programme. For example, after a Dataset has been loaded, and its documentation is known to ADDATI, a request to inspect the data (done by clicking on the *data* button), displays a window like that shown in figure 1.4.

The editor's bottom bar always shows the line, the column, the ordinal number and the name of the variable where the cursor is located. If the type of the variable is [CATEGORIAL](#), also the meaning of its code under the cursor is shown. The information is automatically updated by clicking on another field, or if the cursor is moved using the arrows.

Besides, while carrying out some operations ADDATI sometimes calls the editor to let the user examine a part of the output file that needs attention in order to decide how to continue the analysis.

ADDATI Editor - File: HH-00.bin.dat

File Edit Search Format About

2	5	235.5	14	2	2	56	3	4	1	41	75	2	2	0	1	0	2	2	0	2
2	5	235.5	14	2	2	80	3	2	2	00	00	0	2	0	0	2	1	2	0	1
2	6	235.5	14	2	2	45	2	3	2	00	00	0	1	0	1	2	2	1	0	2
2	7	235.5	14	2	1	28	2	1	1	74	36	1	1	1	3	1	0	2	0	2
2	7	235.5	14	2	1	53	2	2	1	12	22	4	2	0	1	2	1	3	0	2
2	5	235.5	14	2	1	52	2	1	1	61	01	3	2	1	0	0	1	3	0	2
2	5	235.5	14	2	1	43	2	3	1	61	01	4	1	0	0	2	1	2	0	3
2	5	235.5	14	2	1	30	2	4	1	72	28	1	1	0	3	0	0	2	0	1
2	6	235.5	14	2	1	57	2	3	1	52	52	3	2	0	2	2	0	2	0	2
2	3	235.5	14	2	1	64	2	5	1	41	75	2	2	0	0	0	1	2	0	1
2	4	235.5	14	2	2	56	3	5	1	42	92	1	2	0	1	0	2	1	0	2
2	3	235.5	14	2	1	62	2	3	1	83	60	3	2	0	0	0	1	2	0	1
1	7	165.8	14	2	2	62	2	5	1	41	75	2	2	0	0	3	1	3	0	1

Ready

lin : 5 col : 16 field: 5 - Agricultural Indicator (categ): 2 = "Non-agricultural Househo

Figure 1.4 – Data visualisation in the internal editor **ADDAEDIT**

In Chapter 2 some elementary statistical definitions and notions will be supplied, necessary to understand the operations available in ADDATI, and to interpret their results. How to perform such operations is the subject of the chapters that will follow.

Chapter 2 – Some basic notions

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 introduces some elementary statistical notions, particularly relevant for a first inspection of a set of territorial data (check for errors, get an idea of the reasonableness of values), and to understand the functionalities present in the Utility Menu.

Further terminological elements, and some more definitions, preliminary to Multivariate Analyses (Principal Components Analysis, Analysis of the Correspondences, Clustering) will be introduced later on.

2.1 – Multivariate Analyses and scale of measure

When a multivariate analysis is devised, the preparation of the table is certainly the most important and delicate task.

Data are usually organised in a *descriptive table*, whose rows represent a set of statistical units (say, administrative districts, or households, or firms...), described by some indicators (the columns) directly observed or suitably constructed, qualitative or quantitative (e.g., a set of variables relative to socio-economic or demographic characters).

Generally speaking, the variables included in the table should 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 too often necessary owing to data unavailability- but also nothing more, as the inclusion of an inappropriate variable can bias results in an undesired way. The choice of the variables must be based on substantial assumptions, for 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 on 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 described by a line of the data table) are considered according to a *plurality of aspects*, among which some a-priori unknown network of relationships is supposed to exist, that depends on the *particular context of the analysis* (this means that similarly-denominated variables do not have in general the same structure of relationships in India and in Kenya...).

The usual path analytical path explores this network of relationships limiting itself to the linear ones; the dimensionality of the phenomenon is reduced by dropping only a small part of information in an optimal way. The units are then clustered according to their similarity, globally defined taking into account all elementary aspects.

The variables considered can be measured on different scales. The path of analysis that is chosen (the statistical tools that are used), must match the scale on which variables are measured.

When the 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 express all of them according to the same type of scale (option **Utilities**→**Create new Variables/Datasets**). This is often not yet sufficient: in addition, in the case of 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 2-1 - The scales of measure used for the variables.

The table 2-1 lists the types of scale. It is important to be able to recognise which one of them is used for a variable, in order to set up a methodologically correct analysis.

Another important fact for the analyst is to recognise whether the statistical units, each described by a row of the data table, are **directly-observed elementary objects** (persons, households, firms...), or **complex objects**, obtained by **grouping underlying elementary units**, directly observed. This may for example be the case of **administrative units** (Municipalities, Census Tracts) whose description is often obtained by cumulating elementary data derived from *individual household questionnaires*. As the two types of units referred above are different, they often require different elaboration methods.

Quantitative (or continuous) variables

They assume numeric values, expressed in a given unit of measurement or sometimes a-dimensional; arithmetic operations make sense. Per-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.

2.1.1 – Types of Variables

In order to deal with them correctly, ADDATI needs to know the scale of measure of the variables included in the Dataset being loaded. The information must be stored in [the documentation file](#), where for each variable one of the following four types must be specified: QUANTITATIVE, CATEGORIAL, COUNT or ID.

2.1.2 – QUANTITATIVE variables

They take *numeric values*, expressed in an appropriate unit of measure, or sometimes a-dimensional. **Arithmetic operations make sense**. Per-capita income, NDVI values, mm. of rainfall, population, activity rate, the dwelling surface, etc. are examples of quantitative variables.

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

2.1.3 – CATEGORIAL (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 on codes**, except counting their frequency. 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 its scale homogeneous to that of other qualitative variables in the same table. This operation causes the loss of a (usually small) amount of information, but does not change too 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 split in suitable sub-intervals (classes).

The table 2.2 shows an example. 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.

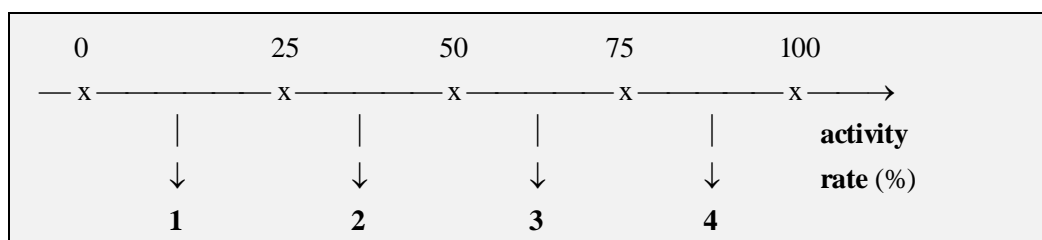


Table 2.2 – Example of the recoding of a quantitative variable into a four-category variable

The above recoding operation is clearly inappropriate in many cases: for example, if the activity rate actually ranges from 17% to 45%, the third and fourth classes are empty. This forces us to define our recoding rule *after determining* the minimal and maximal values for the variable concerned. If we still want four classes and equal length intervals, we get the recoding of table 2.3.

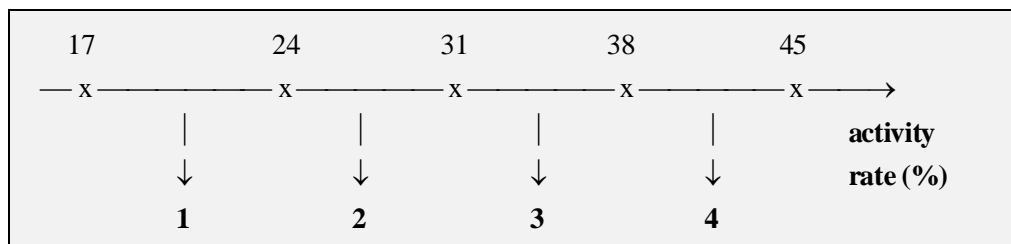


Table 2.3 – The above example, limited to the [17..45] per cent range of values.

Let us now suppose that most municipalities have an activity rate between 25 and 36%: they will fall in the two central classes, and the two extreme ones will 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 differences actually existing amongst the many municipalities belonging to class 2, or to class 3.

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 (quantiles)*. This can be done automatically in ADDATI, when distributions are computed or QUANTITATIVE variables recoded.

Even though numbers are used, the codes '1' to '4' used for the categories have here a non-numeric meaning: 4 is not the double of 2 and 'A' through 'D' would do as well. However, when a continuous variable is split the underlying order is partially maintained: '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'...'n'] to n types of crop: a variable defined as "district's prevalent crop" assumes a value in [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 2.4 shows the values obtained by recoding in binary form the activity rate of table 2.3. As cases are grouped in four categories (classes), four binary variables are necessary.

