Excel2Csv

version 1.3

End-user manual

Andrea Santilli (andrea.santilli@istat.it)

Davide Cervini (davide.cervini@istat.it)

Summary

Introduction

What Excel2Csv is

The Excel2Csv application is a software that allows users to export data from statistical tables stored inside Excel spreadsheets to comma-separated value files (csv) that can be later interpreted by data loading programs, such as OECD.Stat's Entry Gate or the SDMX Builder & Loader.

For this task to be accomplished, the application must receive hints about the location of the table inside the spreadsheet and its characteristics. The information that shall be provided the application includes the sheet number, the table boundaries, the boundaries of the axes and what cell the actual data start from.

How Excel2Csv operates

The Excel2Csv program is designed so that it can detect empty rows and columns in a table and eventually read merged cells as well. Dates and times can be read directly from the Excel format or a string representing them. Should a time reference variable show non-standard values, its structure can be mapped or casted to a valid SDMX time format.

Excel2Csv is also able to read tables representing n-dimensional data cubes or their subsets. Since data cubes can be multidimensional and tables only have two dimensions, the resulting representation must show one or more dimensions on each axis to show the cubes appropriately. For this reason, Excel2Csv was made able to detect such dimensions with the user’s intervention.

Output customisation

The output csv structure is entirely user-customised. Thus, end-users decide what field separator shall be used, what position each table dimension must occupy inside the output file and what format a file must be encoded to. It is also possible to override both the local decimal separator and the maximum number of decimal digits a numeric value can show, and to also set the application up so that it won’t try to parse the data it reads, in case a table *purposely* shows non-numeric values. Finally, a header containing the names of each dimension can be added to the top of the file contents as well.

Variable mapping

Apart from the time mapping capabilities of the program, end-users can finally include a variable value code and its corresponding description into the same table cell so that Excel2Csv application can thus validate the cell contents and extract the codes only. In this way, only the latter are reported in the output data file. Obviously, cell contents must respect a given format.

Since tabular forms which Excel2Csv operates on often represent simplified slices (or “subsets”) of original n-dimensional cubes they refer to, it is not unusual for a table designer to leave a few dimensions out of the table by setting them constant and thus showing the other ones only. For this reason, Excel2Csv is also featured with ability to let users add those constant dimensions back into a csv layout, so that the resulting Csv can be kept compliant to the original data cube schema.

Finally, Excel2Csv is also able to remove dimensions that an end-user might want to hide from the output csv, either because it might not report useful information, or it could just contain referential metadata that can be stripped off a data csv file.

Supported formats

Excel2Csv internally uses the NPOI library to read spreadsheet files. As a result, the formats that are currently being supported are the old Excel xls binary file and the newest xlsx one (in the 2007 zipped OOXML archive form, not the 2003 standalone xml format). Other formats might be added later.

About the csv output formats, the application supports all the encodings also supported by the .Net platform. Record endings are marked by the CR-LF byte sequence. In case the application reads text written in Arabic or Hebrew, the resulting portions of the strings in the record fields shall be read right-to-left, while the whole record itself shall be read left-to-right.

About right-to-left readable languages, only Arabic and Hebrew are currently officially supported. Other eventual RTL languages might give unexpected results. If you'd like other RTL languages to be officially supported, please send us a list.

Future developments

At the time being, there are plans to make Excel2Csv able to match the variable descriptions of a dimension with the ones that can be found inside an SDMX code lists. In this way, it could be possible to retrieve a valid variable code by just matching its description with the descriptions in the SDMX code list. On the contrary, there are also plans to extract valid SDMX code lists and their Multilanguage descriptions from a database file. Recently, the application has been featured with the ability to extract multi-language code lists from an Excel file in a way similar to the data extraction process.

Finally, it is being planned to add a new feature that makes it possible to guess an SDMX Data Structure Definition from a given table and output it to a valid SDMX-ML file.

Step-by-step guide

At the first run, Excel2Csv appears as follows. Please consider that there might be variations between different versions in the way the application looks.

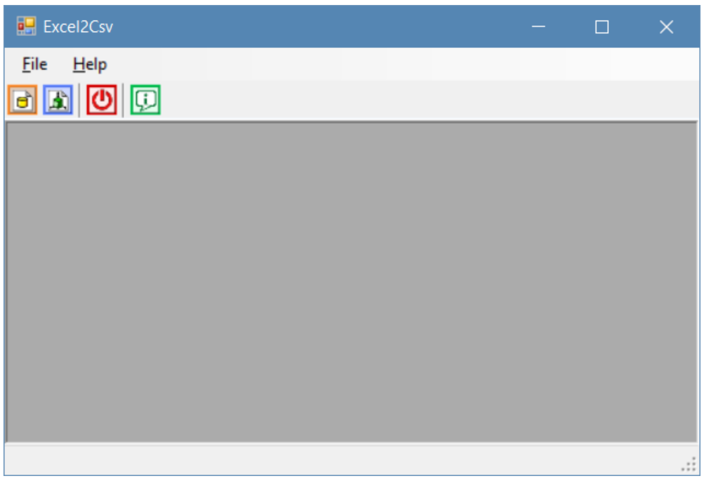
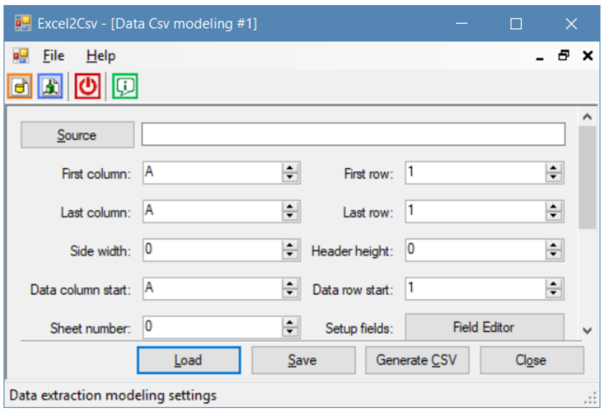


Figure 1: the main window at the first run

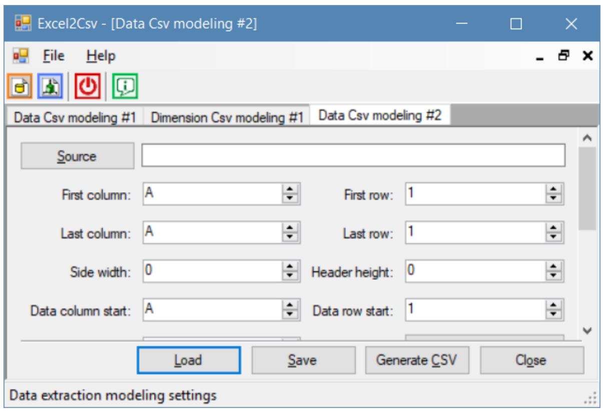
The window with an empty workspace shown in the image is a plain Multiple Document Interface (MDI) parent, that is a window allowed to contain other windows.

From here, we can start new sessions for extracting either data or code lists by just selecting "*New data window*" or "*New dimension window*" from the "*File*" menu. Should several session windows be kept opened and maximised at the same time, the parent window shows a tab row at the top of its workspace.



First column: A
First row: 1
Last column: A
Last row: 1
Side width: 0
Header height: 0
Data column start: A
Data row start: 1
Sheet number: 0
Setup fields: Field Editor
Data extraction modeling settings

Figure 2: only one opened session; tab bar hidden



First column: A
First row: 1
Last column: A
Last row: 1
Side width: 0
Header height: 0
Data column start: A
Data row start: 1
Data extraction modeling settings

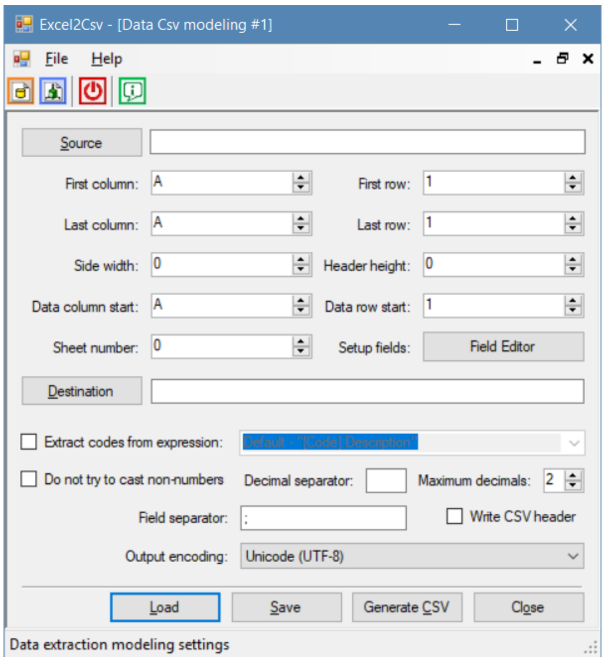
Figure 3: tab bar visible with several child windows

Of course, the keyboard shortcuts that are usually available for MDI windows - such as *ctrl + tab* and *ctrl + shift + tab* for cycling through the children, *ctrl+f4* for closing the current child, or *alt + hyphen* for showing the child system menu - are still valid on Excel2Csv.

We are now going to take a detailed look at both the operational modes of the application.

The data extraction mode

In order to start a new data extraction session, we can open a new data window by either selecting the “*New data window*” entry from the “*File*” menu or by clicking the first button on the button bar of the main window. This is how the application will look like:



First column:	A
First row:	1
Last column:	A
Last row:	1
Side width:	0
Header height:	0
Data column start:	A
Data row start:	1
Sheet number:	0
Setup fields:	Field Editor
☐ Extract codes from expression:	Default - "[Code] Description"
☐ Do not try to cast non-numbers	
Decimal separator:	
Maximum decimals:	2
Field separator:	;
☐ Write CSV header	
Output encoding:	Unicode (UTF-8)

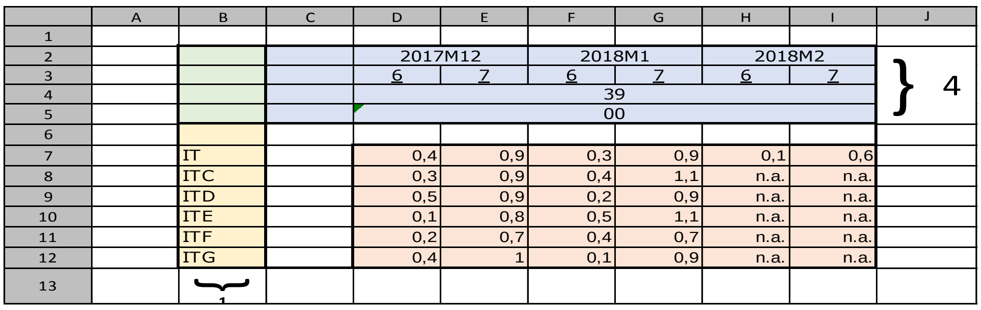
Data extraction modeling settings

Figure 4: the data extraction window stretched enough to show all its controls

Before taking the first step, it is important to keep in mind that a spreadsheet workbook might contain several sheets, each one containing several tables in its turn. Hence, we must give Excel2Csv the right coordinates that identify the table we need to export in an unambiguous way.

It is thus fundamental to pass the application the following information:

1. The full path name of the spreadsheet file we want to read (obviously);
2. What sheet contains the table we want to extract;
3. What cell the table starts from;
4. What cell the table should end at;
5. The area that shall be considered as pertaining to the table axes;
6. What cell the actual data start from.
7. The full path name of the output csv file.



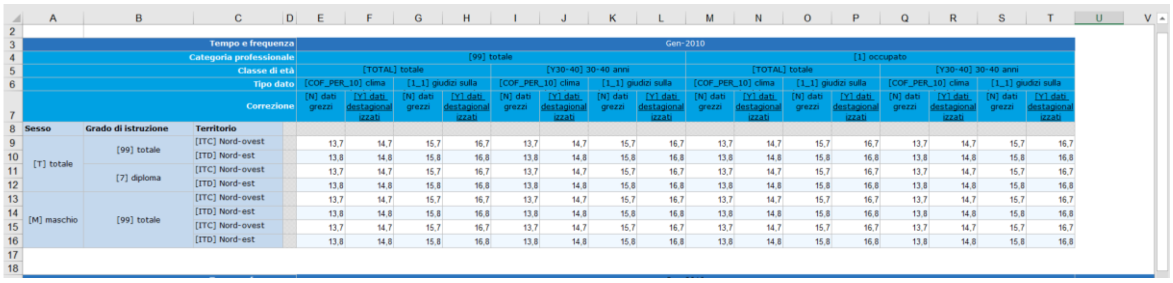
	A	B	C	D	E	F	G	H	I	J
1										
2				2017M12		2018M1		2018M2		
3				6	7	6	7	6	7	
4				39						} 4
5				00						
6										
7		IT		0,4	0,9	0,3	0,9	0,1	0,6	
8		ITC		0,3	0,9	0,4	1,1	n.a.	n.a.	
9		ITD		0,5	0,9	0,2	0,9	n.a.	n.a.	
10		ITE		0,1	0,8	0,5	1,1	n.a.	n.a.	
11		ITF		0,2	0,7	0,4	0,7	n.a.	n.a.	
12		ITG		0,4	1	0,1	0,9	n.a.	n.a.	
13		$\underbrace{\quad}_{1}$								

Figure 5 shows a sample table from which we can guess the necessary basic information we need for the extraction: the beginning and the end of the table are marked by the ***B2*** *and* ***I12*** *cells respectively. The header is* ***4 cells tall*** *and the side is* ***1 cell wide****. Data starts from cell* ***D7****.*

We need to provide both the table axis area and the data start cell either because extra unimportant information might be reported between the axis area and the data area or because the portion of data of our interest doesn't start at the beginning of the data area, thus we must tell the application what the data area boundaries are and what lines and/or rows we want to keep on considering as axes.

In Excel2Csv, point 1 is expressed by an integer value starting from 0; point 2, 3 and 5 are passed by writing the column and the row (respectively starting by "A" and "1") for each cell we must express; about point 4, it is necessary to give both the height of the header and the width of the side.

Let's take the following sample table:

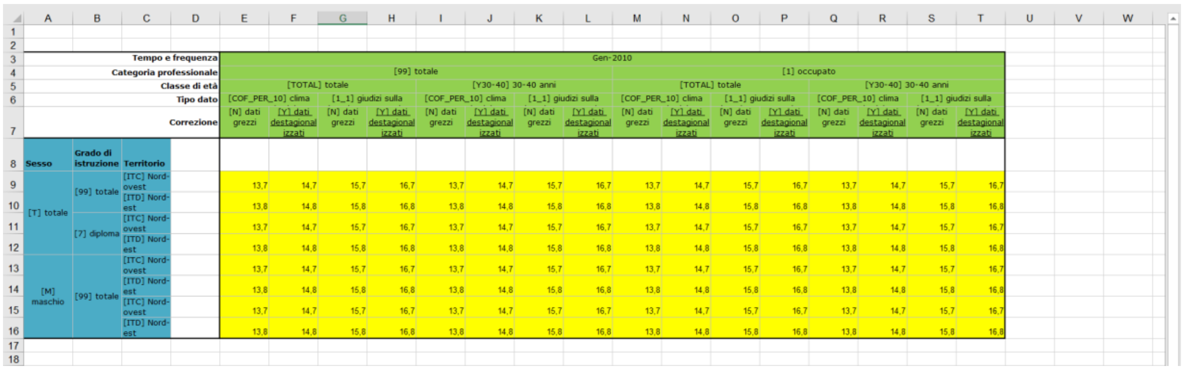


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
2																				
3			Tempo e frequenza		Gen-2010															
4			Categoria professionale		[99] totale								[1] occupato							
5			Classe di età		[TOTAL] totale				[Y30-40] 30-40 anni				[TOTAL] totale				[Y30-40] 30-40 anni			
6			Tipo dato		[COF_PER_10] clima		[1_1] giudizi sulla		[COF_PER_10] clima		[1_1] giudizi sulla		[COF_PER_10] clima		[1_1] giudizi sulla		[COF_PER_10] clima		[1_1] giudizi sulla	
7			Correzione		[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati
8	Sesso	Grado di istruzione	Territorio																	
9	[T] totale	[99] totale	[ITC] Nord-ovest		13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7
10			[ITD] Nord-est		13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8
11		[7] diploma	[ITC] Nord-ovest		13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7
12			[ITD] Nord-est		13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8
13	[M] maschio	[99] totale	[ITC] Nord-ovest		13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7
14			[ITD] Nord-est		13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8
15			[ITC] Nord-ovest		13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7
16			[ITD] Nord-est		13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8
17																				
18																				

Figure 6: our sample table

Our table starts from cell A3 and ends at cell T16. The height of the header takes 5 lines and the side takes 3 columns. Finally, the actual data starts from cell E9. Let's suppose our table is contained inside the sheet number 0, which is the first tab that usually appears at the bottom of Excel.

