Specification of EpiData Template file structure

An EpiData Template is a flat text file containing structured text lines, which can be used to create the contents of a dataset.

Contents of the file is case independent, character set must be UTF-8 to be handled corrctly. There are only two types of lines in the file:

- Comment lines, which can be for descriptions of the contents, version information, documentation, web site reference or organisational information. Comment lines ALWAYS starts with a #
- Content information lines used to specify the file. These define sections, fields, valuelabels , headings and language used. A specifier must appear before it is used subsequently, therefore e.g. A section must be defined before used.

The sequence of section and field definitions in the template defines the sequential structure of the database created.

Content of the template file

Usually one would start with a few lines of comments indicating version and purpose etc.

```
# Epidata template file
# Version 2.2 created March 2nd 2010
```

Following this remaining lines would be either further comments or one of six specification types (d1 – d6). The title **must** always be used before any other – since this also contains the language used:

D1	title	This line contains the title of the project and must start with "title". Only one such line must exist.
D2	valuelabels	These lines specify a valuelabel for a set of valuelabels. Line MUST start with "valuelabel"
D3	sections	These lines specify a named section. Line MUST start with "section" and cannot specify the main section – it is always present.
D4	fields	These lines specify a single field and its attributes, this includes eg. a valuelabel and the associtated section. Line MUST start with "field".
D5	headings	These lines specify a single heading and the associated section. Line MUST start with "heading".
D6	set	These lines modify the content of a field and adds/changes attributes for the field. Line MUST start with "set".
D7	translate	These lines specify a translation for a given text. It is possible to translate a section, a heading, a field, and individual valuelabels. Line MUST start with "translate"

Each of these have a special format, which MUST be followed. Parts of the specification are separated with tab.

D1: Title

This line specifies the default language of project file and defines a title for the project. Only a single line may be specified within the file and it MUST be the first non-commented line in the file (ie. no other specification is allowed before this line).

format:

```
"title" <"language ident"> <"Title for the project">
```

Example:

```
"title" "en"    "This is my first EpiData template file"
```

D2: Valuelabels

These contain definitions for a single value label in the project. A value label gives an explanation for a value entered, e.g. 1 for “Female”. All value labels with the same name is called a “value label set”. The first time a given name is used a new value label set is defined.

All value labels used in any field definition must be defined in the same template file.

Format:

```
"valuelabel" <"set name"> <"type"> <value> <"label text for the value"> [missing]
```

Where “valuelabel” is the keyword, name is specified by the user, value is the desired value and missing is a keyword indicating that this value is used for missing. <type> is the same specifier as used for field definitions - only three types are allowed i: integer, f: float, s:string.

Example:

```
"valuelabel"    "yn"    "i"     1       "No"
"valuelabel"    "yn"    "i"     2       "Yes"
"valuelabel"    "yn"    "i"     8       "Irrelevant"    "missing"
"valuelabel"    "yn"    "i"     9       "Unknown"       "missing"
```

Which would specify the value label ”“yn”” with the values 1,2,8 and 9 with the texts shown and values 8 and 9 defined as missing values. The “i” indicates the integer type. If a string type is define the <value> MUST be written in quotes.

D3: Sections

These lines specify a single logical section placed on the main section. Each line can specify a unique caption and a width of the section. The caption of the section can later be used in the definition of fields to indicate where a given field is placed. The height is calculated automatically based on the number of fields and heading lines contained.

Format:

```
"section"   <"caption"> <width>
```

Example:

```
"section"   "Dairy"  500
```

D4: Fields

These contain the definition for a single field in the project. Each field is specified with a type, assigned to a section, given a name and accordingly a question text. Optionally a field can be associated with a set of valuelabels and whether to show the valuelabel on the form. If not set of valuelabels should be associated with the field, then leave it blank.

Format:

```
"field" <"section caption"> <"fieldtype*"> <format> <"variable name"> <"question"> [<"valuelabel name"> [”show
```

Fieldtype	Is a character defining the type of the field. The list of characters are:
	
	basic data fields:
	i = Integer
	f = Float
	s = String
	u = Uppercase String
	d = Date (DMY)
	m = Date (MDY)

	y = Date (YMD)
	t = Time
	b = Boolean
	
	Fields assigned a value automatically:
	a = Auto Increment
	n = Auto Date (DMY)
	o = Auto Date (MDY)
	p = Auto Date (YMD)
	z = Auto Time

Format	Define the length (or integers + decimal) of the field. For fields with a predifined length (time, date, etc.) set to 0. Float format descriptions contain three parts, first the number of digits before the period, then a period and then the number of digits after the period. A float cannot have 0 decimals. Only integers can have 0 decimal digits.
Valuelabel name	
“show”	If specified on the line, the label associated with the entered value (during entry) will be shown right of the field.

