Paper 068-2019

Python applications in SAS® programming for medical device clinical trial studies

Jianlin Li, Q2 Business Intelligence; Sherry Cao, Abbott;

ABSTRACT

As statistical analysts supporting global medical device regulatory submissions, we routinely deal with requests to generate SAS Table, Figure, and Listing (TFL) in English and the local language, e.g. Chinese or Japanese. We also need to prepare batch files to execute SAS codes to increase programming efficiency. File folder cloning and SAS source code searching and replacement in bulk are also necessary steps for many reporting activities. Text processing is common to many of daily tasks like what are mentioned above.

As a modern general purpose and high-level programming language, python is very powerful, especially in manipulating text data. In this paper, we present a few python applications to improve efficiency and accuracy in SAS programming work: (1) Generic Unicode conversion and handling in SAS programs to support non-English languages; (2) SAS dataset program and SAS TFL program execution batch file generation; (3) Structured folder generation.

SAS macros can be created to generate input parameters and configuration files for Python programs. This enables SAS programmers, without knowing Python languages, to finish tasks by simply calling SAS macros. With its simple syntax rules and readability, Python has more potential applications that could make SAS programmers' lives easier.

INTRODUCTION

Python is one of the high-level programming languages which has been becoming a trend in data mining and processing. Its simple syntax rules and its portability across multiple platforms make it easier for programmers to keep the code base readable and application maintainable. Python especially excels at mining and handling text data.

SAS is widely used in clinical trial data analytics and regulatory reporting in pharmaceutical and medical device companies. SAS programmers play an important role in clinical trial activities. As opposed to module interdependencies of other language applications, SAS programs for regulatory reporting and science publications are usually small in size with similar folder and code structures among multiple studies. This creates the opportunity for other high-level languages like Python to help prepare SAS programming environment setups, generate reusable codes, auto clone SAS programs, and streamline code sharing among SAS programmers. In this paper we use a few examples to show how Python is used to boost SAS programming productivity and accuracy.

PYTHON APPLICATION I: DICTIONARY BASED UNICODE SAS PROGRAM GENERATION

The medical device market in Asia has been growing rapidly in recent years. Medical device vendors are required to meet regulatory compliance requirements by foreign governments. SAS is used to build reports to fulfill part of the requirements. However, Asian countries have various requirements for local language translation on Tables, Figures, and Listings in regulatory submission. It is a common practice in the industry to replicate an English-based SAS program and in a SAS Unicode environment, convert relevant text strings to Unicode, then execute and generate reports in local languages (e.g. CJK languages in Asia). A typical procedure for generating foreign language TFLs is:

1. Open an English version SAS program using generic SAS
2. Open the same original SAS program using Unicode version SAS
3. Open the English TFL file
4. Open the foreign language TFL mock up file
5. Compare the English version TFLs and the Foreign language mock ups for differences and find phrases to be replaced
6. Manually update the SAS program which has been opened in Unicode SAS
7. Save the Unicode version of the SAS program
8. Repeat Steps 1-7 by hand for each SAS program in the project

```
proc format;
  value grpfmt
   1 = "<b>Pre-procedure<br>&#160&#160 Target Vessel</b><br>&#160&#160
       &#160&#160 &#160&#160 LAD"
   . . .
   54 = "&#160&#160 &#160 &#160 In-Segment<br> &#160&#160 &#160 &#160
        &#160&#160 &stb1"
   55 = "&#160&#160 &#160 &#160 In-Scaffold <br> &#160&#160 &#160 &#160
        &#160&#160 &stb1"
    . . .
   ;
quit;

proc format;
  value grpfmt
   1 = "<b>手技前<br>&#160&#160 標的血管</b><br>&#160&#160 &#160&#160
&#160&#160 LAD"
   . . .
   54 = "&#160&#160 &#160 &#160 セグメント内<br> &#160&#160 &#160 &#160
&#160&#160 &stb1"
   55 = "&#160&#160 &#160 &#160 スキャフォールド内 <br> &#160&#160 &#160 &#160
&#160&#160 &stb1"
   . . .
   ;
quit;
```

Above are the English and Japanese versions of some sample SAS code. The manual procedure stated in the previous paragraph is tedious and highly error prone because usually textual translation is scattered everywhere in a SAS program and it is often easy to miss a phrase. For example, 'In-Segment' should be translated to 'セグメント内'; while 'In-Scaffold' should be translated to 'スキャフォールド内. A SAS programmer who does not know Japanese might confuse the two phrases. He might make a translation error, which would be discovered at a later stage of the project.