So, let's recolour our table in a less confusing way:

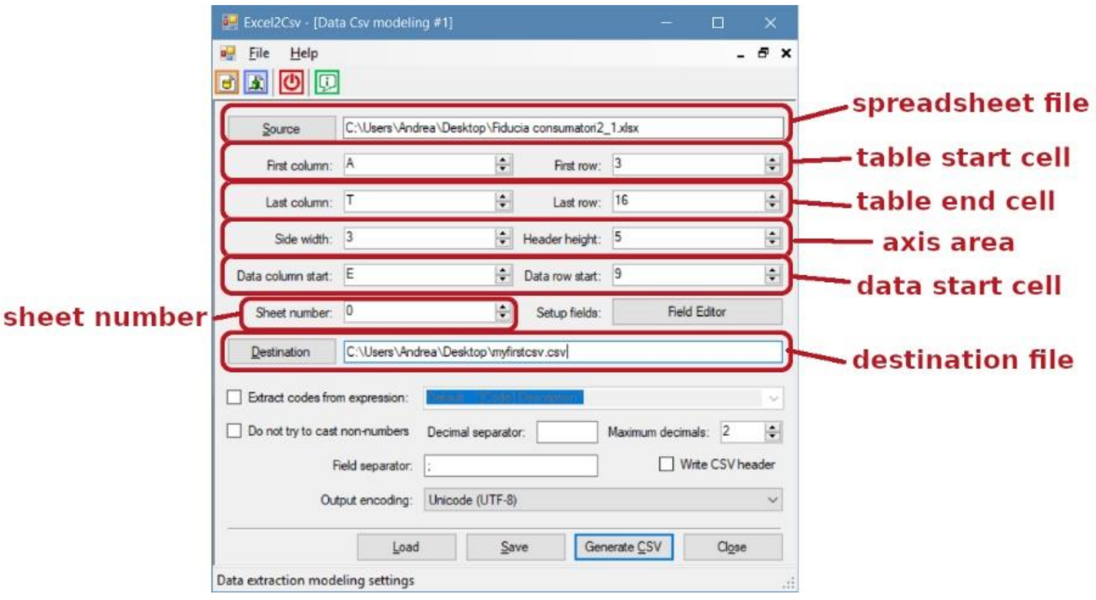


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1																				
2																				
3			Tempo e frequenza		Gen-2010															
4			Categoria professionale		[99] totale								[1] occupato							
5			Classe di età		[TOTAL] totale				[Y30-40] 30-40 anni				[TOTAL] totale				[Y30-40] 30-40 anni			
6			Tipo dato		[COF_PER_10] clima		[1_1] giudizi sulla		[COF_PER_10] clima		[1_1] giudizi sulla		[COF_PER_10] clima		[1_1] giudizi sulla		[COF_PER_10] clima		[1_1] giudizi sulla	
7			Correzione		[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati
8	Sesso	Grado di istruzione	Territorio																	
9	[T] totale	[99] totale	[ITC] Nord-ovest		13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7
10			[ITD] Nord-est		13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8
11		[7] diploma	[ITC] Nord-ovest		13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7
12			[ITD] Nord-est		13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8
13	[M] maschio	[99] totale	[ITC] Nord-ovest		13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7
14			[ITD] Nord-est		13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8
15			[ITC] Nord-ovest		13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7
16			[ITD] Nord-est		13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8
17																				
18																				

Figure 7: the table in figure 2 recoloured in a less confusing way

As it can be seen, the cyan-coloured area is the table side, the green one is the header, and the yellow area represents the data area. The header contains 5 axes of the data cube the table represents, and the side 3.

In this case we are supposed to pass Excel2Csv our information by filling the text boxes of the main window in the following way:



Source: C:\Users\Andrea\Desktop\Fiducia consumatori2_1.xlsx
First column: A   First row: 3
Last column: T   Last row: 16
Side width: 3   Header height: 5
Data column start: E   Data row start: 9
Sheet number: 0
Destination: C:\Users\Andrea\Desktop\myfirstcsv.csv
☐ Extract codes from expression:
☐ Do not try to cast non-numbers   Decimal separator: Maximum decimals: 2
Field separator: ;   ☐ Write CSV header
Output encoding: Unicode (UTF-8)
Data extraction modeling settings

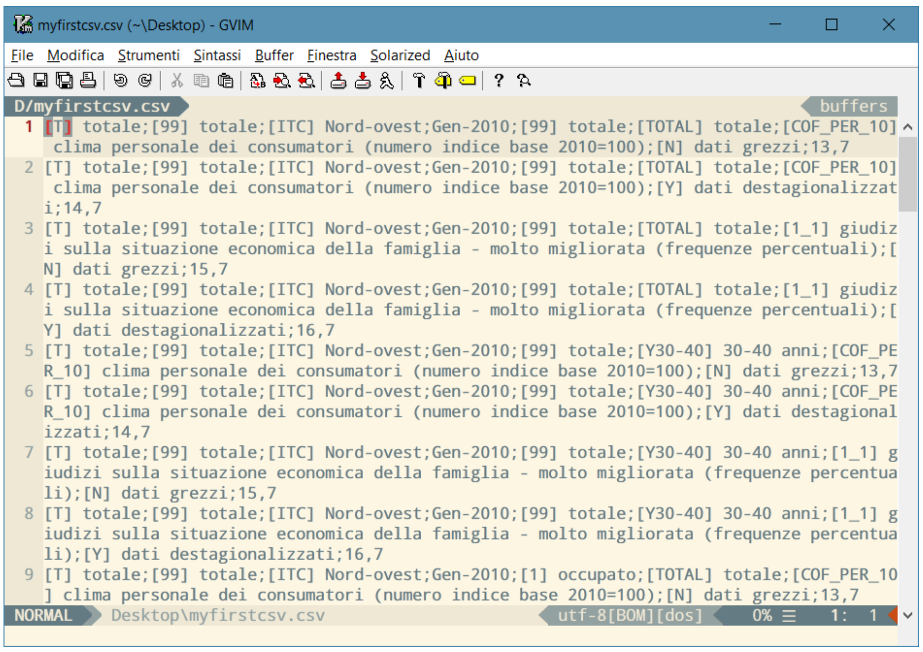
Figure 8: a sample of the correct settings for the table in Figure 2

Both the source and the destination files can be selected by just using the file selector dialogs that are shown by pressing the respective buttons.

Note: spreadsheet documents written in right-to-left languages, such as Arabic or Hebrew, show the sheet labels in a reversed order. Hence, in such documents the sheet "0" is the rightmost tab and the last tab corresponds to the leftmost one. While this fact might sound ordinary for some people, keeping it in mind might save unaccustomed users quite a few headaches.

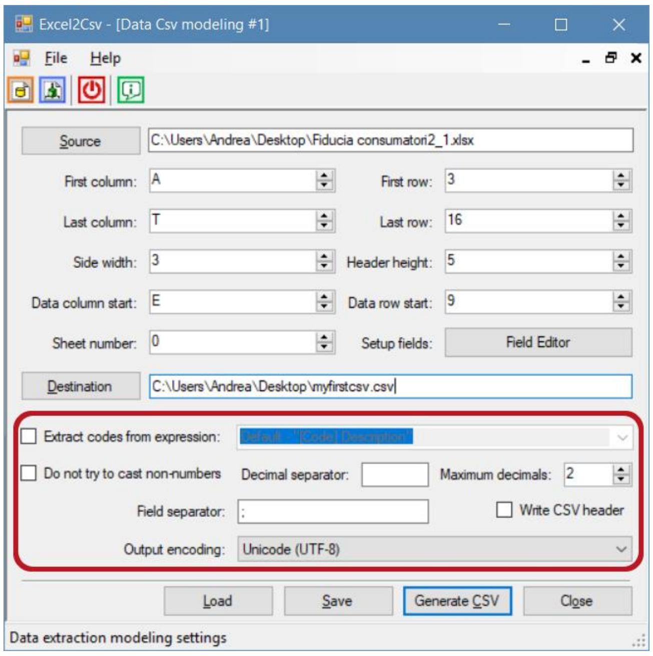
An initial raw output: observations and fine tuning

So far so good, we could already produce our csv file by pressing the OK button. The output csv file shows the following contents:


myfirstcsv.csv (~\Desktop) - GVIM
File Modifica Strumenti Sintassi Buffer Finestra Solarized Aiuto
D/myfirstcsv.csv
buffers
NORMAL
Desktop\myfirstcsv.csv
utf-8[BOM][dos]
0%
1:   1

```
[T] totale;[99] totale;[ITC] Nord-ovest;Gen-2010;[99] totale;[TOTAL] totale;[COF_PER_10]
 clima personale dei consumatori (numero indice base 2010=100);[N] dati grezzi;13,7
[T] totale;[99] totale;[ITC] Nord-ovest;Gen-2010;[99] totale;[TOTAL] totale;[COF_PER_10]
 clima personale dei consumatori (numero indice base 2010=100);[Y] dati destagionalizzat
i;14,7
[T] totale;[99] totale;[ITC] Nord-ovest;Gen-2010;[99] totale;[TOTAL] totale;[1_1] giudiz
i sulla situazione economica della famiglia - molto migliorata (frequenze percentuali);[
N] dati grezzi;15,7
[T] totale;[99] totale;[ITC] Nord-ovest;Gen-2010;[99] totale;[TOTAL] totale;[1_1] giudiz
i sulla situazione economica della famiglia - molto migliorata (frequenze percentuali);[
Y] dati destagionalizzati;16,7
[T] totale;[99] totale;[ITC] Nord-ovest;Gen-2010;[99] totale;[Y30-40] 30-40 anni;[COF_PE
R_10] clima personale dei consumatori (numero indice base 2010=100);[N] dati grezzi;13,7
[T] totale;[99] totale;[ITC] Nord-ovest;Gen-2010;[99] totale;[Y30-40] 30-40 anni;[COF_PE
R_10] clima personale dei consumatori (numero indice base 2010=100);[Y] dati destagional
izzati;14,7
[T] totale;[99] totale;[ITC] Nord-ovest;Gen-2010;[99] totale;[Y30-40] 30-40 anni;[1_1] g
iudizi sulla situazione economica della famiglia - molto migliorata (frequenze percentua
li);[N] dati grezzi;15,7
[T] totale;[99] totale;[ITC] Nord-ovest;Gen-2010;[99] totale;[Y30-40] 30-40 anni;[1_1] g
iudizi sulla situazione economica della famiglia - molto migliorata (frequenze percentua
li);[Y] dati destagionalizzati;16,7
[T] totale;[99] totale;[ITC] Nord-ovest;Gen-2010;[1] occupato;[TOTAL] totale;[COF_PER_10
] clima personale dei consumatori (numero indice base 2010=100);[N] dati grezzi;13,7
```

Figure 9: the output file contents as shown in a text editor



Excel2Csv - [Data Csv modeling #1]

Source: C:\Users\Andrea\Desktop\Fiducia consumatori2_1.xlsx

First column: A
First row: 3
Last column: T
Last row: 16
Side width: 3
Header height: 5
Data column start: E
Data row start: 9
Sheet number: 0
Setup fields: Field Editor

Destination: C:\Users\Andrea\Desktop\myfirstcsv.csv

☐ Extract codes from expression: Default - "[Code] Description"
☐ Do not try to cast non-numbers
Decimal separator:
Maximum decimals: 2
Field separator: ;
☐ Write CSV header
Output encoding: Unicode (UTF-8)

Data extraction modeling settings

Figure 10: finer output tuning settings

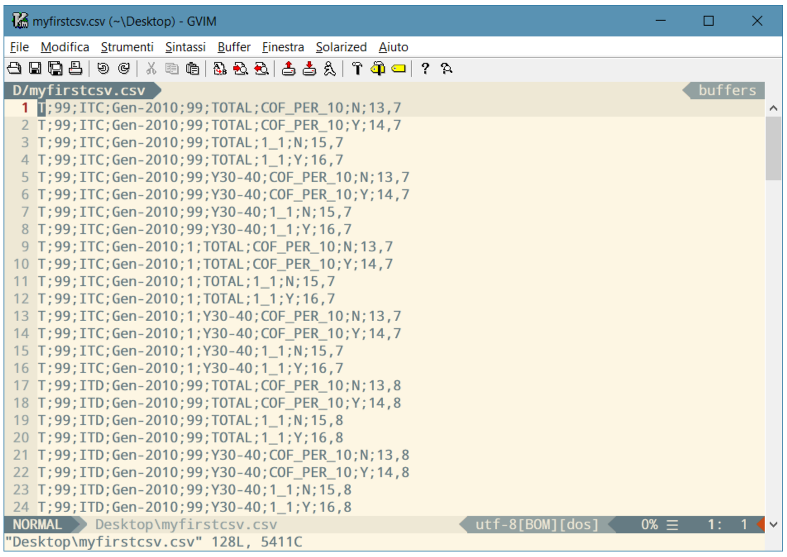
Let's take the first written record:

[T] totale;[99] totale;[ITC] Nord-ovest;Gen-2010;[99] totale;[TOTAL] totale;[COF_PER_10] clima personale dei consumatori (numero indice base 2010=100);[N] dati grezzi;13,7

There are a few details that are worth being noted about this record:

1. By default, Excel2Csv reports the variable values shown on the side first, then the ones on the header and finally the observation value. We will later see how to change this order.
2. About the separators, each record field is separated by a semicolon by default. It is quite obvious how to change this setting from the interface.
   What might be less obvious is the reason why the actual data show a comma as a decimal separator, despite its related box being empty. This is because the system we've been running Excel2Csv on is localised in Italian, thus it used the Italian settings for the representation of real numbers.
   Had we run the application on a UK localised system, the value would have shown a "point" as the decimal separator, and in such case, we would have obtained the value "13.7" instead of "13,7".
   Of course, specifying a different decimal separator in the related box overrides this setting.
3. Another worth-noting detail is about the number of digits in the value. Since we only had one digit in the original table and we specified a maximum amount of 2, the output file only reports one, without adding any zeroes at the end. Please notice that when decimals get cut away, the number is rounded instead of being truncated.
   Anyway, this setting *never* enforces a higher precision than the one that is found in the table.
4. The output file was written in the 8-bit Unicode encoding with Byte Order Mark, as correctly shown in the figure, but it is possible to change this setting. The multiple-choice box, located beside the "output encoding" label, shows a list of encodings that are supported by the .Net platform and that can be used to encode the output file.
   Finally, please notice that line endings are marked the DOS/Windows way (CR-LF).

