Translating PASW Statistics Viewer Output Using the Translator Package

May 20, 2009

Updated Sep 21, 2009

Introduction

PASW Statistics is translated by SPSS Inc into many languages. However there is sometimes a need for output in a language not standardly provided. The translator package addresses this need. It provides code that can read a set of translation definition files and replace text in pivot tables, headings, titles, and outline contents where text is found using the defined translation. This is accomplished using Python scripting or an extension command, but producing or applying a translation does not require the translator to do any programming. This mechanism can also be used to override parts of a standard translation. User interface translation is not provided by this package.

Requirements

This package requires at least PASW Statistics version 17.0.2 and the Python programmability plug-in. With version 19.0.0, footnote translation does not work properly if there is more than one footnote.

Translation Scope

Text in pivot table row, column, and layer labels[1], and text in titles and captions can be translated as can text in outline titles and descriptions. Text in charts, trees, and other objects cannot be translated by this package. Since pivot table dimensions often include text originating in the data, dimensions of particular table types can be excluded from translation.

Text that exactly matches the translation dictionary can be translated by a simple replacement mechanism. If a label includes parameters that vary from instance to instance, the text can be translated using a special regular expression notation that substitutes the parameter(s) into the translated expression.

The translation materials can be defined in terms of any available input language. The translator defines the keys in terms of whatever input language they prefer. A translation into, say, French Canadian might be based on the SPSS-supplied French translation, for example.

The Translation Mechanism

This package provides two ways to perform translation. The first way is to set the translator.py module as an autoscript to trigger translation each time an instance of a

[1] In a layer label, the dimension name (the first part) can be translated but the category values (the part after the ":") cannot.

translatable object is created. This can be a base autoscript, that is, one that is invoked for every creation event, or it can be done by assigning the translator package to one or more specific pivot table subtypes.

The second way is to execute the SPSSINC TRANSLATE OUTPUT extension command included in this package. When executed, the Viewer contents selected by the syntax are all translated, subject to the limitations listed above. It does not track whether output has already been translated. It will try to translate it again. Typically that is harmless, but if a translated term happens to match a key in the dictionary, it may be unintentionally further translated.

Both ways use the same translation files and Python scripting, but they have different performance characteristics. Using autoscripting introduces a delay for each creation event in the autoscripting scope. Using the extension command avoids this but, of course, the output remains in the original language until the user executes the command.

Defining a Translation

To create a translation, you prepare a set of one or more *translation definition files*. Each file consists of one or more *sections*. A section is defined by a line starting with the section name in square brackets. Within each section, translations are defined in the form

key=value

where key is a word or phrase to be translated, and value is the translated text to replace it.

The section names are the OMS table-subtype names or the special section named GLOBALS. The subtype-specific sections are first searched for a matching key according to the table subtype being translated. If no match is found, including the case where there is no section matching the subtype, the GLOBALS section is searched. If the key is still not matched, the text remains as it was.

One implication of this is that text that is outside a pivot table, such as a heading or outline item, must have its translation defined in the GLOBALS section. Another implication is that all tables that have the same subtype will have the same translation definition.

Following is an example of a translation definition.

```
# This defines the translations for the Case Processing Summary table
[caseprocessingsummary]
Mean=Translation of Mean
mean=Translation of mean
N=translation of N

[frequencies]
Total=Totale
```

- Lines beginning with "#" or ";" are considered to be comments and ignored.
- Key matching is case sensitive: *Mean* and *mean* are different keys.

- Text outside any section is ignored
- The colon (":") can also be used as the separator between the key and the value. The separator characters cannot occur within a key.

Small example GLOBALTRANS.ini and LOCALTRANS.ini files are included in the translator package available from SPSS Developer Central.

The OMS Subtype

Every pivot table has an OMS (Output Management System) subtype. There are two ways to find the subtype. The first is to use *Utilities>OMS Identifiers*. This dialog lists every possible subtype for each specific command. The second way is to produce an instance of the table and right click on the outline item for that table. Choose *Copy OMS Table Subtype* to copy the type name to the clipboard. Only the second mechanism can be used for extension commands that generate tables with a subtype that is not one of the standard ones.

OMS subtype names can contain blanks and be in mixed case. When creating sections, write the name in lower case without any blank characters.

