Survey Solutions: Lookup tables

Sergiy Radyakin

February 12, 2016

Contents

1 Lookup tables structure and syntax

1.1 Lookup tables structure

Lookup tables in Survey Solutions are rectangular tables of numeric values, which may be utilized in the enabling and validation conditions. Each questionnaire may define multiple lookup tables of various dimensions and each such table must be assigned a unique name. The values from the tables may be extracted from such tables by specifying three elements: table name, row code, and column name.

Lookup tables may be assigned names as necessary, following the same requirements as applied for a question's variable name. Every lookup table must contain a column named *rowcode*, which must contain integer values identifying individual rows. There may be no duplicates and no missing values in this column. These values don't have to be consequtive or start from 1.

The content of a lookup table may contain fractions (non-integer values) and may contain missing values, as long as at least one value that is not missing.

$rowcode$	$colname_1$	$colname_2$		$colname_k$
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.

The values retrieved from a lookup table are of type *double?* and may be used in the syntax wherever constants of type *double?* are permitted.

1.2 Lookup tables preparation

In many cases the information for a lookup table comes from another system or already exists as a file. To prepare the lookup table file specifically for Survey Solutions software we need to make sure that:

1. the file is saved in tab-delimited text format;
2. the key column (variable) is called specifically *rowcode*;
3. other column names are satisfying the variable naming conventions;
4. there are no string columns in the file;
5. number of rows and columns in the file satisfies the limits for lookup tables.

1.3 Lookup tables syntax

`referencePrices[7m].col5`
: Obtain value from line corresponding to key 7 of the column named *col5* of lookup table *referencePrices*. The letter *m* is intentional and must follow the numeric constants specified in this context.

`referencePrices[(long)itemcode].lastmonth`
: Obtain value from line corresponding to key *itemcode* of the column named *lastmonth* of lookup table *referencePrices*. Note that we cast the type of the *itemcode* to *long* to permit search by this code in the lookup table.

`referencePrices[@rowcode].minimum`
: Obtain value from line corresponding to key *@rowcode* (code of the current row in the roster) of the column named *minimum* of lookup table *referencePrices*.

`self.InRange(referencePrices[@rowcode].minimum,`
`referencePrices[@rowcode].maximum)`
: Verify the value of the current [numeric or categorical single-select] question is between the minimum and maximum bounds extracted from the lookup table *referencePrices* by key corresponding to the current row code of the roster.

`referencePrices.Keys.Contains(self)`
: Verify that the lookup table *referencePrices* contains a key corresponding to the value of the current [numeric or categorical single-select] question.

`referencePrices[107m].maximum??300`
: Verify that the lookup table *referencePrices* retrieves the maximum price for item 107, or 300 if it doesn't.

2 Lookup tables: examples of use

2.1 Validation of prices

In this example we will look at a simple price survey monitoring the fuel prices at different gas stations. The unit of the survey is a gas station, and the following four kinds of fuel are monitored and the prices are expected to be in the givem ranges:

Fuel code	Fuel type	Min. price	Max. price
101	Regular	1.65	1.8
102	Plus	1.72	1.82
103	Supreme	1.95	2.15
201	Diesel	1.75	1.99

We place this reference information into a tab-delimited file:

rowcode	minprice	maxprice
101	1.65	1.80
102	1.72	1.82
103	1.95	2.15
201	1.75	1.99

Note that we have skipped the name of the fuel, since we will be utilizing the fuel code to locate the proper reference prices. We have also made sure that our key variable (the fuel code) is named specifically *rowcode* and that the column names satisfy the naming conventions for variables.

In Survey Solutions we add a new lookup table, load the above prepared file, and name it *RefPrices.*

Our fuel prices will be collected in a fixed roster *Fuel*, with codes corresponding to the fuel types above. There will be only one numeric question *price* in the roster with the following validation:

```
price.InRange(RefPrices[@rowcode].minprice,
       RefPrices[@rowcode].maxprice)
```

We accompany the validation rule with an error message *"Error. Fuel price is out of range"* that will be displayed to an interviewer if the entered price is not in the interval of prices corresponding to that fuel type.

2.2 Validation of yields

One common task in agricultural surveys is making sure the reported yields are recorded correctly. This can be complicated because the yields vary greatly by crop kind, as well as can be measured in non-standard units both for area (hectares, acres, square meters, etc) and amount of the crop (tons, bushels, buckets, sacks).

First we will look at the example of validating the standardized yield. Then we will look at complications.