5. Since each dimension variable seems to show values in the form “[Code] Description” (see, for instance, “[T] Totale” or “[N] dati grezzi”), we can tick the related “*Extract codes from*” box so that the application can use codes only for writing the records. By activating that option, the combo box at the side is enabled but, since it defaults to a “[Code] description” type of expression, we can keep that setting as it is. We will later see how to fine tune the code extraction process. By writing an output again, we can obtain a resulting file as the following one:


myfirstcsv.csv (~\Desktop) - GVIM
File Modifica Strumenti Sintassi Buffer Finestra Solarized Aiuto
D/myfirstcsv.csv
buffers

```
T;99;ITC;Gen-2010;99;TOTAL;COF_PER_10;N;13,7
T;99;ITC;Gen-2010;99;TOTAL;COF_PER_10;Y;14,7
T;99;ITC;Gen-2010;99;TOTAL;1_1;N;15,7
T;99;ITC;Gen-2010;99;TOTAL;1_1;Y;16,7
T;99;ITC;Gen-2010;99;Y30-40;COF_PER_10;N;13,7
T;99;ITC;Gen-2010;99;Y30-40;COF_PER_10;Y;14,7
T;99;ITC;Gen-2010;99;Y30-40;1_1;N;15,7
T;99;ITC;Gen-2010;99;Y30-40;1_1;Y;16,7
T;99;ITC;Gen-2010;1;TOTAL;COF_PER_10;N;13,7
T;99;ITC;Gen-2010;1;TOTAL;COF_PER_10;Y;14,7
T;99;ITC;Gen-2010;1;TOTAL;1_1;N;15,7
T;99;ITC;Gen-2010;1;TOTAL;1_1;Y;16,7
T;99;ITC;Gen-2010;1;Y30-40;COF_PER_10;N;13,7
T;99;ITC;Gen-2010;1;Y30-40;COF_PER_10;Y;14,7
T;99;ITC;Gen-2010;1;Y30-40;1_1;N;15,7
T;99;ITC;Gen-2010;1;Y30-40;1_1;Y;16,7
T;99;ITD;Gen-2010;99;TOTAL;COF_PER_10;N;13,8
T;99;ITD;Gen-2010;99;TOTAL;COF_PER_10;Y;14,8
T;99;ITD;Gen-2010;99;TOTAL;1_1;N;15,8
T;99;ITD;Gen-2010;99;TOTAL;1_1;Y;16,8
T;99;ITD;Gen-2010;99;Y30-40;COF_PER_10;N;13,8
T;99;ITD;Gen-2010;99;Y30-40;COF_PER_10;Y;14,8
T;99;ITD;Gen-2010;99;Y30-40;1_1;N;15,8
T;99;ITD;Gen-2010;99;Y30-40;1_1;Y;16,8
NORMAL  Desktop\myfirstcsv.csv            utf-8[BOM][dos]   0% ☰   1:  1
"Desktop\myfirstcsv.csv" 128L, 5411C
```

Figure 11: the output file we obtain by considering codes only

The first record that we considered earlier is transcoded to a much shorter form as follows:

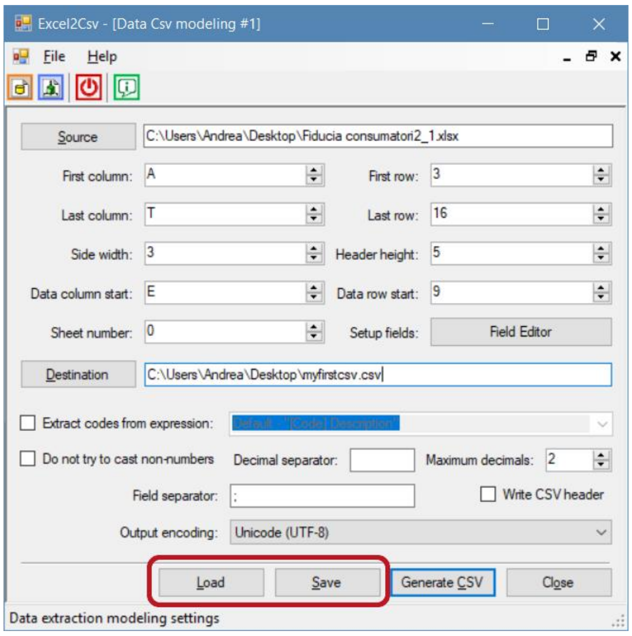
```
T;99;ITC;Gen-2010;99;TOTAL;COF_PER_10;N;13,7
```

6. Finally, we can also include a csv header to the file but, in order to do so, we first need to label each dimension correctly. We will accomplish this task later.
7. Although this is not visible from the example we shown, the software is also able to accept cells containing the “NaN” string (meaning “*not a number*”) and consider it a numeric value without giving errors.
8. In case your loader application can also load non-numeric values by design, it is finally possible to make Excel2Csv jump the test that checks whether a read value is a number by ticking the “*Do not try to cast non-numbers*” check box.

Loading and saving settings

The further we proceed through this guide, the more we will feel the need to save our settings to be later restored, either to momentarily suspend our activity or to repeat similar actions on similar tables in the future.

Doing this is quite straightforward in Excel2Csv: just either press the “Save” button to save all the current settings to an Xml file or press the “Load” button to load the settings from a previously saved file.


Excel2Csv - [Data Csv modeling #1]

Source	C:\Users\Andrea\Desktop\Fiducia consumatori2_1.xlsx		
First column:	A	First row:	3
Last column:	T	Last row:	16
Side width:	3	Header height:	5
Data column start:	E	Data row start:	9
Sheet number:	0	Setup fields:	Field Editor
Destination	C:\Users\Andrea\Desktop\myfirstcsv.csv		

☐ Extract codes from expression: Default - "[Code] Description"

☐ Do not try to cast non-numbers   Decimal separator: Maximum decimals: 2

Field separator: ;   ☐ Write CSV header

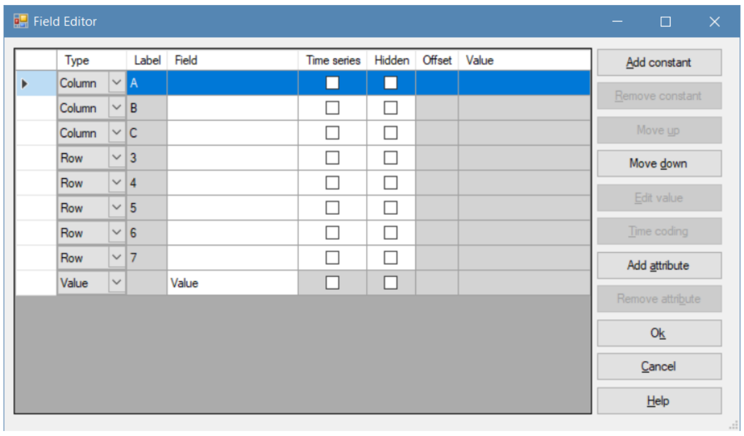
Output encoding: Unicode (UTF-8)

Data extraction modeling settings

Figure 12: the location of the "Load" and "Save" buttons

Dimension settings: the field editor

So far, we have been skipping the parameter setup process specifically related to each dimension. We are going to do this now by pressing the "*Field editor*" button. The following is the dialog window that appears by doing so:


Field Editor

Type	Label	Field	Time series	Hidden	Offset	Value
Column	A		☐	☐		
Column	B		☐	☐		
Column	C		☐	☐		
Row	3		☐	☐		
Row	4		☐	☐		
Row	5		☐	☐		
Row	6		☐	☐		
Row	7		☐	☐		
Value		Value	☐	☐		

Figure 13: the field editor still containing no settings

Axis references

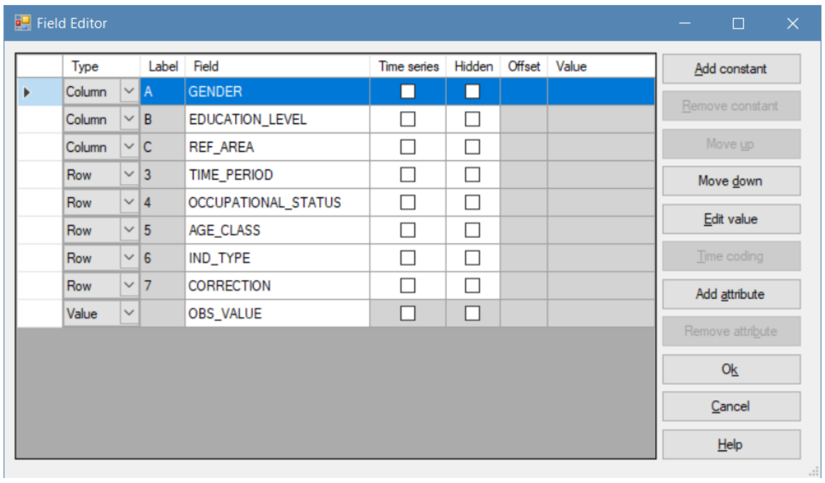
Each row in the editor shows a record field and is linked to a part of the table: the rows that show "Column" as Type refer to the axes found in the table side (each one addressed by the "Label" field), the ones with "Row" refer to the axes found in the header and finally we have the field dedicated to the actual observation value.

So, the field marked with type "Column" and label "A" refers to the axis represented along the column A of the table; the one marked with type "Row" and label "3" is linked to the axis located along the row 3, and so on.

Field labels

The column “Field” is used to label each record field with a dimension code or the name of the observation value field. When this feature is used in combination with the “Write CSV header” setting (located on the main window), a meaningful header is added at the top of the output csv.

Suppose that we want to label each record field in the following way:



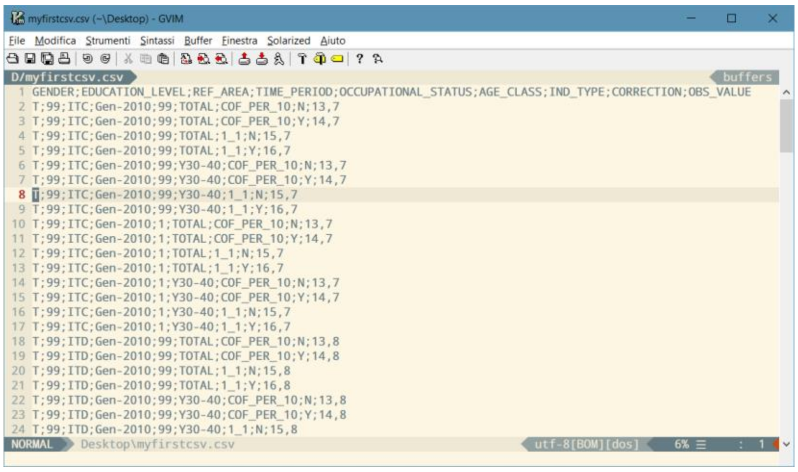
Type	Label	Field	Time series	Hidden	Offset	Value
Column	A	GENDER	☐	☐		
Column	B	EDUCATION_LEVEL	☐	☐		
Column	C	REF_AREA	☐	☐		
Row	3	TIME_PERIOD	☐	☐		
Row	4	OCCUPATIONAL_STATUS	☐	☐		
Row	5	AGE_CLASS	☐	☐		
Row	6	IND_TYPE	☐	☐		
Row	7	CORRECTION	☐	☐		
Value		OBS_VALUE	☐	☐		

Figure 14: labelling the record field columns

and that we tick the “Write CSV header” option:

Figure 15: enabling the csv header

What we obtain is an output file beginning as follows:


myfirstcsv.csv (~\Desktop) - GVIM
File Modifica Strumenti Sintassi Buffer Finestra Solarized Aiuto
D/myfirstcsv.csv
buffers
NORMAL
Desktop\myfirstcsv.csv
utf-8[BOM][dos]
6%

```
GENDER;EDUCATION_LEVEL;REF_AREA;TIME_PERIOD;OCCUPATIONAL_STATUS;AGE_CLASS;IND_TYPE;CORRECTION;OBS_VALUE
T;99;ITC;Gen-2010;99;TOTAL;COF_PER_10;N;13,7
T;99;ITC;Gen-2010;99;TOTAL;COF_PER_10;Y;14,7
T;99;ITC;Gen-2010;99;TOTAL;1_1;N;15,7
T;99;ITC;Gen-2010;99;TOTAL;1_1;Y;16,7
T;99;ITC;Gen-2010;99;Y30-40;COF_PER_10;N;13,7
T;99;ITC;Gen-2010;99;Y30-40;COF_PER_10;Y;14,7
T;99;ITC;Gen-2010;99;Y30-40;1_1;N;15,7
T;99;ITC;Gen-2010;99;Y30-40;1_1;Y;16,7
T;99;ITC;Gen-2010;1;TOTAL;COF_PER_10;N;13,7
T;99;ITC;Gen-2010;1;TOTAL;COF_PER_10;Y;14,7
T;99;ITC;Gen-2010;1;TOTAL;1_1;N;15,7
T;99;ITC;Gen-2010;1;TOTAL;1_1;Y;16,7
T;99;ITC;Gen-2010;1;Y30-40;COF_PER_10;N;13,7
T;99;ITC;Gen-2010;1;Y30-40;COF_PER_10;Y;14,7
T;99;ITC;Gen-2010;1;Y30-40;1_1;N;15,7
T;99;ITC;Gen-2010;1;Y30-40;1_1;Y;16,7
T;99;ITD;Gen-2010;99;TOTAL;COF_PER_10;N;13,8
T;99;ITD;Gen-2010;99;TOTAL;COF_PER_10;Y;14,8
T;99;ITD;Gen-2010;99;TOTAL;1_1;N;15,8
T;99;ITD;Gen-2010;99;TOTAL;1_1;Y;16,8
T;99;ITD;Gen-2010;99;Y30-40;COF_PER_10;N;13,8
T;99;ITD;Gen-2010;99;Y30-40;COF_PER_10;Y;14,8
T;99;ITD;Gen-2010;99;Y30-40;1_1;N;15,8
```

Figure 16: output csv file containing a header

The header is now:

GENDER;EDUCATION_LEVEL;REF_AREA;TIME_PERIOD;OCCUPATIONAL_STATUS;AGE_CLASS;IND_TYPE;CORRECTION;OBS_VALUE

Implicit constant dimensions

At the time being, a cell under the “Type” column can have five possible values:

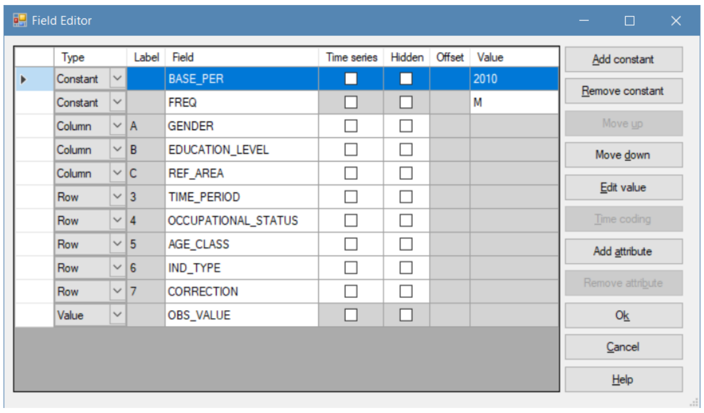
- Column
- Row
- Value
- Constant
- Attribute

being the last two special types reserved to dimensions with constant values that are not reported in the table we want to extract and to cell attributes respectively.

Regarding the example above, suppose that we want to add other dimensions that the table author excluded from the table simply because they have a constant value.

In this case, what we must do is press the “Add constant” button and a newly created field will appear in the editor. We can assign a constant value to this dimension by filling its cell under the “Value” column. Constants are the only fields that can have a “Value” assigned in our dialog window. Of course, we can also label the dimension itself as we already did for the other ones.

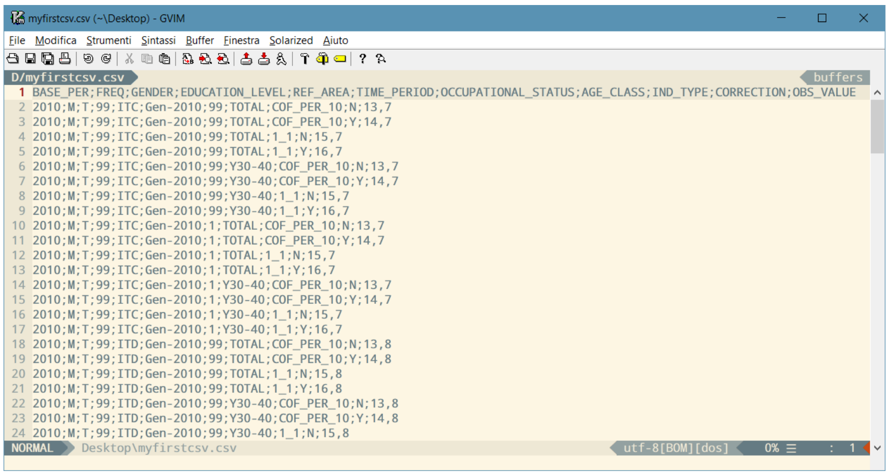
So, let’s add a couple of constant dimensions more: “FREQ” with value “M” and “BASE_PER” with value “2010”:



Type	Label	Field	Time series	Hidden	Offset	Value
Constant		BASE_PER	☐	☐		2010
Constant		FREQ	☐	☐		M
Column	A	GENDER	☐	☐		
Column	B	EDUCATION_LEVEL	☐	☐		
Column	C	REF_AREA	☐	☐		
Row	3	TIME_PERIOD	☐	☐		
Row	4	OCCUPATIONAL_STATUS	☐	☐		
Row	5	AGE_CLASS	☐	☐		
Row	6	IND_TYPE	☐	☐		
Row	7	CORRECTION	☐	☐		
Value		OBS_VALUE	☐	☐		

