Project management in DIGRAM 4.09

Svend Kreiner

November 2019

Table of contents

1 Introduction

A DIGRAM project consists of three files with raw, recoded and compressed data and a series of files with information on variables, categories, graphical models and other aspects relating to the statistical analysis in DIGRAM.

It is a well-known fact that it is difficult to transfer data from a general statistical program like SAS, STATA or SPSS to a stand-alone program like SCD/DIGRAM. There are at least two reasons for this. The first is that we have to interact with the stand-alone program in ways that are very different from the way we work with the standard program. The second – and probably worst – is that the stand-alone program requires data in a way which is very different from the way it is organized and coded in the standard program. The problems with reorganizing data for the stand-alone program is in many cases the main reason why we abstain from using stand-alone programs even in cases where it is obvious that the facilities of the standard programs are inferior to what is offered by the stand-alone program. It simply takes too long to (re)learn exactly how the stand-alone program wants its data served when we want to use it because it offers facilities that we cannot find in the standard program.

SCD/DIGRAM is neither better nor worse than other stand-alone programs, but I have tried to make it as easy as I could think of in the following ways:

1) All DIGRAM's project files are simple text files. It is therefore easy to read the contents of the project files using any text editor, if you need to remind yourself about the data structure of DIGRAM projects. SCD itself has a text editor you may use for that purpose. It is extremely simple, but it is there.

2) Data is organized as text files. For this reason, it is also relatively easy to redefine the contents of DIGRAM project files. The SCD text editor will not let you change the definitions of a project while you are working on the project in SCD, but a number of other facilities for that purpose are available in DIGRAM, and you will be able to see and check what has happened.

3) What is difficult – even with ordinary text editors – is to define the project variables from scratch, because the formats of the project files defining variables and categories may not feel natural to first-time users of DIGRAM. To make things as painless as possible, I have therefore included an import procedure that reads data from standard programs and where the only thing that you have to be with is the selection of the variables that you want to include in the project. DIGRAM assumes 1) that all variables either are categorical or has to be categorized on their way into DIGRAM, and 2) that the variables are organized in a number of recursive blocks referring to causal and/or temporal structure. It will also be up to you to provide this information, but apart from that, the import procedure will take care of

everything else to make sure that all the project files are organized and formatted in the right way.

4) Instead of starting from scratch by going back to the data in your standard program, you may also use SCD/DIGRAM to generate new DIGRAM projects as revised versions of already existing projects in several ways.

5) And finally, you may use SCD to export data from DIGRAM projects to a number of other stand-alone programs in order to ease the pains of having to relearn how these programs want their data, and having to find a way to generate the file or files for the programs.

The SCD user guide from 2003 described some of these facilities. Since then, several project management facilities have been added to SCD. The purpose of this note is to bring you up-to-date on these.

Chapter 2 describes the project files. Sections 2.3 and 2.4 are perhaps the most important parts to take a look at, because they provide information on how the categories of the variables are defined, and because you at some point may want to change the definition of the variables. Apart from that, I suggest that you skip Section 2 if you are a new user, because it provides information that we rarely need to look at.

Chapter 3 describes what happens when you start DIGRAM. Compared to the 2003 version, the most important additions are that DIGRAM automatically opens the project that you worked on the last time you used DIGRAM and that it describes what you need to do if you want to change the default options of DIGRAM.

Chapter 4 describes how to import data from other programs, so this is probably **the first Section you need to look at if you are a new or inexperienced user**.

Chapters 5 and 6 describe how to change the project variables.

Chapter 7 provides a little information on the so-called project graph, a Markov graph defining a chain graph model for the variables.

Chapters 8, 9, and 10 describes how to create new DIGRAM projects based on the same data and information as the current project.

Chapter 11 finally provides a little information on how to export data from DIGRAM to other programs.

2 Project files

We distinguish between four types of project files:

1) Files that SCD requires in order to define a DIGRAM project.
2) Two files with recoded data that SCD creates the first time a project is opened.
3) Optional project management files created by the user either before the analysis starts or during the analysis of the data.
4) Other files created by the user.

DIGRAM projects are characterized by a specific project name that appears as the first part of six project files[1]. DIGRAM need the first three files to initialize and creates the next three files during initialization.

Table 2.1 gives a complete list of the project files.

Table 2.1. Overview of files associated with DIGRAM

	File type	Comments
Required project files	DAT	The file with raw data
	DEF	Basic project definitions
	VAR	Variable definitions
Project files created during initialization	CAT	Category labels
	SYS	The recoded data
	TAB	Tabulated data
Optional Project files	GRF	The project graph
	CMD	Commands
	INI	Commands for initialization
	GAM	A matrix with partial γ coefficients
	BEL*	A belief matrix for guided model search
	TXT*	A file with information on variables
	EXA*	A file with a table
Other project files	REP*	Report files

*: These files are not described in this version of the note on project management

The user guide from 2003 describes the DAT, DEF, VAR, CAT, SYS, TAB, and GRF files on pages 20-32. The latest version of DIGRAM is, of course, compatible with previous versions. I therefore suggest that you read these pages before you continue, even though some of it is repeated on the following pages.

[1] The exception is the DAT file with data that do not need to have the same name as the project.

2.1 The DAT file

The default name of the file with data is “Projectname.DAT”. The DAT file contains a data matrix with variables in columns and persons in rows. The format is free with variables delimited by spaces. Decimal numbers may be given as European numbers with commas (e.g. 17,3) or as American numbers with dots (17.3).

The DAT file may contain more variables and more persons than we want to include in the DIGRAM project. The information on which persons and which variables to include in the DIGRAM project must appear on the DEF and VAR files, respectively.

2.2 The DEF file

The “Projectname.DEF” file defines the data matrix in the DAT file in terms of

the number of variables/columns per case (not more than 1600 per case),
two types of filters or conditions for selection of cases,
the name of the DAT matrix if it differs from the default name.

The first record of the DEF file must contain the number of variables in the dataset (not the number of variables you are going to include in the DIGRAM project).

Subsequent records should contain filter conditions (one per record) given by

a variable number,
a minimum value
a maximum value.

There is no limit to the number of filters that you may include on the DEF file.

Only cases, for which all values of filter variables belong to the intervals defined by the corresponding minimum and maximum values (both included), will be used in the analysis.

Following the type of filter described above you may also include restrictions on the number of permitted column values. To do this you have to include a SELECT command followed by a list of columns with associated values:

SELECT
COL1 <integer values>
COL2 < integer values>

A person is only included in the project if the value of the column variable is equal to one of the integer values associated with the column.

In some cases the two sets of criteria define the same restrictions. “17 2 2”, for instance, is the same as SELECT followed by “17 2”, just as “17 1 3” is the same as SELECT followed by “17 1 2 3” if the variable in column 17 only takes integer values. The main reason for including the second type of restrictions is to be able to select person for a project if values on a variable takes values from an interrupted sequence of values. SELECT followed by “17 1 3” excludes persons with Column variable 17 = 2.

Finally you may change the name of the DAT file associated with the program you must enter two lines on the DEF file:

```
FILES
D <DAT file name>
```

The first line tells the program that information on files will follow on the next line, while the “D” on the next line indicates that the following name is the name of the DAT file.

The DEF file of the UKU project contains nothing but the number of columns in the data file (5). If we had only been interested in a project containing cases without side effects prior to treatment, and if the data file was not called UKU.DAT, but CASES.DAT then the contents of the DEF file should have been as shown in Figure 2.1.

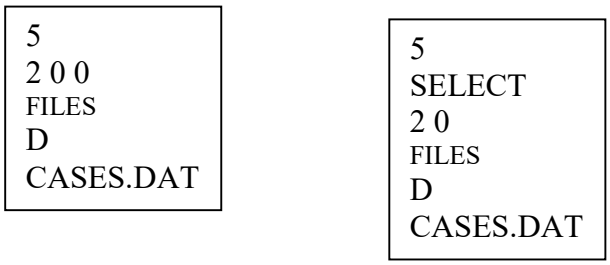


Figure 2.1. The contents of hypothetical DEF files for a UKU project of cases with no side effects prior to treatment and with data from a file called CASES.DAT

2.3 The VAR file

The VAR file contains information on the project variables. Note that the project variables do not have to appear in the same order in the project as in the DAT file. The information on the VAR file should be as follows:

1. The number of project variables (no more than 59 in version 3.34 of the program[2]).

[2] Use the “SHOW L” command if you want to check the limitations in your version of DIGRAM.

2. Definition of project variables. To define a project variable the following information is required on two separate lines:

   A ***variable label*** - a single letter, which may be used for later reference to the variable during the analysis.

   A ***column number*** referring to the data matrix from which the variable is selected.

   The ***number of categories*** of the project variable (no more than 57 in the version 2.19).

   The ***variable type*** (2:nominal/3:ordinal categories).

   Minimum and maximum values of the original variable and one or more cut points defining the categories of the project variables. Cut points always define intervals including upper category boundaries.

3. The number of recursive blocks.

4. Cut points partitioning the project variables into the recursive blocks.

5. Up to 10 lines of Comments. (Optional).

6. Variable names up to 8 characters per variable (optional, but recommended).

Comments and variable names are optional and interchangeable. Otherwise the information should be given in exactly the order presented above.

A couple of conventions and rules must be followed when the VAR file is written.

The ***number of project variables*** must appear on a single record at the top of the file.

The information on project variables should appear *space-delimited* on two records per variable:

Record 1: <Label> <Column number> <Number of categories> <Variable type>
Record 2: <Minimum> <Cut-point(1)>...<Last cut-point> … <Maximum>

Standard variable types are: 2 = nominal categories and 3 = ordinal categories

The categories defined by the cut-points are as follows:

Category 1:	minimum	≤	values	≤ cut-point(1)
Category 2:	cut-point(1)	<	values	≤ cut-point(2)
	etc. etc. until			
The Last category:	The last cut-point	<	values	≤ Maximum

All values less than minimum or greater than maximum will be regarded as missing values.

The ***number of recursive blocks*** should appear on a separate record.

Variables must be defined in the correct order with the ultimate response variables first and explanatory variables later. This means that recursive blocks may be reference to recursive cut-points indicating the last variables in the recursive blocks. The recursive cut-point may appear on separate records or on one record separated by blanks. If the last recursive cut-point is less than the number of project variables, the remaining variables will always be treated as purely explanatory variables.

The final part of the VAR file contains ***comments*** and ***variable names***.

Variable names must be initiated by a record containing the word "VARIABLES" (or at least the first three characters), followed by records containing variable labels and names separated by blanks.

Only the first 8 characters of the variable names will actually be used, but you may of course include as many as you like on the VAR file.

A record containing "COMMENTS" will be taken to indicate that the contents of following records should be treated as comments. Only the first ten lines of comments will be used. Immediately after having read the last recursive cut point, the program will be in COMMENT mode and will remain so until "VARIABLES" appear.

Figure 2.2 shows the contents of the UKU.VAR file. The variables defined in Figure 2.2 are as follows.

The project uses all the variables in the DAT file, but the order of the project variables is not however the same as the order columns in the data matrix.

The Label of the first project variable is D:

D 5 3 3
0 0 1 3

This variable is the fifth variable in the data matrix. The variable has three ordinal categories defined by collapsing of categories 2 and 3 of the original variable.

The categories of D are given by

Minimum = 0
Cut-points = 0 and 1
Maximum = 3.

The three categories of D are therefore defined by the following values of variable 5 in the original data matrix:

Category 1 : 0 (not present)

Category 2 : 1 (mild degree)
Category 3 : 2-3 (moderate to strong degree)

The name of variable D may be found in the list of variable names at the bottom of the file. Variable D is "STATUS 3".

A total of five project variables, D, C, B, A, T are defined in this way. The column numbers of the original data matrix are 5, 4, 3, 2, and 1.

```
5
D 5 3 3
0 0 1 3
C 4 3 3
0 0 1 3
B 3 3 3
0 0 1 3
A 2 3 3
0 0 1 3
T 1 2 2
1 1 2
4
1 2 3 5
Ratings of side effects in a comparative drug treatment.
Side effects are rated pre-treatment (STATUS 0), one, two
and three weeks after treatment (STATUS 1-3).
Ratings were scored:
0 : not present      1 : mild degree
2 : moderate degree  3 : strong degree
The last two categories have been collapsed for the
analysis by DIGRAM.
Treatment was by one of two drugs.
VARIABLES
T TREATMENT
A STATUS 0
B STATUS 1
C STATUS 2
D STATUS 3
```

Figure 2.2 The contents of the UKU.VAR file defining five project variables. Comments have been added after the definition of the recursive structure. The lines following the "VARIABLES" record define the variable names.

The recursive structure is defined immediately after the variables in terms of the reference numbers:

```
4
1 2 3 5
```

The project has four recursive levels or blocks defined by four recursive cut-points in the sequence of variables.

Level 1 : D Level 2 : C Level 3 : B Level 4 : A and T.

The recursive structure reflects underlying temporal structure of the variables:

$$D \leftarrow C \leftarrow B \leftarrow AT$$

2.4 The CAT file

"Projectname.CAT" contains the names of the categories of the project variables. You may create this file before you open the project for the first time. If you have not done so, DIGRAM will create it and let you fill in the information on the categories[3] if you want to do it. This only happens when you open the project for the first time. After that, you have to invoke the "CATEGORIES" command, right click on the variables in the project graph, or use your favourite text editor if you want to change the names of the categories.

The CAT file contains category names for all project variables. CAT files are optional, but recommended. They may be created and modified before you start DIGRAM or at any time during an analysis.

Each record of a CAT file must contain a label of a project variable a value indicating a category and the corresponding category name. The category names on the file may be as long as you wish, but DIGRAM will only use the first eight characters.

```
D 1 None
D 2 Mild
D 3 mod+strong
C 1 None
C 2 Mild
C 3 mod+strong
B 1 None
B 2 Mild
B 3 mod+strong
A 1 None
A 2 Mild
A 3 mod+strong
T 1 Drug1
T 2 Drug2
```

Figure 2.3 The contents of the UKU.CAT file

2.5 SYS and TAB files

SCD creates two files with recoded project data ("Projectname.SYS" and "Projectname.TAB") the first time a project is opened. The SYS file contains a data matrix with project variables recoded

[3] See Section 5 on how to fill in the information on the categories (category labels).

according to the variable definitions on the VAR file. Categories are integer coded starting with 1 corresponding to the first category. Missing values are coded 0 in the SYS file.

The Tab file contains the same information as the SYS file, but in a more compact form.

The format of the SYS and TAB files are as described in the user guide from 2003 and will therefore not be described here.

Creating SYS and TAB files can be time consuming for very large data sets. For this reason, you can now follow what goes on in the status panel below the DIGRAM main dialog form.

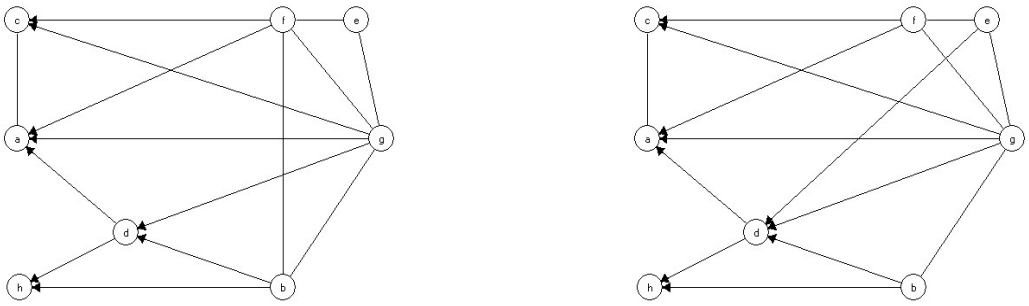
The SYS and TAB file will be recreated if you change the way the variables are categorized of if you change the recursive structure of the variables. If you, for some other reason, are concerned that there is something wrong with the files you can use the **MAKE** command to recreate the files.

2.6 The GRF files

You may save the current graphical project model on "Projectname.GRF" as described in the user guide from 2003. If it exists, the model on the GRF file will be read together with the project definitions and data when you reopen a project that you have already done some work on. If the GRF files does not exists when the project is invokes, DIGRAM suggests that you create a GRF file with an initial graphical model defined by screening of relationships among variables.

During the analysis of data, you may change the current project model so that it differs from the model you have saved on the GRF file. To compare the current project model with the model that you have saved on the GRF file you must select "Compare the current model and the model on the GRF file" on DIGRAM's Model menu as shown in Figures 2.4 and 2.5.

Figure 2.4 shows a hypothetical model on the GRF file (Figure 2.4a) and the current graph as recognized by DIGRAM (Figure 2.4b).



(a) (b)

Figure 2.4. The model on the GRF file (a) and the current model (b)

Figure 2.5 shows you where to find the menu item that you can use if you want to compare the two graphs without having to draw them.

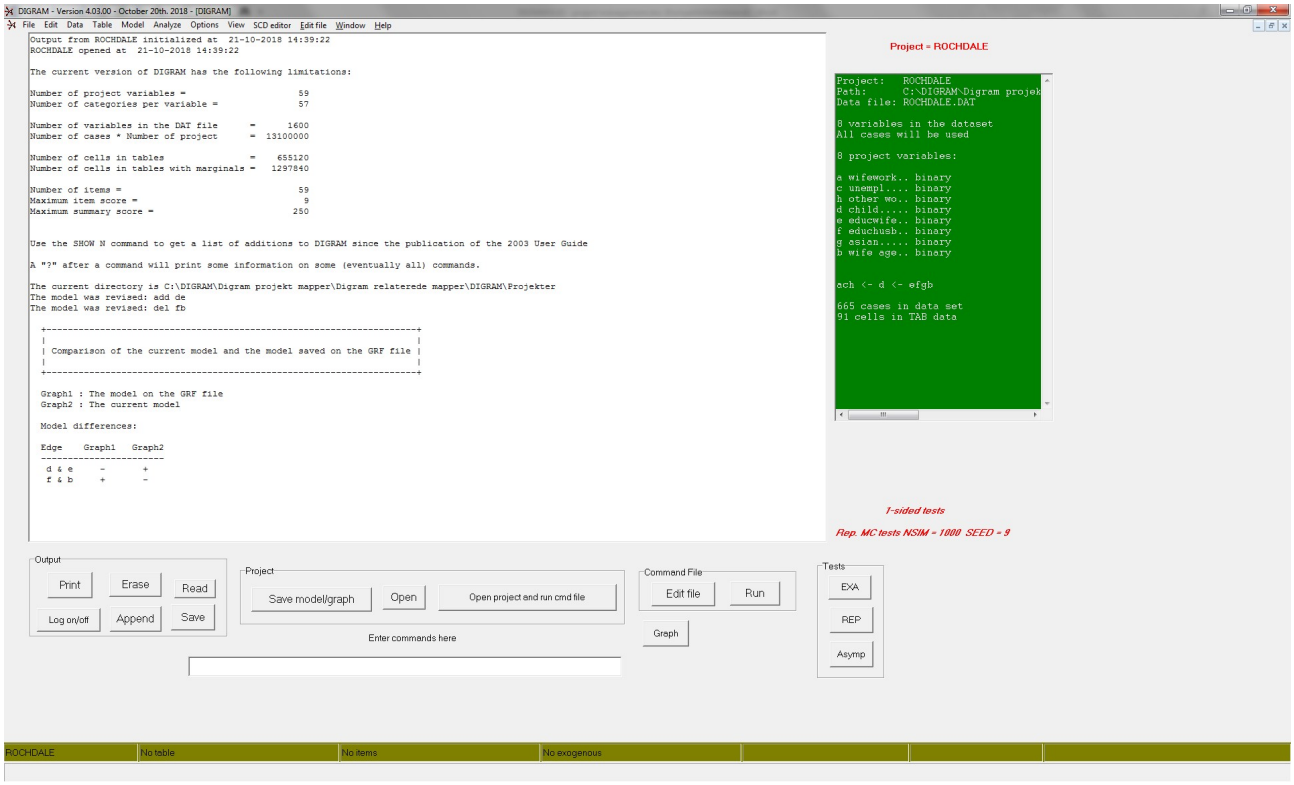


Figure 2.5. Comparing the current model to the model on the GRF file

The result of the comparison of the two models is shown in Figure 2.6, where it can be seen that the edge between b and e has been added to the model during the analysis, whereas the f-b edge was removed.

```
+-------------------------------------------------------------------------+
|                                                                         |
| Comparison of the current model and the model saved on the GRF file     |
|                                                                         |
+-------------------------------------------------------------------------+

Graph1 : The model on the GRF file
Graph2 : The current model

Model differences:

Edge     Graph1    Graph2
----------------------
 d & e      -         +
 f & b      +         -
```

Figure 2.6 Comparison of the model on the GRF file and the current model defined in DIGRAM

2.7 The GAM file

In addition to the project files described in the user guide from 2003, you may now save the current estimates of the partial gamma coefficients on the "Projectname.GAM" file.