Our reference data might look like the following table (hypothetical data):

Code	Name	Dry	Irrigated
101	Wheat	1.75	2.00
102	Corn	1.70	4.70
103	Rye	1.50	1.50
104	Rice	4.62	4.62
105	Barley	2.20	2.20
106	Chickpea	1.80	1.80
107	Drybean	1.50	1.50
108	Lentil	1.00	1.00
109	Potato	13.90	13.90
110	Onion	9.20	18.60
111	Gr.pepper	16.00	16.00
112	Tomato	32.40	32.40
113	Cucumber	16.70	16.70
114	Sunflower	1.15	1.70
115	Groundnut	2.40	2.40
116	Cotton	0.93	0.93
117	Sugarbeet	40.29	40.29
118	Tobacco	0.90	0.90
119	Melon	10.40	18.30
120	Alfalfa	11.00	11.00
121	Fodder	3.10	3.10
122	Soybean	1.60	1.60
123	Sesame	1.25	1.25

The two last columns of the table indicate the yield expected in dry and irrigated conditions for the crops where irrigation makes a difference, and it is expressed in metric tons per hectare.

Now, let's verify that the crop yield indicated by a respondent in our survey is within 10% of the value indicated in the reference table.

Our survey will include three questions:

1. categorical single-select question for whether the field is irrigated (variable name: *irrstatus*, categories: value 1 corresponding to the status *"irrigated"* and value 2 corresponding to the status *"dry"*),

2. categorical single-select question for the kind of crop (variable name: *crop*, categories defined according to the table above), and

3. numeric question for the crop amount (in metric tons) harvested from one hectare (variable name: *yield*).

We will prepare a tab-delimited file with the reference data resembling the above table with the following changes: we will remove the text column (there should be no string content in the reference data), and we will rename the first column from *Code* to *rowcode*, as this is the keyword that the software expects in every lookup file.

rowcode	Dry	Irrigated
101	1.75	2.00
102	1.70	4.70
103	1.50	1.50
104	4.62	4.62
105	2.20	2.20
106	1.80	1.80
107	1.50	1.50
108	1.00	1.00
109	13.90	13.90
110	9.20	18.60
111	16.00	16.00
112	32.40	32.40
113	16.70	16.70
114	1.15	1.70
115	2.40	2.40
116	0.93	0.93
117	40.29	40.29
118	0.90	0.90
119	10.40	18.30
120	11.00	11.00
121	3.10	3.10
122	1.60	1.60
123	1.25	1.25

The resulting lookup file we will load into the list of the lookup tables of the questionnaire, name it as *RefYield*, and refer to its values in the validation expression of the yield question:

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

There are, of course, alternative ways of describing the same condition, for example by defining two macros: `$yi=RefYield[crop].Irrigated` and `$yd=RefYield[crop].Dry` we can rewrite the above as:

```
(irrStatus==1 && yield.InRange($yi*0.9, $yi*1.1)) ||
    (irrStatus==2 && yield.InRange($yd*0.9, $yd*1.1))
```

Alternatively we can first pull out the values from the reference table:

```
$y=((irrStatus==1)?RefYield[crop].Irrigated:RefYield[crop].Dry)
```

then make the ranges around them:

```
yield.InRange($y*0.9,$y*1.1)
```

Either way, we are verifying that the yield is within 10% of the reference data corresponding to the particular crop and particular irrigation status.

2.3 Conversion of non-standard units

Now suppose in the previous example instead of yield expressed in standard units (tons per hectare) the respondent may provide information on the total ammount harvested and area cultivated. Any of these values may be expressed in non-standard units, for example area can be expressed in acres or square meters.

For example, the first numeric question for area may have a variable name *area* and the second categorical single-select question may have a variable name *aunit*.

We can load another lookup table that would provide conversion factors for non-standard units:

unit	code	factor
hectare	101	1
are	102	0.01
acre	103	0.404686
sqkm	104	100
sqm	105	0.0001
sqmile	106	258.999

To convert from non-standard area units to hectares we define the lookup table *AreaConv* with columns *rowcode* and *factor*:

```
rowcode  factor
101      1
102      0.01
103      0.404686
104      100
105      0.0001
106      258.999
```

Once we have the units table defined, we can use it as follows:

```
$AreaHa=AreaConv[aunit].factor*area
```

The resulting macro *AreaHa* can be used in expressions to denote area of cultivation expressed in hectares.

