NaPiRE 2018 Survey: Coding Instructions

Introduction

Generally, there are two types of questions allowing free text answers (**free text questions**):

- questions expecting short free text answers (**short free text questions**; asking for a specific information item), and
- questions expecting long free text answers (**long free text questions**; asking for reasoning/elaboration).

The **coding procedures** to be followed depend on the type of question concerned, and they are detailed below. **Examples** can be found on page 3, the **variable assignments and processing modes** are specified on page 4 and page 5 (using your initials). Further information is provided in the HTML files contained in the **supplement** folder. Please direct any **questions and comments** to:

Corinna Coupette (corinna.coupette@campus.lmu.de)
Daniel Mendez (daniel.mendez@tum.de)

Caution: The csv files mentioned below are distributed with semicolons as separators and double quotes as quotechars so that they can be readily read by the standard Excel configuration, and performing the validation steps in Excel will likely be easier for you than working with the csv files directly. If you desire a different format, please contact us.

Short Free Text Questions

Short free text questions can be grouped into two categories:

- **List-supplementing questions:**
  Respondents are shown a list of options in a previous question. They choose "Other (please specify)" and enter a short free text.
- **List-supplanting questions:**
  Respondents are asked to enter a short free text directly, with no prior list of options or guidance regarding what values to specify.

Both types of questions are preprocessed using similar procedures, and the coding of responses should normally result in the **assignment of exactly one code (also referred to as a tag) to each answer**. The only **exceptions** to this basic rule in the current survey are **v_20**, which asks respondents about their RE certifications, and **v_194**, which asks for RE problems respondents missed in the problem list.

Code Assignment

For one individual short free text question, start with one csv file containing three columns (**code file**):

1. the response identifiers ("lfdn"),
2. the responses ("v_<variable_number>"), and
3. the tags ("tag", empty).

For **list-supplementing questions**, there are two processing modes (see *01_prepare_short_vars.html* for the rationale behind this):

1. **"Code Back"**:
   a. For each row in the code file, **check** whether the response provided fits one of the codes from the list of options provided in the previous question;
   b. If you find a suitable pre-existing code, **assign** that code; else assign **NotCodeable**
2. **"New Tags"** (this is "Code Back" with a different else branch):
   a. For each row in the code file, **check** whether the response provided fits one of the codes from the list of options provided in the previous question;
   b. If you find a suitable pre-existing code, **assign** that code;
   c. For the responses remaining without codes:
      i. **Group** the remaining responses by their content;
      ii. **Find** names for the groups that are similar in structure and level of abstraction to the pre-existing tags;

iii. **Assign** all responses in the individual groups the name of their group as a code.

If an answer is empty (e.g., " ", ".", or "x"), assign the code **NotAnswered** (not the code NotCodeable).
For **list-supplanting questions**, the only processing mode available is analogous to **step c.** from the "New Tags" option detailed above:

1. **Group** all responses by their content;
2. **Find** names for the groups (reflect terminological distinctions made in the research community);
3. **Assign** all responses in the individual groups the name of their group as a code.

For **v_20** and **v_194**, if you want to assign **multiple codes** to an answer, **copy and paste the row** holding the answer as many times as needed to have one row per tuple ("lfdn", "v_<variable_number>", "tag"). **Do not** make lists of tags in a single row.

If you want to **organize** your new tags in a **taxonomy**, please follow the **tag format** and **communication procedure** specified for the long free text answers.

Long Free Text Questions

As the coding of long free text questions is different from the coding of short free text questions, we will generally refer to it as **tagging**. Tagging differs from coding in two important respects:

- The tags are not a set of disjoint concepts but rather organized as a **forest of facets** (a hybrid of the facet and tree taxonomies); each facet is represented by a hierarchical tree of tags, which may be several levels deep.
- **More than one tag may be assigned** to each answer (and this is the rule rather than the exception [as for the short free text answers]).

For the purposes of storage, tags will be Strings with the following structure:
<level1>:<level2>:<level3>_<level4>, where all elements but the first level identifier are optional and where – for the purposes of this survey – a hierarchy no deeper than two levels is generally desirable.
Examples of tags:

- **people:communication** (to characterize an RE problem's cause);
- **text:quantified** (to characterize the documentation of an NFR requirement).

Tag Assignment

For one individual long free text question, start with one csv file containing three columns (**code file**):

1. the response identifiers ("lfdn"),
2. the responses ("v_<variable_number>"), and
3. the tags ("tag", empty).

Note: This general structure is slightly different where respondents' answers depend on answers to previous questions (in this case, these answers are included for your convenience, and some columns have different names).

To assign tags to the responses for a single long free text question, you generally follow the procedure for list-supplanting short free text questions:

1. **Group** all responses by their content;
2. **Find** names for the groups (reflect terminological distinctions made in the research community);
3. **Assign** all responses in the individual groups the name of their group as a tag.

