Regional Training Course on Computer Assisted Personal Interviewing (CAPI) For Agricultural Surveys and Price Reporting

Bangkok, Thailand

Survey Solutions: Advanced Designer (Overview)

Sergiy Radyakin
sradyakin@worldbank.org

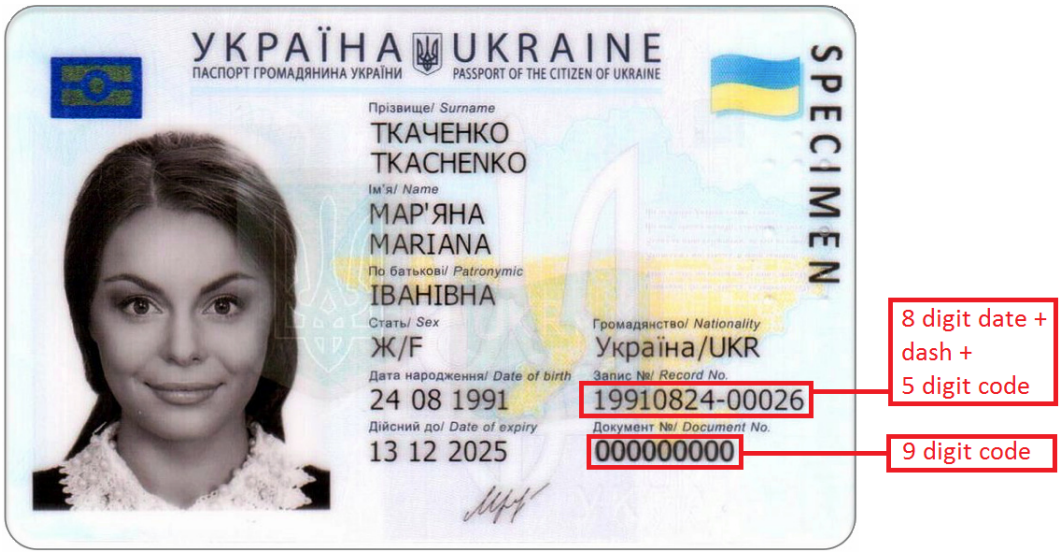
Development Data Group (DECDG),
The World Bank

October 3-7, 2016

Masking

- Masked edit extends functionality of the text question;
- Masked edit allows specifying a pattern (mask) for data entry;
- The mask (pattern) reflects the format of the value to be entered;
- Typical use is entering various ID numbers, phone numbers, etc.

Example



- These two identifier numbers may not be entered as “numeric” type.
- The first ID in the example is a composite ID comprised of the date code and unique code combined with a dash in the middle.
- Second ID requires leading zeroes. Numeric questions will not retain leading zeroes!

Example 2

- This passport number is not a number to be entered as "numeric" type.
- It consists of two letters and six digits. Letters may not be entered into a numeric field.