Recently we had a Japanese regulatory reporting project involving translation of around 70 SAS programs. We started a new approach using Python to build up terminology translation dictionary in this Japanese project, and then used a SAS macro to execute the Python script.

Below are some of the key steps in generating Unicode SAS programs for Japanese reporting.

1. Install Python 3 compiler and Python IDE PyCharm on MS Windows system.
2. Put all the SAS programs in a folder
3. Use SAS code to call the Python program to process all the SAS programs in the folder
4. Generate new Unicode SAS programs using the same original name and put them to a new folder
5. Review and make corrections on the newly generated Unicode SAS programs
6. Enrich the Python dictionary with the newly found Japanese phrases in the above step.
7. Run the SAS programs using SAS Unicode version to generate Japanese TFLs.

A SAS macro is used to incur the Python code:

```
*Initiate Python code to convert regular SAS programs to Unicode SAS
program;
%macro convertutf(py=, inpath=, outpath= );
    *Executing Python Script;
    data _null_;
          x &py "&inpath./*.sas" "&outpath";
    Run;
%mend;

%let fix_unicode=C:\Users\lijx142\PycharmProjects\sas2\jpn_unicode.py;
%let inp=C:\Users\lijx142\Documents\My SAS Files\Unicode;
%let outp=C:\Users\lijx142\Documents\My SAS Files\Unicode\Output;
%convertutf(py=&fix_unicode, inpath=&inp, outpath=&outp);
```

A SAS programmer can use macro *convertutf()* to do preliminary processing on SAS programs in bulk.

The key part of the Python conversion code is to build up a dictionary:

```
def replace (input_line):
  keywords = {
    u'Mean ± SD': u'平均値（標準偏差',
    u'Median': u'中央値',
    u'min, max': u'最小値、最大値',
    u'Pre-procedure': u'手技前',
    u'Target Vessel': u'標的血管',
    u'Calcification': u'石灰化',
    u'Post-Procedure': u'手技後',
    u'Distal': u'遠位部',
    u'Proximal': u'近位部',
    u'In-Segment': u'セグメント内',
    u'In-Scaffold': u'スキャフォールド内',
        'tblseq=&seq,': u'tblseq=&seq, charset=UTF8,'
        . . .
    }

    working_line = str(input_line)
    for key in keywords:
        if working_line.find(key) > 0:
            mapped_value = keywords[key]
            working_line = working_line.replace(key, mapped_value)
    return working_line
```

In the “keywords” python dictionary shown above, we list the English phrase and its corresponding Japanese translation in pairs. One key point worth mentioning is to use ENCODING = 'utf-8' and 'cp1252' for opening input and output IO files respectively in Python 3 such that the text encoding can be handled properly:

```
    with io.open (full_target_file, 'w+', encoding=ENCODING) as f_output:
    with open (source_file, 'r', encoding='cp1252') as f_input:
```

Similarly, for another language, we can also build up a corresponding dictionary. For example, a Chinese dictionary in Python will look like this:

```
def replace(input_line):
    keywords = {
  u'Mean ± SD': u'均数 ± 标准差',
        u'Median': u'中位数',
        u'Range (min, max)': u'极差 (最小值, 最大值)',
        u'Confidence Interval': u'置信区间',
        u'Target Vessel': u'靶血管',
        u'Circumflex or Ramus': u'左回旋支或分支',
        u'Aneurysm: u'动脉瘤',
        u'Calcification': u'钙化',
        u'Tortuosity': u'弯曲',
        u'Eccentric': u'偏心性',
        u'Thrombus': u'血栓',
        u'Bifurcation': u'分叉'
        . . .
}
```

The Chinese dictionary can be used for Chinese regulatory reporting projects.

Usually after the Unicode conversion SAS macro is executed, there might still be some English phrases not translated yet because those phrases are not yet in the dictionary. At this point, a manual and iterative process should be used to review the intermediate generated SAS programs, identify those words, and translate them into the proper language binding until the Python keyword dictionary is complete for this project. The dictionary grows as more SAS programs are processed. Gradually, a good capacity of technical terminologies for the same cluster of medical devices can be covered and maintained.

