PharmaSUG 2023 - Paper AA-###

Automation of validation result checking by Python

Qianqian Li, Sanofi (Beijing) Pharmaceutical Co. Ltd.

Qing Zou, Sanofi (Chengdu) Pharmaceutical Co. Ltd.

Zhimin Liu, Sanofi (Beijing) Pharmaceutical Co. Ltd.

ABSTRACT:

During the analysis of clinical trial data in the industry, batches of data QC comparison results are faced every day to ensure accuracy and quality of a huge number of Datasets/Tables/Listings/Figures. In the past, the results of the comparison were manually checked one by one which is time-consuming. To improve efficacy and accuracy of checking comparison results, most companies produce SAS® data set of the comparation results with the SAS® program, and save those comparation results into sas7bdata, Rich Text Format (RTF) or portable document format (PDF) file. Nevertheless, results of sas7bdat format are unable to be directly inspected, and the SAS® program is helpless to results of the DOC and PDF Output checking. Therefore, the third-party procedures and tools have been developed gradually by the technicians and users of major companies to address the shortcomings SAS® program. These procedures also contain shortage itself, inability of reading PDF format and low incompatibility of program development environment for checking QC validation results. This python program was developed for not only automatically checking and listing QC validation results as well as program and output modification date into the excel sheet, but also for a better compatibility of program development environment and effectively supportive for both RTF and PDF file reading.

KEY WORDS

SDTM, ADaM, QC validation, python, RTF file, PDF file

INTRODUCTION:

SAS® macro can automatically input the modification dates and read RTF files into SAS® datasets for further checking and archiving purpose. But it is a real challenge to converted RTF files into SAS® datasets, due to there are no RTF control word to identify the page, the title, the table header. To keep the body of the table, SAS® program need to be further modified to remove page numbers, remove subgroups, remove the title, the table header and the footnotes. Which cause the low effeteness of checking bunches of validation results. then try to split each row into different columns (1).

Furthermore, the SAS® program is helpless to results of the DOC and PDF Output checking (2). Therefore, the third -party procedures and tools have gradually been developed by the technicians and users of major companies to address the shortcomings SAS® program. One of these procedures and tools is VBA embedded in Excel (3). But extracting data by VBA from PDF file are highly reply on the Acrobat.exe application through which the PDF file could be opened (4). Moreover, only the paid version of Adobe Acrobat has more advanced features that allow users to create, edit, export, and sign PDF files (5) and Adobe Acrobat also require the compatible VBA version. (The Adobe Acrobat 10.0 Type Library can be used with VBA to programmatically work with Adobe Acrobat) (6, 7)**.** Based on above shortage, now data retrieval by VBA is limitedly applied to only certain format of documents, for an example RTF files.

And with the usage of various program development environment which show weak compatibility of VBA, the use of Python in clinical trial data analysis will become an inevitable trend nowadays. Because Python show advantages in certain aspects, it is fairly convenient to handle the RTF and PDF files in Python, which is tedious to deal with in SAS. Advantages of Python include its independence on embedment in specific software programs, it is open-source nature, it own generation of various development packages, and Python program can also be executed using system commands.

This article developed a python program (.py file) by using PyPDF2, openpyxl, sas7bdat package to quickly read PDF, RTF, excel, sas7bdat files. Compare results could be extract directly from PDF, RTF files, both validation results and date of modification could be summarized into data frame. Eventually summarize results could be export into a new sheet of original excel file. Moreover, py file could be run by multiple interfaces: batch run in various program development environment (SAS® Enterprise Guide V8.3). Moreover, openpyxl package could read .xlsm file with VBA macro imbedded, so the original VBA macro will not be destroyed. By contrast VBS and SAS® edited by SAS® are difficult to retain the original VBA macro.