Mask structure

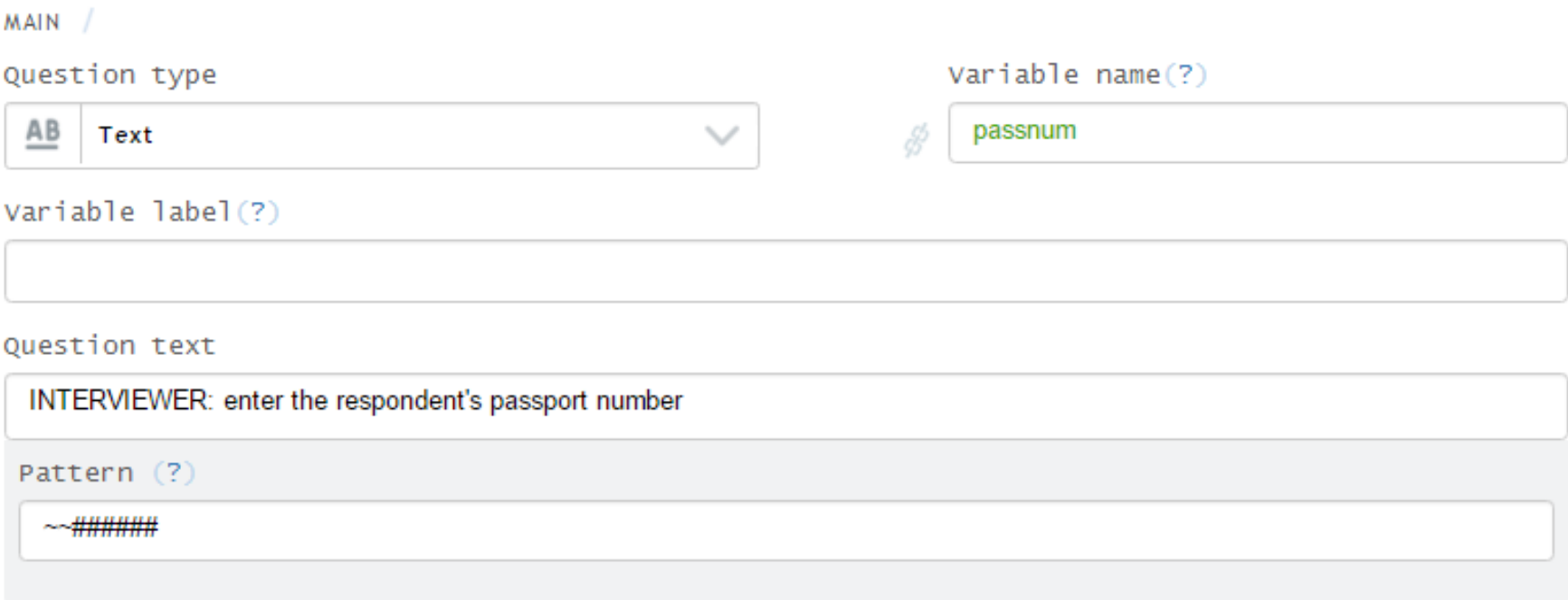
- Rules for mask specification:
  - a tilde (~) represents one alpha character (A..Za..z);
  - a hash sign (#) represents one numeric character (0..9);
  - a star (*) represents one alphanumeric character in that position;
  - any other characters entered in the pattern will be visible on the screen during data entry, while the mask characters will be rendered as underscores (_).
- So the patterns appropriate for the examples above are:
  - for the record number: ######-######
  - for the document number: #########
  - for the passport number: ~~######

Example

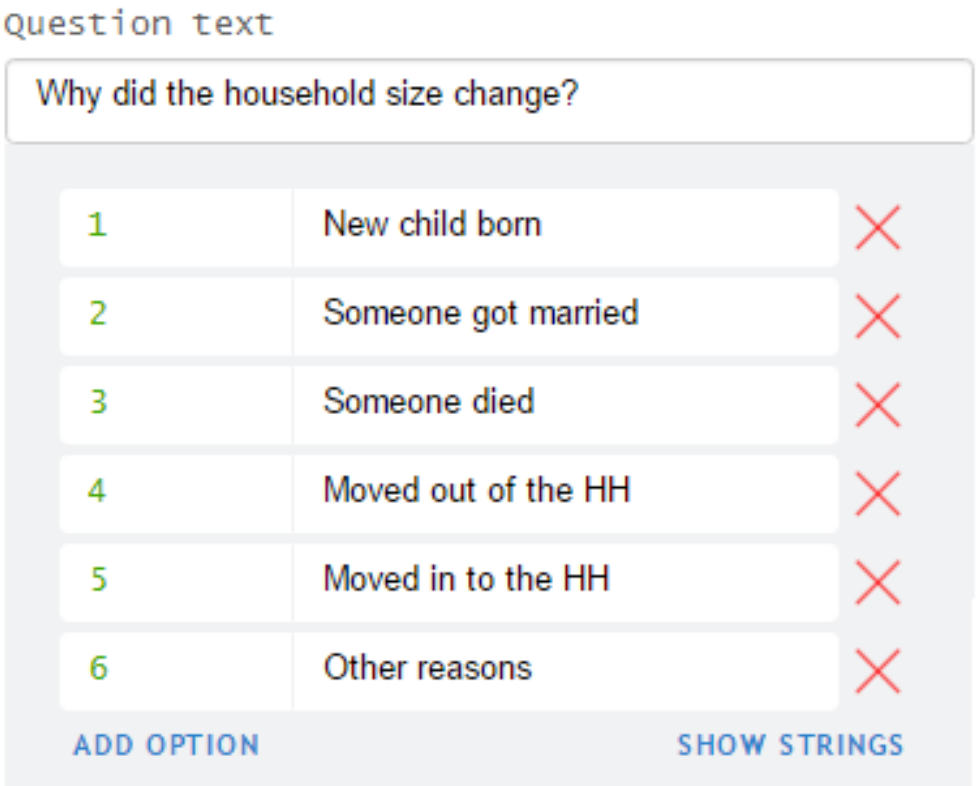
- Note that:
  - pattern enforces fixed length of content;
  - pattern enforces "hard" validation.
- If using it for e.g. phone numbers, make sure all (potential) phone numbers are of the same length.

