PharmaSUG China 2019 – Paper CC-036

# Create your SAS programs automatically using python

Linjie Jia, Sanofi Aventis;
Lily Zhang, Sanofi Aventis

## ABSTRACT

As a statistical programmer, no matter which role you play in your daily work, leading or validating, you need to create/validate many SAS programs for tables, listings and figures. Especially in CSR, there would be hundreds of SAS programs need to be created when the study begins or when folder new created. However, most of us choose to copy and paste the names from excel file, and then begin to programming. Obviously, it takes much time and also need to be careful to avoid the typo issues. In this paper, a new method raises up to create SAS programs in your folder base on the spreadsheet (such as QC tracking) automatically and correctly. Three functions are listed as below for this automatically work:

1. All the SAS programs will be created automatically at once.
2. The SAS programs will be automatically named as you want.
3. The header can be designed at first and applied automatically for all.

Moreover, the new method is implemented by a few lines of python code. And all what you need to do is just filled in a simple configuration file and double click on the executable file, and then you can start your formal programming.

## INTRODUCTION

As we all know, a statistical programmer need to create and validate many tables, listings, and figures in every study. However, it is annoying to create many programs and modify the header one by one. We need to find a lot of information from several files to get them, and paste it to corresponding programs. Now, we can make everything works automatically. In this paper, we will use some configuration files and python code to create SAS programs. First, it will show you the reason for choosing python, not SAS code or any other programming languages. Then, the flow of the entire process using a simple example is described.

## REASON FOR CHOOSING PYTHON

Python language is a general-purpose interactive, high-level and object-oriented programming language. It supports the structured and functional methods, and Object-Oriented Programming Languages are probably used for scripting. Compared with SAS, python is not as authoritative as SAS in statistical analysis, but it has its own advantage.

Firstly, as some people said that SAS is not a real programming language, while python is. It is versatile with the basic features of a programming language, in addition to basic data types, integers, float, Booleans, strings, etc. It also has the complex data structure, lists, tuples, and dictionaries, which make it more flexible to accomplish your idea.

Secondly, python can support IO reading and writing very well, thus the manage process threads will be very convenient.

Thirdly, python standard library, it mainly reflects the advantages of python processing strings. Due to the versatile properties and good support for regular expressions, it couldn't be more suitable for processing text.

## AN EXAMPLE OF THE METHOD

### PREPARE FILES

Firstly, we need to prepare some files:

1. A spreadsheet that contains your program names, output names, table number, input data or any other variables that you want to display in your SAS programs. And you don't have to create it

especially, some files you have created before the programming can be used directly. Like tracking sheet, a spreadsheet used for QC tracking, and REFLIST, a spreadsheet used for adding title and footnote. An example file is as bellow:

| Output | Title of output | Program Name | Output Name | QC program Name | Input da |
|---|---|---|---|---|---|
| 1.1.01 | Patient disposition in Phase 3 pool (T2 | dis_dispo_r_t | dis_dispo_s1_r_t | v_dis_dispo_r_t | AdaM.adsl |
| 1.1.02 | Patient disposition in Phase 2/3 pool ( | dis_dispo_r_t | dis_dispo_s2_r_t | v_dis_dispo_r_t | AdaM.adsl |
| 1.1.03 | Patient disposition in Renal impairment | dis_dispo_r_t | dis_dispo_s3_r_t | v_dis_dispo_r_t | AdaM.adsl |
| 1.1.04 | List of patient exposed to double blind | dis_trt_notrand_l | dis_trt_notrand_l | v_dis_trt_notrand_l | AdaM.adsl |
| 1.1.05 | Kaplan-Meier plot of time to treatment | dis_disc_km_r_f | dis_disc_km_any_s1_r_f | v_dis_disc_km_r_f | AdaM.adsl |
| 1.1.06 | Kaplan-Meier plot of time to treatment | dis_disc_km_r_f | dis_disc_km_any_s3_r_f | v_dis_disc_km_r_f | AdaM.adsl |
| 1.1.07 | Kaplan-Meier plot of time to treatment | dis_disc_km_r_f | dis_disc_km_ae_s1_r_f | v_dis_disc_km_r_f | AdaM.adsl |
| 1.1.08 | Kaplan-Meier plot of time to treatment | dis_disc_km_r_f | dis_disc_km_ae_s3_r_f | v_dis_disc_km_r_f | AdaM.adsl |
| 1.1.09 | Analysis population in Phase 3 pool (T2 | dis_population_r_t | dis_population_s1_r_t | v_dis_population_r_t | AdaM.adsl |
| 1.1.10 | Analysis population in Phase 2/3 pool ( | dis_population_r_t | dis_population_s2_r_t | v_dis_population_r_t | AdaM.adsl |
| 1.1.11 | Analysis population in renal impairment | dis_population_r_t | dis_population_s3_r_t | v_dis_population_r_t | AdaM.adsl |