If the same name specifier is used for several fields these will be renamed with an integer following the part being the same, e.g. If more fields have the name “s” they will be named s1 s2 s3 etc. Fields with empty name will be named v1 v2 v3 ….

Examples: A simple integer of width 1 with valuelabel “yn”:

```
"field" "main" "i" 1    "v1" "I = integer"      "yn"    "show"
```

Example of float and string are similar:

```
"field" "main" "f" 3.2  "v2" "f = Float"
"field" "main" "s" 12   "v3" "question type s = String"
```

D5: Heading lines

These lines define a single heading in the project. Headings provide no function and can only be used to display texturial information to the user of the project. The <name> is only used to facilitate translation, see below for details.

If the same name specifier is used for several headings these will be renamed with an integer following the part being the same, e.g. If more headings have the same name “s”, they will be named s1 s2 s3 etc. Headings with empty name will be named h1 h2 h3 ….

Format:

```
"heading"       <"section"> <"name"> <"caption text">
```

Example:

```
"heading" "main" "" "This is my first project!"
"heading" "demography" "" "Please enter the name:"
```

D6: Set

These lines does not define a new entity for the dataset, but is used as a mean to add or change properties for a field.

Format:

```
"set"   "field" <"name"”>       <”set-command”> ....
```

The format is unlike the other constructs, since each set-command takes different argument. The different set-commands are:

Set: "confirm"

Sets the attribute “Confirm Field” on the field.

Format:

```
(none)
```

Example:

```
"set"   "field" "V1"    "confirm"
```

Set: "entrymode"

Defines the field attribute EntryMode.

Format:

```
”musteenter" / "noenter”
```

Example:

```
"set"   "field" "V1"    "entrymode" "mustenter"
```

Set: "range"

Defines a valid entry range for the field.

Format:

```
<min.-value> <max.-value>
```

Example:

```
"set"   "field" "V1"    "range" 0       50
```

Set: "jump"

Defines a single jump for a field. It is possible to define several jumps (with multiple set lines), making it possible to jump to different locations based on different values.

Format:

```
<jump-value> <jump-type> <reset-value>
```

- *Jump-value*: The value which activate a jump. Must be the same type as the type of the field.
- *Jump-type*: Must be one of the following
    1. “skipnext”: Skips the next record.
    2. “exitsection”: Jumps to the field/section following immediately after the current section. If defined for a field on the main section, this is just like “saverecord”.
    3. “saverecord”: Immediately saves records, performing the “reset-value” on all following fields.

It is NOT possible to define a jump to another field. This can only be done in the EpiData Manager program.

- *Reset-value*: Must be one of the following:
    1. “sysmissing”: Skipped field are reset to the system missing (blank).
    2. “maxmissing”: Skipped fields are reset to the maximum defined missing value for the fields. If no missing value is defined (done in a valuelabel set) for a field, the fall-back default is

“leaveasis”.

3. “2ndmissing”: Equivalent to “maxmissing”, except the chosen missing value is the 2nd max define missing value. Fall-back defaults to “leaveasis”.
4. “leaveasis”: Skipped field are left untouched. Data already entered is not reset to another value (or left blank).

Examples:

```
"set"   "field" "V1"    "jump"          0       "skipnext"      "sysmissing"
"set"   "field" "V1"    "jump"          1       "exitsection"   "maxmissing"
"set"   "field" "V1"    "jump"          2       "saverecord"    "2ndmissing"
```

D7: Translation

These lines define a translation to a text in a previous definition. The construct of these lines are somewhat different to the previous definition in the sense that here information is added to already existing definitions, but no new definition is given.

It is possible to translate the caption of a section, the “question” of a field, the caption of a heading and labels of a valuelabel.

Format:

```
"translate"     <"valuelabel/section/field/heading"> <"name identifier"> [value] <"language ident"> <"Text of
```

The special contruct of [value] is to specify which label should be translated and is only used when translating value labels.