Translating Expressions Containing Parameters

The simple substitution method covers most cases, but sometimes translatable text contains values that can vary from instance to instance of a table. For example, the Independent Samples Test table in the T-Test procedure contains the label
95% Confidence Interval of the Difference

In this label "95" is a user-specified value, so using that label as a translation key will not work in general. For situations like this, *regular expressions* can be used to define the translation. These expressions use a special notation to describe the pattern and where to insert the parameters in the translated version. They are very general and are not fully explained here beyond some simple examples. See
http://docs.python.org/howto/regex.html for one explanation.

Regular sections cannot contain regular expressions, but each regular section can have a related section of regular expressions for it. That section has the name of the regular section followed by "-regexp". For example, you might have the following

```
[independentsamplestest]
df=locals translation of df
[independentsamplestest-regexp]
(\d+)% Confidence Interval of the Difference=\1% locals translation of CI
```

In this example, "df" will be translated from the standard section, and the confidence interval text from the regular expression section. Any section, including GLOBALS, can have a related regular expression section.

First, an attempt is made to translate text using the standard local and global sections. If no translation is found, each regular expression key in the corresponding regexp sections, if any, is tried in turn, and the first one that matches produces the translation. The TSCOPE setting of the standard section also applies to the regular expression section.

Both the key and the value are regular expressions in regexp sections, and the keys are case sensitive.

Regular Expression Notation

In a regular expression, certain characters have a special meaning. If these characters are to have their conventional meaning, they must be escaped by preceding them with "\". For example, parentheses are special, so they must be written as \(and \) to treat them as regular text. Certain regular characters have a special meaning when preceded by \. For example, \d refers to any digit character.

The special characters that you might encounter in the text are

Character	Meaning
.	any character
^	start of the text
$	end of the text
*	zero or more repetitions. E.g., ab* matches a followed by zero or more b characters
+	one or more repetitions
?	zero or one repetition
()	a group
[]	a set of characters
|	A|B matches either A or B

These must be escaped with \ or used to have their regular expression meaning.

In the example above, (\d+) defines a group of one or more digits. The group is indicated by the parentheses, \d denotes a digit, and the + means one or more. In order to use a parameter in the translation, it must be enclosed in parentheses in the key and referred to by \number in the translation.

A string that contains a variable name or label might be

> The variable *fred* has a non-Gaussian distribution.

To match this treating "fred" as a parameter that will vary, you could write

> The variable (.+) has a non-Gaussian distribution.

This matches the text with any variable name, actually any text, and the parentheses define a group, here group 1.

In the translated text, you can refer to the group(s) in the key as \1, \2, … numbered left to right in the pattern. Thus this text might be translated as

> Die Variable \1 hat eine nicht-Gaussian Verteilung.

The variable name parameter, which is group 1, is substituted for \1 in the translation.

Taking The first example further, while (\d+) defines a group of one or more digits, the user-entered value might have a decimal. Either period or comma might be the decimal point. To capture this form requires a more complicated regular expression.

(\d+[.,]*\d*)

This means a group of one or more digits, \d+, followed by zero or more occurrences of period or comma, [.,]*, followed by zero or more digits. Inside square brackets, you don't escape special characters such as ".".

Another way to accomplish this would be

([.,\d]+)

which would match any string of digits, periods, and commas.

Finally, the expression

(^.+)%

would match the start of the text up to but not including a percent sign. The percent sign would not be part of the group.

Organizing Translation Definitions into Files

The translator module looks for definition files as follows.

The file GLOBALTRANS.ini is expected to contain global terms within a GLOBALS section. These definitions will be used if no more-local definition is found.

The file LOCALTRANS.ini consists of one or more sections specific to OMS subtypes.

Files of the form OMS-subtype.ini, for example, frequencies.ini, can define local sections appropriate to a particular subtype. They must still include the section name.

All of these files are optional. The first two are always read if present. The third type is read only when an instance of that subtype is found in the Viewer. If no files are found, then nothing is translated.

The GLOBALTRANS file is intended for terms shared across subtypes and for items not in a pivot table. All of the type-specific sections could be placed in LOCALTRANS.ini, or these could be placed in their own files. The way sections are distributed will affect performance, since the first two will be read frequently, while the third types will be read only as needed.