Use the **GAM** command to calculate the gamma coefficients under the current model and then invoke **SAVE G** to save the matrix with gamma coefficients on "Projectname.GAM". A gamma matrix corresponding to the model on the GRF file in Figure 2.4b could look like this where gamma values for missing edges per definition are equal to 0.

```
   999 -0.554  0.000 -0.587  0.000  0.342 -0.737  0.000
-0.554    999  0.000  0.000  0.000 -0.631  0.524  0.000
 0.000  0.000    999 -0.530  0.000  0.000  0.000  0.740
-0.587  0.000 -0.530    999  0.000  0.000  0.674 -0.939
 0.000  0.000  0.000  0.000    999  0.490 -0.670  0.000
 0.342 -0.631  0.000  0.000  0.490    999 -0.579 -0.413
-0.737  0.524  0.000  0.674 -0.670 -0.579    999 -0.627
 0.000  0.000  0.740 -0.939  0.000 -0.413 -0.627    999
```

Figure 2.7. Contents of a GAM file corresponding to the graph in Figure 4a

If the gamma coefficients are available, DIGRAM creates the GAM file when you exit and saves the model, but the values on the GAM file will not automatically be read by DIGRAM the next time you reopen the project. To read the content of the GAM file, you must first enter a **READ G** command. Remember that the Gamma coefficients are calculated under a specific model. You will be warned if the set of partial gamma coefficients does not correspond to the current model as DIGRAM sees it when you use the **READ G** command, in which case you have to invoke the **GAM** command to get the correct estimates.

2.8 CMD and INI files

A CMD file is a file – a batch file if you like – containing a set of commands to be executed. The current version of DIGRAM recognizes two types of command files. CMD files are ordinary command files, which you can edit by clicking on the "Edit file" and/or execute and "Run" buttons in the DIGRAM main form (Figure 1). The "Projectname.INI" file is a command file that will be executed every time you open the project.

You may have several CMD files associated with a project and you may call them whatever you like. When you click the "Run" button, DIGRAM will show you all the files with a CMD extension including the "Projectname.CMD" file if it exists. If you have CMD files saved with other extensions you have to browse the project folder to find it.Finally, you can also invoke the **RUN <filename>** command to execute the commands on a CMD. The default CMD file is "Projectname.CMD". The commands on this file will be executed if you invoke the **RUN** command without a file name.

3 Opening and initializing projects

Let me briefly summarize what happens when you start DIGRAM, because it is important that you know exactly what happens when you open a DIGRAM project, and because Version 4.03 differs with regard to start behaviour from the version described in the user guide from 2003.

Starting DIGRAM

When you invoke DIGRAM, the DIGRAM main dialog form is opened directly and the previous project that you worked on will be reopened. If the project for some reason is not found or if you want to work on another project, you have to click on OPEN and find the project yourself.

When the project is open (whether or not it is the previous project, another project that you have worked on before or a completely new project) DIGRAM executes all the commands on the "Projectname.INI" file if it exists. In this way, the INI file gives you an opportunity to reset DIGRAM's default parameters in the way that suits you the best.

Initializing projects

The first things that happens, when you open a new project, is that DIGRAM creates the SYS and TAB files with recoded project data.

Next, DIGRAM creates a saturated Markov graph of a chain graph model for the complete set of project variables and saves the model on the GRF file. Before it saves the model, DIGRAM asks whether you want to screen the relationships between the variables before the model is saved. This may take a little time if the project has a large number of variables, but I recommend that you do so anyway.

Finally, DIGRAM creates a file "DIGRAM.CMD" in the folder with the program (not necessarily the project) containing name of the folder with the project and the name of project you have just opened so that DIGRAM can find the project again, the next time it is started.

Reopening projects

When you reopen an existing project, DIGRAM 1) executes the commands on the INI file, 2) reads the contents of the TAB file for easier data access, and 3) reads the contents of the GRF file to define the current project.

Working with projects

The only thing that I need to tell you here is that DIGRAM writes all output to a temporary file named "Projectname.TMP" before it is written to DIGRAM's output fields. The temporary file is disposed when you exit DIGRAM, except when DIGRAM crashes (which might happen if you do not treat it kindly). If the temporary file exists when DIGRAM is invoked, DIGRAM is supposed to

read it and copy the contents to the output field of DIGRAM main dialog form so that you can see where you were when you provoked a breakdown.

Exit from DIGRAM

DIGRAM asks you whether you want to save the output, if it believes that there is output that you have not already saved. It also asks whether you want to save the project graph if this differs from the model on the GRF file, so that the information on the project graph is up to date when you return to the project.

4 Importing data from other programs

The previous versions of DIGRAM assumed that you either prepared the DEF, VAR and DAT files yourself, used the SPSStoDIGRAM5 program to import data from SPSS or Excel to DIGRAM, or used a SAS macro to create the files for you. You can still do it in this way, but the current version of DIGRAM has an **IMPORT** command that attempts to make it a little easier to get data into DIGRAM.

Before you invoke the **IMPORT** command, you only have to do one or two things.

The *first* step is to recode the variables so that they have exactly the categories that you want to use in DIGRAM. If you, for instance, have an Age variable with range = 18-87 you have to categorize this variable either before or when you transfer Age to DIGRAM. To do this, you either have to create the categorized variable when you prepare in your data set for import and then tell DIGRAM to use this categorical version of the Age variable, or you have to tell DIGRAM how to do the categorization when you import the data. In practice, it turns out that the second option is much simpler than the first option, so this is what we recommend. Recode Age into a categorical AgeCat variable. If Agecat has three categories (e.g. 1 = 18-39, 2 = 40-59, 3 = 60-87) the only information you have to provide is that 1 is the minimum and 3 the maximum value of AgeCat.

The *second* step is to save your data as a text file with data delimited by either commas, semicolons, tabs or spaces and with the variable names in the first record. Since nothing else is needed, and since all serious statistical programs[4] (and Excel) can create csv files with data delimited by comams or semicolons, you should have no problems creating such a file.

To create a DIGRAM project from such a file you have to do the following.

1) Invoke the **IMPORT** command.
2) Find the file with data.
3) Select the variables and tell DIGRAM what the range is, whether or not you want DIGRAM to categorize the variables in the DIGRAM project, and whether the variable is nominal or ordinal.
4) Enter the name of the new DIGRAM project.
5) Check that variable definitions are valid.
6) Create the project.

If your data originated in an SPSS file, you may read the category names from an exported output file with the file info from SPSS. If you data did not come from SPSS or if you do not care to use such a file, you have to define the category names yourself as described in Section 5.

Having done all this, you only need to return to DIGRAM's main form and open the new project.

[4] In SPSS you should save data as a Tab-delimited file. In STATA you can use the "outsheet variable-list filename, c" to create a comma separated file with the variables in then variable-list.

We illustrate the import procedure with data on the physical functioning subscale of SF36. That is with the same kind of data as the PF3 project described in the guided tours through the item analysis in DIGRAM, except that data comes from a different health survey.

The data for the new project originated in an SPSS file called Sundby36.sav with a subset of variables from the health survey. The data on Sundby36.sav was subsequently saved on a comma-separated file called Sundby.csv that will be used to create a DIGRAM project called SB3[5].

In addition to the PF items of the subscale measuring physical functioning, the Sundby36 files also include information on age, gender and self-reported health (SRH).

The responses to the PF items are coded 1 = Limited a lot, 2 = Limited a little, and 3 = Not limited as in the PF3 project. Gender is 1 = Male and 2 = female and self-reported health is 1 = Very good, 2 = Good, 3 = fair, 4 = Bad, and 5 = Very bad. Age is not categorised in the Sundby36 data. The range is from 18 to 94 years.

The PF items and gender will be imported as they are found in the Sundby36 data. During import, bad and very bad health will be collapsed into one category, and age will be categorized in the same way as in the PF3 project[6] with an additional fifth category with elderly more than 69 years of age.

The import of data for the SB3 project proceeds as follows.

Step 1: Find the file with data and open the file. The variables in the data appear in the field to the left. The right hand field gives you some information on the data. Figure 4.1 on the next page shows the import dialog that is invoked by the **IMPORT** command after you have found the Sundby36.csv file.

[5] The SB3 project is included together with the PF3 project among the DIGRAM projects for item analysis.
[6] The age categories in the PF3 projects are 1 = 16-29, 2 = 30-44, 3 = 45-59, and 4 = 60-69.

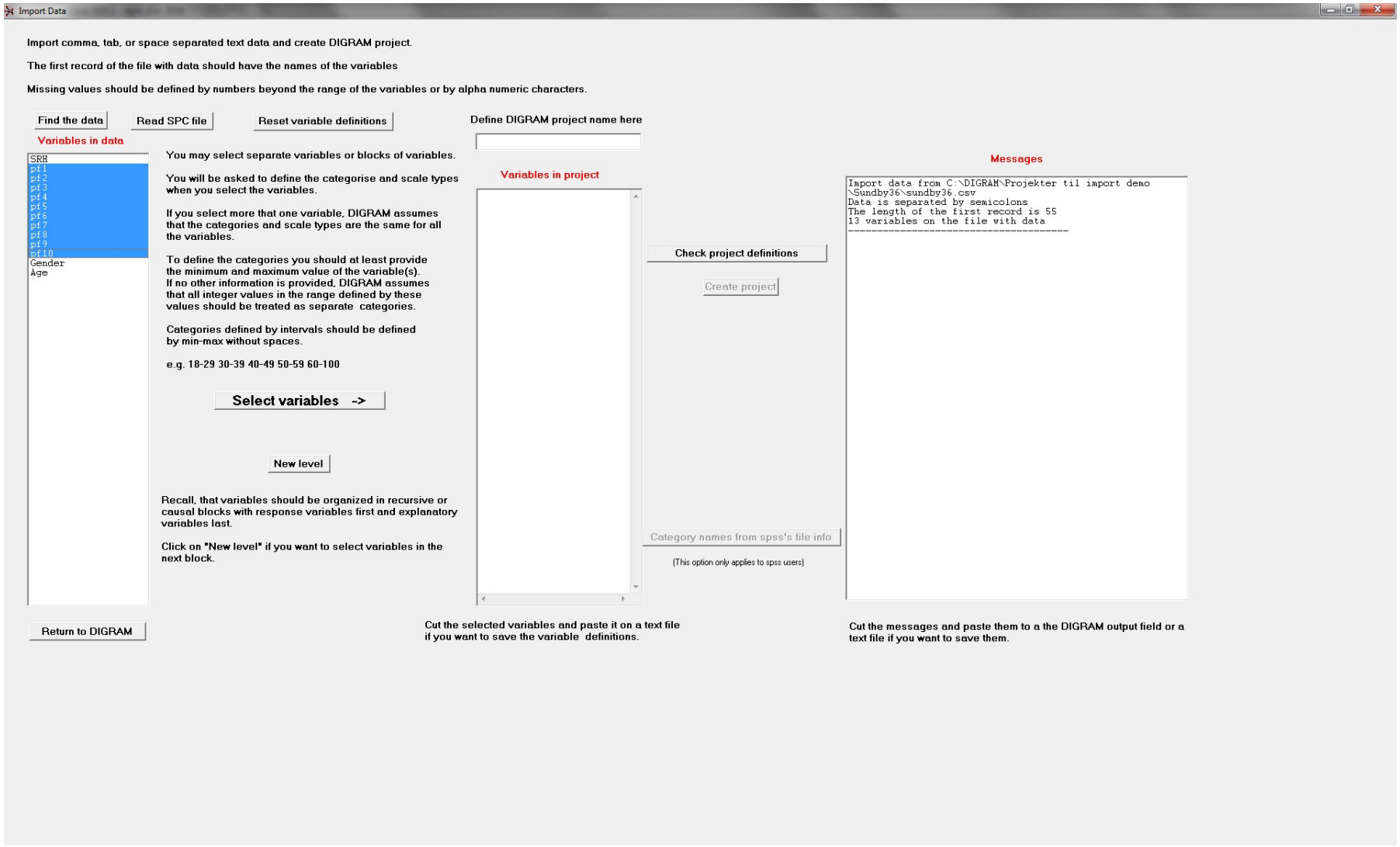


Figure 4.1. The import data dialog after the data has been found. The PF items have been selected, but not transferred to the field with project variables.

Step 2: Select the variables you want to use and click on the '->' button. Do this as many times as you need. Recall that DIGRAM assumes that there is a block recursive (maybe causal) structure in data and that the ultimate response variables have to come first and the explanatory variables last. Click on the 'New level" button if you want to define a new block.

When you select variables, DIGRAM asks how the should be categorized and whether variables are ordinal (default) or nominal. Figures 4.2 - 4.5 shows the dialogs used for this purpose. The information will apply for all the variables you have selected.

You have to define the range and the categorization before the scale type. The range and categories can be defined in three ways

By two integers (e.g. '1 3' as in Figure 4.1) defining the minimum and maximum of the variable. In such cases, DIGRAM assumes that all integervalues from minimum to maximum defines categories. This is the easiest way to define variables. It assumes that you already have recoded the variables that you want to include in the project.

By integers and integer intervals (e.g. '1 2-5', '1-2' 3 4 5', '1-2 3 4-5'). DIGRAM requires the numbers to be presented in increasing order, and will try to define these categories as consistently as possible.

By minimum, cutpoints and maximum in the same way as on DIGRAM's VAR files (see Section 2.3). This is the most difficult way and you would probably want to avoid it.

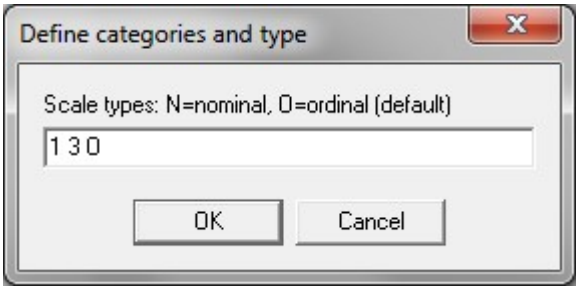


Figure 4.2. Definition of selected PF items. Three ordinal categories with codes from 1-3.

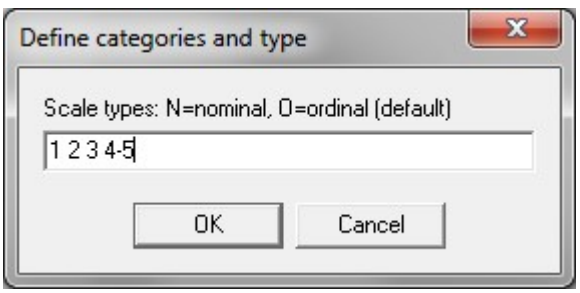


Figure 4.3. Definition of SRH. Four ordinal categories, with categories 4-5 collapsed into one category. Since ordinal categories are default, we do not need to include an O at the end.

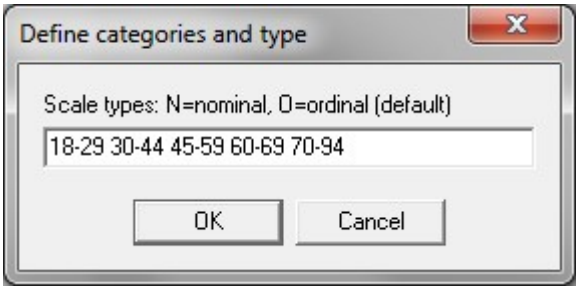


Figure 4.4. Definition of age five age categories.

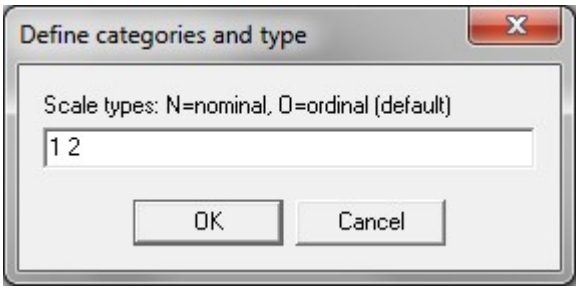


Figure 4.5. Definition of age five age categories.

DIGRAM assumed that the user provide information on the recursive (e.g. defined by causality or temporal structure) among variables. You therefore have to start with the response variables and finish with the explanatory variables. When the next variable to be entered is at a lower recursive level than theprevious variables the, you have to click in “new level” before you select the next variable.

In Figure 4.6, I have defined a project with the PF items and SRH in the first recursive block, and Age and Gender in the second.

When the variables have been selected, you have to enter the name of the new DIGRAM project and then click on “Check project definitions” to find out whether you have made any errors and whether you have selected a name of an existing DIGRAM project. If no errors have been made, the “Create project” button will be enabled. If errors were found you can edit the information in the “Variables in project” field and try again.