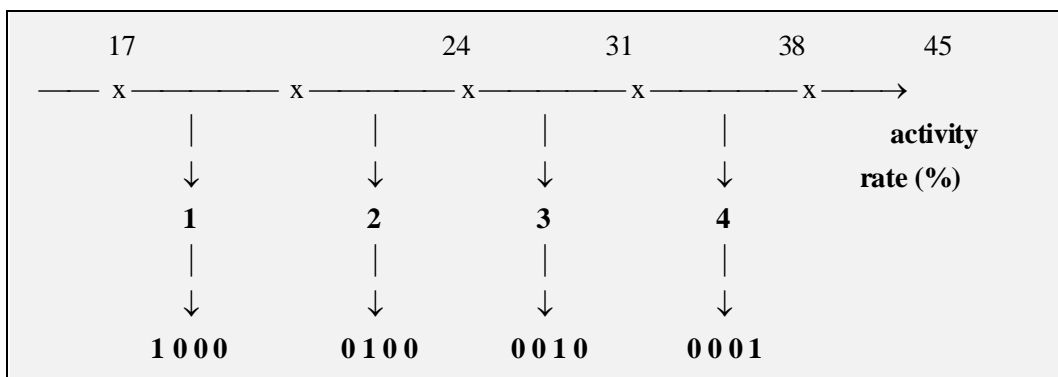


Table 2.4 - The example of table 2.3, followed by a conversion to binary form. Four binary variables are created: the one that corresponds to the category taken by the unit assumes value 1.

2.1.4 – Variables of type COUNT

Like QUANTITATIVE variables they also take numeric values, but these are only integers, not decimals. These variables *count* statistical units (persons, households, dwellings...). For examples, for a set of administrative Districts variables like the number of persons less than 15 years old, or of those with a primary school licence, or the number of occupied dwellings **are of type COUNT**.

The operations that can be carried out on QUANTITATIVE variables hold also for COUNT variables, but the difference is clear.

In general, COUNT variables are *converted into percentages* over an appropriate total before processing them. It makes little sense to compute the *mean of the absolute values* of some COUNT variable (e.g., the population with a high educational level in each District of a Region), because this number is approximately proportional to the District's global population, and its dependence on the differences existing among Districts is only a second-order effect. Instead, *it will make sense to convert for each District this absolute value into a percentage over the District's total population, then compute the mean of these percentages, weighing each of them with the District's population*. Doing so, *the Region's true mean* is obtained.

	Population < 15 years	Population 15-64 years	Population >64 years
District 1	1000	3000	1500
.....
District i	15000	30000	15000
.....
District n	10000	40000	20000

Table 2.5 – Example of cross-tabulation that issues a description in terms of COUNT variables.

Consider the table 2.5. It can be seen as a Dataset that describes n Districts of a Region (the units) by splitting their population in three classes of age (the variables). For each District, these values *count* the persons in each class of age, so they are of type COUNT.

From another point of view, that assumes the inhabitants of the Region as the elementary units, the table can be seen as the result obtained by cross-tabulating, *over the Region's population*, two categorical variables: the District where each person resides, with n categories, and his/her class of age, with three categories.

The fact is general: the frequencies on the categories of a categorical variable, computed at the maximum level of disaggregation, can be considered as COUNT variables in a description that focuses on the aggregate units (here, the Districts).

The difference between QUANTITATIVE and COUNT variables is not too critical: in the practice the two types can often be confused without problems. Yet, this is not always the case, and their difference should be clear.

2.1.5 - Variables of type ID (identifiers of the statistical units)

They take different values in the different statistical units, therefore we call them *variables*, but actually they are *labels*, i.e. *names associated to the statistical units*: the ordinal number, the name or the statistical code of a Municipality are *identifiers*. The cartographic code used to perform in ArcView a *joining* operation between a statistical Dataset and the shapefile that records the spatial location of the statistical units is an ID.

As for some variables both type QUANTITATIVE and COUNT can fit, and the more convenient one is declared, so an ID can sometimes be declared as CATEGORIAL. It depends on the analyst's intentions.

An ID can have a 1-to-1 correspondence with the statistical units: an example is offered by the numeric codes assigned to every Municipality by the National Statistical Bureau, which are all different. Sometimes, however, variables declared as ID are simply variables on which the analyst intends to perform no arithmetic operation. Thus, the Dataset that describes all the Municipalities of a Country usually includes a 'Region' field that can be declared as ID. It is actually *a variable with a certain number of categories*, whose codes are the names of the Regions in that Country. Therefore it could (more precisely, *it should*) be declared CATEGORIAL. From the programme's viewpoint, in both cases the values are loaded as alphanumeric strings; the only difference is that if a variable is declared as ID it cannot be used in distributions and cross-tabulations, while it can if it is declared CATEGORIAL. This is all.

2.2 - Standardisation (normalisation) 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. Therefore, it must 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 well-known 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 in the unit i . We will mostly deal with units to which a **weight** is associated, that represents their relative importance in that particular analysis. In ADDATI different weights can be used for different variables.

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 the weights applied to the units. Once normalised, **weights sum up to 1**: each of them measures as a percentage the importance of the related unit, i.e. the share of the overall system it represents (as an example, think of the Municipalities of a Region...)

The **weighted average** (or **mean**) of variable x over I is given by the well-known formula

$$\bar{x} = \text{mean}(x) = m_1 \cdot x_1 + m_2 \cdot x_2 + \dots + m_n \cdot x_n = \sum_i m_i \cdot x_i \quad (2.1)$$

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). In the particular case of n units all having the same weight, the common value of the weight is

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

and the (2.1) becomes the well known **simple average**:

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

The **mean** 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

$$\text{var}(x) = \sigma^2(x) = m_1 \cdot d_1^2 + \dots + m_n \cdot d_n^2$$

where $d_i = x_i - \bar{x}$ is the difference between the variable's value in the unit i and its average value.

The **variance** is obtained by adding up the squares of these differences for all the units, each weighted with the weight assigned to the unit.

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. To this purpose, the value x_{ij} taken by the variable j in the unit i is **standardised** (the term **normalised** is also used), i.e. transformed as follows:

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

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

Note When a Dataset is loaded, the mean, standard deviation, minimum, maximum and some other statistical information are at once computed and stored for each **QUANTITATIVE** and **COUNT** variable. These values are updated should the user changes the parameters that control their computation (e.g., the weight to be used).

The mean, the standard deviation and the total sum of **QUANTITATIVE** or **COUNT** variables can be used in the expressions provided to construct new variables (**Utilities**→**New Variables/Datasets**) or in the conditions that control the selection of a particular segment of units.

The **ADDATI** function **PCA** (Principal Component Analysis: **Analyses**→**Principal Components Analysis**) performs in an automatic way a standardisation of the continuous variables in input, so as to assign to each of them the same importance in the analysis.

This means that by default **the correlation matrix is diagonalised**. In alternative, the user can request to diagonalise the **matrix of variances-covariances**.

2.2.1 - 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 2.6.

	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 2.6 - Sample data for three administrative units.

Would it be correct to compute the activity rate *of the overall system* as shown below?

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

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 different contributions to the computation of the statistics relative to the overall system;
- each region is a fraction of the overall system, and contributes to it in proportion to 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**) **mean** is an indicator of the central tendency of a distribution:

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

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

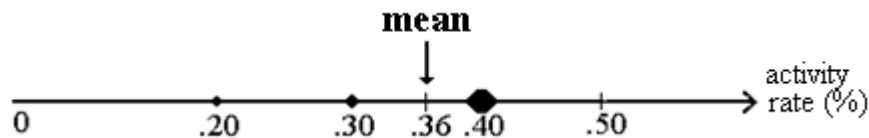


Figure 2.1 - The activity rate values of the three regions represented on an axis. The size of each point is proportional to its population. The average falls in the centre of gravity of the three weighed points.

The variance of the activity rate for the three regions is

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

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

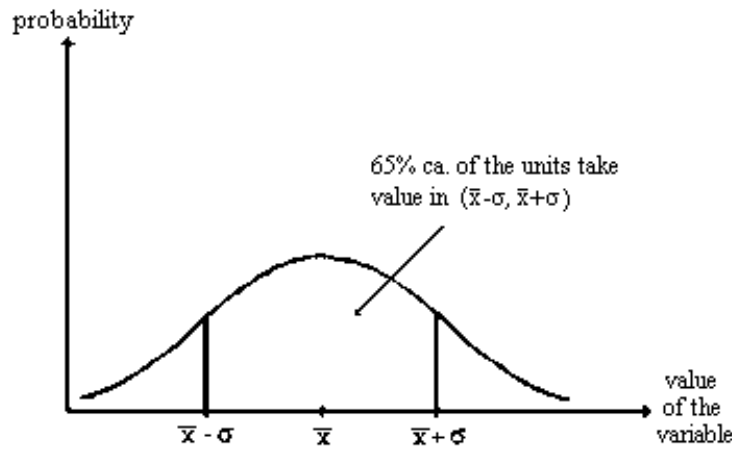


Figure 2.2 - The *normal* (or *Gaussian*) distribution and the meaning of σ .

The **standard deviation** σ (sigma) measures the distribution's dispersion. The figure 2.2 shows the distribution of a normal variable and the meaning of the standard deviation.

2.2.2 – Example 2 – Quantitative variables: normalisation

The table 2.7 is referred to a town divided in five districts Q_i ($i = 1...5$), each with its population pop_i . As the overall population pop_{tot} is 187,234 persons, the generic district i has a weight m_i , shown in the table, equal to

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

The following four variables have been measured in each district, and processed 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.

	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 2.7 - Indicators of well-being in five districts

It can be immediately observed that:

- Q4 appears to be the best district, with the highest level of *status* and *high_ed*, and the lowest levels of *poor* and *crowd*.
- It is difficult to directly appreciate the meaning of 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.