REQUIREMENTS:

Software: Python 3.8
One excel file which contains program and qc file information.

package	version	usage
pandas	1.5.2	It provides an efficient multi-dimensional container of generic data, which were extracted from either PDF,RTF or sas7dbdat file in this study
PyPDF2	2.12.1	The python PDF library capable of splitting, merging, cropping, and t ransforming the pages of PDF files
JupyterLab	3.1.18	The web-based user interface for Project Jupyter
Openpyxl	3.0.10	It is used to perform excel tasks such as read data from excel file, or write data to the excel file etc.
sas7bdat	2.2.3	The Python module of reading sas7bdat files

DEFINE "QC_DATE" FUNCTION:

```
qc_date(file_type     = ,
        studypath    = ,
        path_excel = ,
        excel_sheet =,
        dataset = ,
        lead_date = ,
        qc_pgm = ,
        qc_output_date = ,
        qc_status = ):
```

file_type: "DATA", "OUTPUT" option could be entered. This represents for triplets of SDTM/ADAM data set, or triplets of OUTPUT respectively.

studypath: This represents the path of the study triplets.

path_excel: This represents the path and the name of the excel file(xlsx,.xlsm), which contains program and qc file name. For an example: 'QC_tracking.xlsm', '_ref_csr.xlsx'

excel_sheet: This represents the sheet name of the excel file. For an example: 'Tracking DS', '__ref'

dataset: This represents the column name containing all Lead programs name in the certain sheet of excel file. For an example: “Program Name”

lead_date: This represents the lead program modification date which this qc_date function will return. lead_date is in datetime format of ‘yyyy-mm-dd hh:mm:ss’. For an example: “2023-04-19 03:42:08”

qc_pgm: This represents the column name containing all QC programs name in the certain sheet of excel file. For an example: “QC program Name”

qc_output_date: This represents the QC output (rtf/pdf format) modification date which this qc_date function will return. qc_output_date is in datetime format of ‘yyyy-mm-dd hh:mm:ss’. For an example: “2023-04-19 04:42:08”

qc_status: This represents the final QC output (rtf/pdf format) status which this qc_date function will return. qc_status contains “Pass QC”, "not pass" options

READ EXCEL FILE：TRANSFORMED EXCEL FILE INTO DATA FRAME FORMAT

By using panda package, information of excel sheet was transformed into data frame format for the further data manipulation. The missing value from the loaded data will be filled with an empty string value “” by using function (data.fillna(value="",inplace=True)). The subsequent iteration program will reset the name of each column to avoid any missing lead rows in this excel sheet.

```
##################loading excel file and name column with dataset ##################
path1=os.path.expanduser('~')
excl = load_workbook(path1+studypath+path_excel, read_only=False,keep_vba=True )
data = pd.DataFrame(excl[excel_sheet].values)
data.fillna(value="",inplace=True )
  #reset column name
  for indexs in data.index:
     df_row=data.iloc[[indexs]]
     if   dataset in df_row.values:
     df =data.set_axis(data.iloc[indexs], axis='columns', copy=False)
```

GET MODIFICATION DATETIME ： LEAD SAS7BDAT DATASET AND RTF/PDF FILE OF QC OUTPUT RESULTS

For qc output result. qc_pgm_rtf stands for file name and path of QC output results, which will be determined by file_type (param 1 in qc_data funtion). If file_type is 'DATA', then qc_pgm_rtf will be ‘qc_pgm’.rtf (Column 2 in excel, param 7 in qc_data funtion), If file_type is 'OUTPUT', then qc_pgm_rtf will be "qc_"+ ‘qc_pgm’.rtf (Column 2 , param 7). QC output results modification datetime will be produced by qc_date() function and return it to corresponding column (Column 4 , param 8)

Refers to lead program. row[dataset] (Column 1 , param 5) represent sas7bdat dataset lead program produced. If file_type is 'DATA' then row[dataset] represent either SDTM or ADAM

data set, which will be determined by row[dataset] start with "ad" or not. The modification datetime of lead sas7bdat dataset will be produced by qc_date() function and return it to corresponding column (Column 3 , param 6)