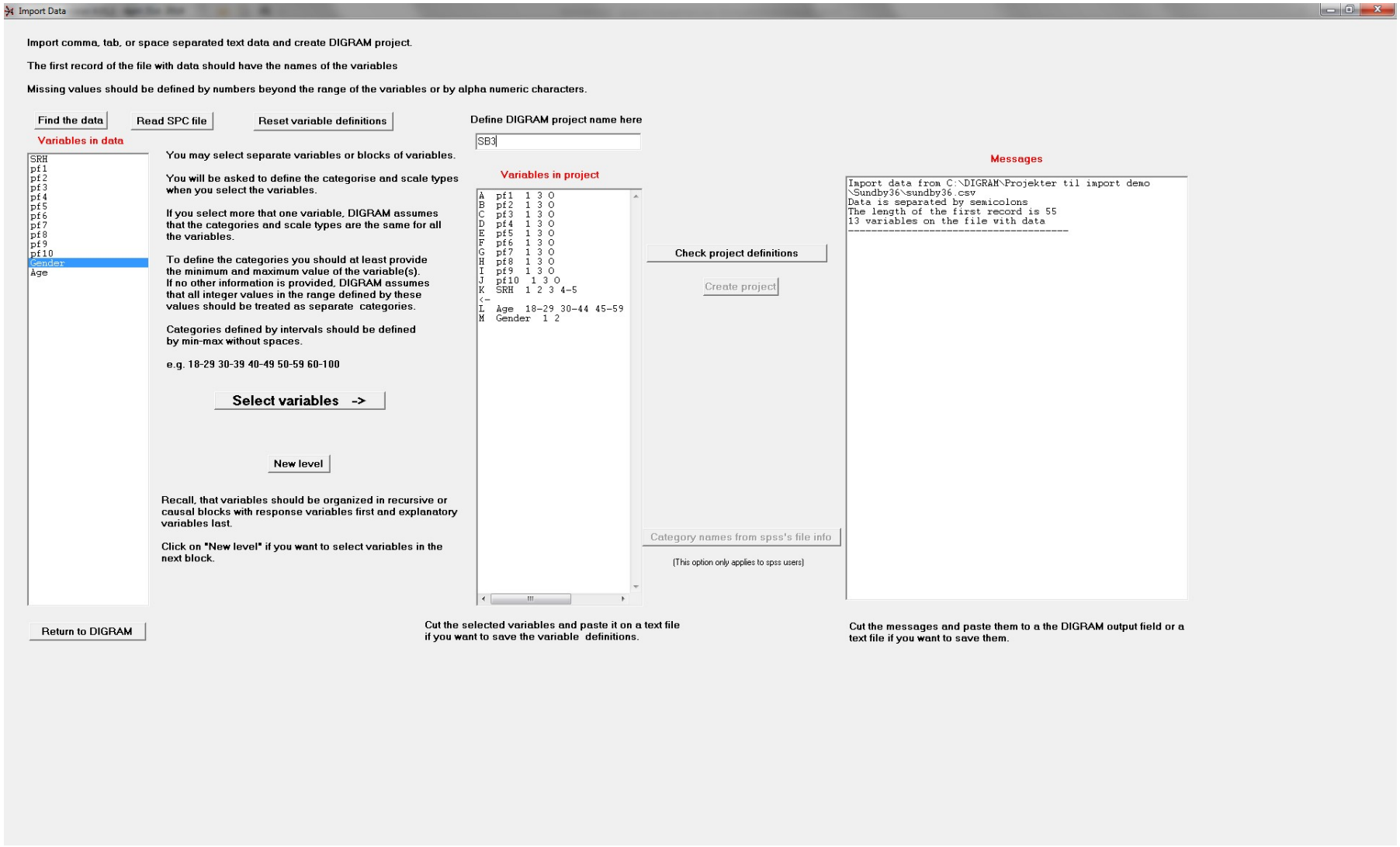


Figure 4.6 The import dialog after definition of variables and project name

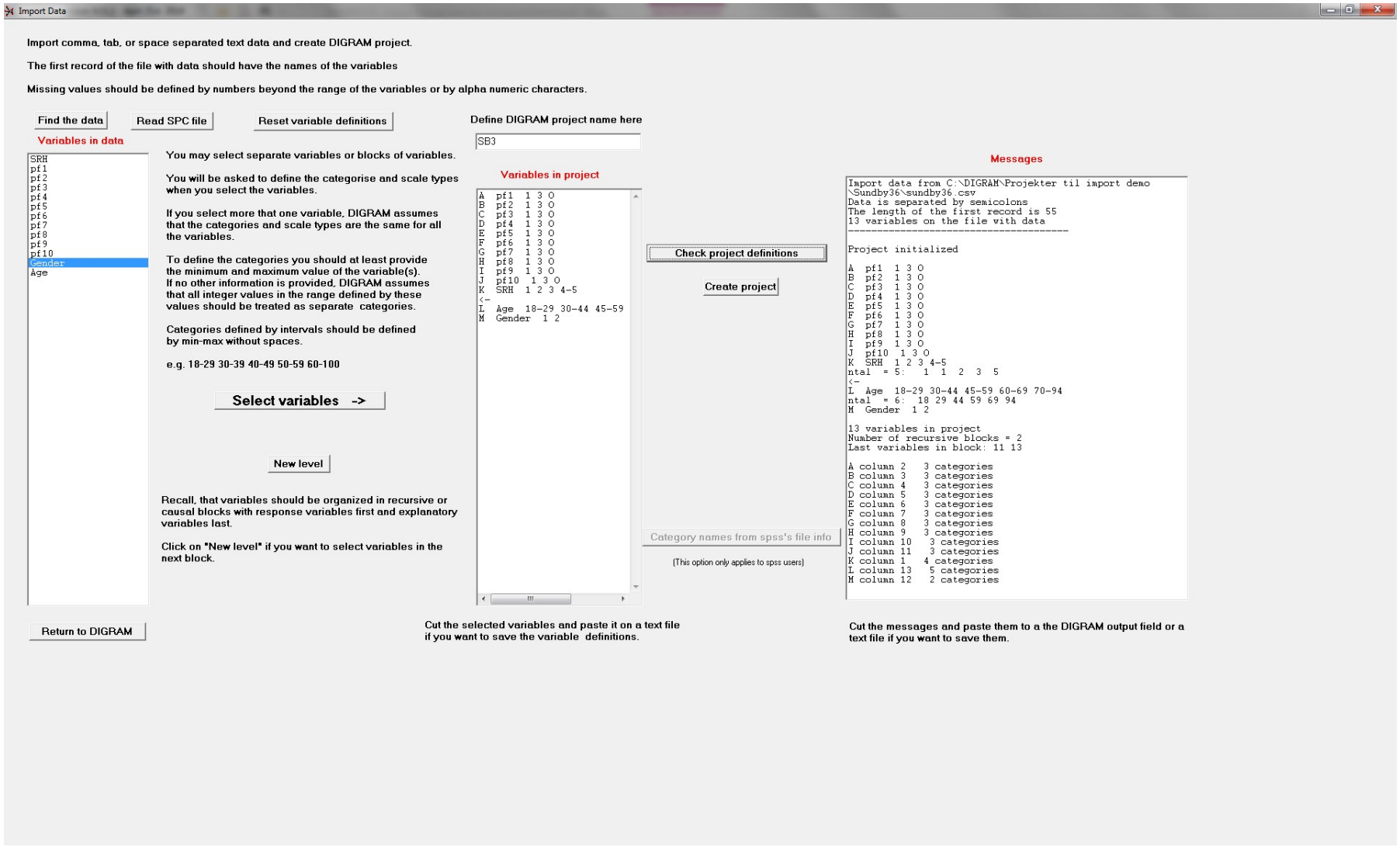


Figure 4.7 The import dialog after the check of the project definitions. The "Create project" button has been enabled because the definitions were accepted.

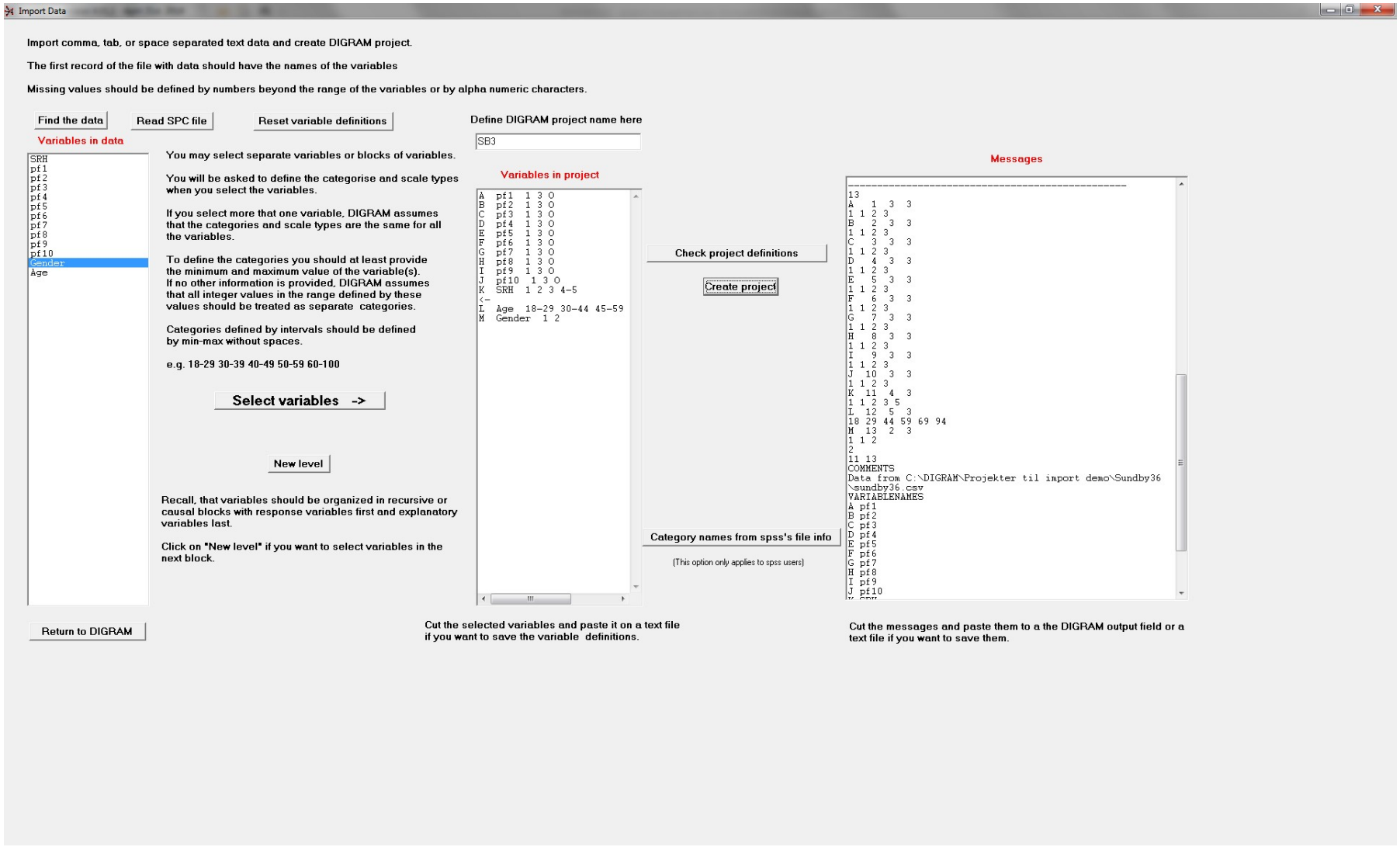


Figure 4.8 The import dialog after the project has been defined. Scroll down the messages to find the info included on the VAR file of the new project. The "Categories from spss's file info" button has been enabled. You may use this if data came from SPSS and if you have exported the File Info to a text file.

Since no problems were found, the “Create project” button was enabled. All that remains is to press this button and to return to DIGRAM unless you want to create a new project with variables from the from the same data set. If you want to do this you should click on “Reset variable definitions” and start again.

When you return to DIGRAM and open the new project, you will be asked to define and modify the categories. Figure 6.1 shows how. Note that DIGRAM has created categories defining intervals of values. All the rest you have to do yourself (see Chapter 6) unless you have a text file with the file info from spss. If such a file is available, you have to use the “Create category names from spss” before you return to DIGRAM.

And finally. Just to be sure that no errors were made, do not forget to compare the frequencies of the variables in the project with the frequencies of the variables in the original data set, once you have returned to DIGRAM.

5 Changing variable definitions

You can change the definitions of the project variables in three different ways.

1) You may change them using your preferred text editor.

2) You may use the Change Variables or Change Categories options on the Data menu in DIGRAM

3) You may change them by right-clicking on the variables in the project graph

4) Or you may keep the old DIGRAM project and instead generate a new project

The first option is not recommended, since any errors you make will not be recognized until the next time you try to access the project in DIGRAM. Changes to the variable definition may also mean that you will have to create new SYS and TAB files; problems that will not be recognized by DIGRAM, which assumes that the SYS and TAB files are correct. It is not difficult to create these files using the MAKE command, but it is up to you to do this[7].

Options 2) and 3) are clearly to be preferred, since DIGRAM checks that the changes to the variables are consistent, and automatically creates new SYS and TAB files if this is required.

[7] If in doubt, always use the MAKE command if you are unsure about the existing SYS and TAB files. No harm will be done if it wasn't necessary since the MAKE command in this case creates new copies of the existing files.

6 Revising category names

Assume that you want to change the names of the categories on the CAT file. One way to do this would be to right-click on the variables and in the project graph (see Chapter 7) where you may change the definition of single variables. Another and simpler way would be to invoke the CATEGORIES command enabling the category dialog form shown in Figure 6.1.

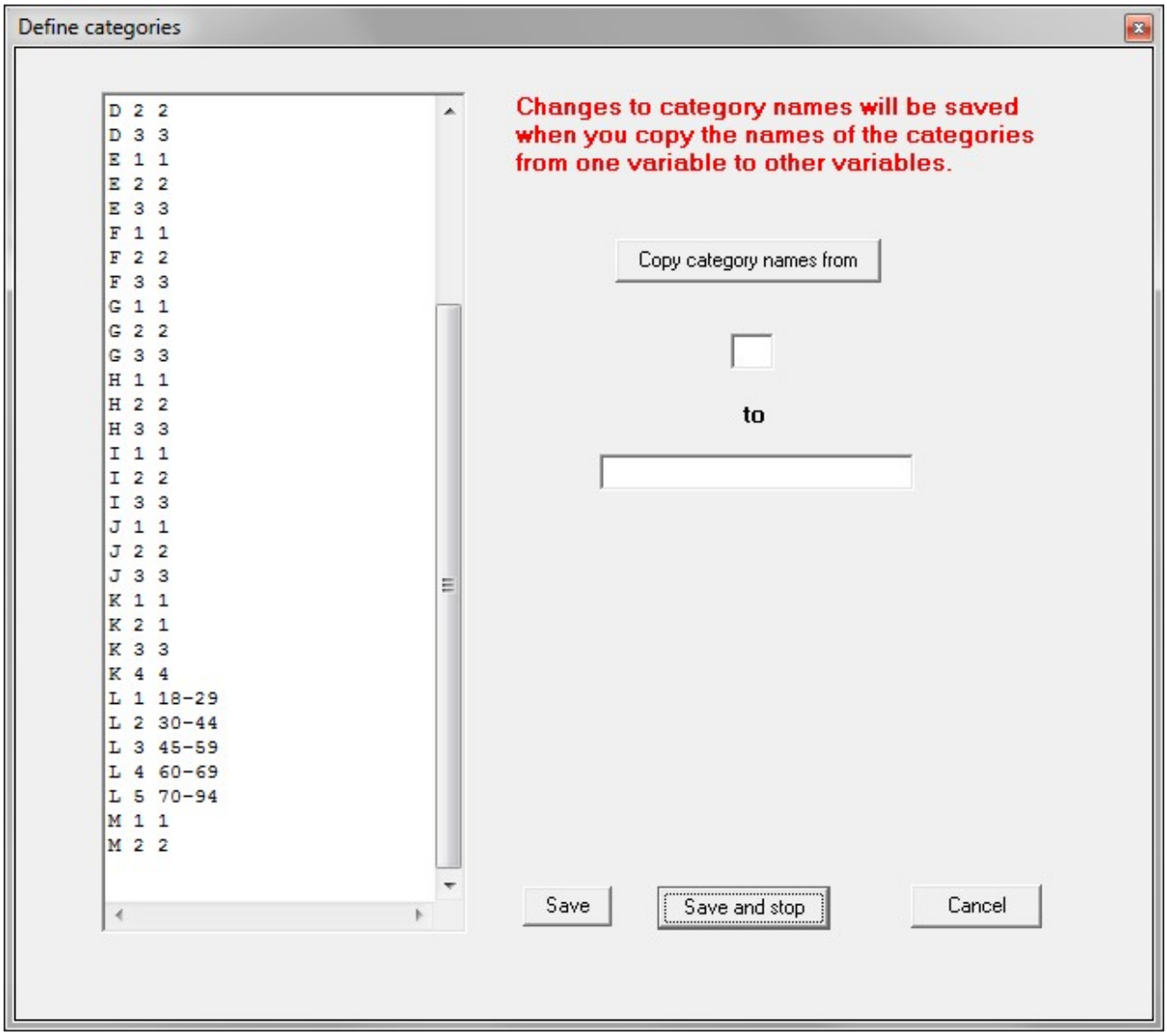


Figure 6.1 The category dialog as invoked after import of SB3

The SB3 project with the variables shown in Figure 6.1 contains ten polytomous items A-J with three response categories 1 = “A lot”, 2 = “A little”, and 3 =”No limits”, and Self reported health (K) with four response categories 1 = “very good”, 2 = “Good”, 3 = “Fair”, and 4 = “Bad”. Age L) has five categories “18-29”, 30-44”, “45-59”, “60-69”, and “70-94”. Finally Gender (M) is coded 1=“Male” and 2=“Female”. Figure 6.1 show you the categories after import to DIGRAM. The age

categories are defined during import, but the rest you have to define yourself using the catergory dialog.

To change the categories of single variables you just have to enter the nmaes in the list and click on “Save” as shown in Figure 6.2

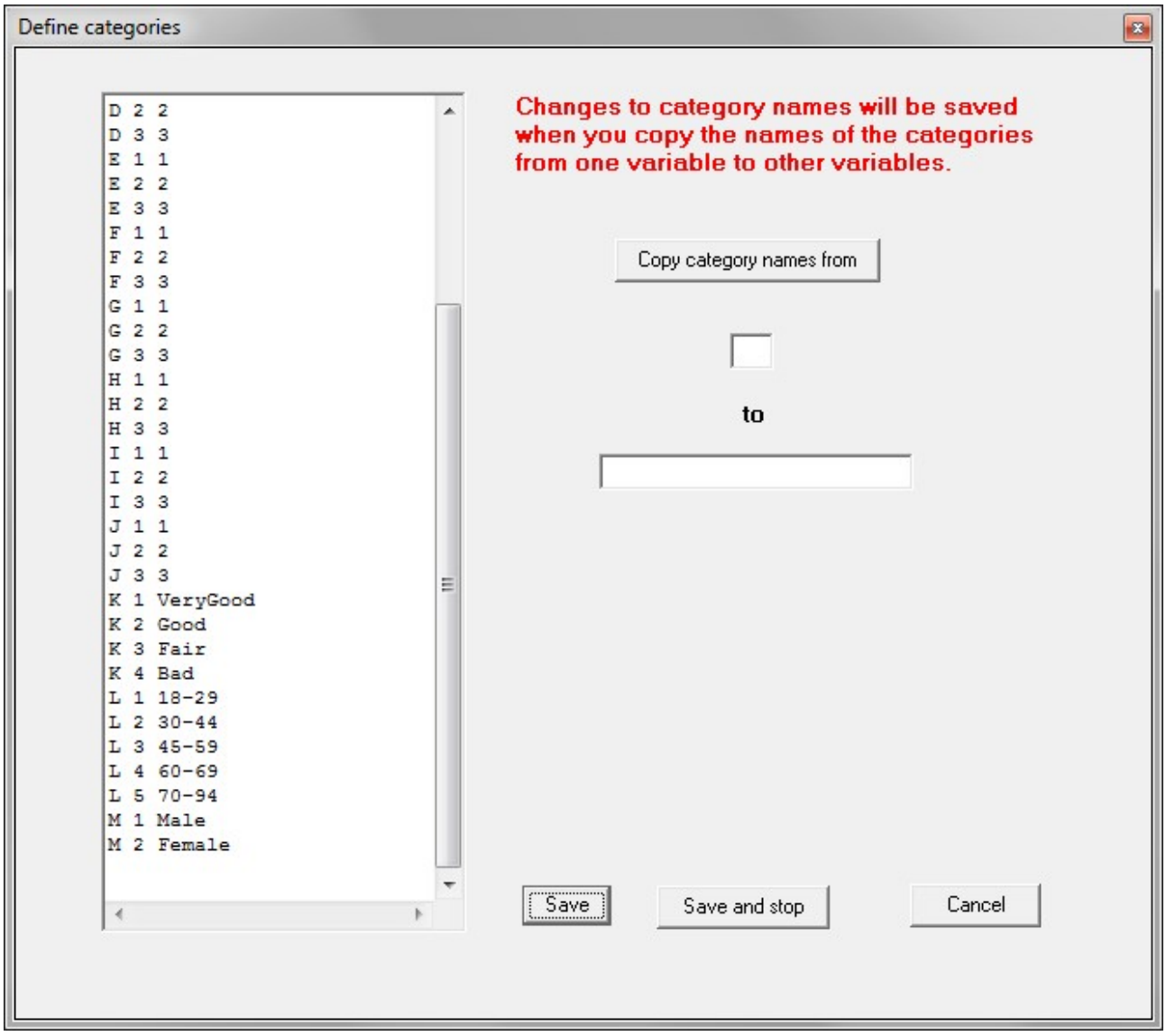


Figure 6.2 The category dialog with information on Age and gender categories.