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

$$y_{ij} \leftarrow x_{ij} - \bar{x}_j \quad (2.2)$$

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 overall mean.

The (2.2) 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.

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. The inspection might be quite long and tedious to carry out.

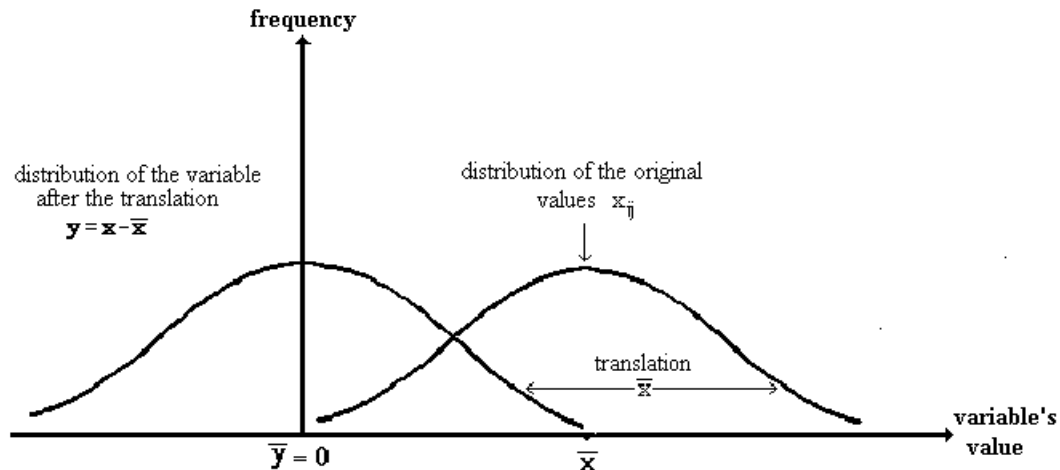


Figure 2.3 - Distribution of the values of x before the translation, and distribution of the corresponding centred variable y .

For example, the table 2.7 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 would be 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 y (the difference with respect to the mean for each variable) by its standard deviation σ . This affects more the differences 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 by writing 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 initial 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 2.8 - The values of table 2.7 expressed as *z-scores*

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

The table 2.8 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* near to zero, i.e. with behaviour close to the system's average.

Let us represent the two variables *status* and *crowding* on the two axes of a Cartesian plan: as each district is described by a pair of such values (see table 2.7), 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 2.4a 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.

Perhaps the term “*cloud of points*” is exaggerate when the points are so few, but in a real analysis they are usually many 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 2.4b represents the same cloud, 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 distance among each pair of district-points, that remain the same in both cases.

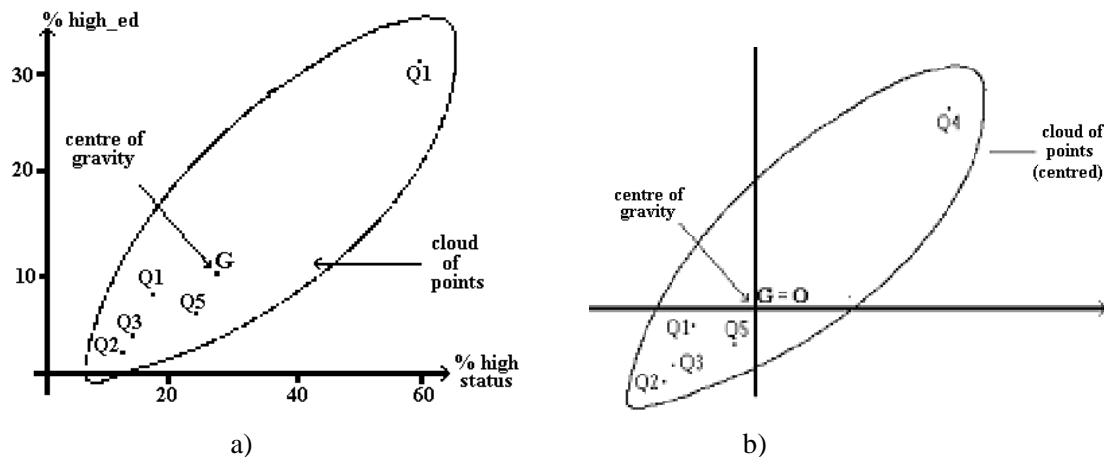


Figure 2.4 - The cloud of points (2.4a), and the same cloud centred (2.4b)

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:

$$\text{cov}(x, y) = \sum_i m_i (x_i - \bar{x})(y_i - \bar{y}) \quad (2.3)$$

Each statistical unit i gives a *positive contribution* to the covariance when the differences of both variables from their respective averages have in i the same sign (i.e., the two variables are in i both above or both below their respective averages, 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 (2.3) becomes

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

The following **extreme** cases can occur:

- **high and positive covariance**: the two variables tend to take **together** values above or below their average, and the cloud has the form shown in fig. 2.5a.
- **high and negative covariance**: the two variables tend to have differences from their respective average that have opposite signs (fig. 2.5b).
- **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. 2.5c).

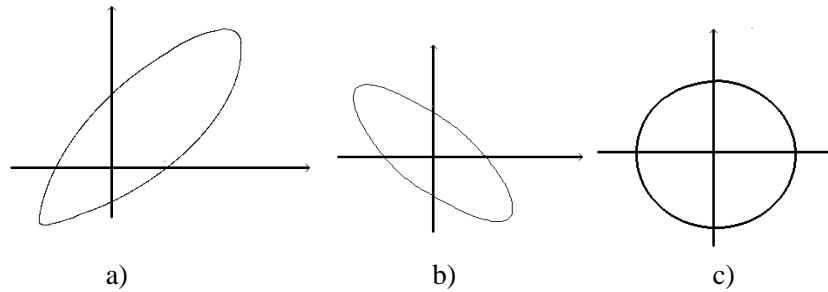


Figure 2.5 – Different types of association between two continuous variables

In particular:

- The covariance of a variable with oneself 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*. It actually depends on how the differences to the mean 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 varied, 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. This value 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 prove 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 that provided by the other.

The table 2.9 shows *the correlations* (*1000) of the four variables of table 2.7.

- The matrix is symmetric: for each pair of variables (i, j) , $\text{corr}(i, j) = \text{corr}(j, i)$. The correlation is *a property of the pair of variables*, measured over a given set of statistical units (the "context", consisting in our example of 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 2.9).

- *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.

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

Table 2.9 – The correlations (*1000) among the four variables.

2.3 – Types of tables

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

2.3.1 - 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*.

2.3.2 – Contingency tables

They are obtained by counting statistical units of the same type (persons, households, acres...) according to the combination of the categories of **two categorical variables** (or even more than two, and the resulting table will have more than two dimensions).

The table 2.10 is a contingency table obtained **by 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.

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

Table 2.10 - An example of contingency table.

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

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.*

We will see later that an *Analysis of the Correspondences* is the appropriate multivariate technique to process contingency tables.

Remember

Any table for which it makes sense to add the values of a line, or those of a column, can be thought of as a contingency table.

Chapter 3. – The FILE Menu

It consists of the items shown in figure 3.1, described in detail in this chapter.

Settings (second item from the bottom) is used to change the programme's default settings. It should positively be use the first time the programme is run, and when it is needed thereafter. [It has already been illustrated in Chp. 1.](#)

Setting the working Directory

The choice of the Working Directory is usually the first operation after launching ADDATI. The first time the programme is run after installation, the Working Directory is the one where the package resides. However, it is better to assume as active the folder that contains the data to be processed.

The option displays a dialog to select the desired directory. The choice is confirmed by a message that appears in the main window.

If the programme happens to complain that it is not able to open some input file, or something similar, this step was probably omitted and the working directory is not the expected one.

The current working directory is remembered by the programme the next time it is run.

Next, the input Datasets to be processed must be opened.

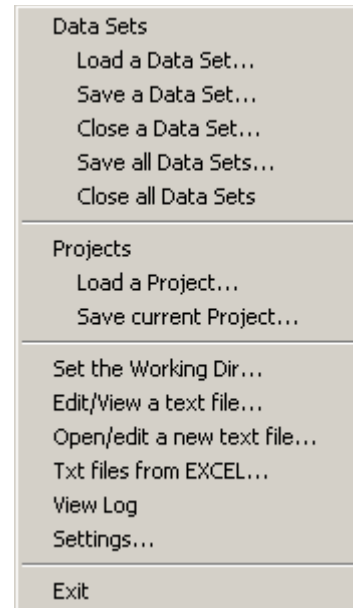


Figure 3.1 – The FILE Menu

3.1 – Datasets: description, opening, closing

The top section of the FILE menu concerns the opening and closing of Datasets.

It has been said that a Dataset consists of a **data file** (with a suitable format), a **documentation file** that describes its contents, and a **label** that identifies it amongst all the loaded Datasets.

3.1.1 – Data files

Each record consists of a set of variables (or indicators) that describe a statistical unit (persons, household, administrative units...). Variables can be separated by spaces, commas or semicolons, known as **field delimiters**. There can also be no separator, and in this case each variable must occupy exactly the same columns in every record, to make it possible to distinguish it from the others.