Figure 17: adding two constant dimensions

We will obtain an output file like the following:


myfirstcsv.csv (~\Desktop) - GVIM
File Modifica Strumenti Sintassi Buffer Finestra Solarized Aiuto
D/myfirstcsv.csv
buffers
NORMAL
Desktop\myfirstcsv.csv
utf-8[BOM][dos]
0%
: 1

```
BASE_PER;FREQ;GENDER;EDUCATION_LEVEL;REF_AREA;TIME_PERIOD;OCCUPATIONAL_STATUS;AGE_CLASS;IND_TYPE;CORRECTION;OBS_VALUE
2010;M;T;99;ITC;Gen-2010;99;TOTAL;COF_PER_10;N;13,7
2010;M;T;99;ITC;Gen-2010;99;TOTAL;COF_PER_10;Y;14,7
2010;M;T;99;ITC;Gen-2010;99;TOTAL;1_1;N;15,7
2010;M;T;99;ITC;Gen-2010;99;TOTAL;1_1;Y;16,7
2010;M;T;99;ITC;Gen-2010;99;Y30-40;COF_PER_10;N;13,7
2010;M;T;99;ITC;Gen-2010;99;Y30-40;COF_PER_10;Y;14,7
2010;M;T;99;ITC;Gen-2010;99;Y30-40;1_1;N;15,7
2010;M;T;99;ITC;Gen-2010;99;Y30-40;1_1;Y;16,7
2010;M;T;99;ITC;Gen-2010;1;TOTAL;COF_PER_10;N;13,7
2010;M;T;99;ITC;Gen-2010;1;TOTAL;COF_PER_10;Y;14,7
2010;M;T;99;ITC;Gen-2010;1;TOTAL;1_1;N;15,7
2010;M;T;99;ITC;Gen-2010;1;TOTAL;1_1;Y;16,7
2010;M;T;99;ITC;Gen-2010;1;Y30-40;COF_PER_10;N;13,7
2010;M;T;99;ITC;Gen-2010;1;Y30-40;COF_PER_10;Y;14,7
2010;M;T;99;ITC;Gen-2010;1;Y30-40;1_1;N;15,7
2010;M;T;99;ITC;Gen-2010;1;Y30-40;1_1;Y;16,7
2010;M;T;99;ITD;Gen-2010;99;TOTAL;COF_PER_10;N;13,8
2010;M;T;99;ITD;Gen-2010;99;TOTAL;COF_PER_10;Y;14,8
2010;M;T;99;ITD;Gen-2010;99;TOTAL;1_1;N;15,8
2010;M;T;99;ITD;Gen-2010;99;TOTAL;1_1;Y;16,8
2010;M;T;99;ITD;Gen-2010;99;Y30-40;COF_PER_10;N;13,8
2010;M;T;99;ITD;Gen-2010;99;Y30-40;COF_PER_10;Y;14,8
2010;M;T;99;ITD;Gen-2010;99;Y30-40;1_1;N;15,8
```

Figure 18: the output file showing the newly created constant dimensions

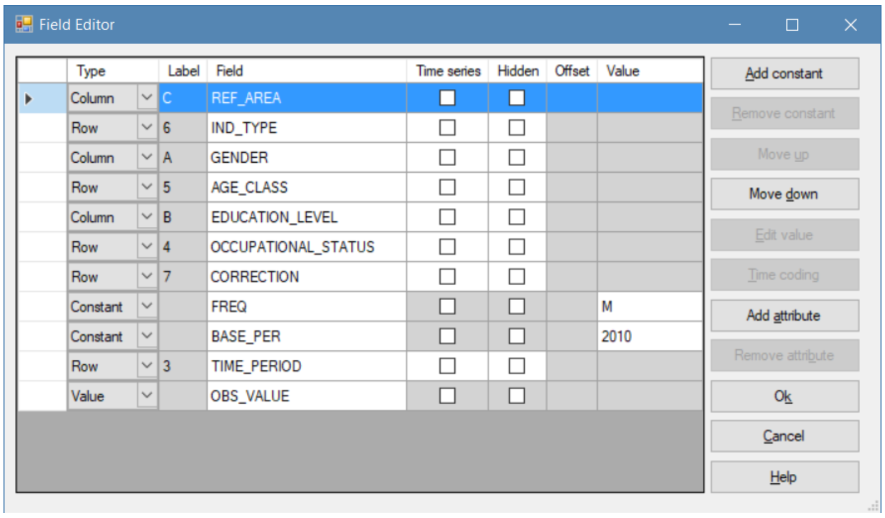
In case we change our mind, or we just made a mistake, it is possible to remove a constant dimension too by selecting the field to be deleted and pressing the “Remove constant” button.

Field ordering

Our output file is already fairly good. Nonetheless, suppose that the loader we want to use expects our csv fields to be ordered differently because the cube in our data warehouse that we need to populate, was built with the dimensions arranged in a different order. In this case we have no choice but to reorder the csv columns to match the dimensions of the cube structure.

To do so, we can select each row representing a csv field, and then move it up and down by pressing the “Move up” and “Move down” buttons respectively. This is a pretty straightforward operation.

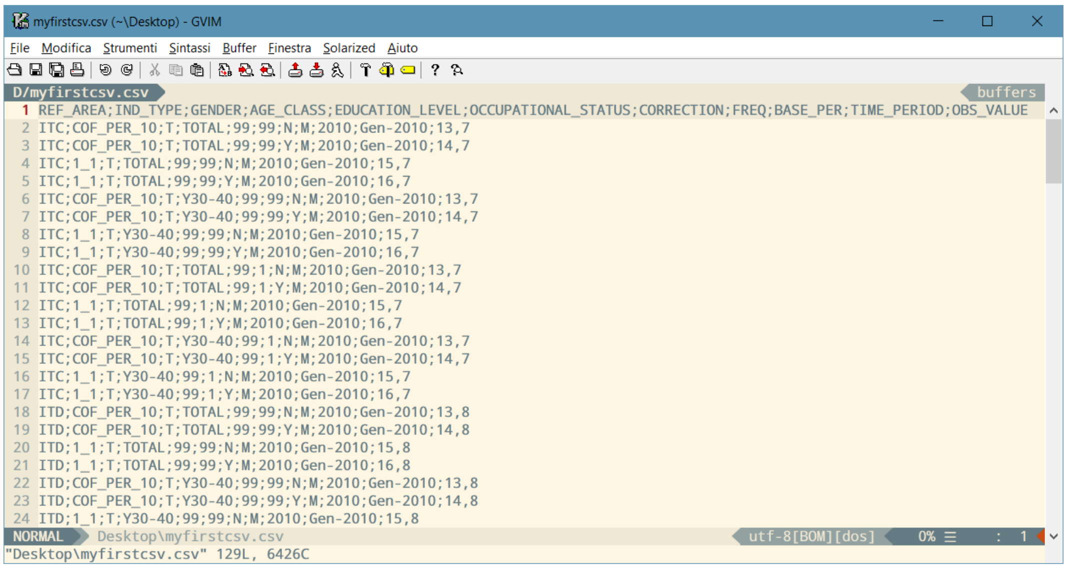
So, let’s reorder the fields as follows:



Type	Label	Field	Time series	Hidden	Offset	Value
Column	C	REF_AREA	☐	☐		
Row	6	IND_TYPE	☐	☐		
Column	A	GENDER	☐	☐		
Row	5	AGE_CLASS	☐	☐		
Column	B	EDUCATION_LEVEL	☐	☐		
Row	4	OCCUPATIONAL_STATUS	☐	☐		
Row	7	CORRECTION	☐	☐		
Constant		FREQ	☐	☐		M
Constant		BASE_PER	☐	☐		2010
Row	3	TIME_PERIOD	☐	☐		
Value		OBS_VALUE	☐	☐		

Figure 19: reordering the output record fields

We will obviously have the following output:



myfirstcsv.csv (~\Desktop) - GVIM
File Modifica Strumenti Sintassi Buffer Finestra Solarized Aiuto
D/myfirstcsv.csv
buffers

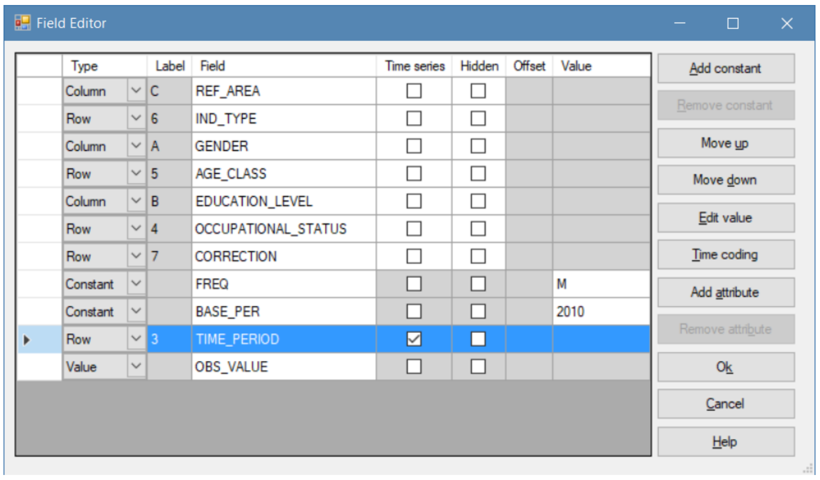
```
REF_AREA;IND_TYPE;GENDER;AGE_CLASS;EDUCATION_LEVEL;OCCUPATIONAL_STATUS;CORRECTION;FREQ;BASE_PER;TIME_PERIOD;OBS_VALUE
ITC;COF_PER_10;T;TOTAL;99;99;N;M;2010;Gen-2010;13,7
ITC;COF_PER_10;T;TOTAL;99;99;Y;M;2010;Gen-2010;14,7
ITC;1_1;T;TOTAL;99;99;N;M;2010;Gen-2010;15,7
ITC;1_1;T;TOTAL;99;99;Y;M;2010;Gen-2010;16,7
ITC;COF_PER_10;T;Y30-40;99;99;N;M;2010;Gen-2010;13,7
ITC;COF_PER_10;T;Y30-40;99;99;Y;M;2010;Gen-2010;14,7
ITC;1_1;T;Y30-40;99;99;N;M;2010;Gen-2010;15,7
ITC;1_1;T;Y30-40;99;99;Y;M;2010;Gen-2010;16,7
ITC;COF_PER_10;T;TOTAL;99;1;N;M;2010;Gen-2010;13,7
ITC;COF_PER_10;T;TOTAL;99;1;Y;M;2010;Gen-2010;14,7
ITC;1_1;T;TOTAL;99;1;N;M;2010;Gen-2010;15,7
ITC;1_1;T;TOTAL;99;1;Y;M;2010;Gen-2010;16,7
ITC;COF_PER_10;T;Y30-40;99;1;N;M;2010;Gen-2010;13,7
ITC;COF_PER_10;T;Y30-40;99;1;Y;M;2010;Gen-2010;14,7
ITC;1_1;T;Y30-40;99;1;N;M;2010;Gen-2010;15,7
ITC;1_1;T;Y30-40;99;1;Y;M;2010;Gen-2010;16,7
ITD;COF_PER_10;T;TOTAL;99;99;N;M;2010;Gen-2010;13,8
ITD;COF_PER_10;T;TOTAL;99;99;Y;M;2010;Gen-2010;14,8
ITD;1_1;T;TOTAL;99;99;N;M;2010;Gen-2010;15,8
ITD;1_1;T;TOTAL;99;99;Y;M;2010;Gen-2010;16,8
ITD;COF_PER_10;T;Y30-40;99;99;N;M;2010;Gen-2010;13,8
ITD;COF_PER_10;T;Y30-40;99;99;Y;M;2010;Gen-2010;14,8
ITD;1_1;T;Y30-40;99;99;N;M;2010;Gen-2010;15,8
NORMAL  Desktop\myfirstcsv.csv                                utf-8[BOM][dos]   0% ☰   :  1
"Desktop\myfirstcsv.csv" 129L, 6426C
```

Figure 20: output csv with rearranged fields

Time encoding: SDMX time mapping

In order to simplify the explanation and use a very small table, we chose to use a table that only shows one possible value for the time series, being this “Gen-2010”. The latter can’t possibly be formatted according to a standard SDMX time syntax, so we need to remap the time period dimension to get valid values.

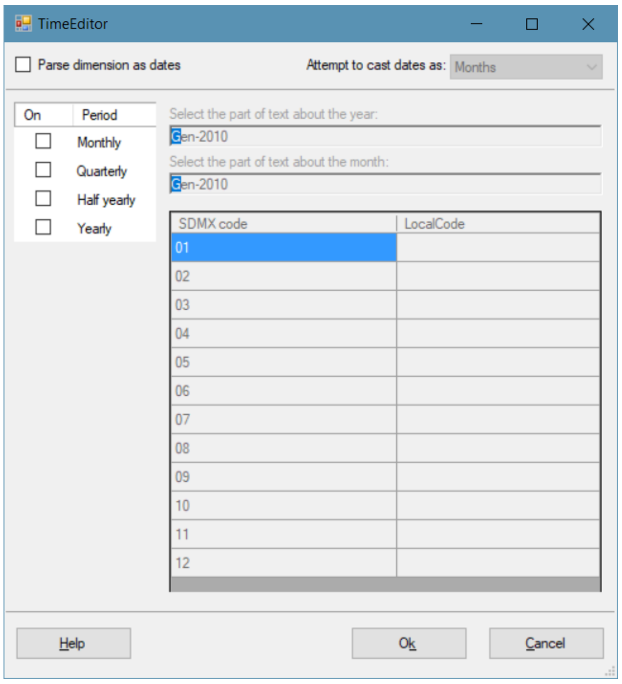
To do so, we first need to select our time series that we called “TIME_PERIOD” and then tell the application that it’s a time dimension by ticking the box under the “Time series” column. This action will enable the “Time coding” button whenever the same row is selected:



Type	Label	Field	Time series	Hidden	Offset	Value
Column	C	REF_AREA	☐	☐		
Row	6	IND_TYPE	☐	☐		
Column	A	GENDER	☐	☐		
Row	5	AGE_CLASS	☐	☐		
Column	B	EDUCATION_LEVEL	☐	☐		
Row	4	OCCUPATIONAL_STATUS	☐	☐		
Row	7	CORRECTION	☐	☐		
Constant		FREQ	☐	☐		M
Constant		BASE_PER	☐	☐		2010
Row	3	TIME_PERIOD	✓	☐		
Value		OBS_VALUE	☐	☐		

Figure 21: activating the "Time series" status for the TIME_PERIOD dimension

Let’s finally press the newly activated “Time coding” button to open a new dialog window:



☐ Parse dimension as dates

Attempt to cast dates as: Months

On	Period
☐	Monthly
☐	Quarterly
☐	Half yearly
☐	Yearly

Select the part of text about the year: Gen-2010

Select the part of text about the month: Gen-2010

SDMX code	LocalCode
01	
02	
03	
04	
05	
06	
07	
08	
09	
10	
11	
12	

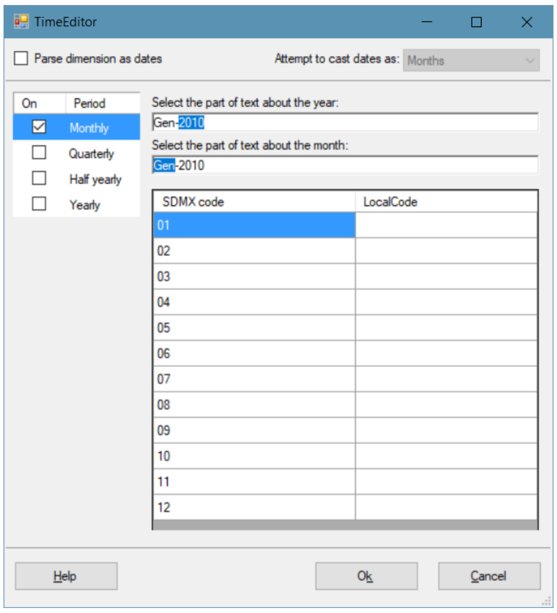
Figure 22: the time editor window

Since our time string is not a valid Excel datetime value, we can leave the “Parse dimension as dates” unselected. We will get back to this part later.

The “Gen-2010” string is supposed to represent a monthly value, being “Gen” the shortened form of “Gennaio”, meaning January in Italian. Thus, in order to properly set up the mapping, we need to tick the check box beside the “Monthly” row, at the left side of the window. This will show a grid of 2x12 cells and both the grid and the text boxes above it will be enabled to editing. Both the text boxes are filled by default with the first value that the reader met while scanning the table axis, but it is possible to edit such value.

What we need to tell the application next is what part of the shown value contains the information about the year and what part identifies the period, which in this case must be a month. We can do so by simply selecting these parts of text in the upper text boxes.