Use os.path.exists() to determine certain file exist or not, so that either PDF or RTF file could be determine by python code itself.

```
######################checking lead program modified date#######################
df[lead_date]=''
df[qc_status]=''
df[qc_output_date]=''

if file_type == 'DATA':
    df[qc_pgm+'_rtf']=path1+studypath+"QC/OUTPUT/"+df[qc_pgm].str.replace('.sas', '' ,regex=True).str.lower()+'.rtf'
    df[qc_pgm+'_PDF']=path1+studypath+"QC/OUTPUT/"+df[qc_pgm].str.replace('.sas', '' ,regex=True).str.lower()+'.pdf'

elif file_type == 'OUTPUT':
    df[qc_pgm+'_rtf']=path1+studypath+"QC/OUTPUT/"+"qc_"+df[qc_pgm].str.replace('.sas', '' ,regex=True).str.lower()+'.rtf'
     df[qc_pgm+'_PDF']=path1+studypath+"QC/OUTPUT/"+"qc_"+df[qc_pgm].str.replace('.sas', '' ,regex=True).str.lower()+'.pdf'

for index, row in df.iterrows():
  if file_type == 'DATA':
        #for SDTM dataset
        if row[dataset] not in ("", None) and row[dataset].lower().startswith("ad")==0:
            row[lead_date]=path1+studypath+"SDTM/DATA/"+row[dataset].lower().strip()+'.sas7bdat'

        #for ADAM dataset
        elif row[dataset] not in ("", None) and row[dataset].lower().startswith("ad"):
            row[lead_date]=path1+studypath+"ADS/DATA/"+row[dataset].lower().strip()+'.sas7bdat'

  elif file_type == 'OUTPUT':
        if row[dataset] not in ("", None) :
            row[lead_date]=path1+studypath+"REPORT/PGM/"+row[dataset].lower().strip()+'.sas'

#for lead program date of modification
  try:
      if row[dataset] == dataset:
          df.at[index,lead_date]=lead_date
      elif row[dataset] == "":
          df.at[index,lead_date]=""
      elif os.path.exists(row[lead_date])!=0:
          df.at[index,lead_date]=time.strftime('%Y-%m-%d %H:%M:%S',time.localtime(os.path.getmtime(row[lead_date])))
      elif os.path.exists(row[lead_date])==0:
          df.at[index,lead_date]=""
  except:
          df.at[index,lead_date]="#NA#"

#for qc results date of modification

   try:
      if row[dataset] == dataset:
            df.at[index,qc_output_date]=qc_output_date
       elif row[dataset] == "":
            df.at[index,qc_output_date]=""
       elif os.path.exists(row[qc_pgm+'_rtf'])!=0:
            df.at[index,qc_output_date]=time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(os.path.getmtime(row[qc_pgm+'_rtf'])))
       elif os.path.exists(row[qc_pgm+'_rtf'])==0 and os.path.exists(row[qc_pgm+'_PDF'])!=0:
            df.at[index,qc_output_date]=time.strftime('%Y-%m-%d %H:%M:%S',time.localtime(os.path.getmtime(row[qc_pgm+'_rtf'])))

  except:
```

```
        df.at[index,qc_output_date]="#NA#"
```

For file_type= 'DATA':

Dataset Name	Variable	Priority (Optional)	Quality Control type	CDISC Compliance Check Required	Statistical Programmer Name	Program Name	Date Ready for QC	Internal Reviewer Name	QC program Name	Modification Requester	Modification against QC plan	Comments of findings of QC	Date of final QC	QC Status
INT_USUBJID					QQ Li	int_usubjid.sas	2023-04-19 03:28:17	Liu	qc_int_usubjid.sas				2023-04-19 03:43:23	Pass QC
INT_VISITS					QQ Li	int_visits.sas	2023-04-19 03:29:51	Liu	qc_int_visits.sas				2023-04-19 03:43:23	Pass QC
DM SUPPDM	ALL VARIABLES		Basic, R	Y	QQ Li	dm.sas	2023-04-19 03:30:17	Liu	qc_dm.sas				2023-04-19 03:43:37	Pass QC
SE SUPPSE	ALL VARIABLES		Basic, R	Y	QQ Li	se.sas	2023-04-19 03:30:28	Liu	qc_se.sas				2023-04-19 03:43:44	Pass QC
SV SUPPSV	ALL VARIABLES		Basic, R	Y	QQ Li	sv.sas	2023-04-19 03:30:39	Liu	qc_sv.sas				2023-04-19 03:43:57	Pass QC
CM SUPPCM	ALL VARIABLES		Basic, R	Y	QQ Li	cm.sas	2023-04-19 03:31:33	Liu	qc_cm.sas				2023-04-19 03:52:30	Pass QC
EC SUPPEC	ALL VARIABLES		Basic, R	Y	QQ Li	ec.sas	2023-04-19 03:32:07	Liu	qc_ec.sas				2023-04-19 03:44:53	Pass QC
EX SUPPEX	ALL VARIABLES		Basic, R	Y	QQ Li	ex.sas	2023-04-19 03:32:50	Liu	qc_ex.sas				2023-04-19 03:52:33	Pass QC

```
qc_date('DATA', 'studypath', 'QC_tracking.xlsm','Tracking DS', 'Dataset Name', 'Date Ready for QC','QC program Name','Date of final QC','QC Status')
```