The format without delimiter is accepted only in input. It can be found in very huge files, that record data at the maximum level of disaggregation (e.g., with each record corresponding to one household questionnaire), often distributed by National Census Bureaus or collecting the data from large Surveys.

When saving a Dataset, ADDATI always uses a separator.

3.1.2 – Documentation files

The contents of a data file are described by the documentation file associated to it. By default, ADDATI expects it to have the same filename as the file it describes, and extension ‘.TXT’, but the user can point to any file whose contents is appropriate. It can be prepared using any text editor, like Windows Notepad or the ADDATI internal Editor, and following the rules listed here below. A limited help to its creation is foreseen, but not yet implemented.

The Datasets created or modified in ADDATI (by copying or recoding existing variables, or computing new ones; or by extracting a condition-based subset of units) are saved together with their documentation file, produced automatically.

3.1.3 – Structure of a Documentation File

It provides the programme with the information necessary to properly process the variables.

The characters '#' or ';' (semicolon), wherever located in a record, mark the beginning of a **comment** that lasts to the end of the line, and **is ignored by the programme**. It is convenient to use comments abundantly, to make clear (to human beings) all that is useful.

Every record different from a comment consists of a keyword, followed by one or more values that specify it. The keywords are the following (**written in red**):

ARCVIEW_FORMAT (optional and obsolete) It specifies if the input data file is written in a format that makes it immediately loadable as a text file by ARCVIEW or EXCEL). In such case, the file can be easily used for joining operations with the .DBF files of attributes that accompany cartographic shapefiles.

The data file must have a first record (**that ADDATI will ignore**) with the columns' headings (variables' names) in double quotes, and it must use the COMMA as field delimiter.

The keyword ARCVIEW_FORMAT is still accepted in the documentation files of the Datasets being loaded. However, when a Dataset is saved the user is requested to enumerate explicitly the features of the Dataset to be written. Actually, the field delimiter could be the COMMA or the SEMICOLON, the first heading record could be present or not...

Warning!

If the ARCVIEW_FORMAT is declared, but a first record of headings is not present in the data file, ADDAWIN will assume the first data record to be that with the headings and will ignore it.

Example:

ARCVIEW_FORMAT YES # or **NO**, the default if this statement is missing.

HEADINGS Used to specify if the first record of the data file consists of the (short) names of the variables, in double quotes and separated by commas. ADDAWIN will ignore its contents, but the presence of such record is useful for a full compatibility with ARCVIEW or EXCEL.

HEADINGS YES # or **NO**, the default if the keyword is not present.

N_UNITS (number of the statistical units, optional)

The programme can automatically determine the number of units (one per record) when it loads the data file. The keyword N_UNITS can be used to load only part of them. It is followed by the number of the records to be loaded. For example:

N_UNITS 3500 # to load only the first 3500 records

MISSING_VALUE (code denoting a missing value, optional)

When the programme is started, it loads from the configuration file ADDATI.INI a default code used to represent missing value. Such a code can be modified by editing [ADDATI.INI](#), or at runtime by choosing from the Menu the **File**→ **Settings** option to open the [configuration dialog](#). **The missing value code read from ADDATI.INI is used for all the Dataset that do not declare a specific one**, which can be done by inserting in the documentation file the instruction

MISSING_VALUE code

where 'code' is the numeric value to be used for the missing values of that Dataset. For example:

MISSING_VALUE -999 # in this Dataset '-999' means that the value is missing

FIELD_DELIMITER (field separator used in the data file)

The number of the variables to be loaded is determined by the programme on the basis of the documentation file's contents.

In each record, the values of the variables are separated by SPACES, COMMAS or SEMICOLONS. These are the only separators that can be used when a Dataset is saved by ADDATI. When a Dataset is loaded, instead, also the absence of any field separator is accepted (**FIELD_DELIMITER** NONE). In this case all records must have exactly the same length, and the values of each variable can be recognized by their place in the record

Example:

FIELD_DELIMITER SPACE # possible values: SPACE, COMMA, SEMICOLON, NONE

As many blocks follow, headed by the keyword **VARIABLE**, as there are variables to be loaded.

The order of the keywords is compulsory. Names and labels must stay between double quotes if they include some internal spaces.

VARIABLE long_name [short_name] type [ncat]

'long_name' is the name of the variable. It must stay between double quotes if it contains some space.

'short name' is an **optional** short version of the name. It is used to head the columns if the Dataset is saved in ARCVIEW format, or if the insertion of a record of headings is explicitly requested. It is also used in the factorial analyses, in order to prevent too confused projections onto the factorial planes.

If a short name is not supplied, the long name is used, truncating it if necessary to 12 characters, and asking the analyst to confirm or to change it.

'type' identifies the variable's scale. Its possible values are CATEGORIAL , QUANTITATIVE, ID o COUNT.

- **CATEGORIAL**: the variable is categorical
- **QUANTITATIVE**: the variable is quantitative, but **it does NOT count units**. It can be a computed variable, a rate, etc.
- **COUNT**: counts some kind of units (persons, households, etc): it takes integer values. For example, the population of a geographica unit, the number of persons with a given

educational level, the number of the dwellings built after 1980, etc.

- **ID:** an identifier of the statistical units, not to be dealt with numerically

'ncat' is the number of the categories, **present (and compulsory) only for variables of type CATEGORIAL**

For pure reasons of convenience, the keyword '**VARIABLE**' can be followed, without any separation space, by an ordinal number: when it encounters 'VARIABLE23' or 'VARIABLE52' the programme interprets it simply as 'VARIABLE', ignoring the number.

QUANTITATIVE or COUNT Variables

The '**VARIABLE**' line **can** be followed by some more lines, **all optional** a da altre righe, tutte opzionali, headed by the keywords **CLASSES**, **THRESHOLDS**, **WEIGHT** and **DECIMALS** (the last one if the variable is **QUANTITATIVE**). If such keywords are missing, the default values specified in ADDATI.INI are applied. They can be inspected and modified from the [configuration dialog](#).

CLASSES is followed by the number of the classes to be used when computing the variable's distribution.

THRESHOLDS can be followed by:

- '**EQUAL_WIDTH**' if the variable's range of values must be subdivided in equal-width intervals;
- '**EQUAL_FREQUENCY**' to obtain *quantiles*, i.e. classes approximately equal-frequent;
- a *sequence of threshold values* set by the analyst, separated by spaces and consistent with the requested number of classes. If they disagree, the thresholds prevail.

WEIGHT specifies the name of the variable to be used as weight when computing the statistics for the current variable. If it is missing, all statistical units have the same weight. The weight can be modified at runtime.

DECIMALS specifies the number of decimals (the default is 0) to be used **in all outputs, and when the Dataset is saved** (not the decimals the variable has in the input data file, that are recognized automatically when the file is loaded).

In the statistics (mean etc.) and in the thresholds the number of decimal is increased by one.

Example:

CLASSES 8 # eight classes requested for the distributions

THRESHOLDS EQUAL_FREQUENCY # the classes requested must be equi-frequent (*quantiles*).

WEIGHT POPULATION # use the population to weight the statistical units

DECIMALS 0 # the values of the variables will be written without decimals

CATEGORIAL Variables

If the variable is categorical, its declaration is followed by as many lines as it has categories. Each line specifies **one category**, and has the following form:

VAL code "label"

where "**code**" is the code of a category, and "**label**" is its name. The double quotes are not compulsory, but are necessary if the label includes some internal space.

Warning: the code is considered an alphanumeric string, not a number: "2" and "02" are not equivalent.

The keyword **WEIGHT** is accepted also for categorial or binary variables; it can make sense when the units measured by the variable have are different in 'size' (e.g., a set of administrative units, weighted with their population or area), or in the case of a Survey, when the sampling factor is used to weight each statistical unit.

The following instruction is present only when the value of **FIELD_DELIMITER** is **NONE**. It informs the programme about the location of each variable in the record:

START_LEN start len

where '**start**' is the 1-based columns (byte) of a record where the value of the variable begins, and '**len**' is its field length, i.e. the number of characters it takes.

An identifier ID is treated as an alphanumeric string, and the programme determines its maximum length at loading.

The keyword '**SKIP**' commands **the skipping of the variables** that the analyst does not want to load. Its form is

SKIP n

and its meaning depends on the field delimiter used.

- if there is a field delimiter (SPACE, COMMA or SEMICOLON), n is **the number of fields** to be ignored.
- if **FIELD_DELIMITER=NONE**, n is **the number of characters** to be ignored when loading.

A '**SKIP**' instruction must obviously stay between two '**VARIABLE**' blocks.

3.1.4 – Example of a documentation file

Some info on the Quito barrios (city quarters). One record per quarter.

N_UNITS 388 # only for clarity, but not necessary as in this case all records are loaded

MISSING_VALUE -99 # in the data file '-99' is used to indicate a missing value

FIELD_DELIMITER COMMA # commas are used as field separators

VARIABLE1 "CLAVE" ID # an identifier for joining operations with cartographic data

VARIABLE2 "UBICACIÓN" CATEGORIAL 3

VAL 1 "FUERA LIMITE URBANO"

VAL 2 "URBANO"

VAL 3 "PROTECCIÓN ECOLOGICA"

SKIP 1 # ignore the variable that follows in the record

The following variable is categorical. When computing the frequencies of its categories, the analyst not

only wishes to know how many quarters each category includes, but also the total population in each

category, as the quarters have different size, and their population can differ a lot.