2.4 Regional bounds

Now, suppose we want to reduce the possibility that the interviewers submit data from wrong enumeration areas. We can examine their location and verify whether they are outside of their designated area by checking if location is within or outside of the area bounds.

Ponta do Sol
Ribeira Grande
Cruzinha
Chã de Igreja
Paul
Corda
Janela
Ribeira da Cruz
MARTIENE
Bonita
Santo Antão
Cirio
Monte Trigo
Porto Novo
Tarrafal de Monte Trigo
Baia das Gatas
Lazareto
Mindelo
Calhau
Sao Pedro
Ilha de São Vicente
Madeiral
Santa Luzia

Suppose the areas of the survey are the 10 islands of Cabo Verde. The following table shows approximate bounds for every island and can be utilized as a lookup table in Survey Solutions:

Code	Island	N	E	S	W
101	Santiago	15.346438	-23.420105	14.890378	-23.789520
102	Fogo	15.045920	-24.292351	14.808020	-24.517349
103	Sao Nicolau	16.678801	-24.019751	16.478319	-24.432390
104	Santo Antao	17.197170	-24.970449	16.907089	-25.358740
105	Sao Vicente	16.923808	-24.850388	16.773644	-25.087967
106	Maio	15.341471	-23.083649	15.114884	-23.237457
107	Boa Vista	16.233794	-22.663422	15.964630	-22.971725
108	Sal	16.861720	-22.868042	16.579604	-22.999878
109	Santa Luzia	16.807992	-24.681644	16.733701	-24.792366
110	Brava	14.904976	-24.661560	14.800111	-24.752884

As before, we prepare the lookup table as a tab-delimited file, skipping the string column (*Island*) and making sure our identifying column is named *rowcode*. We will load the lookup table into our questionnaire and name it *IslandRef*:

rowcode	N	E	S	W
101	15.346438	-23.420105	14.890378	-23.789520
102	15.045920	-24.292351	14.808020	-24.517349
103	16.678801	-24.019751	16.478319	-24.432390
104	17.197170	-24.970449	16.907089	-25.358740
105	16.923808	-24.850388	16.773644	-25.087967
106	15.341471	-23.083649	15.114884	-23.237457
107	16.233794	-22.663422	15.964630	-22.971725
108	16.861720	-22.868042	16.579604	-22.999878
109	16.807992	-24.681644	16.733701	-24.792366
110	14.904976	-24.661560	14.800111	-24.752884

We will run our survey in Survey Solutions' sample mode. In our questionnaire we include the island code (categorical single-select question with variable name *island*) and location question (GPS location question with variable name *hhlocation*). We will prefill the island code in our sample upload file (along with other addressing information), and let the interviewers capture the household location upon arrival to the household. Our questionnaire will include the following condition to verify that the interviewer is on the right island:

```
hhlocation.InRectangle(IslandRef[island].N,IslandRef[island].W,
         IslandRef[island].S,IslandRef[island].E)
```

In Survey Solutions' census mode the question on the code of the island will be assigned by the interviewer, and no changes to the validation condition are required.

2.5 Caloric intake

In consumption surveys it is commonly necessary to verify whether a certain amount of food items is larger than a reasonable threshold for personal consumption. This can be done based on estimated caloric value of the reported items and utilizes reference caloric content information (typically for a large number of items).

We design our questionnaire by placing a numeric-triggered roster *FoodItems* with following questions:

1. *itemcode* - categoric single-select question for the item code;

2. *amount* - numeric question for the amount consumed in kg;

We than ask for the size of the family (*hhsize)* and check whether the consumption agrees with the indicated number of persons.

We use lookup tables functionality to contain the reference data in Survey Solutions. Our table, which we will name *RefCalories*, will contain two columns: *rowcode* for the codes of food items and *CalKg* for amount of calories in 1kg of this food items.

Amount of calories from an individual item may be determined with the following expression:

```
RefCalories[(long)itemcode].CalKg*amount
```

If we want to know the total amount of calories from all items we apply the following expression:

```
FoodItems.Sum(x=>RefCalories[(long)x.itemcode].CalKg*x.amount)
```

We can now come up with a validation condition for the family size:

```
FoodItems.Sum(x=>RefCalories[(long)x.itemcode].CalKg *
                x.amount)/hhsize <= 5000
```

We've picked 5,000 calories as the upper threshold and the interviewer will receive an error message if the resulting caloric intake per person in the household overshoots this threshold. Of course, the error could be in either the consumption amounts or in size of the household.