For file_type= 'OUTPUT':

esultpgname	filename	sect_app	tit_CSR	Programmer N	Internal Rev	QC program Name	KRM	Intext	batch	Date Ready for QC	QC Status	Date of QC output
Y ran_batch_1	ran_batch_1	16.1.6	Listing of partici	Zou	QQ Li	qc_ran_batch_1				2023-05-04 05:19:58	Pass QC	2023-05-10 11:57:08
Y dis_dispo_ori_s_t	dis_dispo_ori_s_t	16.2.1	Participant dispos	Zou	QQ Li	qc_dis_dispo_ori_s_t	Y	Y	Y	2023-04-10 11:59:51	Pass QC	2023-04-27 09:21:28
Y dis_dispo_rev_s_t	dis_dispo_rev_s_t	16.2.1	Participant dispos	Zou	QQ Li	qc_dis_dispo_rev_s_t	Y	Y		2023-04-10 11:59:24	Pass QC	2023-04-27 09:21:28
Y dis_hold_2017_s_l	dis_hold_2017_s_l	16.2.1	Listing of clinica	Zou	QQ Li	qc_dis_hold_2017_s_l			Y	2023-05-09 12:27:36	Pass QC	2023-05-10 12:23:18
Y dis_hold_2020_s_l	dis_hold_2020_s_l	16.2.1	Listing of volunte	Zou	QQ Li	qc_dis_hold_2020_s_l			Y	2023-05-05 08:23:17	Pass QC	2023-05-10 11:57:02
Y dis_screened_cntrysite_a_t	dis_screened_cntrysite_a_t	16.2.1	Distribution of al	Zou	QQ Li	qc_dis_screened_cntrysite_a_t			Y	2022-12-19 03:29:08	Pass QC	2023-05-10 11:57:02
Y dis_endtrt_imp_ori_s_l	dis_endtrt_imp_ori_s_l	16.2.1	Listing of partici	Zou	QQ Li	qc_dis_endtrt_imp_s_l			Y	2023-04-25 11:30:59	Pass QC	2023-05-10 11:57:01
Y dis_endtrt_imp_rev_s_l	dis_endtrt_imp_rev_s_l	16.2.1	Listing of partici	Zou	QQ Li	qc_dis_endtrt_imp_s_l			Y	2023-04-27 04:44:20	Pass QC	2023-05-10 11:57:02
Y dev_critmaj_a_t	dev_critmaj_ori_a_t	16.2.2	Major protocol dev	Zou	QQ Li	qc_dev_critmaj_a_t		Y		2023-04-13 10:23:12	Pass QC	2023-04-27 09:21:28
Y dev_critmaj_a_t	dev_critmaj_rev_a_t	16.2.2	Major protocol dev	Zou	QQ Li	qc_dev_critmaj_a_t		Y		2023-04-13 10:23:12	Pass QC	2023-04-27 09:21:28
Y dev_critmaj_a_l	dev_critmaj_ori_a_l	16.2.2	Listing of partici	Zou	QQ Li	qc_dev_critmaj_a_l				2023-04-25 11:49:22	Pass QC	2023-05-10 11:57:02
Y dev_critmaj_a_l	dev_critmaj_rev_a_l	16.2.2	Listing of partici	Zou	QQ Li	qc_dev_critmaj_a_l				2023-04-25 11:49:22	Pass QC	2023-05-10 11:57:02
N dev_enrol_nottrt_p_l	dev_enrol_nottrt_p_l	16.2.2	Listing of partici	Zou	QQ Li	qc_dev_enrol_nottrt_p_l				2023-03-15 10:49:51	Pass QC	2023-04-27 09:21:28
Y dis_ana_pop_a_t	dis_ana_pop_a_t	16.2.1	Analysis Sets	Zou	QQ Li	qc_dis_ana_pop_a_t		Y		2023-04-26 09:25:04	Pass QC	2023-04-27 09:29:20

```
qc_date('OUTPUT', 'studypath', '_ref_csr.xlsx','__ref', 'pgmname', 'Date Ready for QC','filename','Date of QC output','QC Status')
```