For this reason, the population is used as weight.

VARIABLE4 "FECHA APARECIMIENTO" CATEGORIAL 6

VAL 1 "HASTA 1959"

VAL 2 "1960-1969"

VAL 3 "1970-1979"

VAL 4 "1980-1989"

VAL 5 "1990-1991"

VAL 6 "1992-2003"

WEIGHT "POBLACIÓN" # the weight to be used when computing this variable's statistics

VARIABLE5 "SUPERFICIE" QUANTITATIVE

CLASSES 6 # six classes requested for the distribution; can be changed at runtime

THRESHOLDS 0 500 1000 2000 4000 6000 # the thresholds requested to delimit the intervals

VARIABLE6 "VIVIENDAS" COUNT # counts the dwellings in the quarter

SKIP 1 # ignore one field

VARIABLE8 "POBLACIÓN" COUNT # the quarter's population, used as weight

VARIABLE9 "DENSIDAD" QUANTITATIVE # Population/quarter's area ratio

WEIGHT SUPERFICIE # for this variable, the quarters (units) are weighted with their area

SKIP 2 # ignore two fields

nine values representing different average economic levels of the quarters. Each quarter is assigned to

one level. The labels of the categories simply coincide with the codes used in the data file.

VARIABLE12 "SECTOR ECONÓMICO" CATEGORIAL 9

VAL 1 "1"

VAL 2 "2"

VAL 3 "3"

VAL 4 "4"

VAL 5 "5"

VAL 6 "6"

VAL 7 "7"

VAL 8 "8"

VAL 9 "9"

WEIGHT "POBLACIÓN"

The following variables are of type COUNT. They count how many dwellings have the type of service indicated by the category, and not persons, but the two quantities can be considered approximately proportional. Therefore, there is no need to weight them.

VARIABLE13 "AGUA RED PÚBLICA " COUNT # dwellings linked to the public water pipe

VARIABLE14 "AGUA POZO" COUNT # water from a well

VARIABLE15 "AGUA RIO_ACEQUIA " COUNT # from river

VARIABLE16 "AGUA CARRO REPARTIDOR " COUNT # from a tank

VARIABLE17 "AGUA OTRO " COUNT

#

VARIABLE18 "ALCANTARILLADO RED PÚBLICA " COUNT # sewage

VARIABLE19 "ALCANTARILLADO POZO CIEGO " COUNT

VARIABLE20 "ALCANTARILLADO POZO SEPTICO " COUNT

VARIABLE21 "ALCANTARILLADO OTRO " COUNT

#

VARIABLE22 "BASURA CARRO RECOLECTOR " COUNT # garbage handling

VARIABLE23 "BASURA TERRENO O QUEBRADA " COUNT

VARIABLE24 "BASURA QUEMADA " COUNT

VARIABLE25 "BASURA OTRO " COUNT

#

VARIABLE26 "ELECTRICIDAD SI" COUNT

VARIABLE27 "ELECTRICIDAD NO" COUNT

#

VARIABLE28 "ESTADO LEGAL" CATEGORIAL 7

VAL 1 "BARRIO APROBADO"

VAL 2 "BARRIO APROBADO PREVIA O ANTEPROYECTO APROBADO"

VAL 3 "ILEGAL A" # various illegal situations

VAL 4 "ILEGAL B"

VAL 5 "ILEGAL C"

VAL 6 "ILEGAL C-"

VAL 7 "ILEGAL NO CLASIFICADO"

WEIGHT "POBLACIÓN" # this keeps track of the overall population affected

3.1.5 – Load a Dataset

Clicking on this menu option the dialog of figure 3.2 is displayed:

Clicking on the first '**Browse**' button the files with extension '.DAT' or 'CSV', existing in the working directory, are shown. The other Browse button is for the documentation file, and displays files with extension .TXT. The user can change folder to trace the files wanted.

The button '**Show**' allows the user to inspect the contents of the data file. As the file is not yet loaded in ADDATI, its documentation is not yet known by the programme, and the editor cannot yet display the name of the variable under the cursor.

The button '**Edit**' displays the documentation file, enabling the analyst to **modify** it if necessary before loading it.

The button '**Create**' will ease the creation of a documentation file, but the option is not yet active.

If a .TXT file with the same filename as the data file exists in the working directory, it is automatically proposed as its associated documentation file. The name can be changed, inserting or choosing a different one, whatever its name or extension. The extension '.TXT' is merely indicative; what is important is that the file must have been prepared according to appropriate rules, and actually describe the data file being loaded.

In conclusion, the names of data and documentation files are not relevant, and the default extensions are only advices. What is important is their contents, and the way they are organized.

The Dataset Label – For each dataSet to be loaded the programme requests a label that will be used to identify it in all circumstances. The programme proposes a label by default, derived from the datafile name. The user can confirm or change it.

Hit 'OK'. The Dataset is loaded. The errors encountered, if any, are listed in a file that can be displayed, to correct as needed both the documentation and data files.

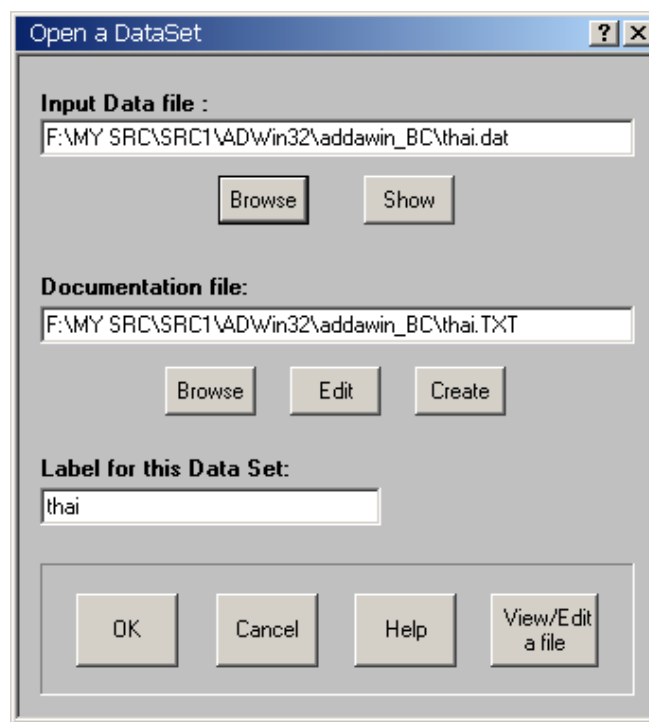


Figure 3.2 – The dialog to open a Dataset

Note *If a severe error occurs (e.g., a record is shorter than what is declared, and the variables' values in it cannot be identified with certainty) the concerned record is discarded.*

If any missing or off-code values are found, they are replaced by the missing value code for the concerned Dataset, specified in its documentation file. If no specific missing value code was indicated, the default missing value code read from the [initialization file](#) is used. The other valid values present in the same record are regularly loaded.

*Even though many errors are found and listed, in general the Dataset is loaded. **It must therefore be closed** before reloading it after doing the necessary corrections.*

3.1.6 – Save/Close a Dataset

The dialog of figure 3.3 is displayed when the user requests to save to file an open Dataset, or close it. The request can refer to **a new DS**, produced during the current work session (e.g., by recoding existing variables), or **to a DS loaded from file**, that can be saved changing its format, e.g. by changing the field delimiter, or the code for missing values.

The operation is defined by the two checkboxes **Save** and **Close**, and carried out when the button 'OK' is hit.

When a DS is closed the label that identifies it is discarded from the DS list. If it is saved, ADDATI writes to disk also the appropriate documentation file.

After saving and/or closing a DS the dialog remains open, to allow the user to do similar operations on the other DS still open. Push the “**Close the Dialog**” button to dismiss the dialog.

Note When a Dataset is saved individually, the **current features** of its variables (number of classes to be used for the distribution, weighting, quantiles, thresholds...) **are also saved** in the documentation file, **and maintained** when the Dataset is re-loaded. That is, the modifications to the parameters introduced while exploring the characteristics of the variables are saved. Instead, if the option "File -> Save all the Datasets" is chosen, only the **new** Datasets created during the work session are saved.

More explicitly: the changes to the parameters introduced when inspecting the distributions of the variables **are maintained** only if the Dataset is saved individually, selecting the option **File→Save a Dataset**.

Field Delimiter - In the output file the variables are aligned in columns and separated by a field delimiter selected from this control. To save a file that can be easily loaded in EXCEL or ARCVIEW, select the COMMA as field delimiter.