Note: The tag assignment process will likely be iterative, and it will result in a taxonomy which can be represented visually (likely: as a collection of trees for different facets, i.e., a forest of facets). For each tag taxonomy you design, please include a visual representation in the ZIP archive you return to us.

In some cases, **several variables** containing long free text answers are best processed **simultaneously**. Where this is the case, you will find a file with **overview** in its name amongst the files in your folder. The purpose of this file is merely to help you craft a taxonomy that is consistent across the related variables with minimum effort – **do not** add tags in the overview file.

Once You Are Done

Please **zip** all your completed code files (along with visual representations of any taxonomies, see below) and **email** them to:

Corinna Coupette (corinna.coupette@campus.lmu.de)
Daniel Mendez (daniel.mendez@tum.de)

Examples of Coding and Tagging Procedures

```
In an imaginary survey, respondents were asked for personal information.

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

List-supplementing short free text answers - Code Back Mode:

v_5 asks respondents to name favorite colors, and the options list included red,
blue, pink, purple, etc., plus: Other (please specify).

Respondents 1 and 2 answered "Other (please specify)", their free text responses are
"magenta" and "violet". Respondent 3 answered "it depends on my mood".
After your processing, the code file will look like this:

lfdn;v_5;tag
1;magenta;pink
2;violet;purple
3;it depends on my mood;NotCodeable

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

List-supplementing short free text answers - New Tags Mode:

v_2 asks respondents to specify their race; with the standard options available.
A non-negligible number of respondents answered "Other (please specify)" and specified
variants of "Hobbit", "Elb", or "Ork", whereas others specified "Dunedain".
You decide to group the Dunedain with the Caucasian Whites but add new
codes for the inhabitants of the Shire and the like.
After your processing, your code file could look like this:

lfdn;v_2;tag
1;hobbit;Hobbit
2;Dunedain;Caucasian White
3;Elb;Elb

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

List-supplanting short free text answers (similar process for long free text answers):

v_7 asked respondents to list their skills, and no options were provided to choose from.
Having looked through the answers, you decide to design a small hierarchy of tags,
differentiating, inter alia, between character traits and magic capabilities.
For respondent 4, you thus assign two tags:

lfdn;v_7;tag
4;"Going invisible; persistence";magic:invisibility
4;"Going invisible; persistence";character:persistence
```

Variable Assignments (Short Free Text Variables)

Variable	Type (if not Supplementing)	Answers	Suggestion	Responsibility
Variable 2		87	`New Tags`	DMF
Variable 3	Supplanting	488	(DONE) `New Tags`	CC
Variable 5		29	`Code Back`	DMF
Variable 15		5	(with `v_14`) `Drop`	-
Variable 18		40	`Code Back`	MF
Variable 19	Supplanting	488	(DONE) `New Tags`	CC
Variable 20	Supplanting	403	`New Tags`	JC
Variable 22		13	`Drop`	-
Variable 35		16	`Drop`	-
Variable 46		24	`Drop`	-
Variable 52		18	`Drop`	-
Variable 60		27	`Code Back`	JC
Variable 67		13	(with `v_66`) `Drop`	-
Variable 81		15	(with `v_80`) `Drop`	-
Variable 96		22	`Drop`	-
Variable 105		11	(with `v_104`) `Drop`	-
Variable 113		43	(also `Drop v_112`) `Code Back`	MF
Variable 164		16	`Drop`	-
Variable 166		18	`Drop`	-
Variable 194	Special	191	`New Tags`	DMF, MK

Variable Assignments (Long Free Text Variables)

Variable	Content	Answers	Suggestion	Responsibility
	Reason Bad/Good Relationship			
Variable 26		3	`Read`	DMF
Variable 27		102	`Code`	DMF
	RE (Dis)Satisfaction		`Code`	DG
	RE Dissatisfaction			
Variable 168		92		"
Variable 169		195		"
Variable 170		91		"
	RE Satisfaction			
Variable 171		93		"
Variable 172		185		"
Variable 173		178		"
	Top Problems: Causes and Effects		`Merge` with `v in range(246,255,2)`, `Sort` (both CC, DONE), then `Code`	DMF, MK
	Top Problems: Causes			
Variable 277		365		"
Variable 278		344		"
Variable 279		322		"
Variable 280		296		"
Variable 281		274		"
	Top Problems: Effects			
Variable 282		334		"
Variable 283		316		"
Variable 284		300		"
Variable 285		274		"
Variable 286		262		"
	Unaddressed Aspect		`Later`	DMF, MK
Variable 297		148		"
	NFR Documentation		`Code`	DMF, MK
Variable 343		129		"
Variable 344		126		"
Variable 345		184		"
Variable 346		61		"
Variable 347		132		"
Variable 348		69		"
Variable 349		173		"
Variable 350		188		"
Variable 351		10	`Drop`	-