PYTHON APPLICATION II: GENERATING BATCH FILES FOR EXECUTING ANALYSIS DATASET (ADS) SAS PROGRAMS, TFL PROGRAMS, AND VALIDATION PROGRAMS

Running SAS programs in batch mode increases efficiency and streamlines multiple SAS program execution. Batch files not only allow SAS programmers to stay in the interactive SAS environment while executing SAS programs in the background, but also to auto-generate SAS Log files and SAS Listing files for future inspection. It is a good practice to create batch files for ADS programs, TFL programs, and Validation programs for each study delivery.

As a powerful text processing language, Python can be used to automatically create batch files for the orchestration of SAS program execution and redirection of List and Log files to designated locations. As shown below, a SAS macro is created to wrap around the Python program that facilitates the batch file creation.

```
*Initiate Python code to create batch files for ADS, TFL, or Validation
program execution;
%macro genbatch(py=, rootpath=, files=);
    *Executing Python Script;
    data _null_;
          x &py "&rootpath" &files;
    Run;
%mend;
%let batch=C:\Users\lijx142\PycharmProjects\sas2\gen_sas_batch.py;
%let rt=C:\Users\lijx142\Documents\My SAS Files\Study101;

%let fl=ADS;
%genbatch(py=&batch, rootpath=&rt, files=&fl);

%let fl=TFL;
%genbatch(py=&batch, rootpath=&rt, files=&fl);

%let fl=VAL;
%genbatch(py=&batch, rootpath=&rt, files=&fl);
```

Again, in the Python program (gen_sas_batch.py), special attention should be paid to text encoding. Here the encoding setting should be ENCODING = 'cp1252' since the Python program is supposed to be executed by generic SAS macro. However, in the output batch file, for foreign language reporting projects, U8 configuration ("…\\nls\\u8\sasv9.cfg") should be used instead of EN configuration ("…\\nls\en\sasv9.cfg").

PYTHON APPLICATION III: GENERATING DIRECTORY STRUCTURES

In SAS programming for clinical regulatory filing, it is very common that environment settings and directory structures are similar among multiple projects. A well-defined folder structure helps SAS programmers collaborate and improves work efficiency.

Python is very handy in creating folder structures. A predefined text file containing the directory structure can be used as the configuration file. The Python program uses the configuration file to create the desired file structure. We created a SAS macro to streamline the entire process. Below is the list of actions in the following SAS macro:

1. Generate text file with the desired layout of file structure.
2. Use the text file as a configuration file to feed the Python program
3. Execute the Python program

```
%macro gendir(py=, rootpath=);
   *Generate Configuration file for the Python program;
   data _null_;
        length line $75;
        *Use TAB to indicate folder layers;
        file "&rootpath\layout.txt" lrecl=4000;
        line="#Layout File for directory structure"; put line;
        line="&rootpath"; put line;
        line='Program'; put line;
        line='ADM'; put '09'x line;
        line='Pgm'; put '09'x '09'x line;
        line='QC'; put '09'x '09'x line;
        ...
   run;

   *Executing Python Script using the above text file as configuration
file;
   data _null_;
        x &py "&rootpath\layout";
   Run;
%mend;

%let rtpath=%str(C:\Users\lijx142\Documents\My SAS Files\Study101);
%let py=C:\Users\lijx142\PycharmProjects\sas2\create_folders.py;
```

The layout of the resulting directory structure is like

```
#Layout File for directory structure
C:\Users\lijx142\Documents\My SAS Files\Study101
Program
        ADM
                Pgm
                QC
. . .
```

At the start of a new study delivery, a desired folder structure can be generated using the macro *gendir*

CONCLUSION

As SAS programmers, we can greatly improve our efficiency and accuracy by eliminating many trivial manual steps in our daily routines when we deploy Python utilities that are tightly integrated with SAS. When a Python interpreter/compiler is installed, SAS macros can help other users utilize the Python applications for process automation in clinical data analysis and reporting.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Jianlin Li
Q2 Business Intelligence
Jianlin.li@q2cro.com
www.q2bi.com

Sherry Cao
Abbott
sherry.cao@abbott.com
www.abbott.com