ID's in double quotes - On request, the IDentifiers can be included in double quotes. Saved this way, they will be loaded as alphanumeric strings by EXCEL or ARCVIEW. This can be useful if a variable of type ID must be used in ARCVIEW for a **joining** operation, and in ARCVIEW the same variable is declared as a string in the file of attributes associated with the theme. **When spaces are used as field separators, double quotes are compulsory for names that**

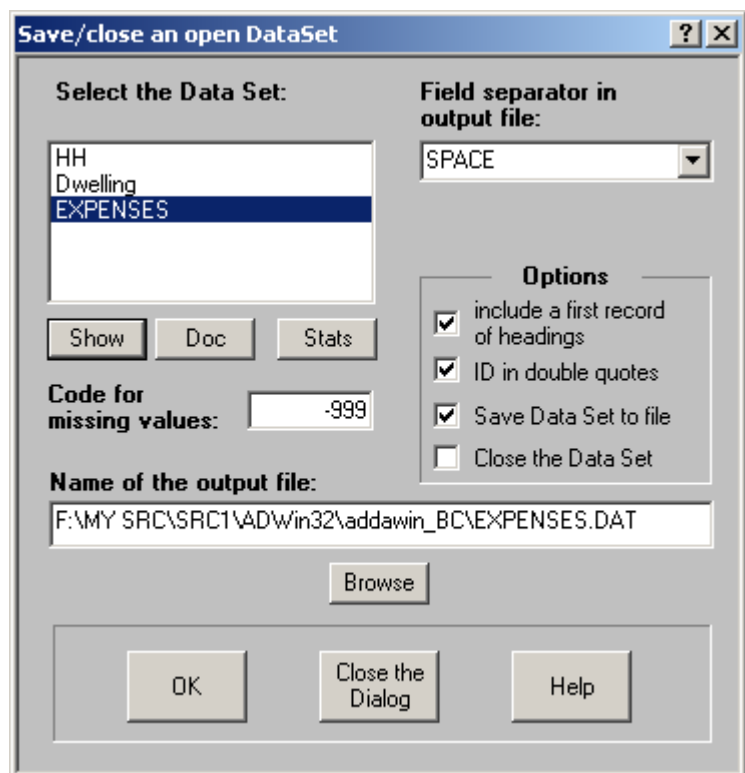


Figure3.3 – The dialog to save a Dataset

include internal spaces.

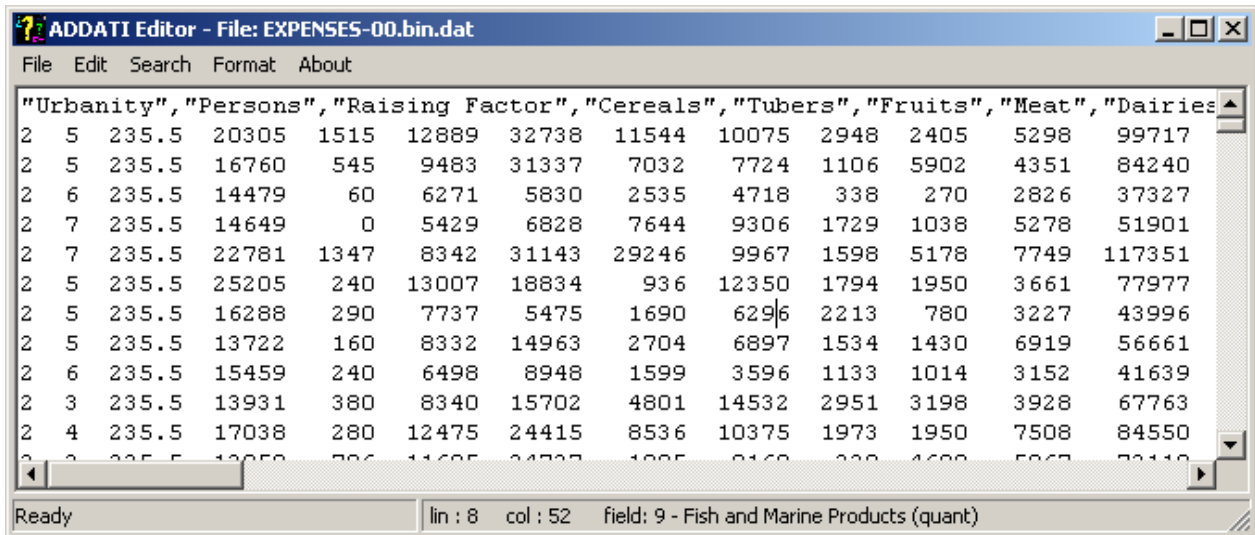
Adding a first record of headings, with the variables' names The extra-record of headings, consisting of the variables' short names, is useful in case the data file is later loaded in EXCEL or ArRCVIEW.

Saving in a format compatible with Excel/ArcView – In order to save a file that can be easily loaded in ArcView or Excel as a text file, the best thing is to write it in .CSV format (Comma-Separated Values). Choose the COMMA as field delimiter, and check the two boxes '**include a first record of headings**' and '**ID in double quotes**'. Better to choose the extension .CSV for the file being written.

The 'Show', 'Doc' and 'Stat' buttons 'Show' and 'Doc' respectively display the data file and its documentation **exactly as they will be saved**. The preferences set by the user in this or other dialogs are enforced (code for missing values, field delimiter, CSV format for Excel/ArcView, etc.).

'Stat' computes and displays some elementary statistics for all the variables in the Dataset. The current weight and control parameters are applied to each variable. The text file with the distributions can be saved with a suitable name.

The window in figure 3.4 was obtained by pushing the '**Show**' button after choosing the space as field delimiter, a first record of headings, ID in double quotes.



		"Urbanity"	"Persons"	"Raising Factor"	"Cereals"	"Tubers"	"Fruits"	"Meat"	"Dairies"			
2	5	235.5	20305	1515	12889	32738	11544	10075	2948	2405	5298	99717
2	5	235.5	16760	545	9483	31337	7032	7724	1106	5902	4351	84240
2	6	235.5	14479	60	6271	5830	2535	4718	338	270	2826	37327
2	7	235.5	14649	0	5429	6828	7644	9306	1729	1038	5278	51901
2	7	235.5	22781	1347	8342	31143	29246	9967	1598	5178	7749	117351
2	5	235.5	25205	240	13007	18834	936	12350	1794	1950	3661	77977
2	5	235.5	16288	290	7737	5475	1690	6296	2213	780	3227	43996
2	5	235.5	13722	160	8332	14963	2704	6897	1534	1430	6919	56661
2	6	235.5	15459	240	6498	8948	1599	3596	1133	1014	3152	41639
2	3	235.5	13931	380	8340	15702	4801	14532	2951	3198	3928	67763
2	4	235.5	17038	280	12475	24415	8536	10375	1973	1950	7508	84550
2	2	235.5	12050	700	11605	24337	10005	8160	3300	4600	5067	72110

Figure 3.4 – The visualization of a Dataset in the format selected for saving it

Code for missing values - This Edit Box displays the missing values code defined for the Dataset highlighted. The value can be modified.

3.2 – Projects – Open and Save

If you have to close a work session and continue later, and there are many open Datasets, the **Project can be saved**. The Project file can be reloaded when necessary, avoiding to have to reload the Datasets one by one, looking for them individually in the folders where they stay. This is all.

There is no advantage when only one Dataset is open, as it takes the same time to open a Project or a Dataset. Some time is saved when there are several Datasets open, especially if they stay in different directories that should be specified one by one.

When a Project is saved or opened a dialog like that in figure 3.5 is displayed. By default, the extension of a Project file is '.APF' (Addati Project File).

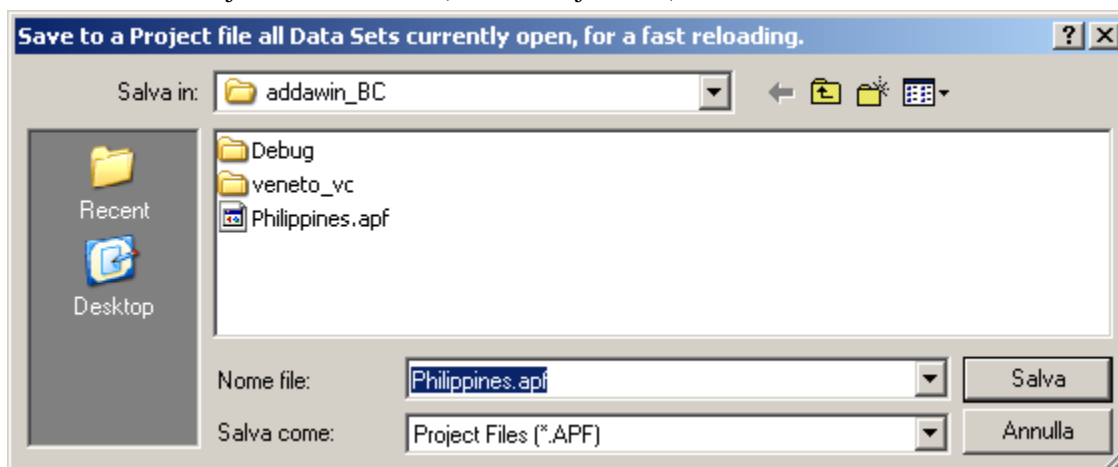


Figure 3.5 – Saving several Datasets as a Project.

3.3 – The other items of the Menu FILE

3.3.1 - Edit/View a text file

This item displays the following dialog.