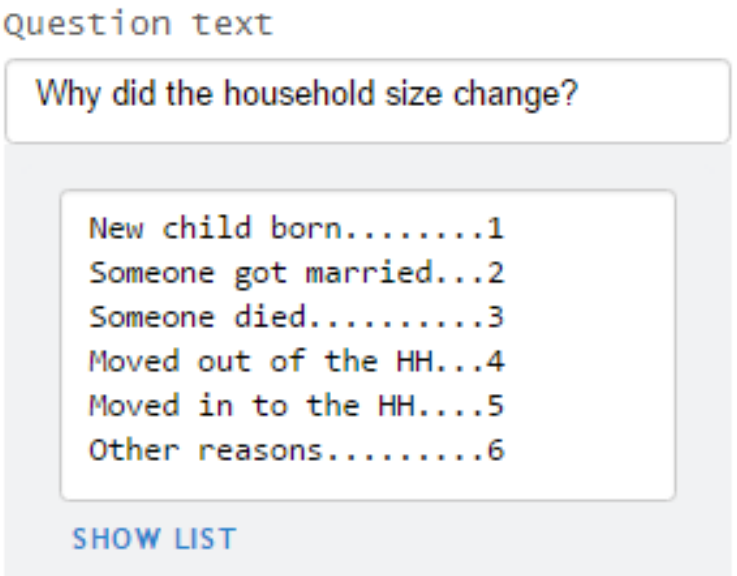
  For example: "*(###) ###-####*" will allow to enter typical USA phone numbers (10 digits) but will not allow entering foreign numbers.

Setting Pattern in Designer



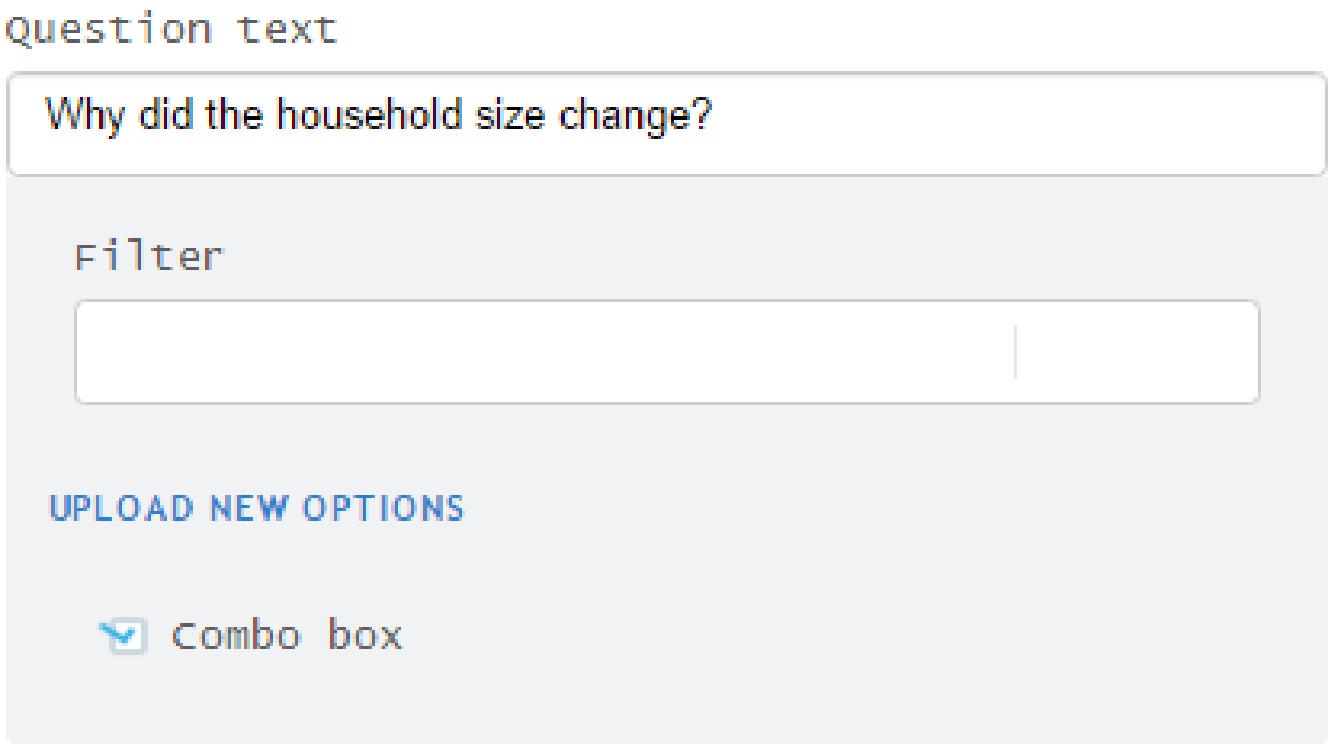
Entering long lists of options





Switch between the alternative views of options with the *"show strings"* and *"show list"* buttons to add/edit them in the most convenient way.