You can do exactly the same for items A-J, but the easy way would be to do it once for item A and then to copy the information on A to the other items. Figure 6.3 shows the first step.

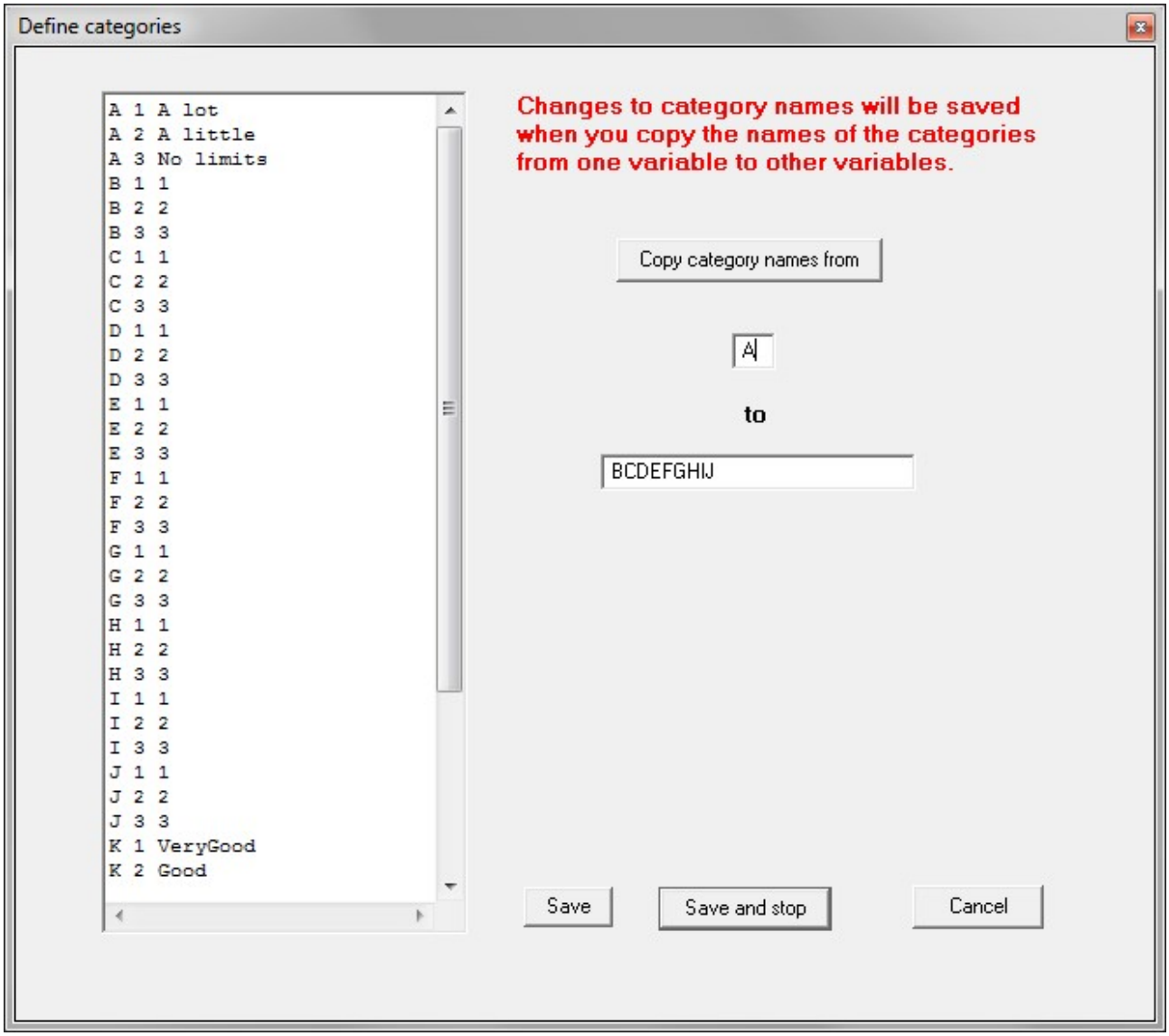


Figure 6.3 The category dialog with information on the categories of A, and the information that you want to copy the categories of A onto the categories of B-J[8].

When you click on the "Copy category names" in Figure 6.3, DIGRAM 1) saves the content of the field with categories before copying, 2) copies the category names, 3) and finally saves the content of the revised category field. Figure 6.4 shows the result.

[8] Instead of writing "BCDEFGHIJ" in Figure 6.3 you could have written "B-J". FIGRAM understands what you want.

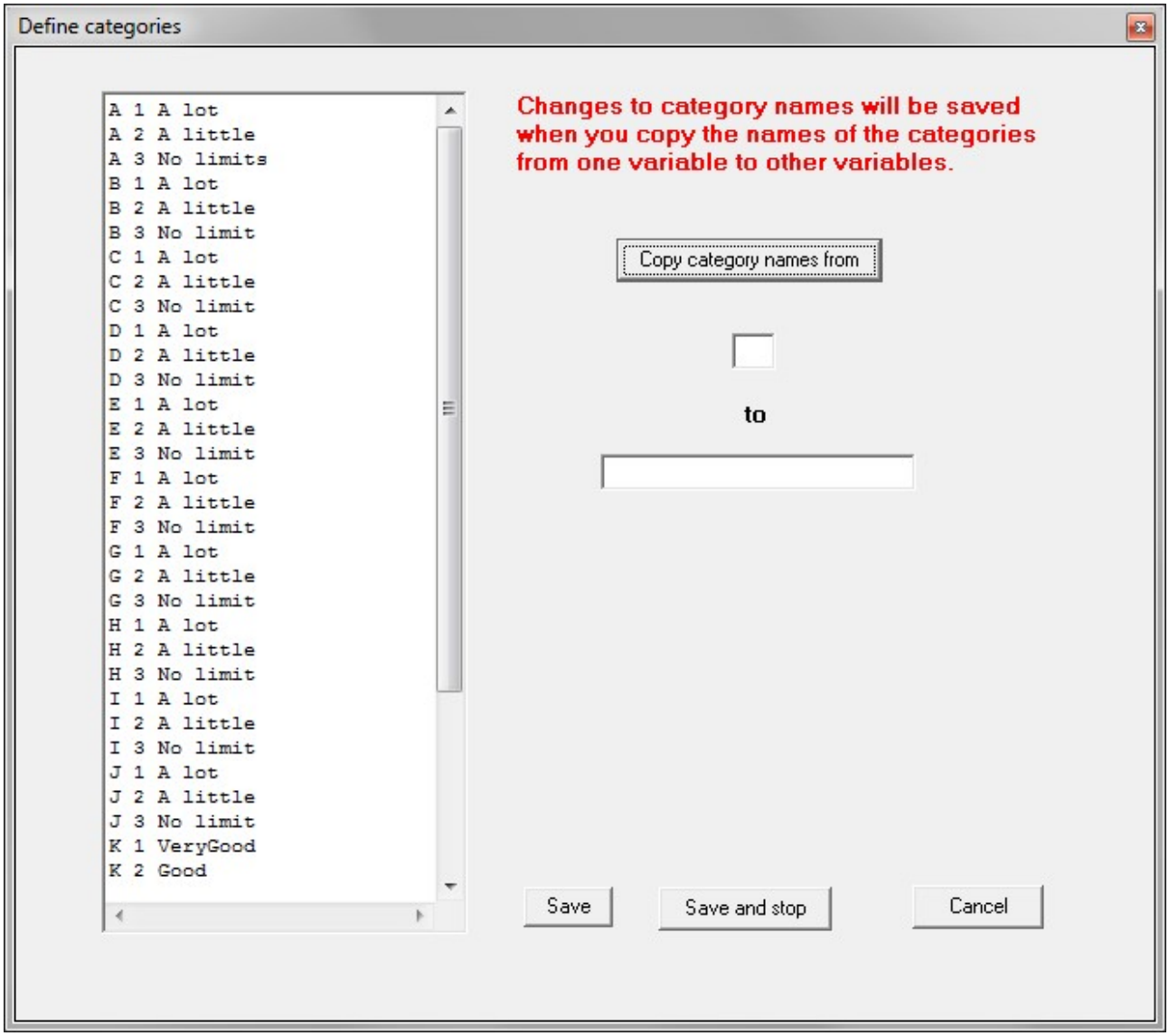


Figure 6.4 The category dialog with information on the categories of B-G after copying.

Important: Remember to save the results before you close the category dialog!

7 Project graphs

A number of extra features, which you may use if you want to change the appearance of a graph have been added to the GRAPH module (for general manipulation of graphs see the guided tour through the Graph module).

Coloured nodes

The COLOR command followed by a list of variables lets you change the colour of the nodes associated with the variables. If no variables are listed after the COLOR command, the program assumes that all nodes have to have a new colour.

To select the colour you have to select among the options presented in the dialog box in Figure 7.1.

Figure 7.1. Colour palette

Dotted and solid lines

Use

DOT <variable pair>

if the edge/arrow linking the two variables should appear as a dotted line and

SOLID <variable pair>

if you want a solid line between the variables.

Neither the colour or the types of edges will be saved on the GRF file.

Rescaling graphs

Figure 7.2 shows the project graph of a chain graph model of the Rochdale project[9].

The type of screen has an impact on how the graph appears when you start the program. The coordinates will automatically be rescaled to fit the graph window, if some of the coordinates lie beyond the boundaries of the window. If the coordinates on the GRF file defines a graph, which is too small for the window you may rescale it if you select the "Rescale graph" option on the FILE menu in the graph mode, as shown in Figure 7.2.

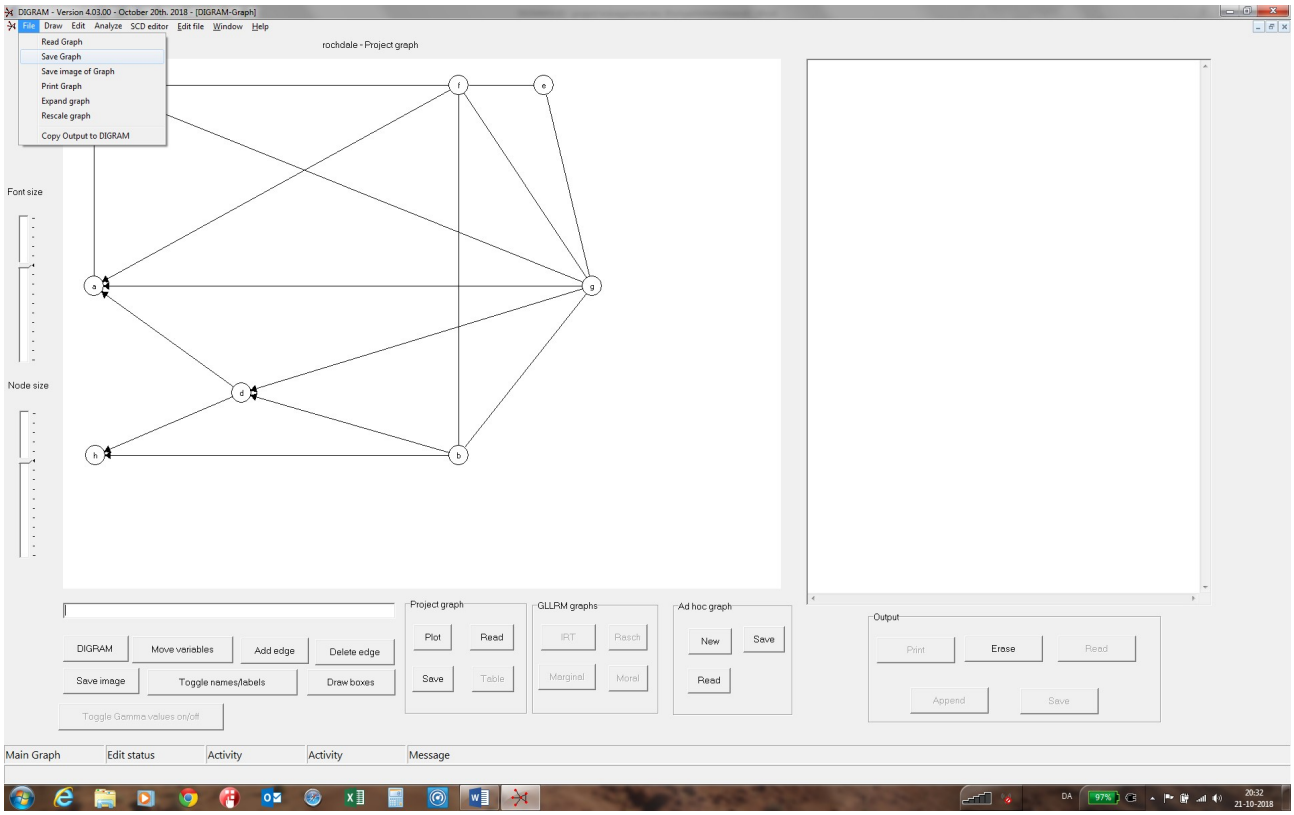


Figure 7.2. Rescaling graphs

Next, you have to select a rescale factor as in Figure 7.3. If the rescale factor is too large for the graph window, the coordinates will be automatically adjusted to fit the window (See Figure 7.4).

Figure 7.3. The rescale factor

[9] Described in the user guide and included with DIGRAM

Inverting graphs

The recursive structure of the graphs in DIGRAM is either organized from right to left (default) or from left to right. If you want to change the orientation of the graph you must use the

INVERT

command.

The result of rescaling (Figure 7.3) and inverting the graph in Figure 7.2 is shown in Figure 7.4.

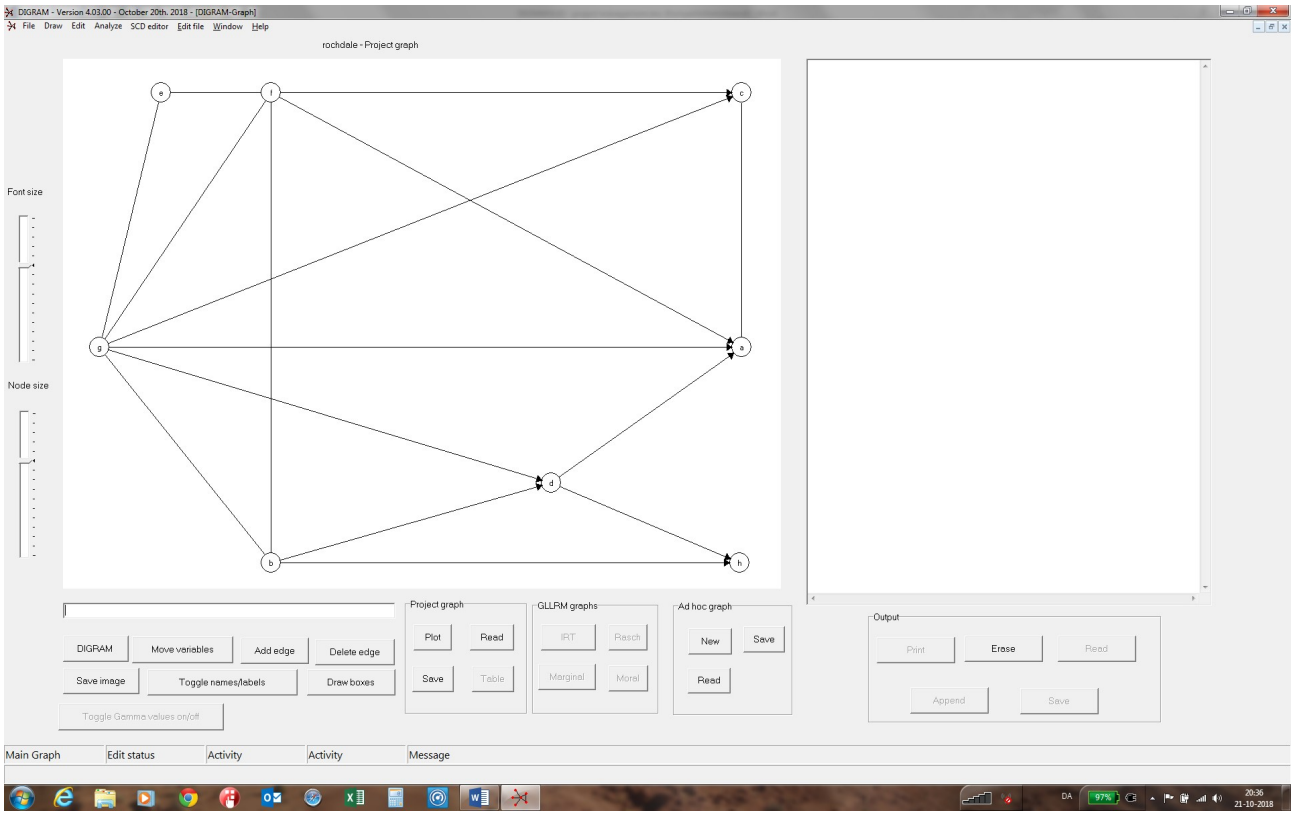


Figure 7.4. The rescaled and inverted project graph

8 Creating new DIGRAM projects

The data menu of the DIGRAM module (Figure 8.1) has several options that will let you

1) Revise variable definitions and categories.
2) Generate new DIGRAM projects.
3) Export project data for analysis in other programs.
4) Export data selected for item analysis to other item analysis programs.

This section is only concerned with new DIGRAM projects. Sections 9 and 10 show you how to define split projects defined by values of a project variable and how to add scores and stacked variables to your projects.