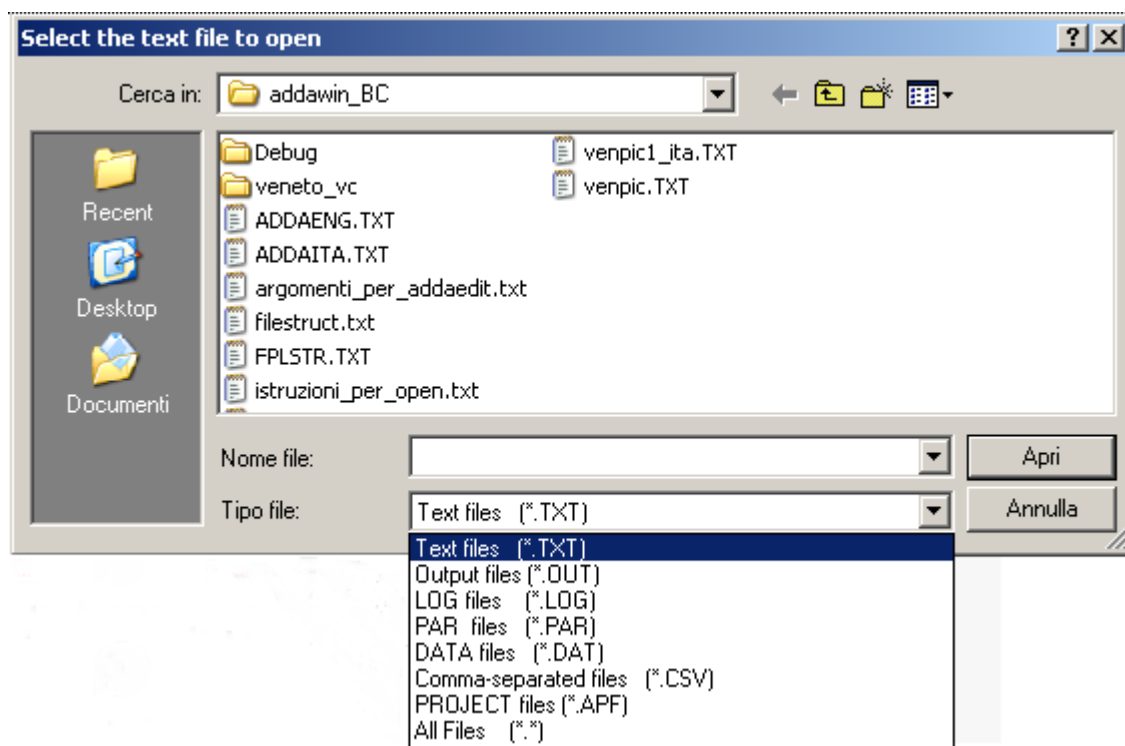


Figure 3.6 – Selection of the text file to be edited.

The text files commonly used in ADDATI have the extensions shown in the fig. 3.7: files ‘.TXT’ (generic text files, or documentation files, or files containing results to be interpreted); ‘.LOG’ (LOG files); ‘.DAT’ or ‘.CSV’ (common extensions for data files). Select an extension to display all the files of that type included in the current folder, then select the one you wish to edit.

Entering ‘NEW’ as the name of the file commands the opening of a new empty file.

The selected file is loaded in **ADDAEDIT**, **ADDATI**’s internal editor.

ADDAEDIT can accept very large files. The line and column (character) where the cursor is located are shown on the status bar.

Besides, if the file being edited is a data file that uses spaces as field delimiters, the ordinal number of the variable under the cursor is also displayed. Data files are automatically recognized if their extension is ‘.DAT’ or ‘.CSV’, but the visualization of the field’s ordinal number can be forced by selecting the item “*Show Field #*” from the Editor’s FILE menu.

As many instances of the editor as necessary can be opened, to inspect/edit several files, or compare their contents. For example, a data file and its documentation, or a data file and the output of an analysis, in order to explore the characteristics of some particular statistical unit...

ADDATI and **ADDAEDIT** are independent applications, and **ADDATI** can be normally used also when the Editor is open.

ADDAEDIT offers some advantages, due to its capability to dialog with **ADDATI**. But we will come back to this later on.

3.3.2 – Conversion of files from EXCEL

This menu item helps the user to import data files from **EXCEL** into **ADDATI**.

ADDATI handles text data files. They must include only data (the .CSV data files can also have a first record of headings with the name of the variables that however is ignored). The complementary information (name and type of the variables, etc.) must be supplied in an associated documentation file. Therefore, EXCEL data must be converted to text format, eliminating all contents different from data.

The **EXCEL** spreadsheet with the data must be saved as a **text file, using as separators tabulations (files .TXT), commas or semicolons (files .CSV – Comma-Separated Values)**. These are two of the options offered by **EXCEL** when the item “Save as” is chosen to save (figure 3.7). **EXCEL** also offers the possibility to create text files in which spaces are used as separators (files .PRN: it is the preceding option in the figure 3.7), and this would be exactly what ADDATI needs. Unfortunately, this option limits to 240 characters the length of the records: this is often insufficient, and requires long and tedious editing operations. Using commas as separators the problem does not exist.

When saving a file in .CSV format, EXCEL may use either commas or semicolons as separators, depending on its current international options. Both are accepted by ADDATI.

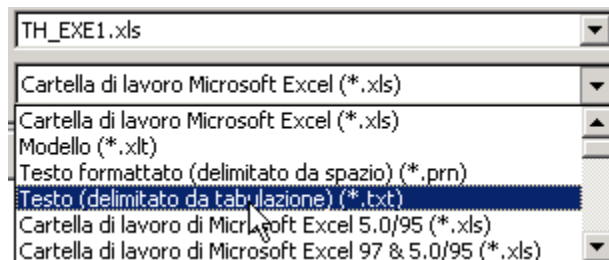


Figure 3.7 – The **EXCEL** option to save data as a tab-separated text file.

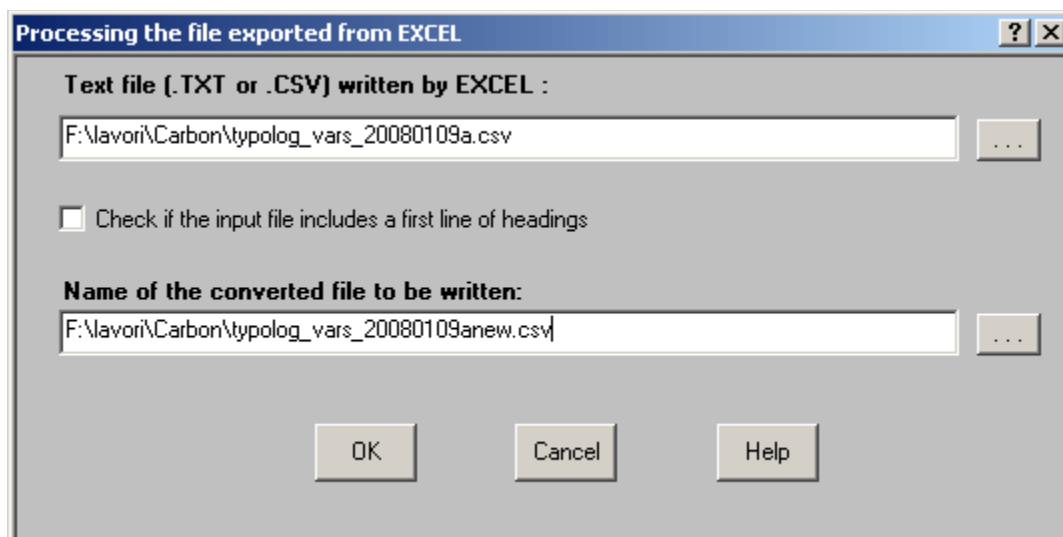


Figure 3.8 – Importing data from **EXCEL**: the initial dialog

After exporting data from **EXCEL** (e.g., in .CSV format), choose the option “*File TXT from EXCEL*” in the **ADDATI FILE** menu. The dialog shown in figure 3.8 is displayed: enter in the upper edit box the name of the file saved from **EXCEL**, or browse to point to it. When the lower edit box is clicked upon, **ADDATI** proposes as output of the conversion process a file with the same name, and extension .CSV or .DAT. Modify the name as suitable, not to overwrite the input file.

Clicking on OK displays the following message:

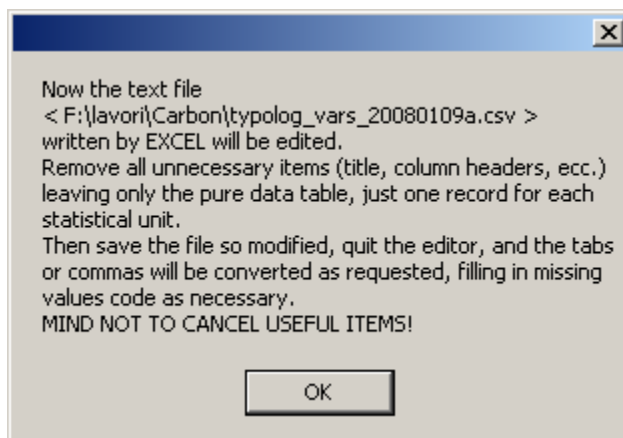


Figure 3.9 – The message that appears before the file to be edited is opened in the editor

Dismiss the information: the file written by **EXCEL** is open in the Editor. Remove all unnecessary elements from it (documentation, labels, notes, etc.), leaving only data and, optionally, a first record with the names of the variables.