In our example, we will have to select “2010” in the first box and “Gen” in the second one, as shown in the picture:



☐ Parse dimension as dates

Attempt to cast dates as: Months

On	Period
☑	Monthly
☐	Quarterly
☐	Half yearly
☐	Yearly

Select the part of text about the year:
Gen-2010

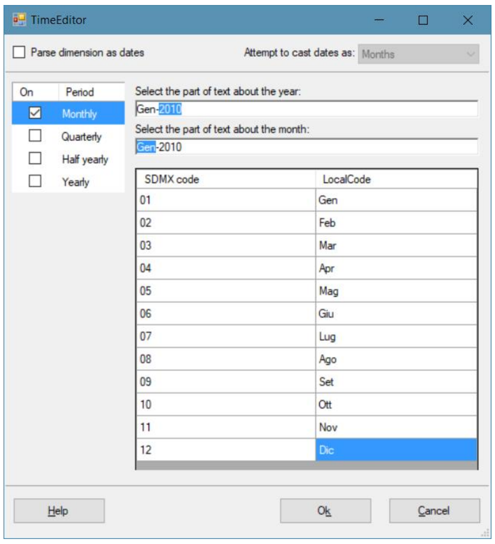
Select the part of text about the month:
Gen-2010

SDMX code	LocalCode
01	
02	
03	
04	
05	
06	
07	
08	
09	
10	
11	
12	

Figure 23: selecting the parts of text regarding the year and the period

Now that we have defined the portion of text specifying the period (month), we need to create a correspondence between the period subtext and the SDMX period values, by using the lower grid.

In our example, we will have to fill the grid as follows:


TimeEditor

Parse dimension as dates [ ] — Attempt to cast dates as: Months

On	Period
✓	Monthly
	Quarterly
	Half yearly
	Yearly

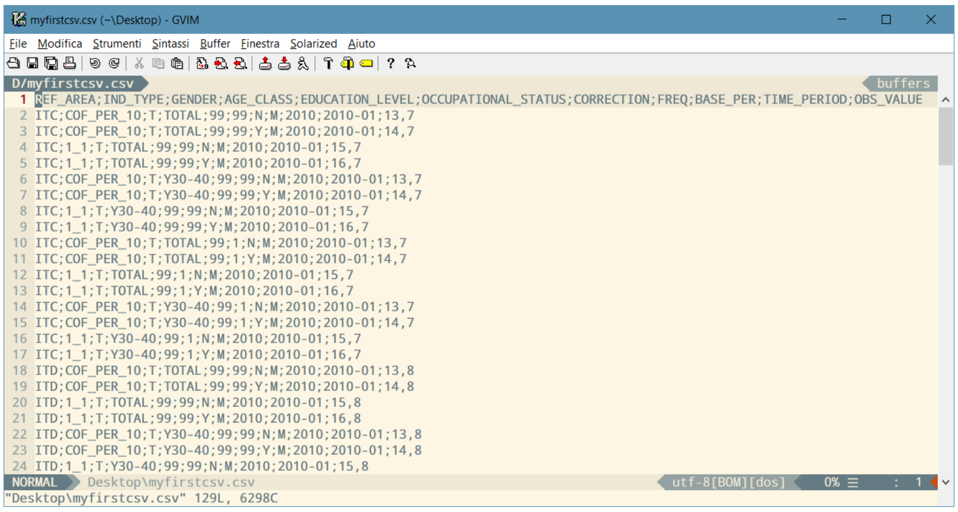
Select the part of text about the year: Gen-2010

Select the part of text about the month: Gen-2010

SDMX code	LocalCode
01	Gen
02	Feb
03	Mar
04	Apr
05	Mag
06	Giu
07	Lug
08	Ago
09	Set
10	Ott
11	Nov
12	Dic

Figure 24: defining a correspondence between the table codes and the SDMX period enumeration

Let’s confirm our settings by clicking OK on both the time and the field editor and let’s refresh our output again. The resulting csv file will be as follows:


myfirstcsv.csv (~\Desktop) - GVIM
File Modifica Strumenti Sintassi Buffer Finestra Solarized Aiuto
D/myfirstcsv.csv
buffers

```
REF_AREA;IND_TYPE;GENDER;AGE_CLASS;EDUCATION_LEVEL;OCCUPATIONAL_STATUS;CORRECTION;FREQ;BASE_PER;TIME_PERIOD;OBS_VALUE
ITC;COF_PER_10;T;TOTAL;99;99;N;M;2010;2010-01;13,7
ITC;COF_PER_10;T;TOTAL;99;99;Y;M;2010;2010-01;14,7
ITC;1_1;T;TOTAL;99;99;N;M;2010;2010-01;15,7
ITC;1_1;T;TOTAL;99;99;Y;M;2010;2010-01;16,7
ITC;COF_PER_10;T;Y30-40;99;99;N;M;2010;2010-01;13,7
ITC;COF_PER_10;T;Y30-40;99;99;Y;M;2010;2010-01;14,7
ITC;1_1;T;Y30-40;99;99;N;M;2010;2010-01;15,7
ITC;1_1;T;Y30-40;99;99;Y;M;2010;2010-01;16,7
ITC;COF_PER_10;T;TOTAL;99;1;N;M;2010;2010-01;13,7
ITC;COF_PER_10;T;TOTAL;99;1;Y;M;2010;2010-01;14,7
ITC;1_1;T;TOTAL;99;1;N;M;2010;2010-01;15,7
ITC;1_1;T;TOTAL;99;1;Y;M;2010;2010-01;16,7
ITC;COF_PER_10;T;Y30-40;99;1;N;M;2010;2010-01;13,7
ITC;COF_PER_10;T;Y30-40;99;1;Y;M;2010;2010-01;14,7
ITC;1_1;T;Y30-40;99;1;N;M;2010;2010-01;15,7
ITC;1_1;T;Y30-40;99;1;Y;M;2010;2010-01;16,7
ITD;COF_PER_10;T;TOTAL;99;99;N;M;2010;2010-01;13,8
ITD;COF_PER_10;T;TOTAL;99;99;Y;M;2010;2010-01;14,8
ITD;1_1;T;TOTAL;99;99;N;M;2010;2010-01;15,8
ITD;1_1;T;TOTAL;99;99;Y;M;2010;2010-01;16,8
ITD;COF_PER_10;T;Y30-40;99;99;N;M;2010;2010-01;13,8
ITD;COF_PER_10;T;Y30-40;99;99;Y;M;2010;2010-01;14,8
ITD;1_1;T;Y30-40;99;99;N;M;2010;2010-01;15,8
NORMAL  Desktop\myfirstcsv.csv                         utf-8[BOM][dos]   0% ☰   :  1
"Desktop\myfirstcsv.csv" 129L, 6298C
```

Figure 25: the output csv with a correctly mapped time series dimension

Whereas the file previously showed “Gen-2010”, now its fields show the “2010-01” string instead, which is a correctly SDMX formatted value representing January 2010.

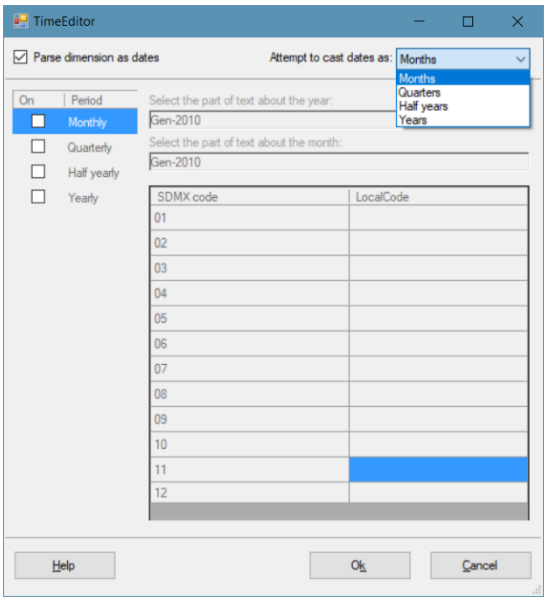
Time encoding: datetime data type casting

In our example we had a string representing a time reference in a non-standard fashion.

Now suppose that for some reason the author of the table wanted to use valid Excel datetime values to express the time series. When I say datetime, I mean a valid Excel data type, not just a string resembling it.

In this case, we need a way to read such values, ditch the time parts, take the year part and cast the rest to valid SDMX periods, which can be monthly, quarterly, half-yearly or yearly.

This task can be accomplished via the time editor by ticking the “*Parse dimensions as dates*” check box and specifying the period we want the month in the date to be casted to:



☑ Parse dimension as dates

Attempt to cast dates as: Months

- Months
- Quarters
- Half years
- Years

On	Period
☐	Monthly
☐	Quarterly
☐	Half yearly
☐	Yearly

Select the part of text about the year: Gen-2010

Select the part of text about the month: Gen-2010

SDMX code	LocalCode
01	
02	
03	
04	
05	
06	
07	
08	
09	
10	
11	
12	

Figure 26: casting valid Excel datetime values to valid SDMX time references

So, if we had a datetime value in the form "02*/10/2017 15:47*" and we wanted to cast it to a monthly series, we would have obtained the "2017-10" value. Had we have wanted to retrieve quarterly values, we would have obtained "2017-Q4" instead, and so on.

Hiding dimensions: excluding row or column labels from the output

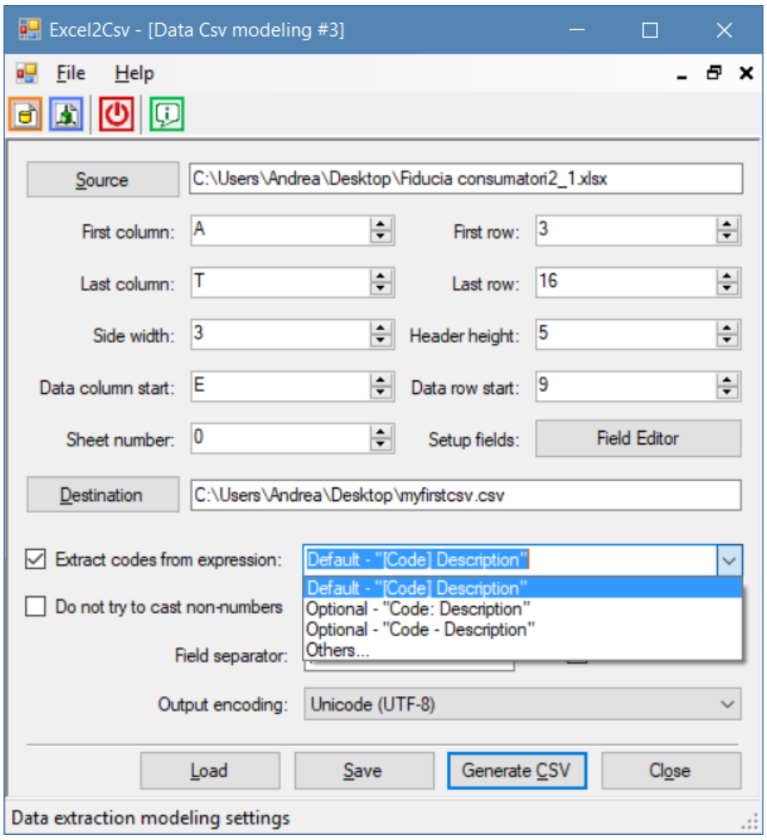
As you surely have noticed, in the field editor there are check boxes under the "hide" column for each field. As it can be easily guessed, by selecting this option it is possible to hide the corresponding dimension that the application detected while reading the file.

Please notice that this operation was added only to give the ability to hide rows/columns from our header/side in case they don't contain any useful information for a data csv, hence such rows/columns must contain referential metadata or no meaningful information at all. Please do not try to use this feature to simply suppress data corresponding to the hidden dimensions, for the application cannot do an SQL-like "*select distinct*" operation on the variable combinations, since they might end up on ceasing to reference data in an unambiguous fashion.

Customising the code extraction: The Expression Editor

Earlier we saw how to enable the code extraction feature in order to fetch only a variable code off a whole cell content containing multiple information. The tuning we automatically chose was the default one, which fetches codes off a "[Code] Description" type expression.

There are other two predefined expressions: the "Code: Description" and the "Code – Description" ones, that are selectable from the combo box aside the code extraction check box.


Excel2Csv - [Data Csv modeling #3]

File Help

Source: C:\Users\Andrea\Desktop\Fiducia consumatori2_1.xlsx

First column: A
First row: 3
Last column: T
Last row: 16
Side width: 3
Header height: 5
Data column start: E
Data row start: 9
Sheet number: 0
Setup fields: Field Editor

Destination: C:\Users\Andrea\Desktop\myfirstcsv.csv

☑ Extract codes from expression: Default - "[Code] Description"

Default - "[Code] Description"
Optional - "Code: Description"
Optional - "Code - Description"
Others...

☐ Do not try to cast non-numbers

Field separator:

Output encoding: Unicode (UTF-8)

Load | Save | Generate CSV | Close

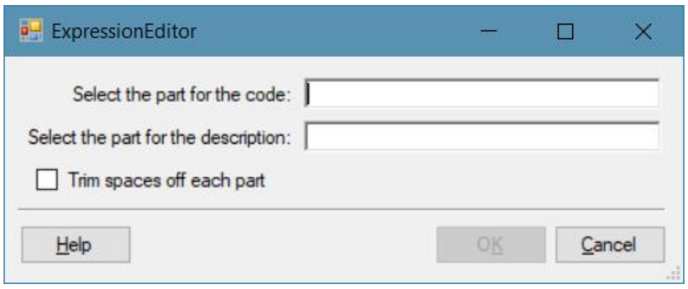
Data extraction modeling settings

Figure 27: list of the available code extraction expressions

To summarise, the parts of the predefined expressions that define the codes are obviously highlighted as follows:

- [Code] Description
- Code: Description
- Code - Description

While these first three options might fulfil the needs of most users, they might not be enough. For this reason, there is a fourth option, represented by the "Others…" label, that opens a whole new window for the expression customisation.


ExpressionEditor

Select the part for the code:

Select the part for the description:

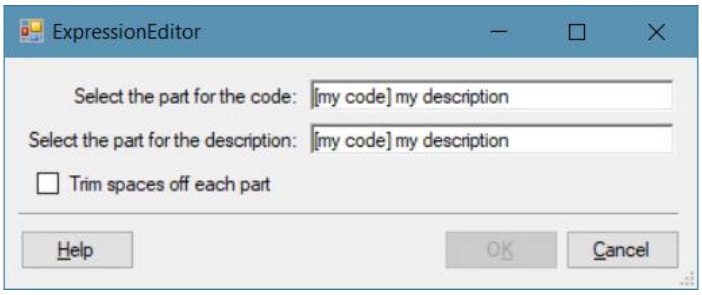
☐ Trim spaces off each part

Help | OK | Cancel

Figure 28: The Expression Editor

By typing in either of the two text boxes, you edit both at the same time. In this way you can highlight different portions of the same text. In fact, this is what we must do here: highlight the part of the text dedicated to the code and the one dedicated to the description.

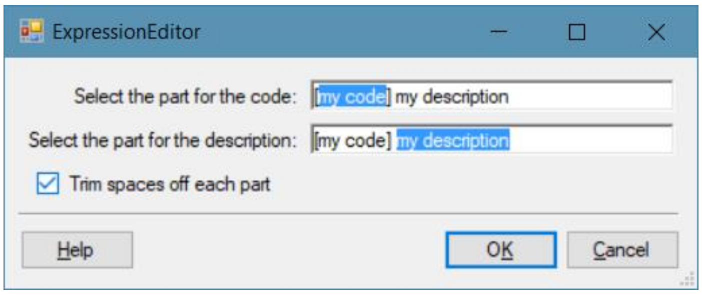
Let’s define some expression that is similar to the default one but that also allows us to see how this window works. Be the chosen expression “[my code] my description”.



Select the part for the code: [my code] my description
Select the part for the description: [my code] my description
☐ Trim spaces off each part

Figure 29: filling either text box edits the other one as well

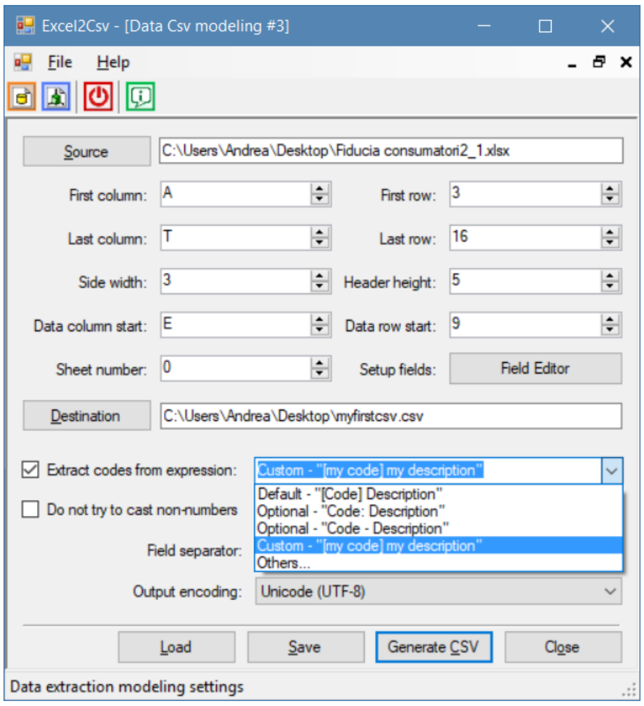
What we need to do next is highlight the code part in the first text box and the description part in the second one. Finally, we can also choose whether extra spaces as the sides of the single parts shall not be considered in our expression.