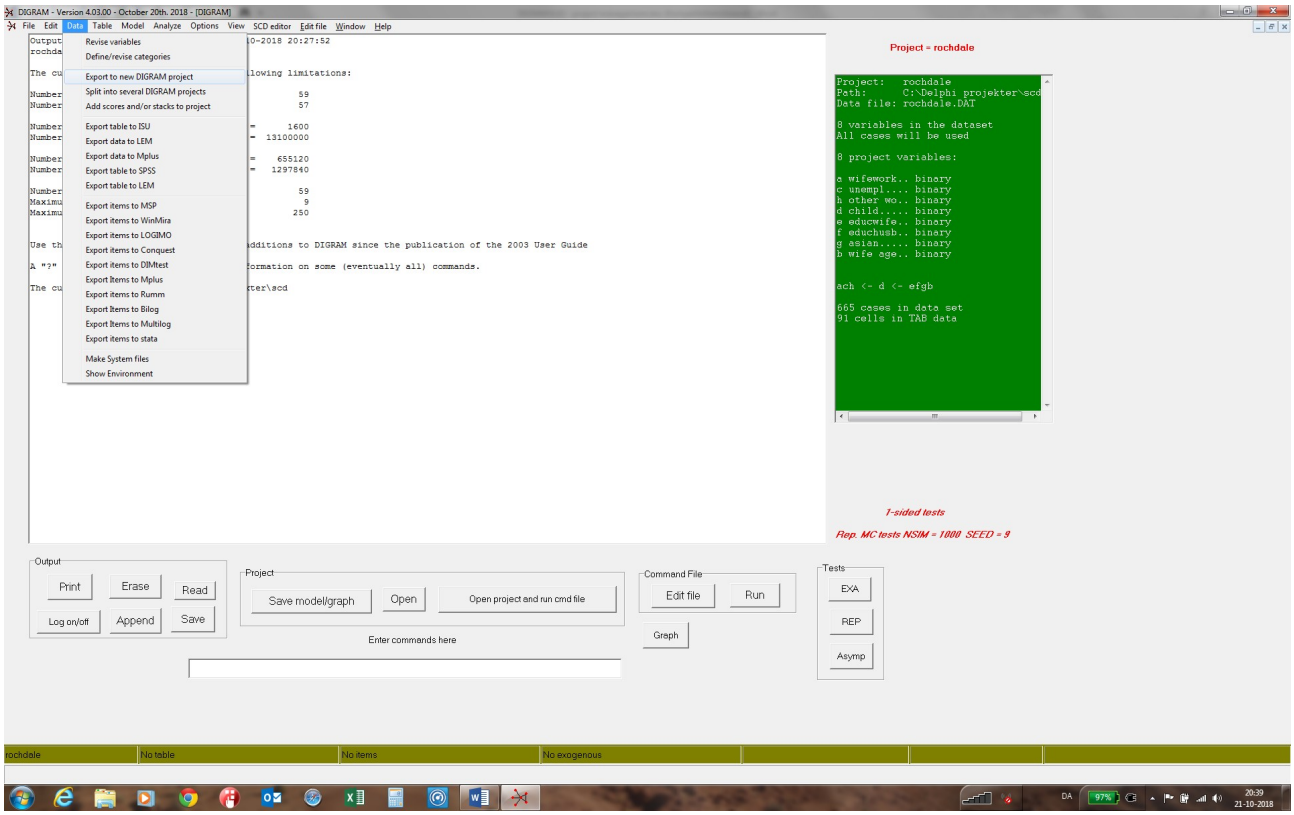


Figure 8.1. The DATA menu

Assume that you want to create a new DIGRAM project aiming at an undirected graphical model of the joint distribution of the variables of the Rochdale project and with the variables in the order described by the labels.

Select “Export to new DIGRAM project” to do this. Having done so, you will be asked (see Figure 8.2) which variables and in which order that you want to include in the project. You may leave the list of variables empty, or you may list a set of variables in the original or in a new order. You may also leave room for new variables by entering a ‘*’ instead of a variable label.

Figure 8.3 shows the dialog, where you define the new project. The definitions of the current project is to be found (written with red letters) in the left column of fields.

Click on “Use variables” if you want to use the list of variables defined in Figure 8.3 or “Use all variables” if you want the variables in the original order. Since the order of the variables has changed, the old recursive structure does not automatically apply. DIGRAM tries to come up with suggestions of a new order, but it is up to you to take that the order is as you wish it to be. Change the definition of the structure if DIGRAM’s suggestion does not please you and correct category labels if you think they could be better than in the original project.

Figure 8.2. The variables of the new project.

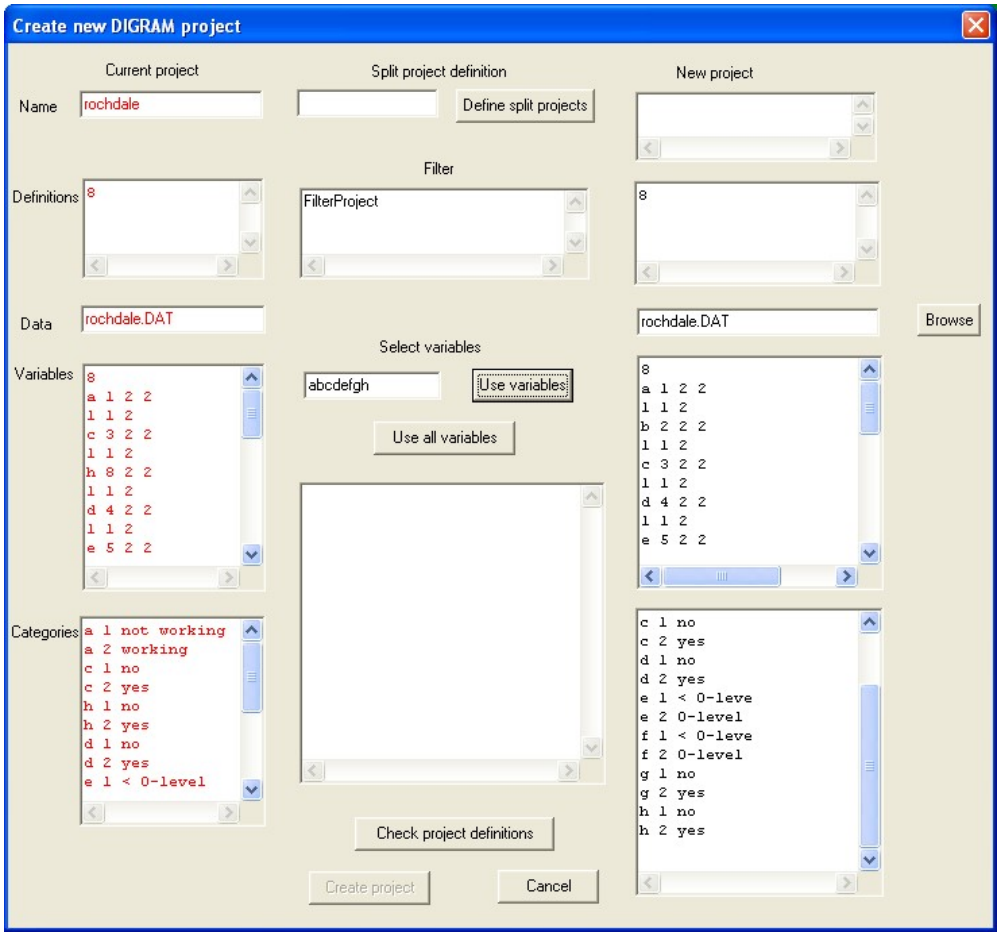


Figure 8.3. The definition of the new project after clicking on the "Use variables" button. Since the order of the variables disagrees with the recursive structure of the Rochdale model, DIGRAM makes no assumption about the recursive structure of the new project

Finally, add a name of the new project in the "New project field) at the upper right hand side of the dialog and click "Check project definitions. Figure 8.4 shows the result. The name of the new project is Whittaker, no errors were found, and the "Create project" button was enabled following which it only remains for you to press the button.

The (sub)graph of the project graph with the variables of the new project will be saved on the GRF file of the new project. The graph may be inappropriate as a model, because the recursive structure of the new model may be different from the order in the original project, but the new graph may serve a convenient starting point for an analysis of the new project.

Finally, note that the SELECT command cannot for the moment[10] be included in the definition of new projects. If you want such commands to be added to the DEF file of the new project you have to do it yourself using the SCD editor or any standard text editor before you open the new project.

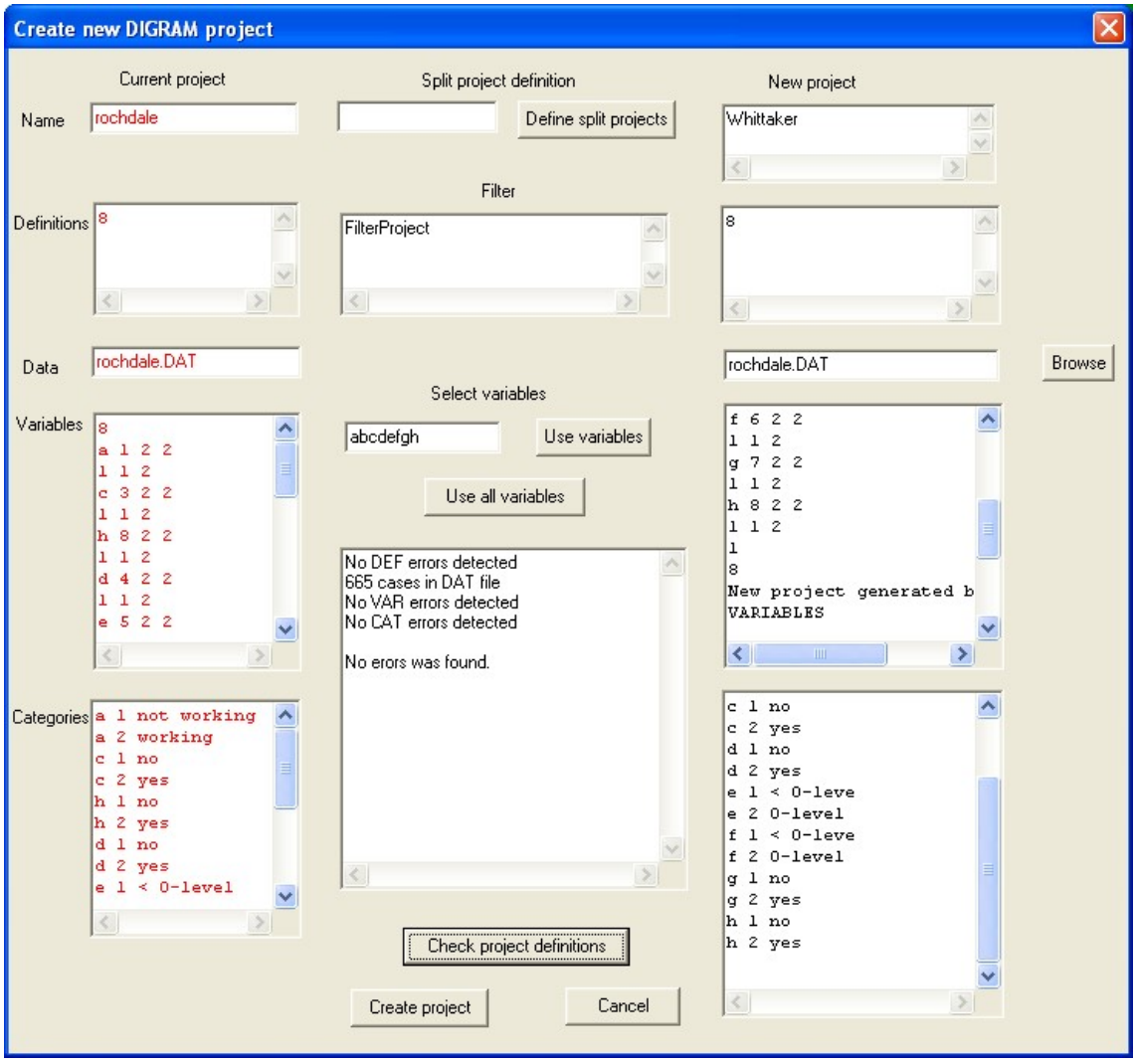


Figure 8.4. The setup after the new project name has been entered in the "New project" field.

[10] Patience! We are working on that

9 Creating split projects

Split projects are subprojects defined by the values of one of the project variables. We will use the DHP[11] project described in the "Item analysis in DIGRAM - Part I" guide to illustrate how to create split projects defined by age.

The item below the "Export to new DIGRAM project" on the list of items on the data menu (Figure 8.1) you will find the "Split into several DIGRAM projects" option. If you select this item, the dialog box shown in Figure 9.1 turns up.

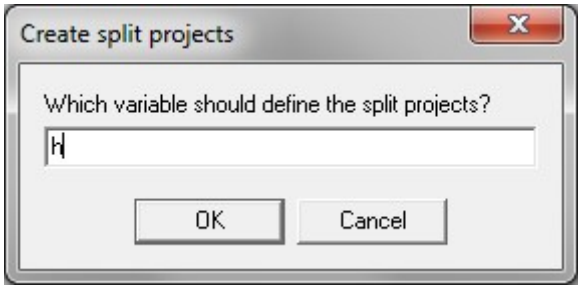


Figure 9.1. Select the variable defining the split projects.

Enter the label of the variable defining the split projects, click on OK and the dialog form Figure 9.2 turns up. Figure 9.2 is the same as Figure 8.3 except that the h label is in the field with "Split project definition". Click on "Define split projects" and DIGRAM defines the projects, suggest that you check the project definitions, and finally, when no errors were found, enables the "Create project" button.

In this case, the result is two projects, AGE1 and AGE2, with persons above and below 60 years of age. The current project graph without the split variable has been copied to the two split projects. It goes without saying that you need to check whether the model defined by this graph fits the data in the split projects.

[11] Age has been dichotomized in this example. The first age group includes persons with less than 60 years of age and the other person with age equal to or larger than 60 years.

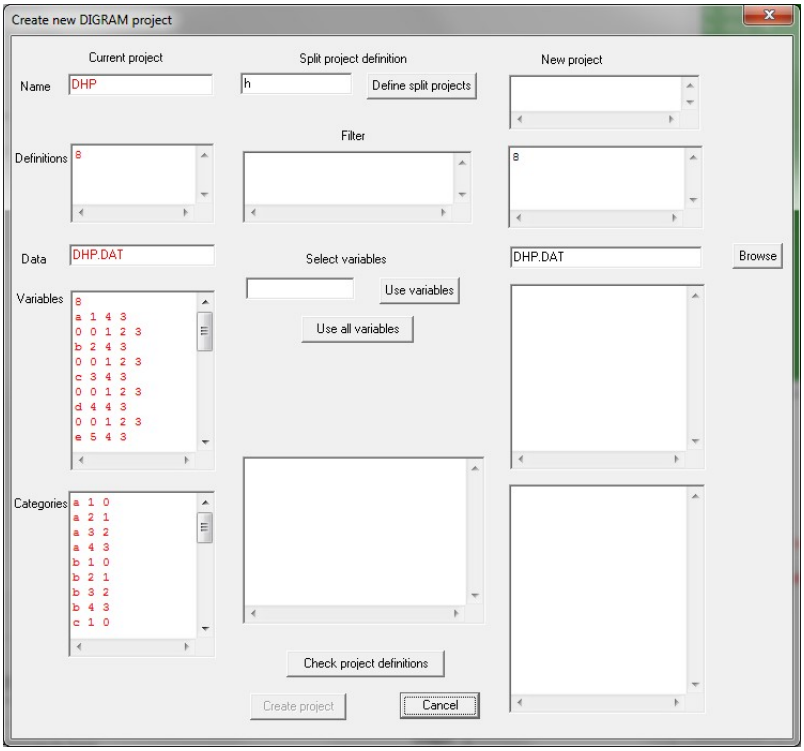


Figure 9.2. Initialization of definition of split projects

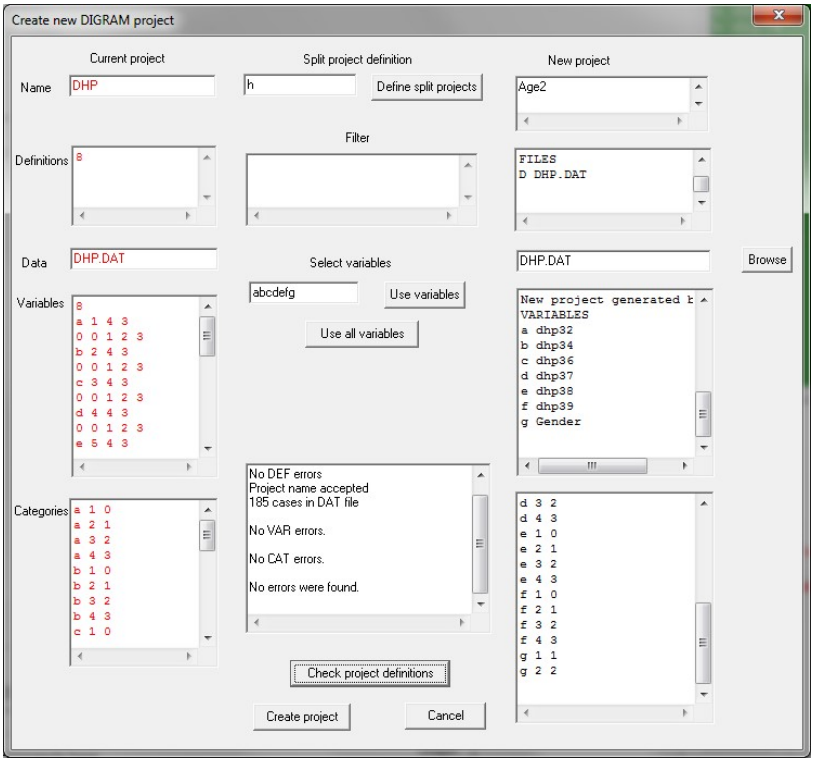


Figure 9.3. Definition of split projects

10 Adding variables to projects

You can create new projects with scores, stacked variables and/or other types of recoded variables to you project. In SCD 4.08, the following options are available.

1) Summary scores over a set of items
2) Import equated scores from a file with person scores
3) Stacked variables
4) Indicator variables
5) Missing variables
6) Random variables

Use the **RECODE** command to invoke these facilities. Figure 10.1 shows what happens when you do this from the PF3 project.

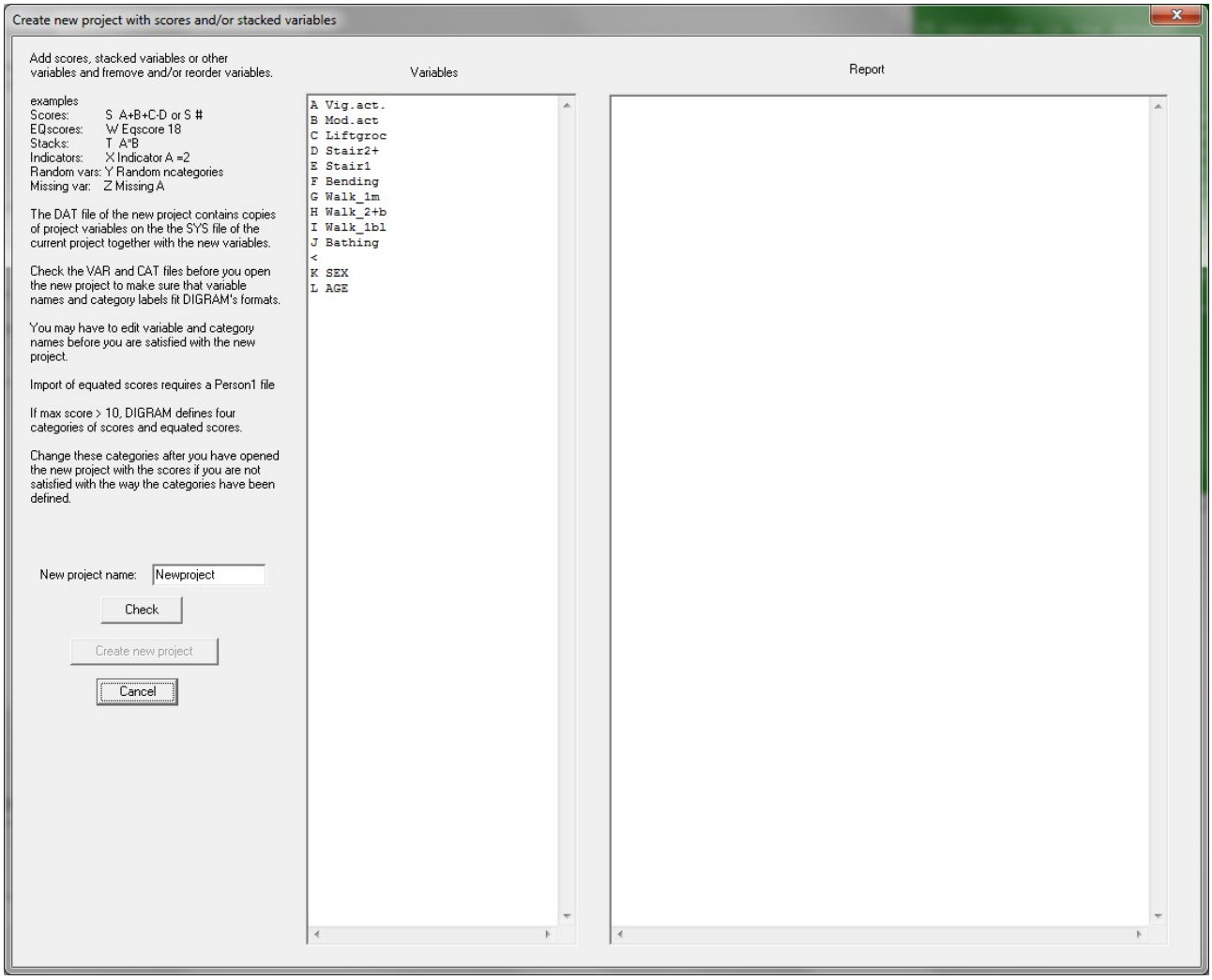


Figure 10.1 Dialog form for creating projects with recoded variables.