Display 1. Sample table Step 1

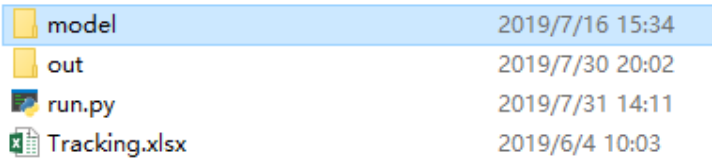
2. A template SAS programs, which will appears in your every new SAS program. You can create a simple SAS programs that contains your header and some common codes. And please attention, you should fill in the part that needs to be replaced like this: ${variable name}. An example is as bellow:

```
 ********************** [START STUDY HEADER]*******************;
 * Property of XXX Company                                     *;
 *                                                             *;
 * Program Name             : ${Program Name}.sas              *;
 * Program Purpose          : Validate ${Output}               *;
 * Compound/Study/ Analysis: xxx/xxx/xxx                       *;
 * Program Author           : xxx                              *;
 * Date Completed           : 22Apr2019                        *;
 *                                                             *;
 * Input Datasets           : ${Input data}                    *;
 * Output                   : ${Output Name}.rtf               *;
 *                                                             *;
 * System                   : SAS version xxx                  *;
 * Revision History                                            *;
************************ [STOP STUDY HEADER]*******************;

proc datasets lib=work nolist kill;run;
%let pgm = ${Program Name};
%let base=${Output Name};
/**********************code body*********************************/

/*************************end*************************************/

data qcd.v_&base;set qc;keep __col_:;run;

data base;set repd.&base;keep __col_:;run;

options device='ACTXIMG';
title "compare result of &base.";
ods pdf file="%sysfunc(pathname(QCO))/v_&base..pdf";
     proc compare base=base compare=qcd.v_&base. listall; run;
ods pdf close;
```

Display 2. Sample code Step 2

### LOAD FILES AND EXECUTE

After preparing these two files, you can create a file structure like the sample file structure, and put your configuration files under the corresponding files.




Display 3. Sample file structure

Just open the executable file run.py and fill in some information needs to be specified, like which variable will be used for your SAS programs, which sheet will be your target configuration in the spreadsheet.

```
## Please fill in your configuration ##############################
colName = 'Program Name'  #the variable used for the program names#
fileName='Tracking.xlsx'  #the name of you configuration file     #
sheetIndex=3              #which sheet would will be used         #
## end of edit                        ##############################
```

Display 4. Sample configuration code

So far, we have finished all the preparation work, after double click the run.py, you can see all the SAS programs in the path you specified.

| Name | Date modified | Type | Size |
|---|---|---|---|
| dis_disc_km_r_f | 7/30/2019 2:02 PM | SAS System Prog... | 3 KB |
| dis_dispo_r_t | 7/30/2019 2:02 PM | SAS System Prog... | 3 KB |
| dis_population_r_t | 7/30/2019 2:02 PM | SAS System Prog... | 3 KB |
| dis_trt_notrand_l | 7/30/2019 2:02 PM | SAS System Prog... | 3 KB |

Display 5. Sample SAS programs

## CONCLUSION

This version is just a beta version, and I will do a graphical interface if most of you think it is a good idea to implement this method.

In addition, it is just a very simple application of python in our daily work, and you don't have to do a lot of repetitive work which will be done by your computer automatically. Therefore, you can create more useful tools if you want.

## CONTACT INFORMATION

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

Linjie Jia
Sanofi Aventis
jialinjie@live.cn

## APPENDIX FULL TEXT OF PROGRAM

The source python code:

```
import sys; print('Python %s on %s' % (sys.version, sys.platform))

## Specify your specific content here#####################################
colName = 'Program Name'              #the variable used for program names #
fileName='Tracking.xlsx'              #the path of the tracking file       #
sheetIndex=3                          #which sheet will be used            #
outPath='out/'                        #the path of your output             #
## end of edit                        ######################################


# step1: read tracking
def read_tracking():
    import xlrd
    excel = xlrd.open_workbook(sys.path[0]+'/'+fileName)
    table = excel.sheets()[sheetIndex]
    print('tracking line count:',table.nrows)
    names= table.row_values(0)
    keyIndex = -1
    for i in range(len(names)):
        if colName == names[i]:
            keyIndex = i
            break
    print(keyIndex)

    table_confs= {}
    for i in range(1, table.nrows):
        line = table.row_values(i)
        d = table_confs.get(line[keyIndex],{})
        for j in range(0,len(names)):
            if j == keyIndex:
                continue
            values = d.get(names[j].lower(),set())
            values.add(line[j])
            d[names[j].lower()] = values
        d[names[keyIndex].lower()]=set([line[keyIndex]])
        table_confs[line[keyIndex]] = d
        ## print(d)
    return table_confs

table_confs = read_tracking()


##step2, replace the specified variable in the model
model_file=[]
import re
with open('model/demo.sas','r+') as f:
    model_file = f.readlines()

print("model file lines ", model_file)

for table_name in table_confs.keys():
    conf = table_confs.get(table_name)
    write_lines=[]
    for line in model_file:
        match = re.findall('\${(.*?)\}', line, flags=0)
        for word in match:
            ############## keep format   ##############
```

```
            index = line.index('${' + word + '}')
            values = list(conf[word.lower()])

            print("key",word,"values ",values)
            tmp = line.replace('${' + word + '}', values[0])
            for x in range(1, len(values)):
                tmp+=(' ' * index) +line[index:].replace('${' + word + '}',
values[x])
            line = tmp
            ################end#######################

        ## create SAS programs and write your code in
        write_lines.append(line)

    print("write table "+ table_name)
    with open('out/'+table_name+'.sas', 'w+') as write_table:
        write_table.writelines(write_lines)


print('end--------')
```