Save the modified file (changing its name if necessary) **and close the editor**: **ADDATI** will convert tabs to spaces (if tabs are used as separators), insert the missing value code when necessary (e.g., if two consecutive tabs or commas are encountered, without a valid value in between), replace with underscores **spaces internal to names**, thus generating valid names. Values are then ordered by drawing them up into columns, in order to ease their inspection. All errors found during this

operation (missing values, records with a number of variables different from what is expected, etc.) are brought to the user's attention.

Mind not to delete important elements. In particular, try to place the cursor at the beginning of a new line after the last one, to be sure that this is actually ended by an End-of-Record. Mind also not to insert some space in this new line, otherwise it will be interpreted by ADDATI as a record with no field, causing an error.

Warning

- It is convenient to eliminate in EXCEL, before exporting the data file, any empty column inserted to separate blocks of variables.
- In **EXCEL**, any ambiguity between 0 and missing values must be avoided. A meaningful 0 value must always be written, while an empty cell will be interpreted as a missing value.
- The last column can create some problems: if some of its cells are empty, **EXCEL** may truncate the writing to the cell that precedes it, without appending a final tab or comma. ADDATI will found a field missing, and raise an error.
To prevent this, the last column should be checked carefully in **EXCEL**, and a zero or a missing value code should be inserted in the empty cells, if any. This can be done with a formula.

The **EXCEL** file can also be saved in **.CSV format** (Comma-Separated Values), using commas or semicolons as separators. In such case they can include a first record of columns' headings (the names of the variables), in double quotes. This format is accepted by **ADDATI**, declaring it suitably in the documentation file. This is also the text format usually accepted by ArcView.

The .DAT or .CSV file output by the conversion must be accompanied by a documentation file, to be prepared before loading the Dataset in **ADDATI**.

Note *As mentioned above, the file saved by EXCEL and submitted to the conversion routine can be of type .TXT or .CSV. The two types are treated differently.*

TXT files, using tabs as separators

The extension .DAT is proposed for the file output by the conversion. Tabs are converted to spaces, inserting missing value codes where necessary.

CSV files, using commas or semicolons as separators

If the extension of the output file is also .CSV the first record with the columns' headings is maintained. Commas or semicolons are used as separators, and missing value codes are inserted where necessary.

*Instead, if the extension provided for the output file is .DAT (or any other) **the first record with the columns' headings is removed**; commas (or semicolons) are replaced by spaces, and missing value codes are inserted where necessary.*

Any empty column present in EXCEL and not removed before exporting is automatically eliminated by the conversion process.

3.3.3 – Show the LOG

The contents of the LOG file is shown, where **ADDATI** stores a copy of the messages displayed in the program's main window.

Chapter 4. – The Utility Menu

The Utility Menu consists of few items, but they enable the user to do quite a lot of things.

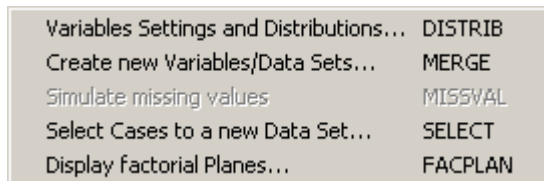


Figure 4.1 – The Utility Menu

The following operations are possible.

- Option **SETVAR**: The features of any loaded Dataset can be explored changing at will, individually for each variable, the parameters that control its distribution: the number of the classes, the thresholds that define them, the weight applied. This permits the user to examine the distribution in detail, easily focusing on intervals where values are concentrated, or singling out possible *outliers* (i.e., abnormal values, for which some kind of error can be suspected).

The distributions can also **be limited to a subset** (a *segment*) **of statistical units**, identified by a particular condition set by the analyst. For example, only the households whose income is greater than a given value; or only the persons living in a given administrative area, etc.

- Option **MERGE**: new Datasets can be created, which can consist of variables copied from DS already open, or new variables created by applying some mathematical operations or logical conditions to existing variables, or variables obtained by recoding existing variables, both quantitative or categorical.
- Option **SELECT**: a new Dataset can be obtained by selecting from an open Dataset only those units that fulfill a condition set by the analyst.
- Option **FACPLAN**: visualization of the projections of variables and statistical units onto the most relevant factorial planes. This utility is used after a Factorial Analysis or a Classification, in order to examine the results. Its use will be illustrated **in another chapter**.

4.1 – Computing variables' distributions

The default parameters used when computing the distribution of the variables (number of classes, thresholds, weights...) are loaded from the programme's initialization file ADDATI.INI, and can be modified from the [Settings dialog](#). They are used for all the variables **for which more appropriate specific values are not indicated** in the documentation file.

When running, this dialog allows the user to change – for individual variables or for groups of them – the parameters that control the computation of some basic statistics, so as to check immediately the effects on the resulting distributions. These changes are permanent, and the distributions according these parameters can then be displayed from all other dialogs.

The changes are valid all along the work session, and can be made permanent in the documentation file if the Dataset is saved to disk.

The figure 4.2 illustrates the meaning of the various controls.

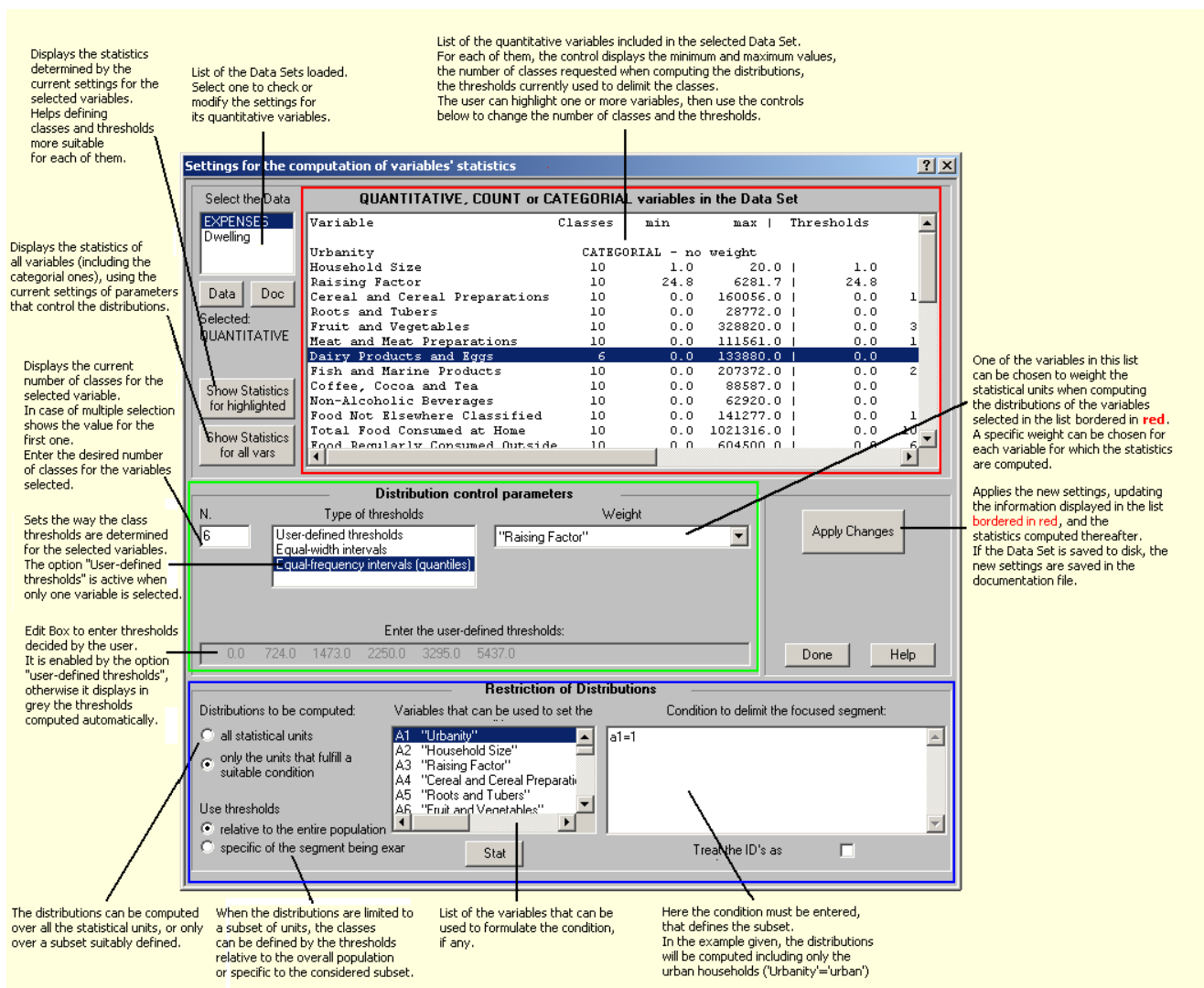


Figure 4.2 – The dialog relative to variables and distributions, commented.

Red frame: minima, maxima and the thresholds displayed for the variables are always those relative to the **whole statistical population**.

Green frame: displays the current preferences for the variable (or variables) selected in the list framed in red.

The preferences can be modified separately for each variable, confirming each change with the **"Apply Changes"** button. The options entered are not active until they are confirmed.

Blu frame: allows the analyst to limit the computed distributions to a particular segment of the population. **The syntax of the condition** used to delimit the desired segment is common to other ADDATI operations.