We will use the PF3 project to illustrate how to create scores and import equated scores and EJH92demo to illustrate the other options.

10.1 The PF3 project

The PF3 project contains 10 items and two exogenous variables. Figure 10.2 shows the IRT graph of a GLLRM for items 2-10 with Age and Sex as exogenous variables.

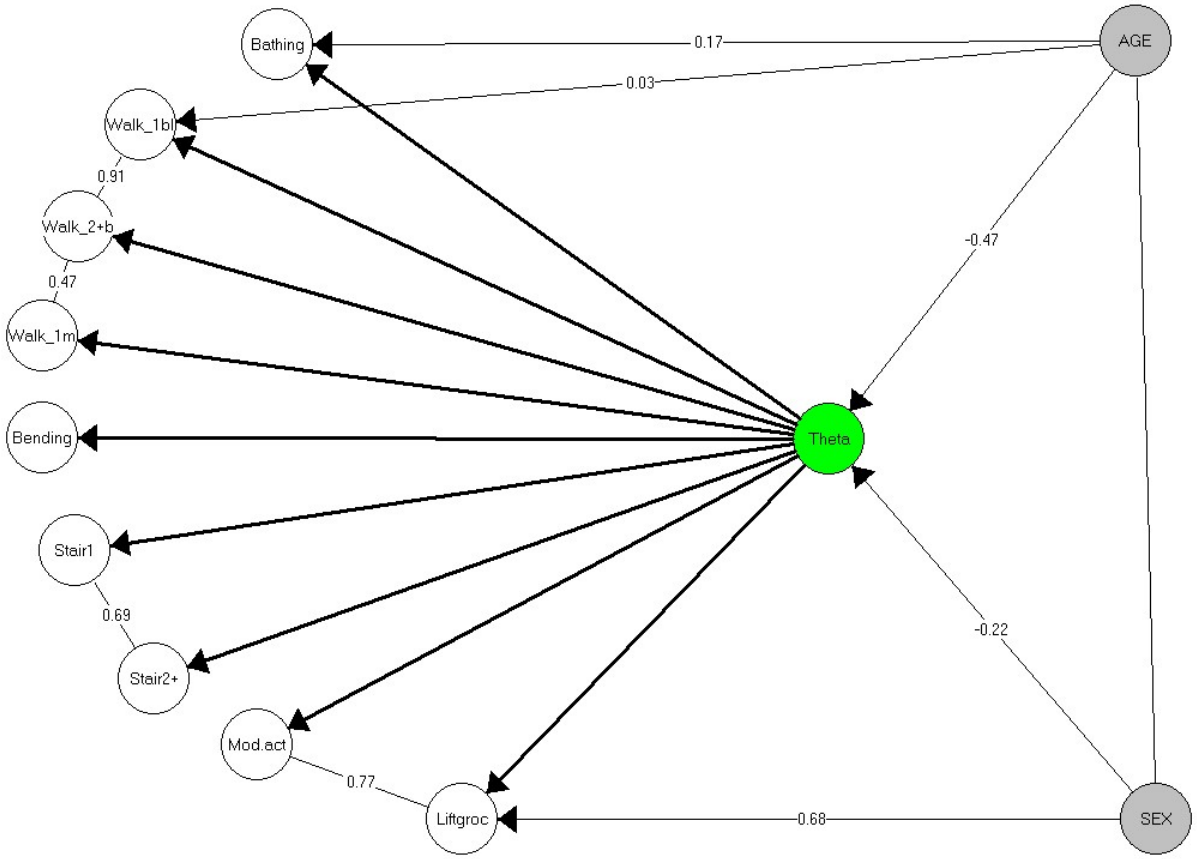


Figure 10.2 IRT graph for the PF3 model.

Figure 10.2 tells us the following.

First, that the total score over items are biased and should be replaced by a DIF equated score where DIF relative to Sec and Age has been taken into account.

Second, that super items summarizing responses to locally dependent items are distributed as partial credit items.

This suggests the following recoding.

First, to include both the total score and the equated score in the project to be able to see the effect of DIF on scores.

Second, to calculate super items defined by dependent items to compare a Rasch analysis with the super items to the analysis of the original items by the GLLRM. In principle the results of the two analyses should be similar since the two models are consistent.

To generate these variables we invoke the RECODE command and define the new variables as shown in Figure 10.3.

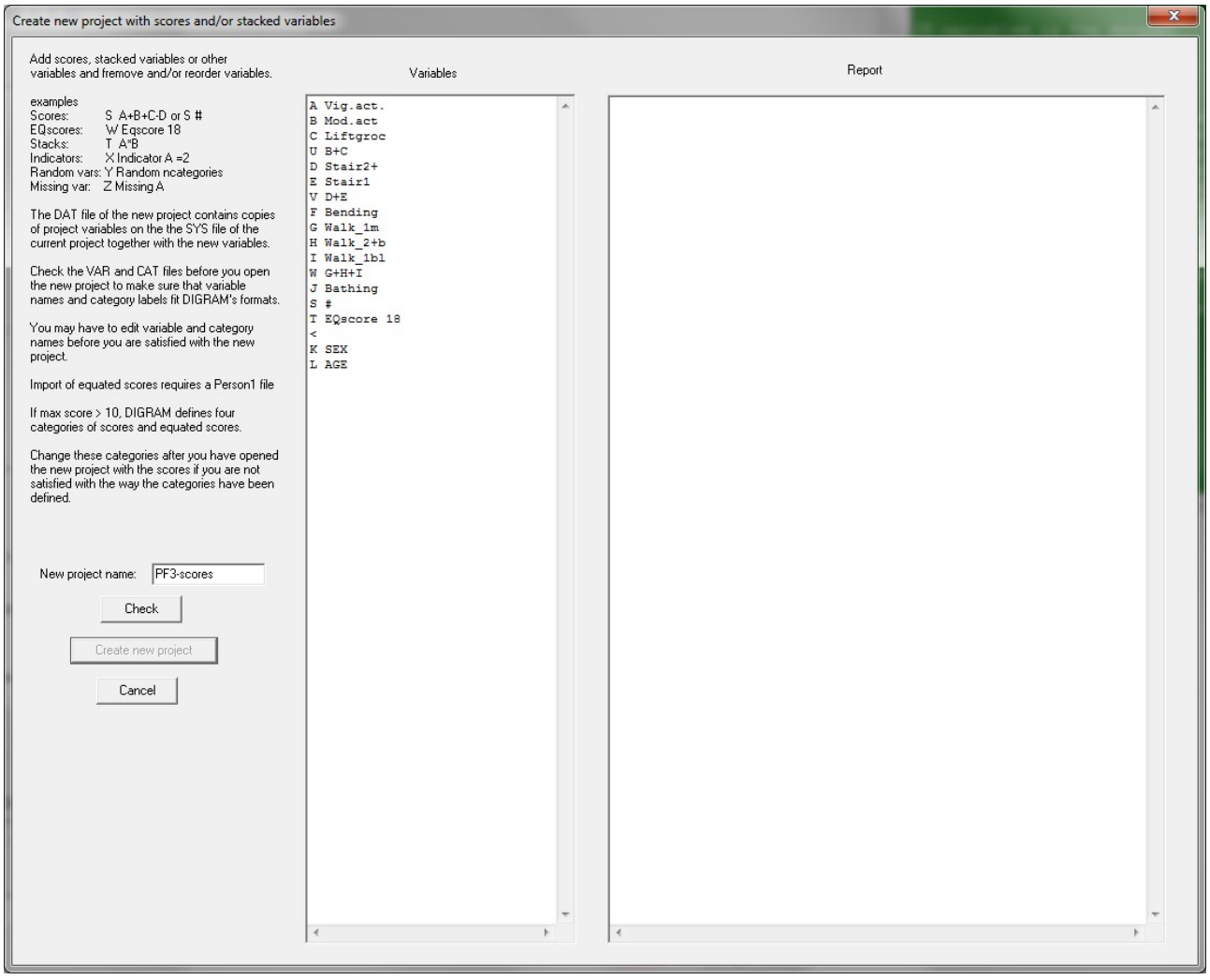


Figure 10.3 IRT graph for the PF3 model.

In Figure 10.3, the line with “S #” defines variable S as the score defined by the current selection of items. If items have not been selected, DIGRAM will remind you that you have forgotten to do so. The “T EQscore 18” line defines a variable with equated scores to be found on the PF3_person1.txt file. Since DIGRAM does not know that how these scores have been defined you have to include information (18) on the maximum score.

line Click on CHECK to test that the variables has been correctly defined and then on “Create New Project” if they have been accepted. The results can be seen in Figure 10.4.

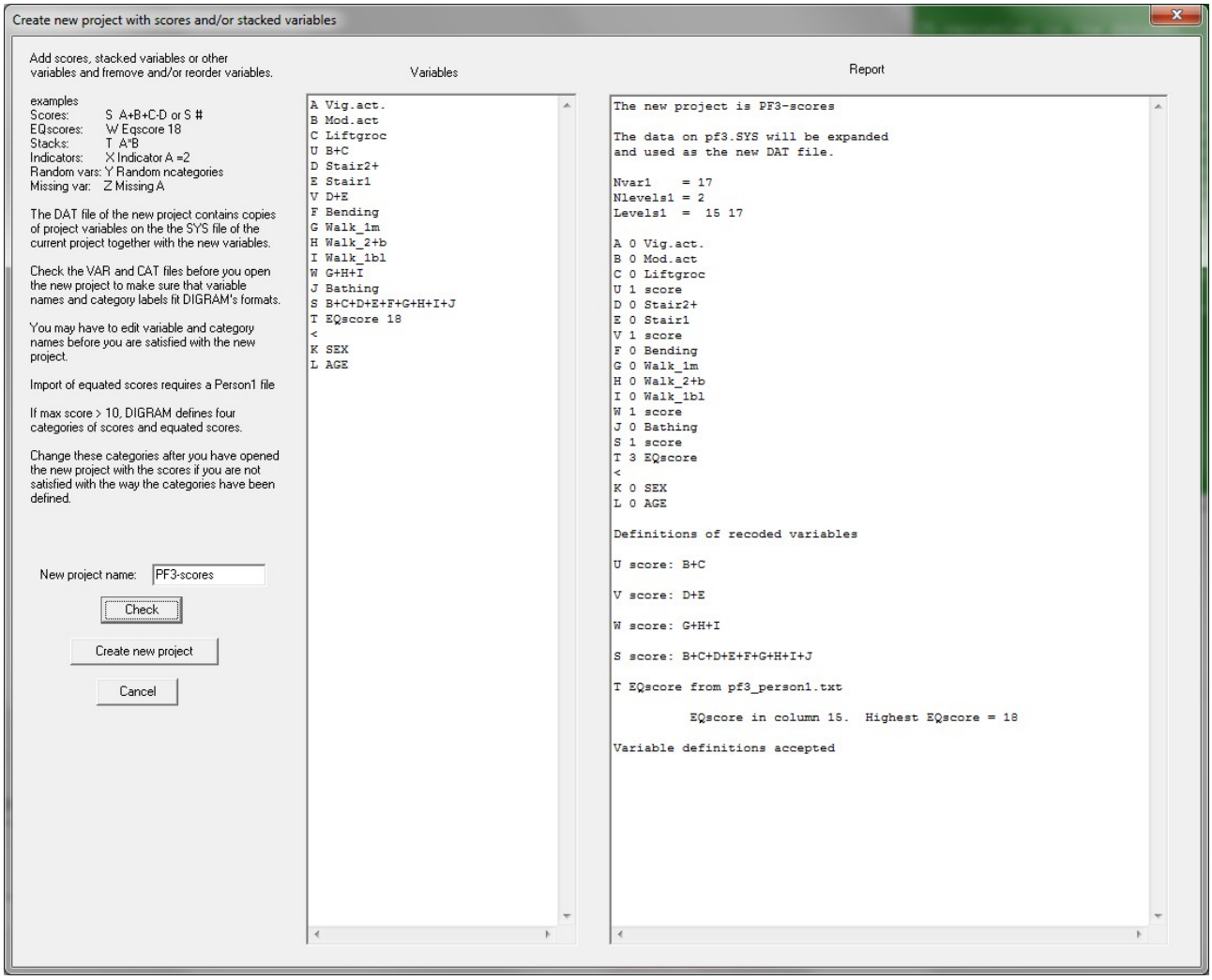


Figure 10.4. The DHP project expanded with scores and super items. The definitions of the new variables have been checked, and the “Create expanded project” button enabled since no errors were found.

Next, click on the “Create expanded project” button. DIGRAM creates the files and closes the DIAGRAM form and leaves the rest to you.

Open the new project, use the VAR command to take a look at the variable definitions and to make the necessary changes. Figure 10.5 shows the contents of the VAR file generated by the RECODE command. Notice the classification of the score (S) and the equated score (T). T is a continuous variable. To make comparison between S and T easier, DIGRAM defines categories rounding the equated scores to integers comparable to the score.

```
17
A 1 3 3
1 1 2 3
B 2 3 3
1 1 2 3
C 3 3 3
1 1 2 3
U 13 5 3
0 0 1 2 3 4
D 4 3 3
1 1 2 3
E 5 3 3
1 1 2 3
V 14 5 3
0 0 1 2 3 4
F 6 3 3
1 1 2 3
G 7 3 3
1 1 2 3
H 8 3 3
1 1 2 3
I 9 3 3
1 1 2 3
W 15 7 3
0 0 1 2 3 4 5 6
J 10 3 3
1 1 2 3
S 16 4 3
0 4 9 13 18
T 17 4 3
0 4.5 9.5 13.5 18
K 11 2 3
1 1 2
L 12 4 3
1 1 2 3 4
2
 15 17
COMMENTS
Data was expanded from pf3.SYS
VARIABLENAMES
A Vig.act.
B Mod.act
C Liftgroc
U B+C
D Stair2+
E Stair1
V D+E
F Bending
G Walk_1m
H Walk_2+b
I Walk_1bl
W G+H+I
J Bathing
S B+C+D+E+F+G+H+I+J
T EQscore
K SEX
L AGE
```

Figure 10.5 The VAR file of the PF3-scores project

Figure 10.6 shows the frequencies of the two scores. The total score is only calculated for persons where all items are observed. The equated score include the expected equated score for persons with missing responses (except of course, if information is completely missing. For this reason there are more persons with missing S score than with missing T score.

```
  +-------------+
  |             |
  | S: B+C+D+E+ |
  |             |
  +-------------+

Reference no.  14
Variable no.   16

This variable was categorized

  Values  S Count     Pct  CumPct
---------------------------------
     0-4  1      21   2.2     2.2
     5-9  2      39   4.0     6.2
   10-13  3      51   5.3    11.5
    7-18  4     856  88.5   100.0

TOTAL   967

Missing:
BELOW    82
=========================================

  +------------+
  |            |
  | T: EQscore |
  |            |
  +------------+

Reference no.  15
Variable no.   17

This variable was categorized

  Values  T Count     Pct  CumPct
---------------------------------
     0-4  1      32   3.2     3.2
     5-9  2      34   3.4     6.5
   10-13  3      64   6.3    12.8
    7-18  4     883  87.2   100.0

TOTAL  1013

Missing:
BELOW    36
=========================================
```

Figure 10.6 The marginal S and T distributions in the PF3-scores project

Figure 10.7 shows the crosstab with the two scores. In this case there is little difference betyween the observed and equated scores. However 24 persons have lower equated than observed scores.

Table 1. The ST distribution.

T	EQscore S:B+C+D+E+ 0-4	5-9	10-13	7-18	TOTAL	
0-4	21	0	0	0	21	
row%	100.0	0.0	0.0	0.0	100.0	
5-9	0	28	0	0	28	
row%	0.0	100.0	0.0	0.0	100.0	
10-13	0	11	38	0	49	
row%	0.0	22.4	77.6	0.0	100.0	
7-18	0	0	13	856	869	
row%	0.0	0.0	1.5	98.5	100.0	X^2 = 2270.5
						df = 9
TOTAL	21	39	51	856	967	p = 0.000
row%	2.2	4.0	5.3	88.5	100.0	Gam = 1.00
						p = 0.000

Figure 10.7 The joint (S,T) distribution in the PF3-scores project

Obviously, you have to change variable names and category labels before the PF3-scores project is up and running. If you intend to do an item analysis of B, U, V, F, W, J, you also have to change items so that all items are defined by the same number of categories. We assume that you know how to do that and therefore leave the project here.

10.2 The EJH92-demo project

Data for this project came from a panel study with background information at age = 18 in 1967 and information on health and socioeconomic variable 25 years later.

