Free2Fix

version 3.0

William D. Richards
School of Communication
Simon Fraser University
Burnaby, B.C. Canada
V5A 1S6
richards@sfu.ca

Free2Fix

Free2Fix is a utility program that accepts free-format data files and converts them to fixed-format files of the type used by MultiNet.

Input

The input file begins with a list of variables that looks like this:

```
variables
  ID, sex, age, height, job-status,
  language, occupation.
```

The first word on the first line should be "**variables**" (in lowercase). The word "variables" must begin in the first column of the line.

Subsequent lines contain names of variables, separated from one another by commas and spaces, like the words in this sentence. If all of your variable names don't fit on one line, end that line with a comma and start another with the next variable name. Use as many lines as you need. The program can deal with up to 200 variables per file.

Following the list of variable names, you may include a list of value labels. This part of the file looks like this:

```
value labels
  sex {1=female, 2=male}
  age {1='10 to 20', 2='21 to 30', 3='31 and above'}
```

The words "**value labels**" must appear first on a line by themselves, in lowercase, starting in the first column.

Each subsequent line begins with the name of a variable. The variable name must be one of the variables named in the "variables" part. Following the variable's name is a "{", which signifies the beginning of the value labels for the variable.

Each value of the variable is associated with its label by following the value with an "=" and then the label.

Ordinarily, a label ends with the first blank space or comma. This works well for values that have one-word labels. If you want to include spaces or commas in a value label, you must put the label in quotation marks. You may use either single quotes 'like this' or double quotes "like this". If you begin a label with a single quote, you must end it with a single quote. The ability to use either single or double quotes will allow you to have labels like "don't drink" or '"really hot"', which produce the following results: don't drink and "really hot".

All labels for a variable must appear on one line. Lines may be as long as 100 characters.

```
value labels
  sex {1=female, 2=male}
  age {1='10 to 20', 2='21 to 30', 3='31 and above'}
begin data
1 1 3
2 1 3
3 2 2
```

The variable labels part is ended by a line that contains only the words "**begin data**" in lowercase, beginning in the first column.

Your data appears after the "begin data" line. The one main restriction you will have to keep in mind is that you cannot use more than one line for a single case.

Put the values one after another in the same order that the variable names appear in the "variables" section at the beginning of the file. Separate each value from the next by spaces, commas, tabs, letters, semi-colons, or other non-numeric characters.

See the sample data file that came with the program "data.ex"

To run the program, type the program's name. It will ask you for the names of your data file and the output file. The program processes one line of variable names or value labels at a time. It displays each line on the screen so you can see what it is doing. It pauses and waits for you to press the enter key until it processes the next line.

Operation of the program is completely automatic, unless you have more than one set of value label settings for any variable. In this case the program will tell you what it found and tell you that it will ignore the second line of settings. It will pause and wait for you to press the enter key before it continues.

If you attempt to provide value labels for variables that do not appear in the list at the beginning, the program will tell you that it encountered an "undefined variable". So you can see where the problem is, it will display the line it is working on, and wait for you to press enter before it continues processing data.

If the program encounters more than 5 undefined variables, it will print a warning on the screen and stop processing.

The program reads through all the data in the file and determines how many digits each variable requires to represent the values in the file. It determines whether or not decimal points are needed, and constructs a format specification for your data. Then it determines which columns each variable will occupy and writes the output file.

If there are not enough values on a line of data, such as in the last two lines in the example below, the values that are present are used for the variables at the beginning of the variable list. The variables at the end of the list — the ones for which there are no values on the line — receive the value "0".

```
begin data                          begin data
  1 1 32          results in:       1  1 32
  2 1 25                            2  1 25
  3 2                               3  2  0
  4   26                            4 26  0
```

Note that the missing value in the fourth line caused "26" — the value that should have been the third one — to become the second one in the output file.

If you want to specify that a value is missing in your data and you don't want the value "0" to be assigned, you can put a "^" in the data file where the missing value would have been. An example is shown here.

```
begin data                          begin data
  1 1 32          results in:       1 1 32
  2 1 25                            2 1 25
  3 2  ^                            3 2
  4 ^ 26                            4   26
```

To see what the program does, look at the file named "data.ex". Run the program with data.ex as the input file

and data.out as the result file. When you compare the two files, you will see how the result is organized. An example file and the output produced for it are shown below.

input file

```
variables
id,sex,age,  height,
    credits weight,job
value labels
sex { 114 = male  27="female"
job {3=full-time, 2=part-time, 1=none}
age {1='21-30', 2='31-50', 3='51-up'}
weight {2=
begin data
1  1 23.5  62    104 134
2  1 22.34   56.126 115 120
3  2  24   124.45    118 195
4,1,^,1124.5,142,182
5m1k30l33.2k143k183
6     2     34    53    119
7     1     28    72    130 191
9,1,33,24.7,140,180
10/2/34/12345./14/181
```

output file

```
id                ( 1- 3)
sex               ( 4- 5)
/
27 female
114 male
/
age               ( 6-11)
/
1 21-30
2 31-50
3 51-up
/
height            (12-21)
credits           (22-25)
weight            (26-29)
begin data
 1 1 23.50    62.000 104 134
 2 1 22.34    56.126 115 120
 3 2 24.00   124.450 118 195
 4 1        1124.500 142 182
 5 1 30.00    33.200 143 183
 6 2 34.00    53.000 119   0
 7 1 28.00    72.000 130 191
 9 1 33.00    24.700 140 180
10 2 34.00 12345.000  14 181
```

Notes:

1. Spacing of variable names in variable list is not important ("id, sex, age, height").
2. Variable names may be separated by commas or spaces in the list ("credits weight, job").
3. Extra spaces around values or labels are ignored ("sex { 114 = male 27="female" 13 = na").
4. Value labels may be separated from one another by spaces or commas.
5. Incomplete value label specifications are ignored ("weight {2=").
6. The closing "}" is not necessary on a value labels line.
7. Data values may be separated by spaces, commas, letters, slashes, tabs, or other non-numeric characters.
8. The number of digits to the right or left of the decimal for a variable is determined by the value for that variable that has the most digits to the right or left of the decimal. For the "height" variable, the second value (56.126) had three digits to the right of the decimal, while the last value (12345.) had five digits to the left of the decimal.
9. If all values for a variable have no decimal points, the resulting variable will be treated as an integer and no decimal points will appear in the results for that variable. In the example, this was the case for "id", "sex", "credits", and "weight."
10. If the number of variable names listed in the beginning of the file is larger than the number of values on all lines of data, the "extra" variables (the ones for which there are no values) will not be listed in the output file. In the example above, there were seven variables named (id,sex,age, height, credits, weight,job), but the maximum number of values per line of data was only six. The last variable, "job", does not appear in the output file.
11. If a line of data does not have enough values to supply data for all the listed variables, the last variable(s) are given the value "0" for that case. See the sixth line of data in the example, where there is no value for "weight".
12. The symbol "^" in the data file means that there is a missing value, and that no value should be assigned to the variable in the output file. See the fourth line of data in the example, where there is a missing value for "age".