PRODUCED QC OUTPUT RESULTS：EXTRACT TEXT FROM RTF/PDF AND COMPARE THEM WITH KEY WORDS

QC outputs results will be read by open (file, mode='r'). Each line will be extracted from RTF file or Each line per page will be extracted by PDFReader for PDF. Then extracted text from QC outputs will be compared with key wards. we specify the four words "Conflicting Types", in Common: 0","but not in", “Unequal” sensitive. If any of these words are detected in a result file, the verification result is “Not pass”, otherwise, “Pass”.

```
##############################checking output results##############################
#Define the key words for determine qc comparation results
regx ="intb"
strg1="Conflicting Types"
strg2="in Common: 0"
strg3="but not in"
strg4="Unequal:"

try:
    if row[dataset] == dataset    :
        df.at[index,qc_status] =qc_status
     elif row[dataset]    == ""    :
```

```
        df.at[index,qc_status]=""
    elif row[dataset] != "" and os.path.exists(row[qc_pgm+'_rtf'])==0 and os.path.exists(row[qc_pgm+'_PDF']) ==0:
        df.at[index,qc_status] =""

#for RTF file
      elif os.path.exists(row[qc_pgm+'_rtf'])!=0 :
        file1 = open(row[qc_pgm+'_rtf'], mode='r' )
        for line in file1:
           df.at[index,qc_status]="Pass QC"
              if search(strg1, line.strip()):
                    df.at[index,qc_status]="not pass "
                    break
              elif search(strg2, line.strip()):
                    df.at[index,qc_status]="not pass"
                    break
              elif search(strg3, line.strip()):
                    df.at[index,qc_status]="not pass"
                    break
              elif search(strg4, line.strip()):
                    match=(re.search(strg4,line.strip())).end()
                    if line[match+1] != '0':
                        df.at[index,qc_status]="not pass"
                        break

#for PDF file
      elif os.path.exists(row[qc_pgm+'_rtf'])==0 and os.path.exists(row[qc_pgm+'_PDF']) !=0:
        PDFFileObj = open(row[qc_pgm+'_PDF'],'rb')
        PDFReader = PyPDF2.pdfFileReader(PDFFileObj)
        pages = PDFReader.numPages

        for i in range(pages):
           pageObj = PDFReader.getPage(i)
           text = pageObj.extractText().split("\n")
           for line in text:
              df.at[index,qc_status]="Pass QC"
              if search(strg1, line.strip()):
                  df.at[index,qc_status]="not pass "
                  break
              elif search(strg2, line.strip()):
                  df.at[index,qc_status]="not pass"
                  break
              elif search(strg3, line.strip()):
                  df.at[index,qc_status]="not pass"
                  break
              elif search(strg4, line.strip()):
                  match=(re.search(strg4,line.strip())).end()
                  if line[match+1] != '0':
                      df.at[index,qc_status]="not pass"
                      break
                  if df.at[index,qc_status] !="Pass QC":
                      break
         PDFFileObj.close()

    except:
          df.at[index,qc_status]="#NA"
df.drop([qc_pgm+'_rtf',qc_pgm+'_PDF' ],inplace=True,axis=1)
```