Select the part for the code: [my code] my description
Select the part for the description: [my code] my description
☑ Trim spaces off each part

Figure 30: selecting parts of the code and the description and enabling the white space trimming

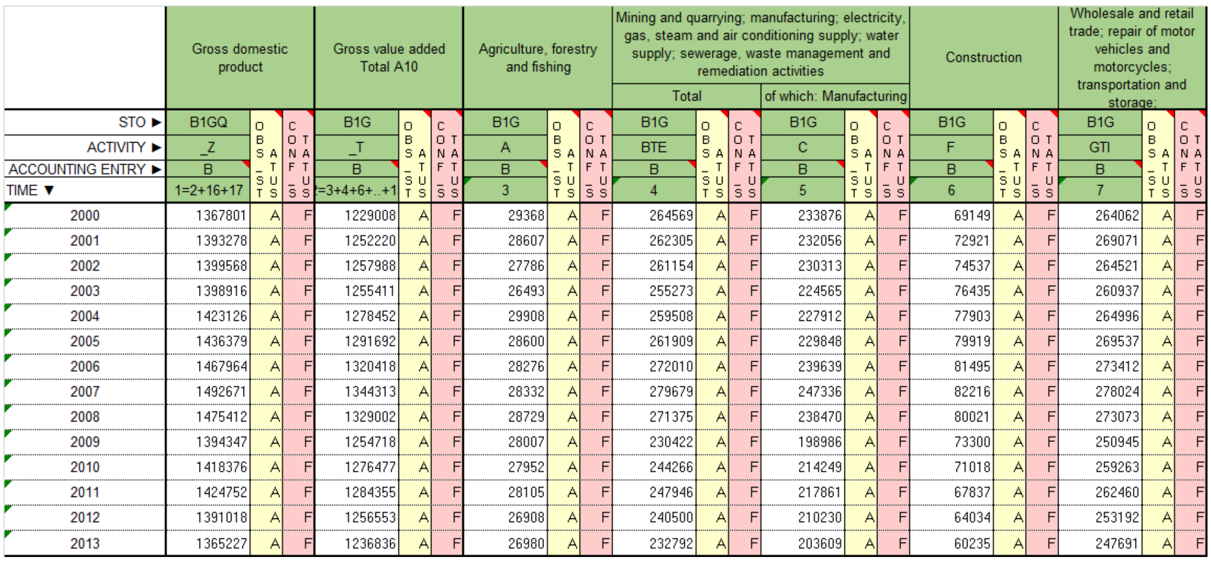
That’s it, by clicking the Ok button a new entry is added to the combo box and is automatically selected. This new setting works in a similar fashion to the default one.



Source: C:\Users\Andrea\Desktop\Fiducia consumatori2_1.xlsx
First column: A
First row: 3
Last column: T
Last row: 16
Side width: 3
Header height: 5
Data column start: E
Data row start: 9
Sheet number: 0
Setup fields: Field Editor
Destination: C:\Users\Andrea\Desktop\myfirstcsv.csv
☑ Extract codes from expression: Custom - "[my code] my description"
Default - "[Code] Description"
Optional - "Code: Description"
Optional - "Code - Description"
Custom - "[my code] my description"
Others...
☐ Do not try to cast non-numbers
Field separator:
Output encoding: Unicode (UTF-8)
Data extraction modeling settings

Cell Attributes

Suppose you have a table like the following one:



	Gross domestic product			Gross value added Total A10			Agriculture, forestry and fishing			Mining and quarrying; manufacturing; electricity, gas, steam and air conditioning supply; water supply; sewerage, waste management and remediation activities		
										Total		
STO ▶	B1GQ	OBS_STATUS	CONF_STATUS	B1G	OBS_STATUS	CONF_STATUS	B1G	OBS_STATUS	CONF_STATUS	B1G	OBS_STATUS	CONF_STATUS
ACTIVITY ▶	_Z			_T			A			BTE		
ACCOUNTING ENTRY ▶	B			B			B			B		
TIME ▼	1=2+16+17			=3+4+6+..+1			3			4		
2000	1367801	A	F	1229008	A	F	29368	A	F	264569	A	F
2001	1393278	A	F	1252220	A	F	28607	A	F	262305	A	F
2002	1399568	A	F	1257988	A	F	27786	A	F	261154	A	F
2003	1398916	A	F	1255411	A	F	26493	A	F	255273	A	F
2004	1423126	A	F	1278452	A	F	29908	A	F	259508	A	F
2005	1436379	A	F	1291692	A	F	28600	A	F	261909	A	F
2006	1467964	A	F	1320418	A	F	28276	A	F	272010	A	F
2007	1492671	A	F	1344313	A	F	28332	A	F	279679	A	F
2008	1475412	A	F	1329002	A	F	28729	A	F	271375	A	F
2009	1394347	A	F	1254718	A	F	28007	A	F	230422	A	F
2010	1418376	A	F	1276477	A	F	27952	A	F	244266	A	F
2011	1424752	A	F	1284355	A	F	28105	A	F	247946	A	F
2012	1391018	A	F	1256553	A	F	26908	A	F	240500	A	F
2013	1365227	A	F	1236836	A	F	26980	A	F	232792	A	F

	Mining and quarrying; manufacturing; electricity, gas, steam and air conditioning supply; water supply; sewerage, waste management and remediation activities			Construction			Wholesale and retail trade; repair of motor vehicles and motorcycles; transportation and storage;		
	of which: Manufacturing								
STO ▶	B1G	OBS_STATUS	CONF_STATUS	B1G	OBS_STATUS	CONF_STATUS	B1G	OBS_STATUS	CONF_STATUS
ACTIVITY ▶	C			F			GTI		
ACCOUNTING ENTRY ▶	B			B			B		
TIME ▼	5			6			7		
2000	233876	A	F	69149	A	F	264062	A	F
2001	232056	A	F	72921	A	F	269071	A	F
2002	230313	A	F	74537	A	F	264521	A	F
2003	224565	A	F	76435	A	F	260937	A	F
2004	227912	A	F	77903	A	F	264996	A	F
2005	229848	A	F	79919	A	F	269537	A	F
2006	239639	A	F	81495	A	F	273412	A	F
2007	247336	A	F	82216	A	F	278024	A	F
2008	238470	A	F	80021	A	F	273073	A	F
2009	198986	A	F	73300	A	F	250945	A	F
2010	214249	A	F	71018	A	F	259263	A	F
2011	217861	A	F	67837	A	F	262460	A	F
2012	210230	A	F	64034	A	F	253192	A	F
2013	203609	A	F	60235	A	F	247691	A	F

Figure 31: A table containing cell attributes

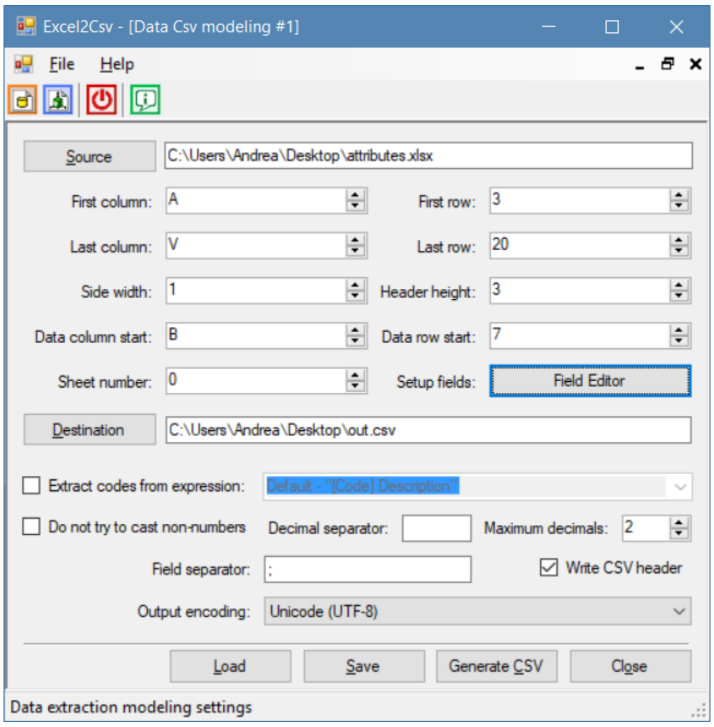
As you can see, the cells in the table data area not only contain data but also codes that refer to the OBS_STATUS and CONF_STATUS attributes.

While attributes might look like plain dimensions in a Csv track or a dataset schema, they don't really contribute to the determination of the data; on the contrary, they are determined together with data according to the relationship:

$$Ds(Dim_1, Dim_2, \dots, Dim_m) \rightarrow (\mathbb{R} \cup \emptyset, Attr_1, \dots, Attr_n)$$

We need a way to retrieve these attributes from the table, do not let the application consider their cells as data cells, and finally report them on the destination Csv as if they were dimensions.

Let's first configure the application in the following way:


Excel2Csv - [Data Csv modeling #1]

File Help

Source	C:\Users\Andrea\Desktop\attributes.xlsx		
First column:	A	First row:	3
Last column:	V	Last row:	20
Side width:	1	Header height:	3
Data column start:	B	Data row start:	7
Sheet number:	0	Setup fields:	Field Editor
Destination	C:\Users\Andrea\Desktop\out.csv		

☐ Extract codes from expression: Default - "[Code] Description"

☐ Do not try to cast non-numbers Decimal separator: Maximum decimals: 2

Field separator: ; ☑ Write CSV header

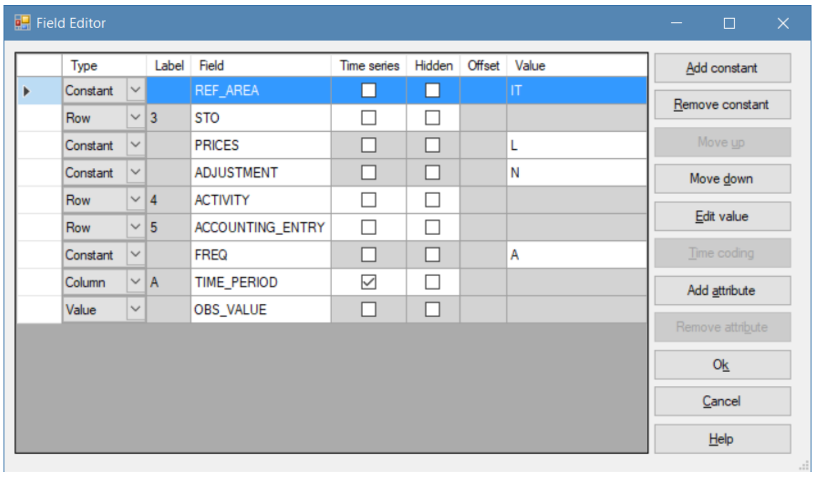
Output encoding: Unicode (UTF-8)

Load Save Generate CSV Close

Data extraction modeling settings

Figure 32: the configuration on the main data window

And the fields as:


Field Editor

Type	Label	Field	Time series	Hidden	Offset	Value
Constant		REF_AREA	☐	☐		IT
Row	3	STO	☐	☐		
Constant		PRICES	☐	☐		L
Constant		ADJUSTMENT	☐	☐		N
Row	4	ACTIVITY	☐	☐		
Row	5	ACCOUNTING_ENTRY	☐	☐		
Constant		FREQ	☐	☐		A
Column	A	TIME_PERIOD	☑	☐		
Value		OBS_VALUE	☐	☐		

Figure 33: the initial field configuration

What we need to do now, is tell the application what offset it can find an attribute at the right of a data column. To do so, we just must click the “Add attribute” button: a new row will be inserted with the default offset value set to 1.

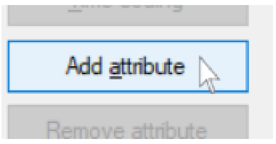
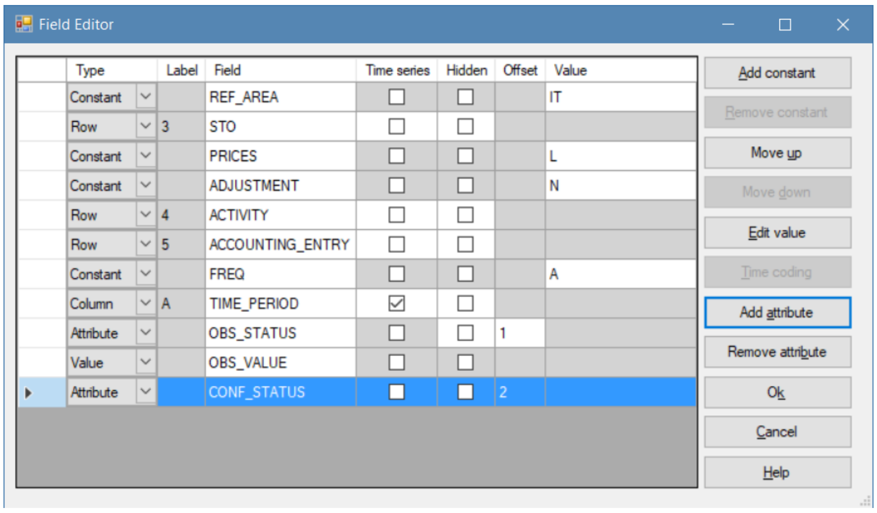


Figure 34: the "Add attribute" button

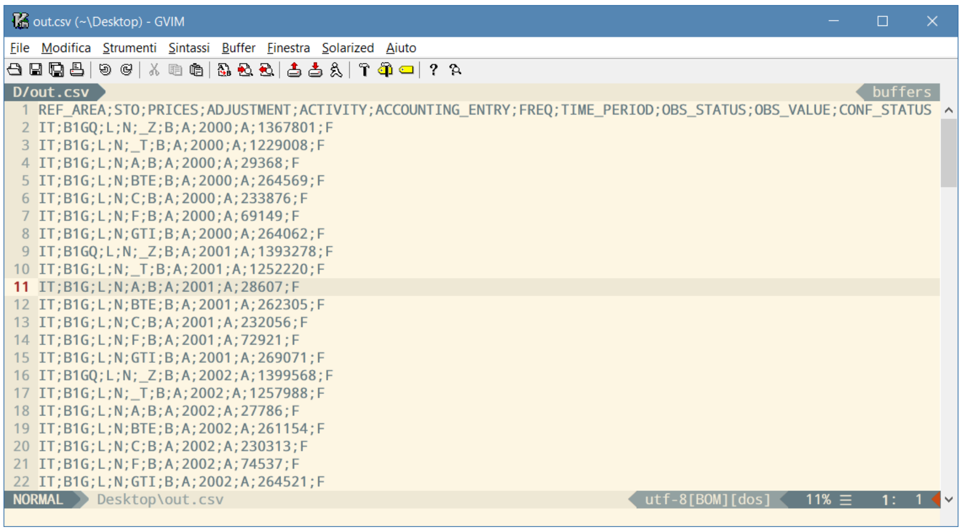
Let’s call it OBS_STATUS. Let’s repeat the operation and call the new attribute CONF_STATUS. As you can see, the new row is being initialized with an offset value set to 2.