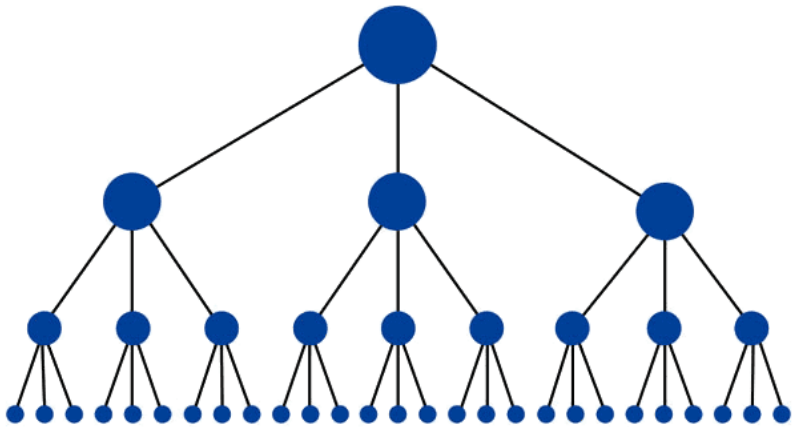
Entering long lists of options



When options are already typed and saved in a file, switch to *combobox* to uload them all together as a file, then switch to another view if desired.

Cascading selection

Cascading selection



- Cascading selection is used when the number of items is large and there is a certain hierarchy among them.
- For example, administrative divisions:
  - State
  - County
  - Town
- Other examples: industries, occupations, products and goods classifications, etc.

Filtered Categorical Questions

- Categorical questions may have filtering condition;
- For every option the program will decide whether it must be available for selection or suppressed;
- Allow preventing the error (compare to validation);
- Filtering condition may refer to any of the previously asked questions:

  ```
  (age>15) || (@optioncode<4)
  ```

  Here options 4 and above will only be available for respondents aged more than 15.
- Especially useful in categorical linked questions, e.g. allow listing all persons eligible based on some criteria: for example listing all females older than 15:

  ```
  (age>15) && (sex==2)
  ```

Lookup tables

Lookup tables:

- allow storing (invariable) reference information;
- used in expressions;
- typical use:
  - conversion between alternative or from non-standard units;
  - currency conversions;
  - caloric content information;
  - etc.

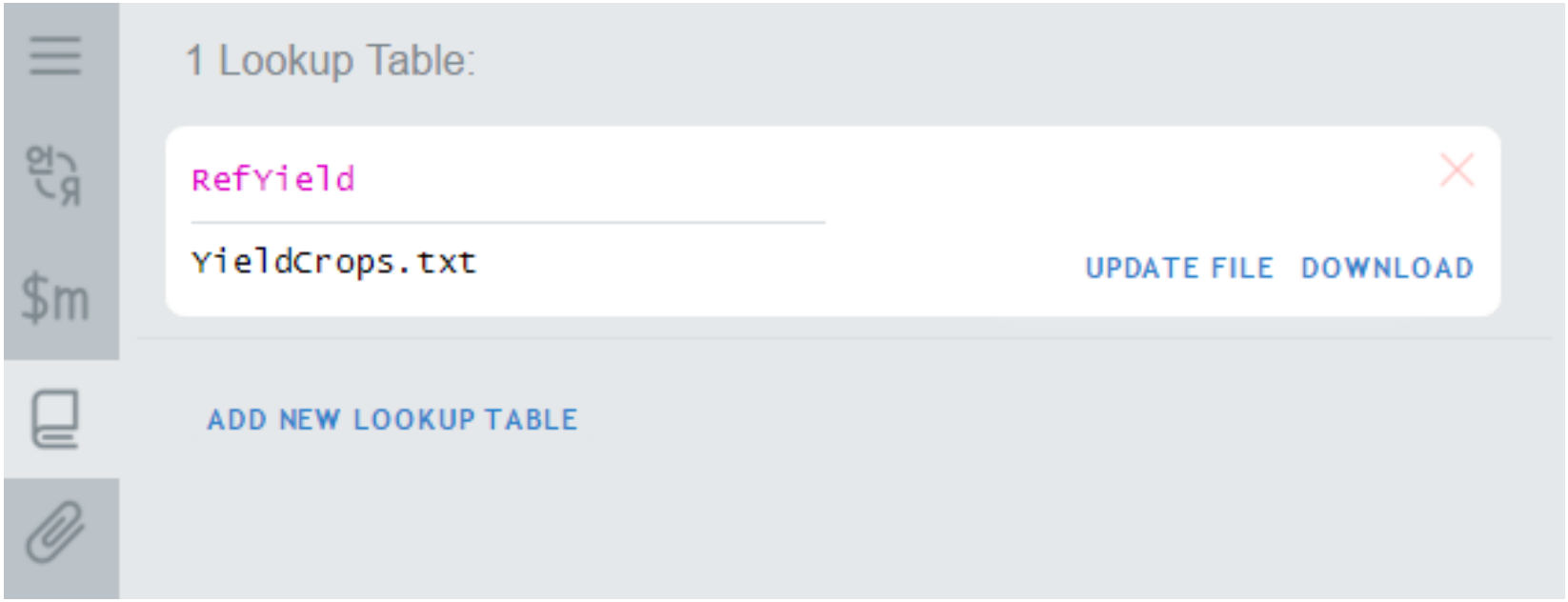
NB: for WHO z-scores use the built-in Z-scores functions.