The eleven variables included fits the chain graph model shown in Figure 10.8

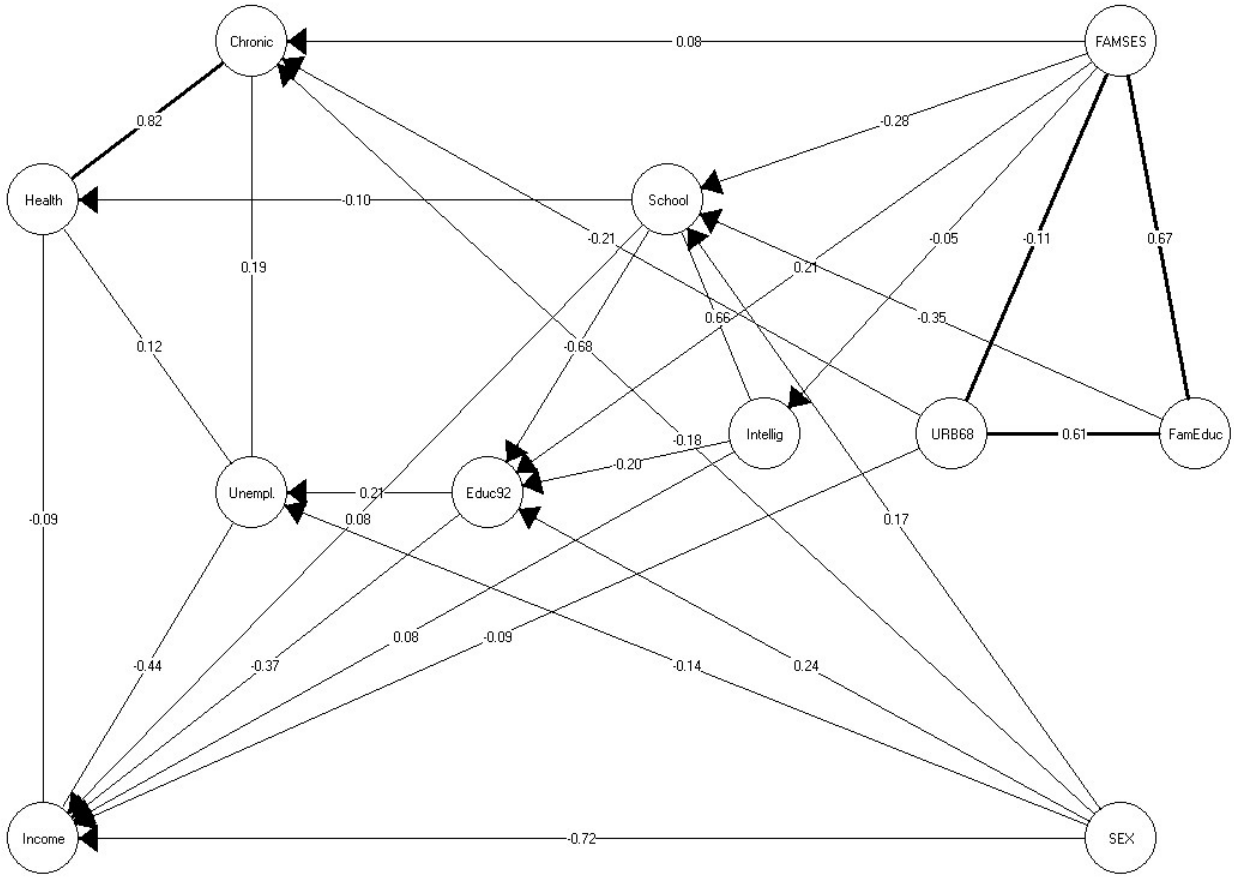


Figure 10.8 Chain graph model for the variables in EJH92demo

We will add the following variables to this project:

1) A variable with information on whether Income (D) is missing.
2) An indicator variable distinguishing between persons without education (F=4) after school
3) A variable defined as a stack of FAMSES (K) and FamEduc (L)
4) A random variable with three categories

All of these can be used for something useful, but the purpose here is simple to illustrate how recoding works.

Figure 10.9 shows the setup and the report after accept of the new variables.

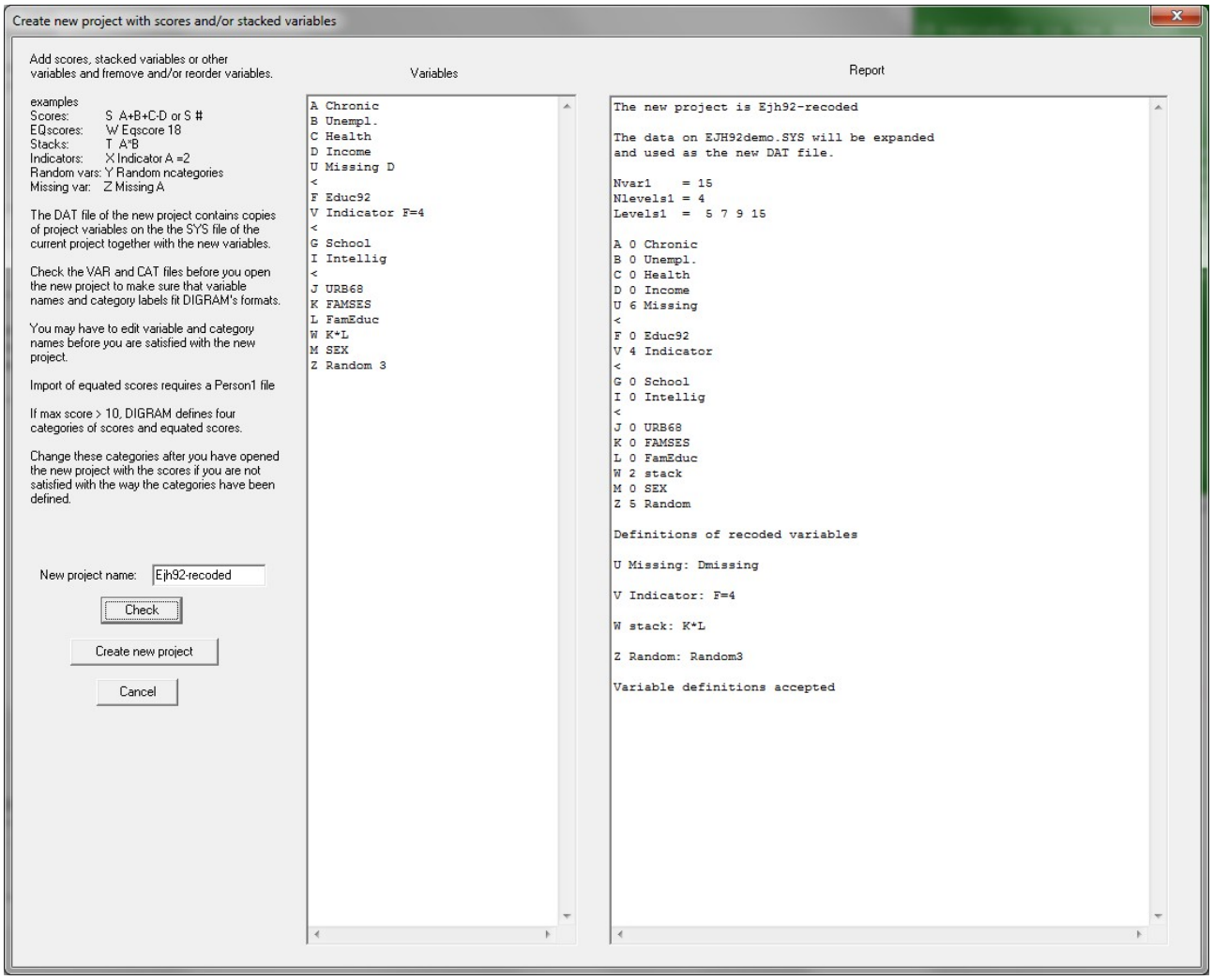


Figure 10.8 Chain graph model for the variables in EJH92demo

Figure 10.9 shows the marginal distributions of the new variables. Once again, it is obvious that you have to work on variable names and category labels, before you use them.

```
  +-------------+
  |             |
  | U: Dmissing |
  |             |
  +-------------+

Reference no.   5
Variable no.   12

            U Count     Pct  CumPct
-----------------------------------
observed    1  2342   74.33   74.33
 missing    2   809   25.67  100.00

TOTAL  3151

=======================================

  +---------+
  |         |
  | V: F=4  |
  |         |
  +---------+

Reference no.   7
Variable no.   13

            V Count     Pct  CumPct
-----------------------------------
    F<>4    1  2143   80.29   80.29
     F=4    2   526   19.71  100.00

TOTAL  2669

Missing:
BELOW   482
=======================================
```

```
  +---------+
  |         |
  | W: K*L  |
  |         |
  +---------+

Reference no.  13
Variable no.   14

            W Count     Pct  CumPct
-----------------------------------
     1*1    1    84    3.41    3.41
     2*1    2    40    1.63    5.04
     3*1    3    20    0.81    5.85
     4*1    4     0    0.00    5.85
     5*1    5     0    0.00    5.85
     1*2    6    13    0.53    6.38
     2*2    7    67    2.72    9.10
     3*2    8    37    1.50   10.61
     4*2    9     6    0.24   10.85
     5*2   10     1    0.04   10.89
     1*3   11    32    1.30   12.19
     2*3   12   134    5.44   17.64
     3*3   13   584   23.73   41.37
     4*3   14   390   15.85   57.21
     5*3   15   123    5.00   62.21
     1*4   16     0    0.00   62.21
     2*4   17    14    0.57   62.78
     3*4   18   345   14.02   76.80
     4*4   19   188    7.64   84.44
     5*4   20   383   15.56  100.00

TOTAL  2461

Missing:
BELOW   690
=======================================


  +-------------+
  |             |
  | Z: Random3  |
  |             |
  +-------------+

Reference no.  15
Variable no.   15

            Z Count     Pct  CumPct
-----------------------------------
 subpop1    1  1036   32.88   32.88
 subpop2    2  1044   33.13   66.01
 subpop3    3  1071   33.99  100.00

TOTAL  3151

=======================================
```

Figure 10.9. Marginal distributions of variables the new variables in EJH92-recoded

10.3 The Fienberg-6-8 project. Recoding variables.

Data for this project came Table 6-8 of Fienberg (1980) summarizing information on 2294 young males who failed to pass the armed forces qualification test in 1963. The purpose of looking at was to test the analysis by log-linear models in DIGRAM. The project included information on the educational level of the respondent and the respondent's father defined by three ordinal categories, Gramma School, Some high school education, and High School graduate. Unfortunately, the father's education included a missing category, for which reason the father's Education had to be treated as a nominal variable.

Analysis of category colalpsibility showed that there was little difference between the distributions of the respondents education when the father's education was missing or Gramma school. To simplify the analysis and to be able to use tests for ordinal categorical data it was decided to replace the father's education with another variable where gramma school and Missing was collapsed into one category.

To do that you have to define a new variable by "F = Recode B 1 2 3 1" with information on the new codes of assigned to the outcomes of variable B.

Figure 10.10 shows how to do it and that the setup was accepted. Figure 10.11 shows the distributions of B before recoding and F after recoding. Notice that you once again have to edit the variable definitions and categories of F to make them presentable including the variable type of F that recoding defines as a nominal variable.

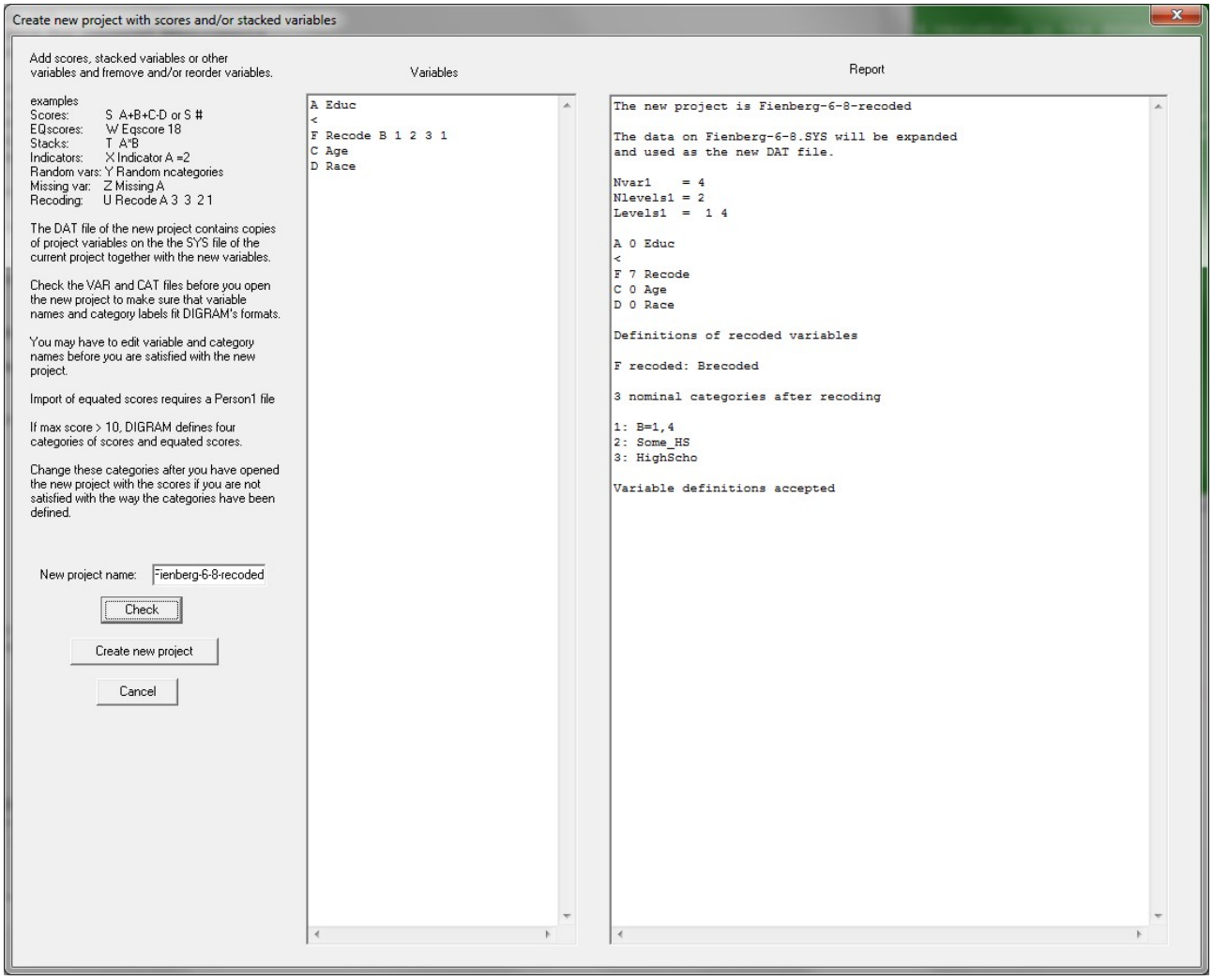


Figure 10.10 Recoding replacing variable B by variable F with three categories.

```
  +-----------+
  |           |
  | B: Father |
  |           |
  +-----------+

Reference no.   2
Variable no.    2

            B Count      Pct  CumPct
-------------------------------------
 Grammar    1   897    39.10   39.10
 Some_HS    2   174     7.59   46.69
HighScho    3   190     8.28   54.97
 Missing    4  1033    45.03  100.00

TOTAL  2294
```

```
  +-------------+
  |             |
  | F: Brecoded |
  |             |
  +-------------+

Reference no.   2
Variable no.    5

            F Count      Pct  CumPct
-------------------------------------
   B=1,4    1  1930    84.13   84.13
 Some_HS    2   174     7.59   91.72
HighScho    3   190     8.28  100.00

TOTAL  2294
```

Figure 10.11. Fathers Education after Recoding

11 Select data for subprojects

You may use the SELECT command to create subprojects from an already existing project containing persons with a subset of the values associated with one of the variables.

We will use the PF3 project described in the guided tours through DIGRAM's item analyses to illustrate how to SELECT persons restricted by age.

The PF3 project contains the following variables

A Vig.act.	B Mod.act	C Liftgroc
1 A lot	1 A lot	1 A lot
2 A little	2 A little	2 A little
3 NoLimits	3 NoLimits	3 NoLimits

D Stair2+	E Stair1	F Bending
1 A lot	1 A lot	1 A lot
2 A little	2 A little	2 A little
3 NoLimits	3 NoLimits	3 NoLimits

G Walk 1m	H Walk 2+b	I Walk 1bl
1 A lot	1 A lot	1 A lot
2 A little	2 A little	2 A little
3 NoLimits	3 NoLimits	3 NoLimits

J Bathing	K SEX	L AGE
1 A lot	1 Male	1 16 - 29
2 A little	2 Female	2 30 - 44
3 NoLimits		3 45 - 59
		4 60 - 69

The parameters to the SELECT command has to define the restriction that has to be satified if persons are included in the subproject.

There are three different ways to do this:

First, by the label of the variable and a set of category numbers. This command first creates a new data file with the persons, but does not change the data. The definition of the variable used for selecting will be changed so that categories that have been deselected do not exist.

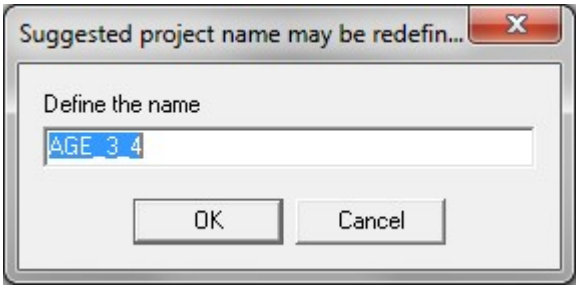
Second, by the label and subsets of categories defined by category numbers connected with a "&",which results in the categories in the subsets will be collapsed into one category in the sub project. This command also creates a new data set, but the variable defining the criteria will be recoded with values referring to the category numbers in the new project.

Finally, by a label and single category number. The dataset includes all persons where the value of the variable correspond to the category number. Since the variable only has one value in the subproject, the variable is eliminated from the set of variables.

The following example will show how it works. Note, that you may include a "=" between the variable and the categories if you think it is convenient to do so.

Use **"SELECT L = 3 4"** to define a project with the two eldest age groups.