Type	Label	Field	Time series	Hidden	Offset	Value
Constant		REF_AREA	☐	☐		IT
Row	3	STO	☐	☐		
Constant		PRICES	☐	☐		L
Constant		ADJUSTMENT	☐	☐		N
Row	4	ACTIVITY	☐	☐		
Row	5	ACCOUNTING_ENTRY	☐	☐		
Constant		FREQ	☐	☐		A
Column	A	TIME_PERIOD	☑	☐		
Attribute		OBS_STATUS	☐	☐	1	
Value		OBS_VALUE	☐	☐		
Attribute		CONF_STATUS	☐	☐	2	

Figure 35: the final configuration of the fields supporting attributes

That’s it, the application will jump the attribute columns while fetching data, and then it will gather the attributes from the respective column offsets.


out.csv (~\Desktop) - GVIM
File Modifica Strumenti Sintassi Buffer Finestra Solarized Aiuto
D/out.csv
buffers
NORMAL
Desktop\out.csv
utf-8[BOM][dos]
11%
1: 1

```
REF_AREA;STO;PRICES;ADJUSTMENT;ACTIVITY;ACCOUNTING_ENTRY;FREQ;TIME_PERIOD;OBS_STATUS;OBS_VALUE;CONF_STATUS
IT;B1GQ;L;N;_Z;B;A;2000;A;1367801;F
IT;B1G;L;N;_T;B;A;2000;A;1229008;F
IT;B1G;L;N;A;B;A;2000;A;29368;F
IT;B1G;L;N;BTE;B;A;2000;A;264569;F
IT;B1G;L;N;C;B;A;2000;A;233876;F
IT;B1G;L;N;F;B;A;2000;A;69149;F
IT;B1G;L;N;GTI;B;A;2000;A;264062;F
IT;B1GQ;L;N;_Z;B;A;2001;A;1393278;F
IT;B1G;L;N;_T;B;A;2001;A;1252220;F
IT;B1G;L;N;A;B;A;2001;A;28607;F
IT;B1G;L;N;BTE;B;A;2001;A;262305;F
IT;B1G;L;N;C;B;A;2001;A;232056;F
IT;B1G;L;N;F;B;A;2001;A;72921;F
IT;B1G;L;N;GTI;B;A;2001;A;269071;F
IT;B1GQ;L;N;_Z;B;A;2002;A;1399568;F
IT;B1G;L;N;_T;B;A;2002;A;1257988;F
IT;B1G;L;N;A;B;A;2002;A;27786;F
IT;B1G;L;N;BTE;B;A;2002;A;261154;F
IT;B1G;L;N;C;B;A;2002;A;230313;F
IT;B1G;L;N;F;B;A;2002;A;74537;F
IT;B1G;L;N;GTI;B;A;2002;A;264521;F
```

Figure 36: the expected results of adding attributes

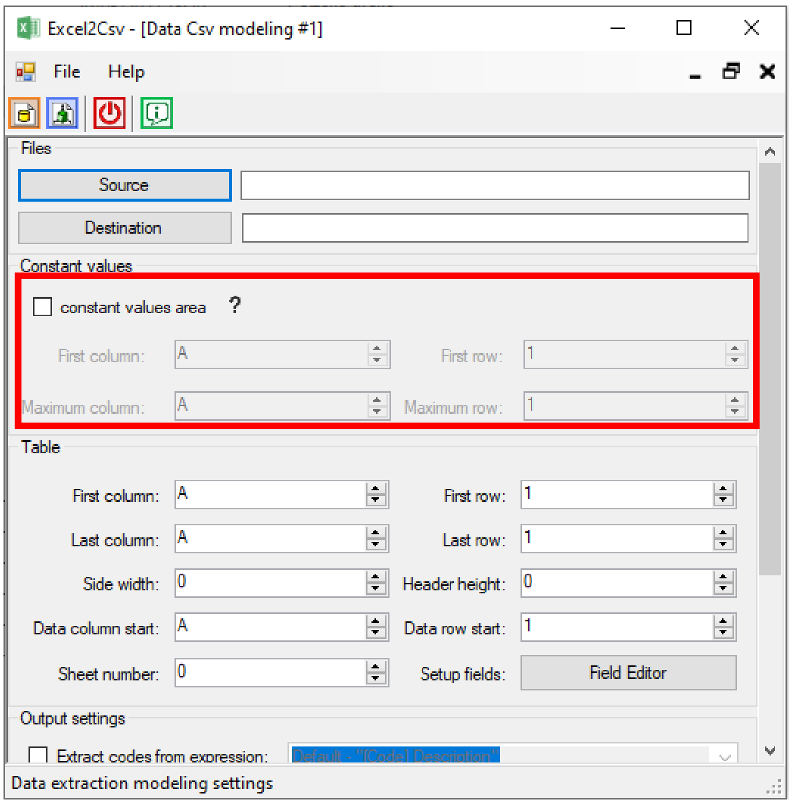
Please also notice that an attribute can be hidden mainly to let the application have an idea of the number of columns it has to jump while fetching data.

Finally, an attribute can be removed by clicking the “Remove attribute” button.

The extraction of constant values from worksheet’s cells

Starting from the version 1.3 the graphical user interface has been modified in order to provide a new functionality: the export, in the csv output file, also of the constant values located into specific Excel worksheet’s cells. This new function and the related updates of the graphical interface don’t interfer with the other functionalities of the software and therefore what has been described so far and is described in the following paragraphs continues to be valid.

After having opened a new data window, is possible to see that there’s a new “constant value area” (inside the red rectangle) for configuring the the extraction of constant values from some specific excel worksheet’s cells: this area is disabled by default



Excel2Csv - [Data Csv modeling #1]

File Help

Files

Source

Destination

Constant values

☐ constant values area ?

First column: A
First row: 1
Maximum column: A
Maximum row: 1

Table

First column: A
First row: 1
Last column: A
Last row: 1
Side width: 0
Header height: 0
Data column start: A
Data row start: 1
Sheet number: 0
Setup fields: Field Editor

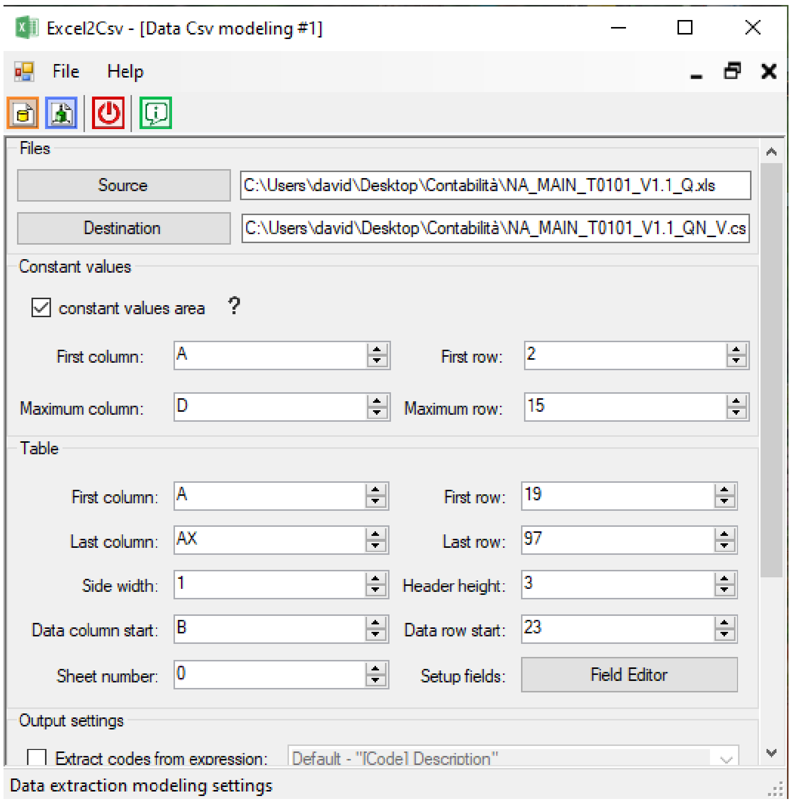
Output settings

☐ Extract codes from expression: Default - "[Code] Description"

Data extraction modeling settings

Figure 37: The new user interface with the ‘constant values’ area

In order to activate this area in necessary to flag the related checkbox, and to populate the fields that indicate the section of the excel worksheet containing the constant value cells (Figure 39): 'First Column', 'First Row', 'Maximum Column' and 'Maximum Row'.



Excel2Csv - [Data Csv modeling #1]

File Help

Files

Source: C:\Users\david\Desktop\Contabilità\NA_MAIN_T0101_V1.1_Q.xls

Destination: C:\Users\david\Desktop\Contabilità\NA_MAIN_T0101_V1.1_QN_V.cs

Constant values

[✓] constant values area ?

First column: A First row: 2

Maximum column: D Maximum row: 15

Table

First column: A First row: 19

Last column: AX Last row: 97

Side width: 1 Header height: 3

Data column start: B Data row start: 23

Sheet number: 0 Setup fields: Field Editor

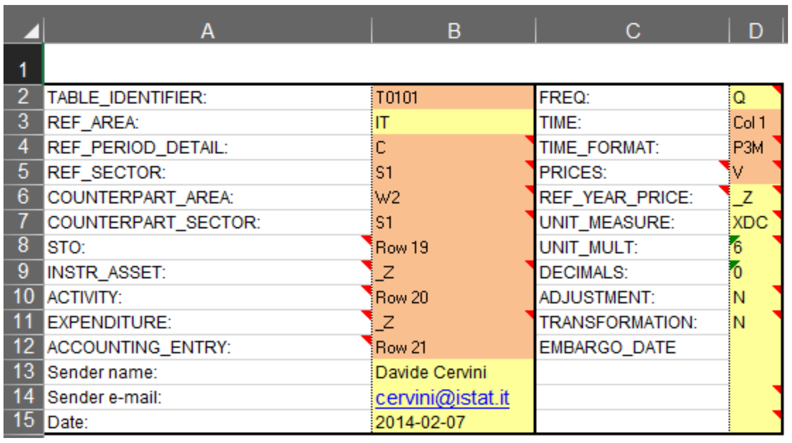
Output settings

[ ] Extract codes from expression: Default - "[Code] Description"

Data extraction modeling settings

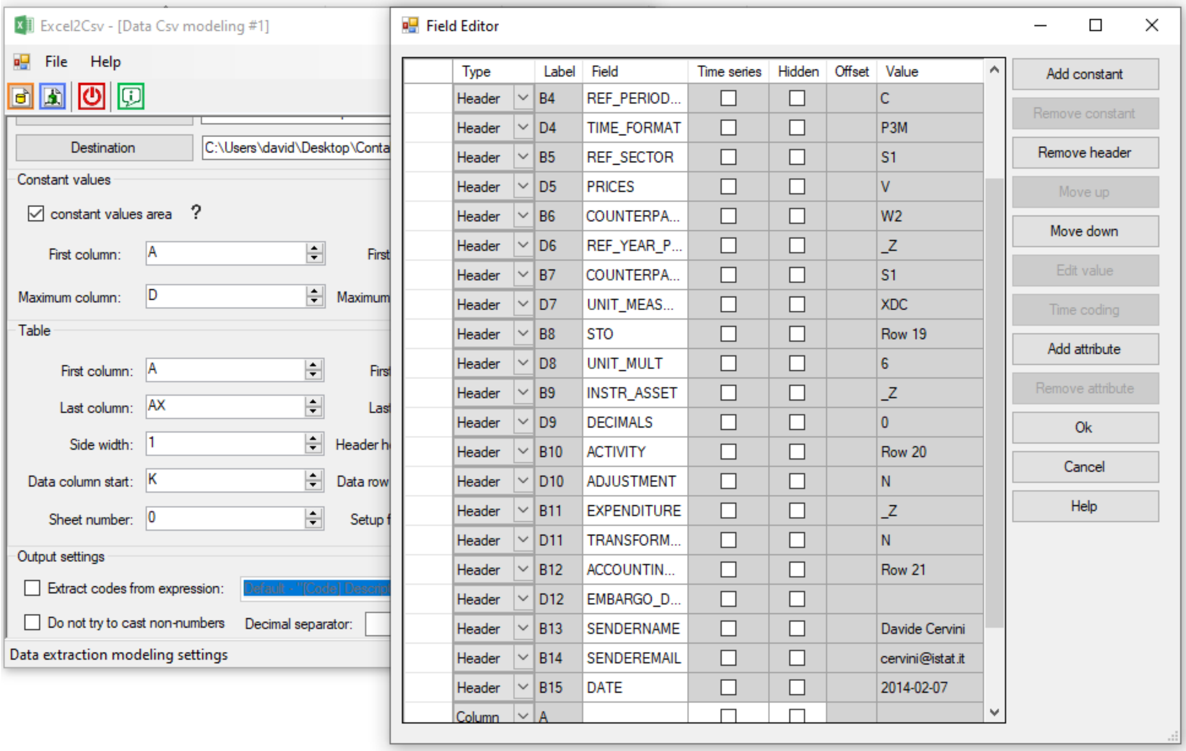
Figure 388: Selecting the area that contains the constant values

Once configured the parameters for indicating the constant value cells area in the excel worksheet, opening the Field Editor, is possible to visualize the insertion of constant value items whose type is 'Header'; for each 'Header' item is possible to edit only the 'Field' entry containing the item's identifier (Figure 40).



	A	B	C	D
1				
2	TABLE_IDENTIFIER:	T0101	FREQ:	Q
3	REF_AREA:	IT	TIME:	Col 1
4	REF_PERIOD_DETAIL:	C	TIME_FORMAT:	P3M
5	REF_SECTOR:	S1	PRICES:	V
6	COUNTERPART_AREA:	W2	REF_YEAR_PRICE:	_Z
7	COUNTERPART_SECTOR:	S1	UNIT_MEASURE:	XDC
8	STO:	Row 19	UNIT_MULT:	6
9	INSTR_ASSET:	_Z	DECIMALS:	0
10	ACTIVITY:	Row 20	ADJUSTMENT:	N
11	EXPENDITURE:	_Z	TRANSFORMATION:	N
12	ACCOUNTING_ENTRY:	Row 21	EMBARGO_DATE	
13	Sender name:	Davide Cervini		
14	Sender e-mail:	cervini@istat.it		
15	Date:	2014-02-07		

Figure 399: The constant values area on the spreadsheet



Destination	C:\Users\david\Desktop\Conta
Constant values	
☑ constant values area ?	
First column:	A
Maximum column:	D
Table	
First column:	A
Last column:	AX
Side width:	1
Data column start:	K
Sheet number:	0
Output settings	
☐ Extract codes from expression:	Default - "[Code] Descri
☐ Do not try to cast non-numbers	Decimal separator:
Data extraction modeling settings	

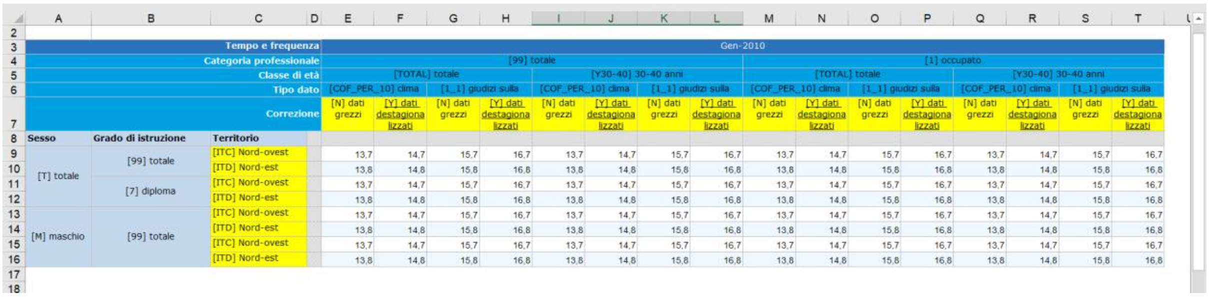
Type	Label	Field	Time series	Hidden	Offset	Value
Header	B4	REF_PERIOD...	☐	☐		C
Header	D4	TIME_FORMAT	☐	☐		P3M
Header	B5	REF_SECTOR	☐	☐		S1
Header	D5	PRICES	☐	☐		V
Header	B6	COUNTERPA...	☐	☐		W2
Header	D6	REF_YEAR_P...	☐	☐		_Z
Header	B7	COUNTERPA...	☐	☐		S1
Header	D7	UNIT_MEAS...	☐	☐		XDC
Header	B8	STO	☐	☐		Row 19
Header	D8	UNIT_MULT	☐	☐		6
Header	B9	INSTR_ASSET	☐	☐		_Z
Header	D9	DECIMALS	☐	☐		0
Header	B10	ACTIVITY	☐	☐		Row 20
Header	D10	ADJUSTMENT	☐	☐		N
Header	B11	EXPENDITURE	☐	☐		_Z
Header	D11	TRANSFORM...	☐	☐		N
Header	B12	ACCOUNTIN...	☐	☐		Row 21
Header	D12	EMBARGO_D...	☐	☐		
Header	B13	SENDERNAME	☐	☐		Davide Cervini
Header	B14	SENDEREMAIL	☐	☐		cervini@istat.it
Header	B15	DATE	☐	☐		2014-02-07
Column	A		☐	☐		

Figure 40: The 'field editor' where all constant values area have been inserted as 'Header'

The code list extraction mode

Let's get back to the table shown in figure 6: there are many axes there that contain both codes and descriptions for the dimensions they refer to.