Example:

```
"translate" "field"     "V1"                    "da"    "Alder på personen"
"translate" "heading"   "H1"                    "fr"    "Âge de la personne"
"translate" "section"   "Demography"    "de"    "Demografie"
"translate" "valuelabel"        ""yn""  "0"     "cn"    "无"
```

Complete sample template file

This file defines the structure of the Bromar – marathon sample data file released with EpiData Analysis.

```
# Epidata sample template file
# Version 1 created 2011-03-15
"title" "en"    "Bromarathon data - example project"
"valuelabel"    "yn"    "i"     1       "Yes"
"valuelabel"    "yn"    "i"     2       "No"    "missing"
"valuelabel"    "yn"    "i"     9       "Unknown"       "missing"

"valuelabel"    "fm"    "i"     1       "Female"
"valuelabel"    "fm"    "i"     2       "Male"
"valuelabel"    "res"   "i"     1       "0-25 km"
"valuelabel"    "res"   "i"     2       "26-62 km"
"valuelabel"    "res"   "i"     3       "63-120 km"
"valuelabel"    "res"   "i"     4       "120+ km"
"valuelabel"    "res"   "i"     5       "Copenhagen"
"valuelabel"    "res"   "i"     6       "East Denmark"
"valuelabel"    "res"   "i"     8       "Other Country" "missing"
"valuelabel"    "res"   "i"     9       "Unknown"       "missing"
"valuelabel"    "rest"  "i"     9       "Unknown"       "missing"
"valuelabel"    "hours" "i"     0       "2.5-3.4"
"valuelabel"    "hours" "i"     1       "3.5-3.7"
"valuelabel"    "hours" "i"     2       "3.8-4.1”
"valuelabel"    "hours" "i"     3       "4.2-6.2"
# Notice that keywords (here "missing") are NOT case sensitive.
"valuelabel"    "hours" "i"     9       "Unknown"       "Missing"

# Now fields are defined sequentially.
"field" "main"          "i"     4 "id"  "Runners number during the Marathon"
```

```
"set"   "field" "id"    "range" 1 4300
"field" "main"          "i"     4 "km"  "Distance from residence (km)"
"set"   "field" "km"    "range" 0 800
"field" "main"          "i"     4 "kmgrp"       "Distance from residence (grouped km)"  "res"
"set"   "field" "kmgrp" "range" 0 800

# New section with demography content.
"section"       "demography"    400
"field" "demography"    "i"     4 "sex"  "Sex of particpant"  "fm"
"field" "demography"    "i"     4 "born" "Year of birth (4 digits)"
"set"   "field" "range" 1930 1985

# Results section
"section"       "results"       450
# float variables indicate number of digits before and after decimal point.
# format specifications must be using period (.) to describe the lengths
"field" "results"       "i"     1 "run" "Completed race ?" "yn"
# If person did not complete race, remaining fields should be cleared.
"set"   "field" "jump" 8 "saverecord" "sysmissing"

"field" "results"       "f"     1.4 "runtime"   "Time in hours, minutes and seconds"
"field" "results"       "f"     1.4 "dectime"   "Completion time (decimal)"
"set"   "field" "range" 2.0 7.0
"field" "results"       "i"     1 "decgrp"      "Completion time (grouped)"             "hours"
```

EpiData Command Line Tool (Template Usage)

The EpiData Command Line Tool[1], in short "epidatacmd", is the tool to use when transforming template files into EpiData Project files (.epx files).

For Windows®m users the command is best envoked using the "Command Prompt" or through a batch script (.bat files) using "epidatacmd template" On Linux either place the program in a directory within your PATH environment or envoke from a terminal using "./epidatacmd template"

It is important to notice that the parameter "template" is added, this is because the Command Line Tool in the future may support many other operations, hence it is necessary to specify what type of operation you wish to perform.

Syntax

The general use of the Command Line Tool is as follows

```
epidatacmd template -i <input-file> -o <output-file>
```

Where <input-file> is the user-defined template and <output-file> is a filename (with .epx extension) that the parsed template should be saved as.

Example

```
epidatacmd template -i cold_food_template.txt -o cold_food.epx
```

This parses the template named "cold_food_template.txt" and produces a EpiData Project file with the name "cold_food.epx"

Additional Options

It is possible to specify a number of options on the command line. These options controls various parts of the parsing and generation of the project file.

All options in the table below MUST be prefix with two dashes, ie. "–" else the program will fail. For instance to specify a new "fieldleft" type:

```
epidatacmd template ... --fieldleft=250
```

Option Name	**Options Value**	**Explanation**	**Default**

sectionleft	Integer	Left position (in pixels) of a section	5
fieldleft	Integer	Left position (in pixels) of a field	300
headingleft	Integer	Left position (in pixels) of a heading	10
fieldfielddist	Integer	Vertical distance (in pixels) between fields	10
fieldheadingdist	Integer	Vertical distance (in pixels) between field and heading	25
headingheadingdist	Integer	Vertical distance (in pixels) between headings	5
hidefieldnames	(none)	Hide all field name on dataform	
backuponclose	(none)	Activates backup feature when closing Manager/EntryClient	
showvaluelabels	(none)	Shows a label besides ALL fields with an associated valuelabel set	

[1] Currently NOT available to MAC users

documentation/templateformat.txt · Last modified: 2011/06/07 12:12 by torsten.christiansen