DIGRAM selects a name for the subproject, but asks whether you want to use another name



The output following the SELECT command looks like this:

```
  +-------------------------------------+
  |                                     |
  | Extract of data from pf3 to AGE_3_4 |
  |                                     |
  +-------------------------------------+

  +-----------------------------------------+
  |                                         |
  | Overview of persons included in AGE_3_4 |
  |                                         |
  +-----------------------------------------+

508 persons

   old                new
```

```
category           category    Frequencies
-------------------------------------------------------------
    3     45 - 59      1        264  52.0%
    4     60 - 69      2        244  48.0%
```

The next step is to open the new project and to check that everything worked well by a “SHOW V” command to see the variables in the subproject and a “FRE L” command check the distribution of L in the subproject.

```
+----------------------------------------+
|  A Vig.act. |  B  Mod.act |  C Liftgroc |
|-------------+-------------+-------------|
|  1    A lot |  1    A lot |  1    A lot |
|  2 A little |  2 A little |  2 A little |
|  3 NoLimits |  3 NoLimits |  3 NoLimits |
+----------------------------------------+

+----------------------------------------+
|  D  Stair2+ |  E   Stair1 |  F  Bending |
|-------------+-------------+-------------|
|  1    A lot |  1    A lot |  1    A lot |
|  2 A little |  2 A little |  2 A little |
|  3 NoLimits |  3 NoLimits |  3 NoLimits |
+----------------------------------------+

+----------------------------------------+
|  G  Walk 1m |  H Walk 2+b |  I Walk 1bl |
|-------------+-------------+-------------|
|  1    A lot |  1    A lot |  1    A lot |
|  2 A little |  2 A little |  2 A little |
|  3 NoLimits |  3 NoLimits |  3 NoLimits |
+----------------------------------------+

+----------------------------------------+
|  J  Bathing |  K      SEX |  L      AGE |
|-------------+-------------+-------------|
|  1    A lot |  1     Male |  1  45 - 59 |
|  2 A little |  2   Female |  2  60 - 69 |
|  3 NoLimits |             |             |
+----------------------------------------+

  +--------+
  |        |
  | L: AGE |
  |        |
  +--------+

Reference no.  12
Variable no.    2

This variable was categorized

  Values  L Count     Pct  CumPct
---------------------------------
```

45 - 59	1	264	52.0	52.0
60 - 69	2	244	48.0	100.0
TOTAL	508			

Use **"SELECT L = 1&2 3"** to define a project with the two youngest age groups collapsed into one group and with the third age group as a separate group.

```
  +--------------------------------------+
  |                                      |
  | Extract of data from pf3 to AGE_1-2_3 |
  |                                      |
  +--------------------------------------+

Persons have been selected for AGE_1-2_3 if AGE =  1 2 3


  +-----------------------------------------+
  |                                         |
  | Overview of persons included in AGE_1-2_3 |
  |                                         |
  +-----------------------------------------+

805 persons

   old                new
category           category    Frequencies
-------------------------------------------------------------------
    1     16 - 29      1        248  30.8%
    2     30 - 44      2        293  36.4%
    3     45 - 59      3        264  32.8%

AGE has been recoded in data to collapse categories

Frequencies of recoded date

category   count  pct
-----------------------
 1       1&2  541  67.2
 2   45 - 59  264  32.8
```

The variables of the subproject follow below. The user has to redefine the category label of the collapsed category herself.

```
+----------------------------------------+
|  A Vig.act. |  B  Mod.act |  C Liftgroc |
|-------------+-------------+-------------|
|  1    A lot |  1    A lot |  1    A lot |
|  2 A little |  2 A little |  2 A little |
|  3 NoLimits |  3 NoLimits |  3 NoLimits |
+----------------------------------------+


+----------------------------------------+
|  D  Stair2+ |  E   Stair1 |  F  Bending |
|-------------+-------------+-------------|
|  1    A lot |  1    A lot |  1    A lot |
|  2 A little |  2 A little |  2 A little |
|  3 NoLimits |  3 NoLimits |  3 NoLimits |
+----------------------------------------+
```

```
+----------------------------------------+
|  G  Walk 1m |  H Walk 2+b |  I Walk 1bl |
|-------------+-------------+-------------|
|  1    A lot |  1    A lot |  1    A lot |
|  2 A little |  2 A little |  2 A little |
|  3 NoLimits |  3 NoLimits |  3 NoLimits |
+----------------------------------------+


+----------------------------------------+
|  J  Bathing |  K      SEX |  L      AGE |
|-------------+-------------+-------------|
|  1    A lot |  1     Male |  1      1&2 |
|  2 A little |  2   Female |  2  45 - 59 |
|  3 NoLimits |             |             |
+----------------------------------------+


  +--------+
  |        |
  | L: AGE |
  |        |
  +--------+

Reference no.  12
Variable no.    2

            L Count     Pct  CumPct
-------------------------------------
     1&2    1   541   67.20   67.20
 45 - 59    2   264   32.80  100.00

TOTAL   805
```

Use **"SELECT L = 2"** to define a project exclusively with persons from the second age group.

```
  +-----------------------------------+
  |                                   |
  | Extract of data from pf3 to AGE_2 |
  |                                   |
  +-----------------------------------+

Persons have been selected for AGE_2 if AGE =  2

  +---------------------------------------+
  |                                       |
  | Overview of persons included in AGE_2 |
  |                                       |
  +---------------------------------------+

293 persons

   old               new
category          category    Frequencies
-------------------------------------------------------------------
    2     30 - 44      1        293 100.0%
```

The subproject has no Age variable because all persons in the subproject are now between 30 and 44 years.

A Vig.act.	B Mod.act	C Liftgroc
1 A lot	1 A lot	1 A lot
2 A little	2 A little	2 A little
3 NoLimits	3 NoLimits	3 NoLimits

D Stair2+	E Stair1	F Bending
1 A lot	1 A lot	1 A lot
2 A little	2 A little	2 A little
3 NoLimits	3 NoLimits	3 NoLimits

G Walk 1m	H Walk 2+b	I Walk 1bl
1 A lot	1 A lot	1 A lot
2 A little	2 A little	2 A little
3 NoLimits	3 NoLimits	3 NoLimits

J Bathing	K SEX
1 A lot	1 Male
2 A little	2 Female
3 NoLimits	

12 Exporting project data to other programs

(to be explained later)

13 Creating projects from tables

Figure 13.1 shows a 4-way table from Fienberg (1979) that we want to use to illustrate loglinear analysis in DIGRAM.

Table 6-8
Observed Cross-Classification of 2294 Young Males Who Failed to Pass the Armed Forces Qualification Test (Talbot and Mason [1975])

Race	Age	Father's Education*	Respondent's Education: Grammar School	Some HS	HS Graduate
White	< 22	1	39	29	8
		2	4	8	1
		3	11	9	6
		4	48	17	8
	≥ 22	1	231	115	51
		2	17	21	13
		3	18	28	45
		4	197	111	35
Black	< 22	1	19	40	19
		2	5	17	7
		3	2	14	3
		4	49	79	24
	≥ 22	1	110	133	103
		2	18	38	25
		3	11	25	18
		4	178	206	81

*1 = Grammar School, 2 = Some HS, 3 = HS Graduate, 4 = Not Available.

Figure 13.1 a four-way table.

To create a DIGRAM project with this table you must either select the “Enter table” or “Read Table from file” option on the Table menu. We suggest you use “Enter table” unless you know exactly what DIGRAM needs to accommodate you. Doing this opens the dialog form shown in Figure 13.2 where you have to enter information on the variables and the cells of the table or read the contents of a file with the information. Figure 13.3 shows the result after the table has been checked and accepted.

The next step is to “Save EXA file and define project”. DIGRAM asks you to save the table on an EXA file. Choose Fienberg-6-8.exa and DIGRAM understands that “Fienberg-6-8” is the name of the new project. Next, DIGRAM asks you to define the categories. Figure 13.3 shows that you have to do it in the same way that you define categories using a CAT command.

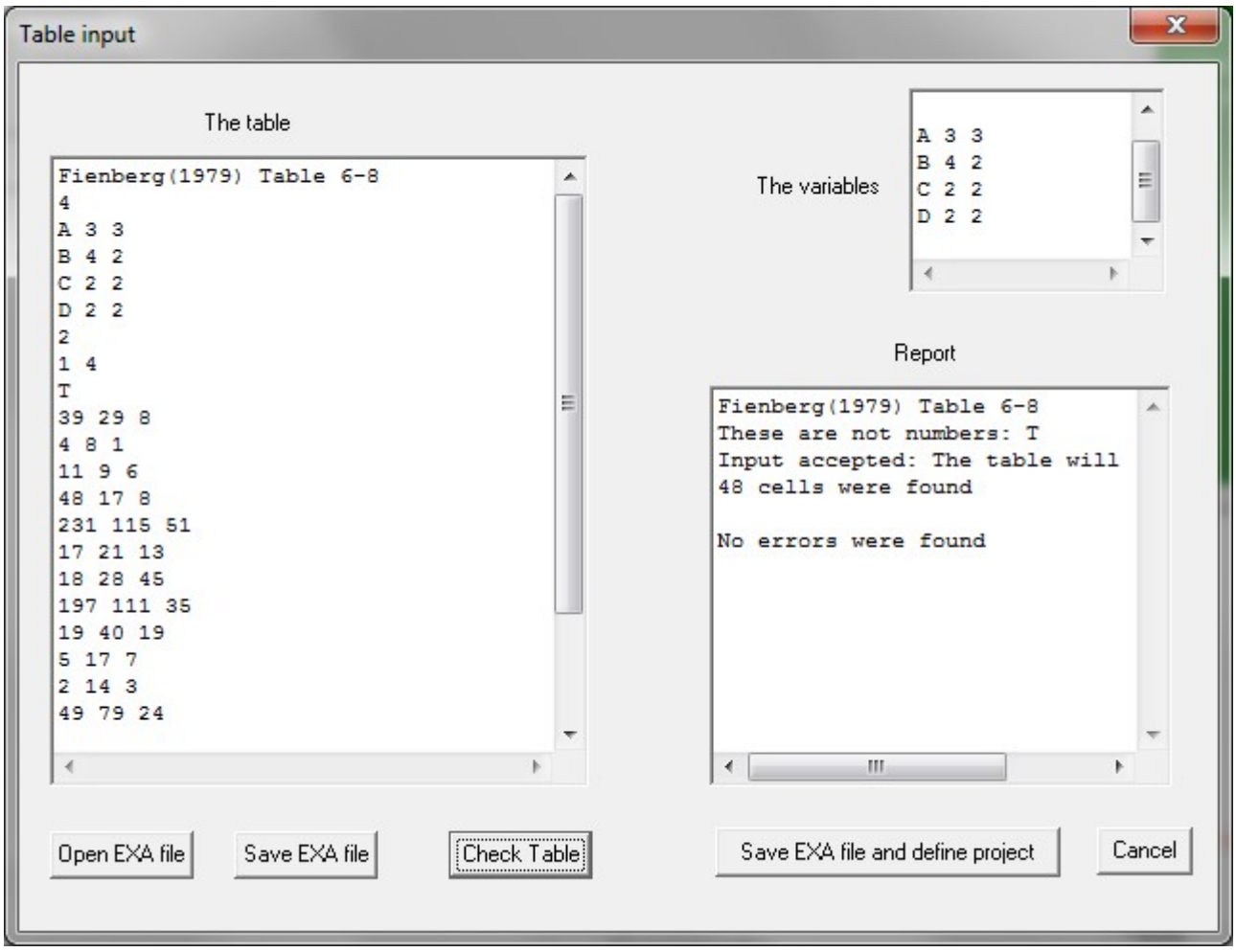


Figure 13.2 The dialog form needed to enter data for a table.

Enter the data as shown in Figure 13.3, check Table, and click on “Save EXA file and define projecy” to create the project. The info will be saved on Fienberg.exa and the data created. Note that the information in the first line is not copied to the VAR file and that you will have to define variable names and category labels yourself.

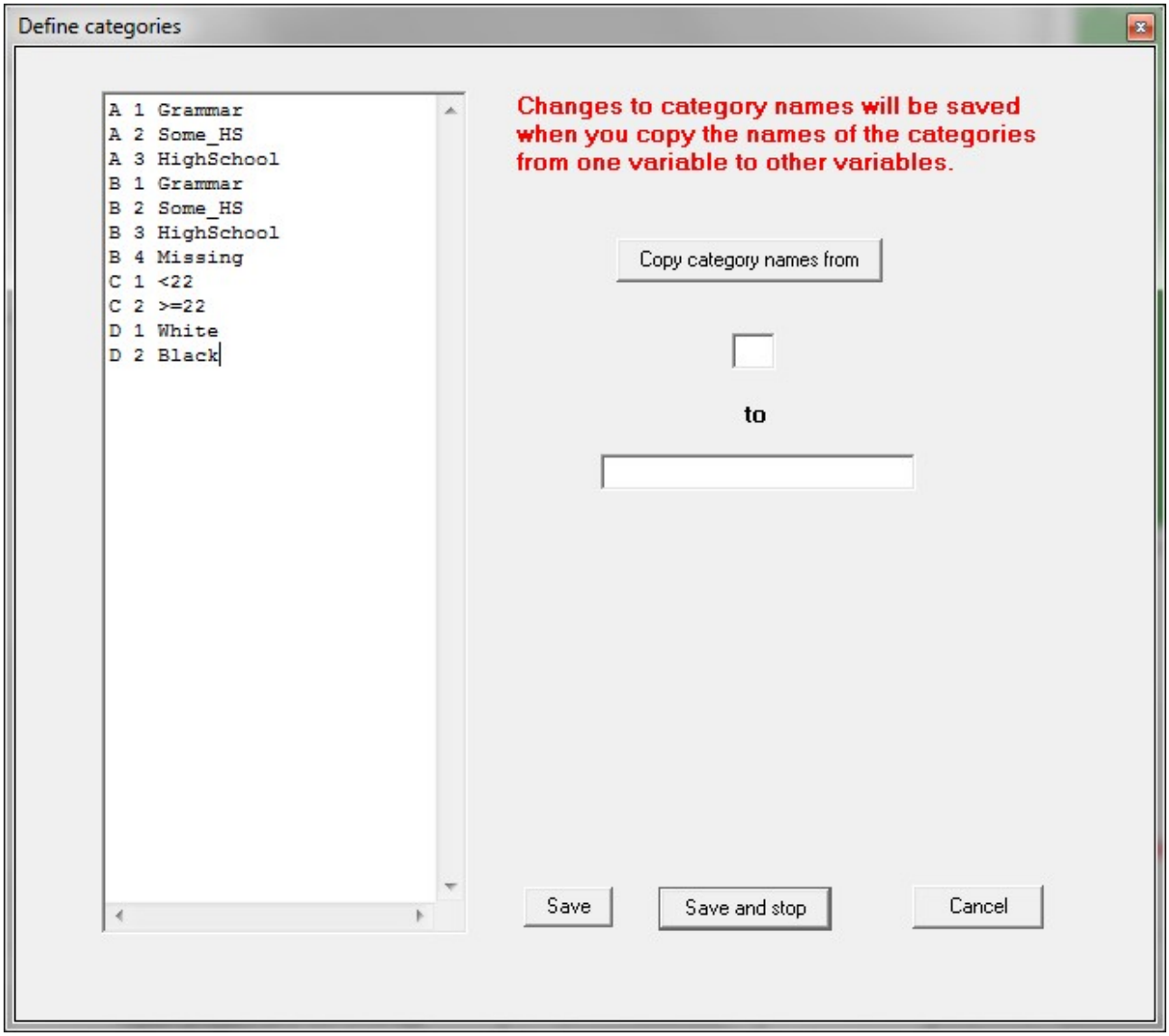


Figure 13.3 The dialog form with the table in Figure 13.1.

Finally, click “Save and stop” following which DIGRAM closes project you ar working on and opens the Fienberg-6-8 project and you will se (Figure 13.4) that DIGRAM assigns variable names VAR1, VAR2, VAR3, and VAR4 to the variables. The only thing you have to do on your own is to redefine the variable names following which you are up and running.

Fiigure 13.5 shows the project graph after the variable names has been corrected and after initial screening of data for an initial graphical model.

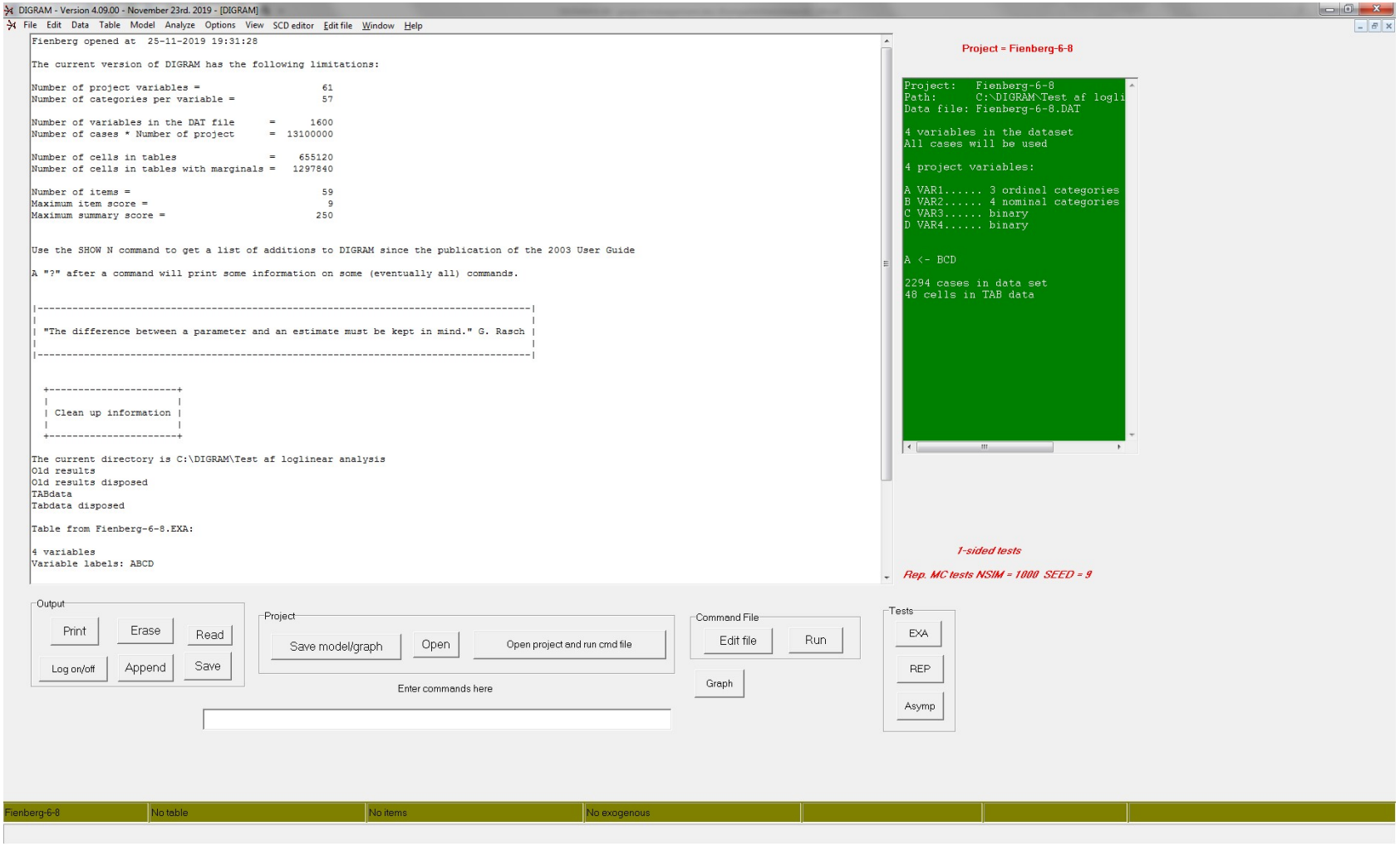


Figure 13.4. Information provided when the Fienberg-6-8 projects is opened.

Screening of the data creates the graphical model shov in Figure 13.4.

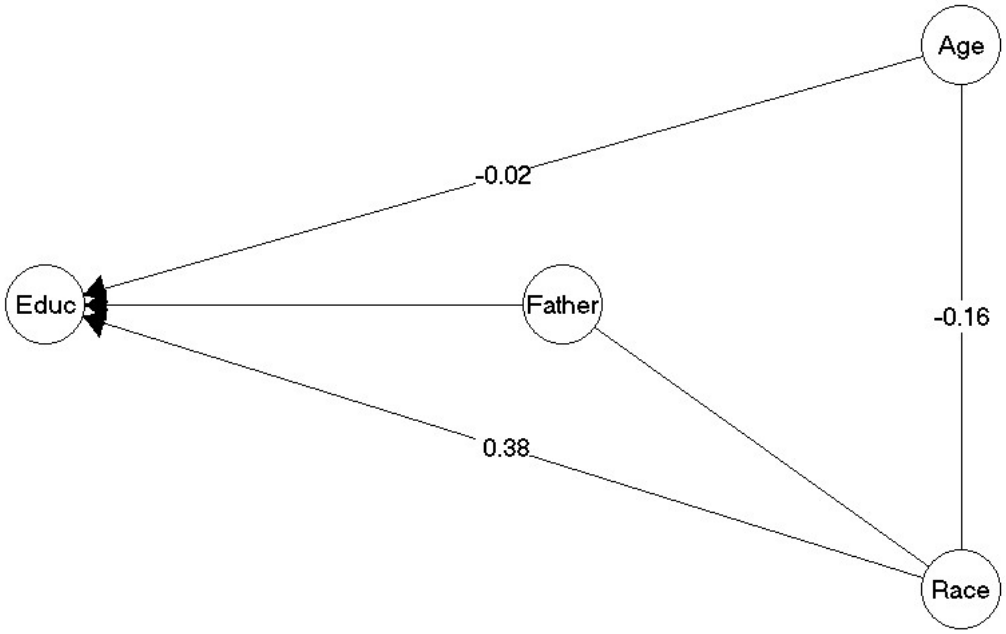


Figure 13.4 A graphical model for the Fienberg project defined by screening