Suppose we now want to extract the code lists from a couple of axes, namely the axis on row 6 and the one on column C. Let's highlight those axes:



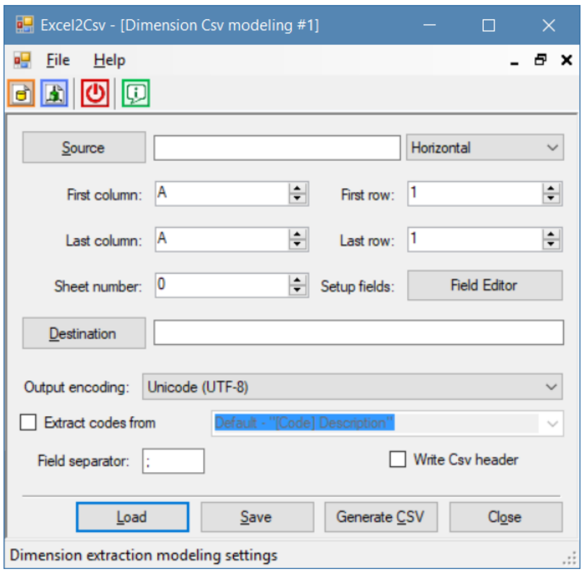
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
2																				
3			Tempo e frequenza		Gen-2010															
4			Categoria professionale		[99] totale								[1] occupato							
5			Classe di età		[TOTAL] totale				[Y30-40] 30-40 anni				[TOTAL] totale				[Y30-40] 30-40 anni			
6			Tipo dato		[COF_PER_10] clima		[1_1] giudizi sulla		[COF_PER_10] clima		[1_1] giudizi sulla		[COF_PER_10] clima		[1_1] giudizi sulla		[COF_PER_10] clima		[1_1] giudizi sulla	
7			Correzione		[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati	[N] dati grezzi	[Y] dati destagionalizzati
8	Sesso	Grado di istruzione	Territorio																	
9	[T] totale	[99] totale	[ITC] Nord-ovest		13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7
10			[ITD] Nord-est		13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8
11		[7] diploma	[ITC] Nord-ovest		13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7
12			[ITD] Nord-est		13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8
13	[M] maschio	[99] totale	[ITC] Nord-ovest		13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7
14			[ITD] Nord-est		13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8
15			[ITC] Nord-ovest		13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7	13,7	14,7	15,7	16,7
16			[ITD] Nord-est		13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8	13,8	14,8	15,8	16,8
17																				
18																				

Figure 41: same table as the one in Figure 6 with the dimensions of our interest highlighted

The first aspect we notice is that the dimension on the header obviously extends horizontally, while the other one does vertically. Hence, if we want to tell the application to extract either, we need to tell it what the direction of the code list is.

The second aspect is that in the case of the dimension on the header, values get repeated over and over. For this reason, we would need to identify the part of the text referring to the code and do some kind of SQL-like *distinct* operation on such values.

To start a code list extraction session, we need to either click the “New dimension window” entry in the File menu or the related button on the application bar. A new dimension window will be opened.


Excel2Csv - [Dimension Csv modeling #1]

File Help

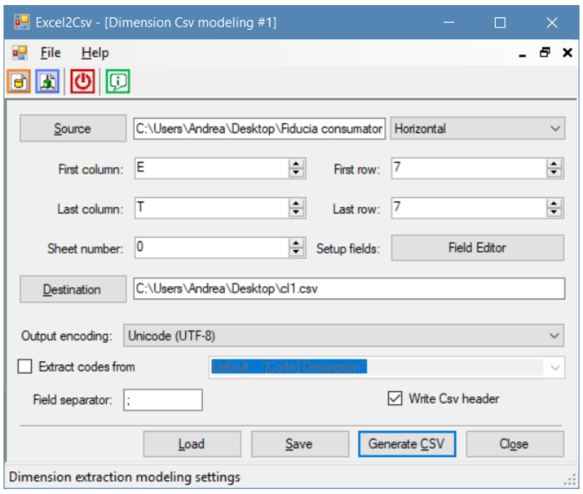
Source: (empty) Horizontal
First column: A First row: 1
Last column: A Last row: 1
Sheet number: 0 Setup fields: Field Editor
Destination: (empty)
Output encoding: Unicode (UTF-8)
☐ Extract codes from Default - "[Code] Description"
Field separator: ; ☐ Write Csv header

Load Save Generate CSV Close

Dimension extraction modeling settings

Figure 42: the code list extraction window

In case we want to extract the code list for the dimension in the header, we need to set up the main interface as follows:


Excel2Csv - [Dimension Csv modeling #1]

File Help

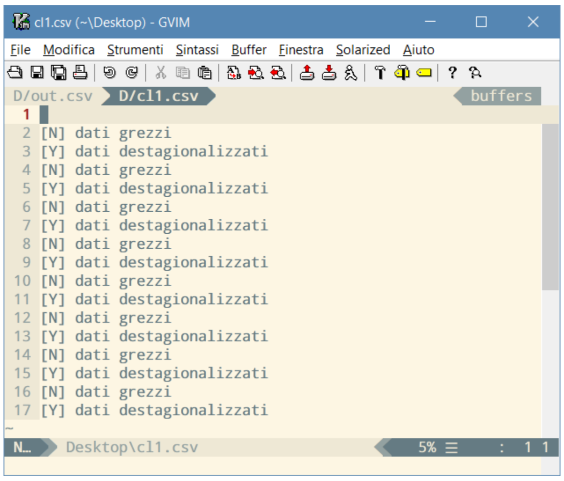
Source: C:\Users\Andrea\Desktop\Fiducia consumator Horizontal
First column: E First row: 7
Last column: T Last row: 7
Sheet number: 0 Setup fields: Field Editor
Destination: C:\Users\Andrea\Desktop\cl1.csv
Output encoding: Unicode (UTF-8)
☐ Extract codes from Default - "[Code] Description"
Field separator: ; ☑ Write Csv header

Load Save Generate CSV Close

Dimension extraction modeling settings

Figure 43: the basic settings for extracting the code list at the header

Now if we click the “Generate CSV” button, we should obtain a csv containing 16 records like the one in the following picture. Notice the first row is empty because we included a header that still wasn’t defined.


cl1.csv (~\Desktop) - GVIM
File Modifica Strumenti Sintassi Buffer Finestra Solarized Aiuto
D/out.csv
D/cl1.csv
buffers
N...
Desktop\cl1.csv
5%
1 1

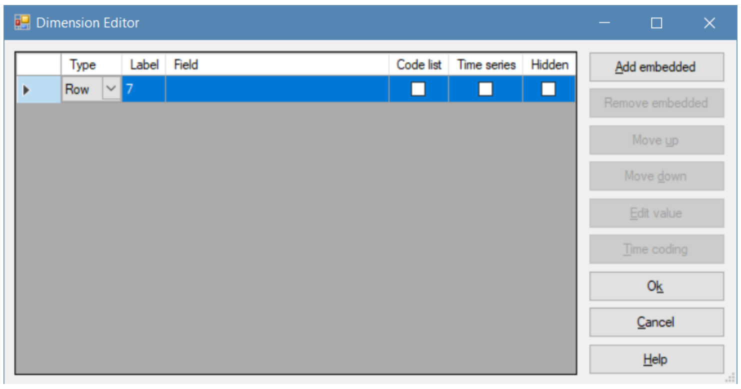
```

[N] dati grezzi
[Y] dati destagionalizzati
[N] dati grezzi
[Y] dati destagionalizzati
[N] dati grezzi
[Y] dati destagionalizzati
[N] dati grezzi
[Y] dati destagionalizzati
[N] dati grezzi
[Y] dati destagionalizzati
[N] dati grezzi
[Y] dati destagionalizzati
[N] dati grezzi
[Y] dati destagionalizzati
[N] dati grezzi
[Y] dati destagionalizzati
```

Figure 44: the initial output code list

The reason why we got so many repeated rows is that the application cannot be sure whether the list it read really contains codes, hence it still cannot remove duplicate records. We're going to do this now.

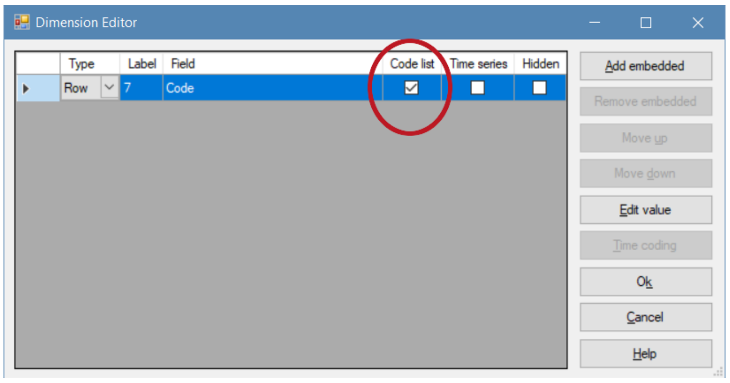
Let's click the "Field editor" button now. We should get only a column as follows:



Type	Label	Field	Code list	Time series	Hidden
Row	7		☐	☐	☐

Figure 45: the dimension editor as it appears in our example

Had we had several columns containing descriptions in many languages, we could have obviously included them by modifying the table boundaries accordingly.

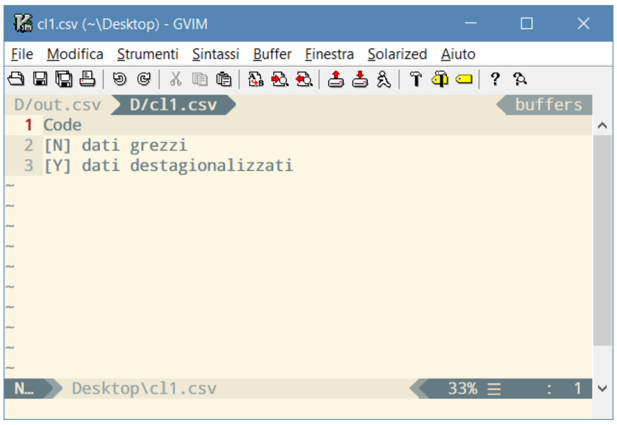


Type	Label	Field	Code list	Time series	Hidden
Row	7	Code	✓	☐	☐

Figure 46: activating the Code list check box

Anyway, let's give our column a name and let's mark it as a code list by clicking the related check box under the "Code list" column. Then let's accept our settings and generate our Csv.

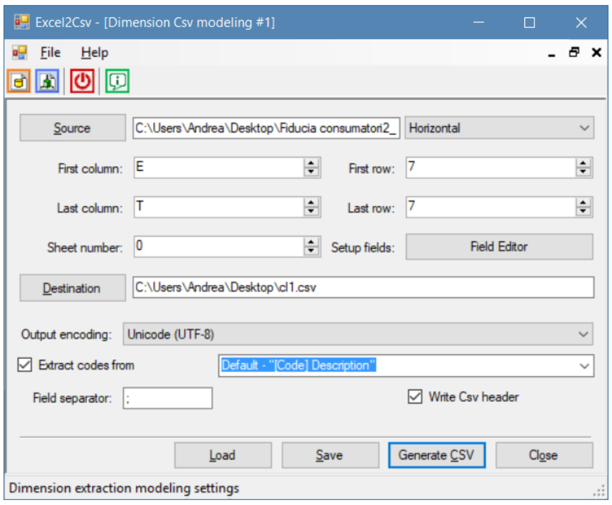
Notice that you can have only one field marked as a code list, although this is not evident in this example. In general, checking such a box for a given field removes the same mark from all other fields.


cl1.csv (~\Desktop) - GVIM
File Modifica Strumenti Sintassi Buffer Finestra Solarized Aiuto
D/out.csv
D/cl1.csv
buffers
N...
Desktop\cl1.csv
33%
1

```
Code
[N] dati grezzi
[Y] dati destagionalizzati
```

Figure 40: the result after setting the column as a code list

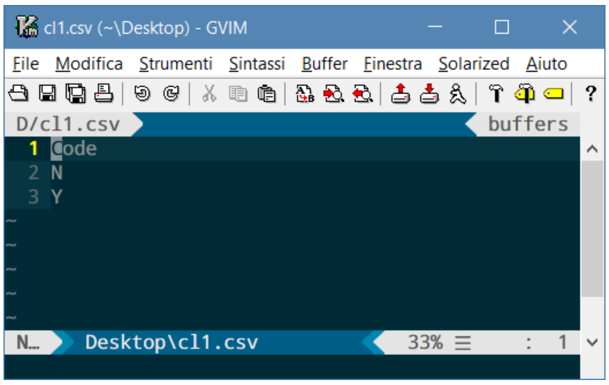
Next step is to separate the codes and the descriptions in two distinct fields. The way we can accomplish this task is very similar to what we already did on data. Since the expressions we have on the sheet are already in the "[Code] description" format, we just need to click on the "Extract codes from…" check box on the main window.


Excel2Csv - [Dimension Csv modeling #1]
File Help
Source
Destination
Field Editor
Load
Save
Generate CSV
Close
Dimension extraction modeling settings

Source	C:\Users\Andrea\Desktop\Fiducia consumatori2_	Horizontal	
First column:	E	First row:	7
Last column:	T	Last row:	7
Sheet number:	0	Setup fields:	Field Editor
Destination	C:\Users\Andrea\Desktop\cl1.csv		
Output encoding:	Unicode (UTF-8)		
☑ Extract codes from	Default - "[Code] Description"		
Field separator:	;	☑ Write Csv header	

Figure 47: enabling the code and description extraction

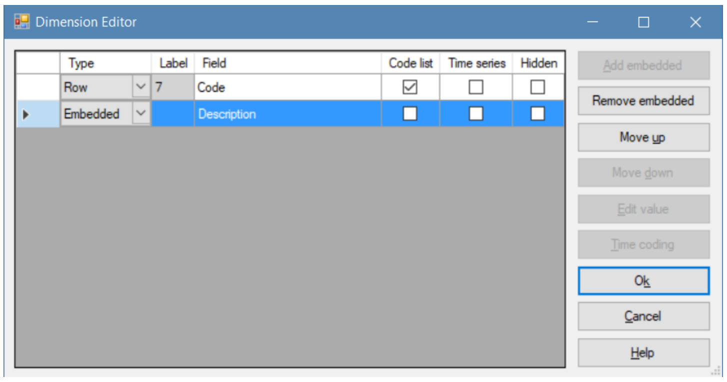
Let's click the "Generate CSV" button now and see what happens.


cl1.csv (~\Desktop) - GVIM
File Modifica Strumenti Sintassi Buffer Finestra Solarized Aiuto
D/cl1.csv
buffers
N...
Desktop\cl1.csv
33%
1

```
Code
N
Y
```

Figure 48: codes only

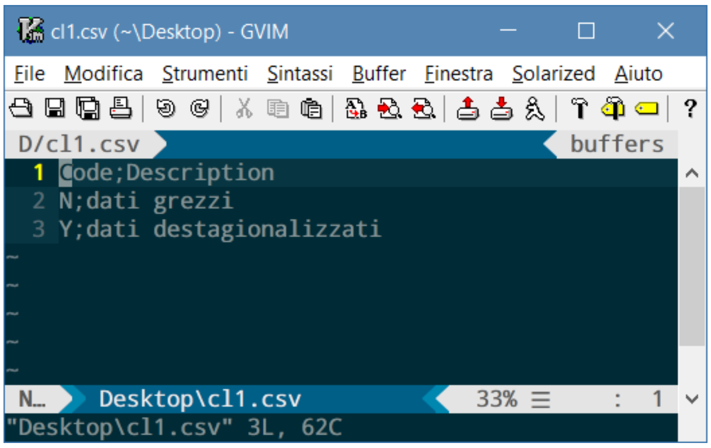
What we get as a result is a simple list of codes without descriptions. To insert the descriptions back in the file, what we need to do is open the dimension editor again and click on “Add embedded”, so that the embedded description can have its own separate Csv field. Let’s call this field “Description” and move it at the bottom of the list.



Type	Label	Field	Code list	Time series	Hidden
Row	7	Code	☑	☐	☐
Embedded		Description	☐	☐	☐

Figure 49: adding an embedded description in its own field

Now we can accept the changes and newly generate the output csv, obtaining the expected result.



```
Code;Description
N;dati grezzi
Y;dati destagionalizzati
"Desktop\cl1.csv" 3L, 62C
```

Figure 50: the complete output

The code list extraction mode obviously also lets you process several columns/rows so that you can consider the latter as optional description fields, maybe intended to have multiple description, one for each intended language.

Final observations

If we want to do some additional tuning for our output file to be, for instance, .Stat compatible, then the default settings for both separators would most likely have to be overridden. However, the software was written with the purpose to keep it as generic as possible. In case you notice the software is not generic enough for your purposes, feature requests are most welcome.

Still about compatibility, please consider that if you must use the application on spreadsheets written in right-to-left languages, then most likely you need to keep the table side axis at the rightmost side of the sheet, for everything is flipped vertically. Also, considering that the output file would have to contain strings written in opposite directions, the beginning and the end of RTL substrings would have to be marked by special bytes, that flip the reading direction throughout the substring they delimit. For this reason, a generic data loader might need to be adapted to ignore such bytes, if so needed.