Structure of a Lookup Table

rowcode	$colname_1$	$colname_2$		$colname_k$
C_1	$V_{1,1}$	$V_{1,2}$		$V_{1,k}$
C_2	$V_{2,1}$	$V_{2,2}$		$V_{2,k}$
C_m	$V_{m,1}$	$V_{m,2}$		$V_{m,k}$

m is up to 5,000;
k is up to 10.

Setup of Lookup Table in Designer



Example

Addressing a value in a lookup table:

```
RefYield[crop].Yield
```

Use of a lookup table reference values in expressions (here to validate the entered yield is within a 10% interval around the reference yield taking into account irrigation status):

```
 (irrStatus==1 && yield.InRange(
RefYield[crop].Irrigated*0.9,
RefYield[crop].Irrigated*1.1))
||
(irrStatus==2 && yield.InRange(
RefYield[crop].Dry*0.9,
RefYield[crop].Dry*1.1))
```

Hidden questions

- Hidden questions are containers to store information for use in enabling conditions and validation;
- Not visible to interviewers;
- Can't be modified by either interviewers or supervisors;
- Can be revealed (if necessary) through text piping.
- Typical use: store previous month prices for validation of current prices (e.g. 10% band around last month price).

Macros and Variables

- simplify complex expressions by breaking them down to smaller parts;
- simpify creation of large questionnaires with repetitive condition;
- hint on the meaning of expression by their name (or description);
- simplify changes to parameters and definitions in the survey;
- may be used to hold survey parameters;
- not exported!

Macros and Variables

Macros	**Variables**
are referred to by *$name*	are referred to by *name*
don't have to be syntactically correct	have to be syntactically correct
fragments of expressions	expressions
don't have a declared type	have a declared type (one of the following: string, long?, double? DateTime?, Boolean?)
computed every time when encountered	computed once every evaluation cycle
have descriptions	don't have descriptions
may not refer to other macros	may refer to other variables
may not be used in text piping	may be used in text piping

Functions

- Functions are syntax elements for performing common data transformation tasks.
- Some functions are standard C# functions, some are written specifically for Survey Solutions.
- Survey Solutions functions include string and list manipulation functions, Z-score functions, date, distance, and other functions.
- See full list online in the documentation.

Functions: Example

For example, to calculate the age of a person in years:

```
FullYearsBetween(borndate, interviewdate)
```

Random Selection

- Optional module is enabled by a condition like the following:

  ```
  (rnd<personindex/N) && (rnd>=(personindex-1)/N)
  ```

  where:
  - N is # of eligible;
  - personindex is person's index among all eligible;
  - rnd is random value between 0 and 1.
- Function Quest.IRnd() returns a random value between 0 and 1;
- Number of eligible persons can be computed with:

  ```
  MEMBERS.Count(p=>p.age>15)
  ```
- Person's index among eligible is:
  MEMBERS.Count(person=>person.@rowcode<=@rowcode)

  ```
  MEMBERS.Count(p=>(p.@rowcode<=@rowcode)
  &&(p.age>15)&&(age>15))
  ```

Random Selection

- In case of eligibility conditions, the number of eligible persons must be determined first!
- Collect the eligibility variables in a different roster!
- To prevent the interviewers from reshaffling the members request their info to be collected in a particular order, for example from oldest male to youngest, then from oldest female to youngest.