A typical arrangement would have a small GLOBALTRANS file with outline terms and very common words or phrases along with either a single LOCALTRANS file containing all the sections or a set of subtype files each with one or a few subtypes. However, many variations are possible. No rules are enforced regarding organization. The only immutable rule is that terms defined in GLOBALTRANS are overridden by matching items in LOCALTRANS or the subtype-specific files. A given type of section should be defined in only one place. So, for example, if the term *factor* has the same translation in most contexts but needs a different translation in one particular table, it could be defined in a section for that subtype and in the GLOBALS section with a different translation.

Local definition sections can contain a special key named *TSCOPE*. Its value can be ROWS, COLUMNS, or ALL. ROWS and COLUMNS limit the scope of the keys in that section to the indicated dimension. ALL is equivalent to omitting TSCOPE from the section. TSCOPE does not apply to the GLOBALS section, however.

All of the translation definition files must be encoded in Unicode UTF-8 and begin with the UTF-8 Byte Order Mark (BOM). Notepad on Windows is one editor that can save files in this format. When saving the file, select UTF-8 in the encoding dropdown control on the file chooser dialog.

Where the Translation Definition Files Are Found

By default, the translation definition files are expected to be in the *extensions* subdirectory of the PASW Statistics installation. However, if an environment variable named *SPSS_TRANSLATOR* is defined, it defines the directory (folder) containing the files, and the extensions subdirectory is not used. In order to have more than one set of definition files or if the user does not have permission to write to the extensions subdirectory when the translation is installed, use this environment variable. The environment variable contents may end with a directory separator or not: both forms will work.

The SPSSINC TRANSLATE OUTPUT Command

This is an extension command that causes the specified contents of the Viewer, subject to the limitations listed above, to be translated. Its syntax is

```
SPSSINC TRANSLATE OUTPUT [FOLDER=folder-specification]
  [SUBTYPE=list of subtypes]
  [PROCESS={PRECEDING* | ALL}]
  [SELECTEDONLY={NO* | YES}]
  [/HELP].
```

FOLDER, if specified, defines the folder where the translation definition files are located. Enclose the name in quotation marks. Otherwise the files are expected to be found based on the SPSS_TRANSLATOR environment variable or in the extensions subdirectory of the installation folder.

SELECTEDONLY = YES causes only the selected items in the Viewer to be translated.

SUBTYPE can specify a list of OMS table subtypes. If given, only tables of those types will be translated. This is ignored if SELECTEDONLY is YES.

PROCESS specifies whether to process only the immediately preceding procedure output or the entire Viewer contents. This is ignored if SELECTEDONLY is YES.

/HELP displays this text and does nothing else.

The package includes a dialog box for this command. After installation, it will appear on the Utilities menu.

Installing a Translation

The translation definition files should be copied to the locations as indicated above. If using the SPSS_TRANSLATOR environment variable, define that as a user or system variable. On Windows, this is done via the Windows Control Panel>System>Advanced>Environment Variables dialog.

The Python files translator.py and SPSSINC_TRANSLATE_OUTPUT.py can be placed anywhere that Python can find them. The extensions subdirectory is one such convenient place.

The SPSSINC_TRANSLATE_OUTPUT.xml file should be placed in the extensions subdirectory or in a directory listed in the SPSS_EXTENSIONS_PATH environment variable.

If using the autoscripting mechanism for translation, use *Edit>Options>Scripts* to make translator.py the base autoscript or apply that file specification to each specific table subtype that should be translated. If there are many of these, a Registry script (on Windows) will facilitate the translations.

On the Mac, Java preferences are implemented using property list (plist) files, which are binary or XML. The file system file for holding these preferencess for a given user is ~/Library/Preferences/com.spss.pasw statistics.plist or appropriate product name. As of OS X 10.4, Apple switched to a binary representation. To convert between binary and XML, use the plutil command:

$ plutil –convert xml1 “com.spss.pasw statistics.plist”

This file can then be edited with a text editor, or use shell scripting to add a large number of autoscript entries.

Table types defined by programs or extension commands do not appear in the Options dialog box but can still be added via the Registry or preferences mechanism.