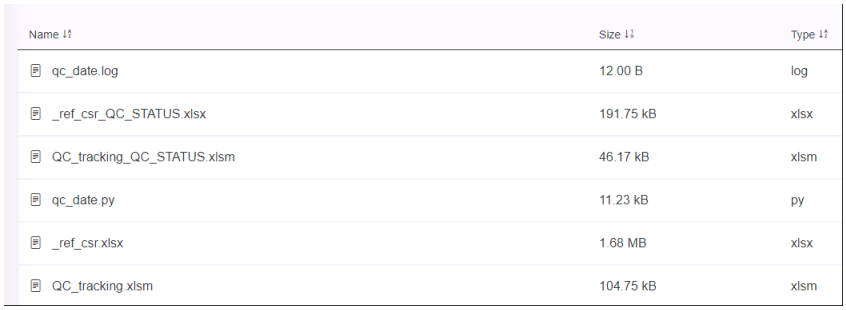
COPY ORIGINAL AND NEW SHEET INTO NEW EXCEL XXX_QC_STATUS

```
##################copy original and new sheet into new excel xxx_QC_STATUS#########
```

```
src = path1+studypath+path_excel
dest = path1+studypath+path_excel.split('.')[0]+'_QC_STATUS.'+path_excel.split('.')[-1]
path = shutil.copyfile(src,dest)
# for .xlsm file, keep VBA
if ".xlsm" in path_excel:
   with pd.ExcelWriter(dest, engine="openpyxl",mode="a",if_sheet_exists="replace", , engine_kwargs={"keep_vba": True}) as write
:
            df.to_excel(writer,qc_status, startrow=0, startcol=0, index=False,header= False)
            wb = openpyxl.load_workbook(dest,read_only=False,keep_vba=True)

# for excel file not is .xlsm , do not keep VBA
else:
   with pd.ExcelWriter(dest, engine="openpyxl",mode="a",if_sheet_exists="replace") as writer:
            df.to_excel(writer,qc_status, startrow=0, startcol=0, index=False,header= False)
            wb = openpyxl.load_workbook(dest,read_only=False)

sheet=wb[excel_sheet]
sheet2=wb[qc_status]

#for final value qc_status column equals to 'not pass' then highlight with red
for cell in row:
   if cell.value == qc_status:
      key_column=cell.column
      break
for key_column, row in enumerate(sheet2.iter_rows()):
    for j, cell in enumerate(row):
        if cell.value in ['not pass']:
            cell.fill = PatternFill(patternType='solid',start_color=Color(index=10))

wb.save(path1+studypath+path_excel)
wb.close()
print('Done.')
```

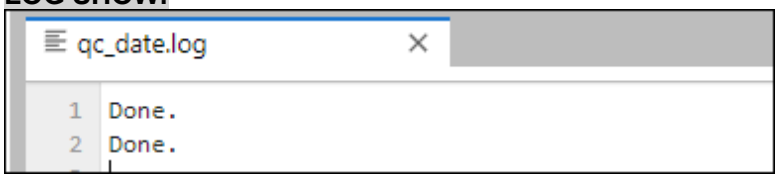
BATCH RUN IN LINUX SYSTEM

Eventually, the following checking files will be produced. _ref_cst.xlsx is the excel file contains all output program and qc information. QC_tracking.xlsm is the excel file contains all SDTM/ADAM program and qc information. Excel files with suffix " _QC_STATUS" are new files with validation results and modification datetime information inside.

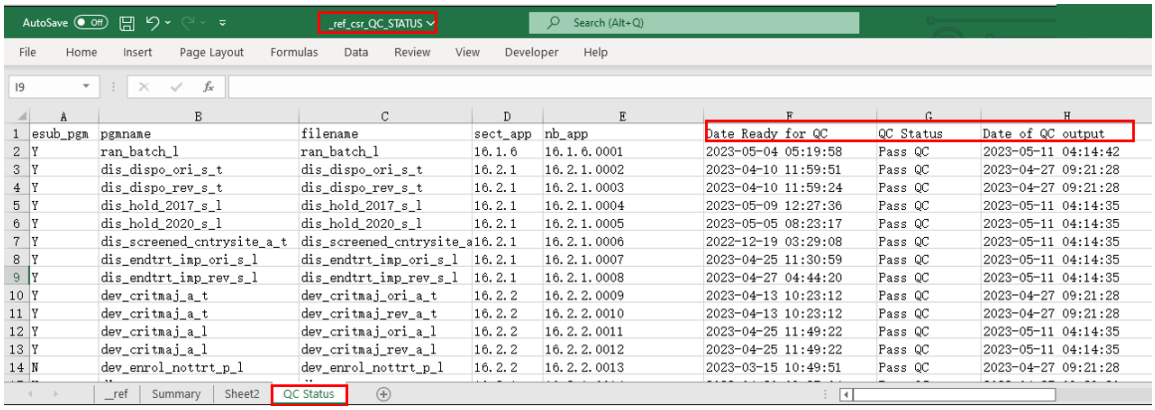


Name	Size	Type
qc_date.log	12.00 B	log
_ref_csr_QC_STATUS.xlsx	191.75 kB	xlsx
QC_tracking_QC_STATUS.xlsm	46.17 kB	xlsm
qc_date.py	11.23 kB	py
_ref_csr.xlsx	1.68 MB	xlsx
QC_tracking.xlsm	104.75 kB	xlsm

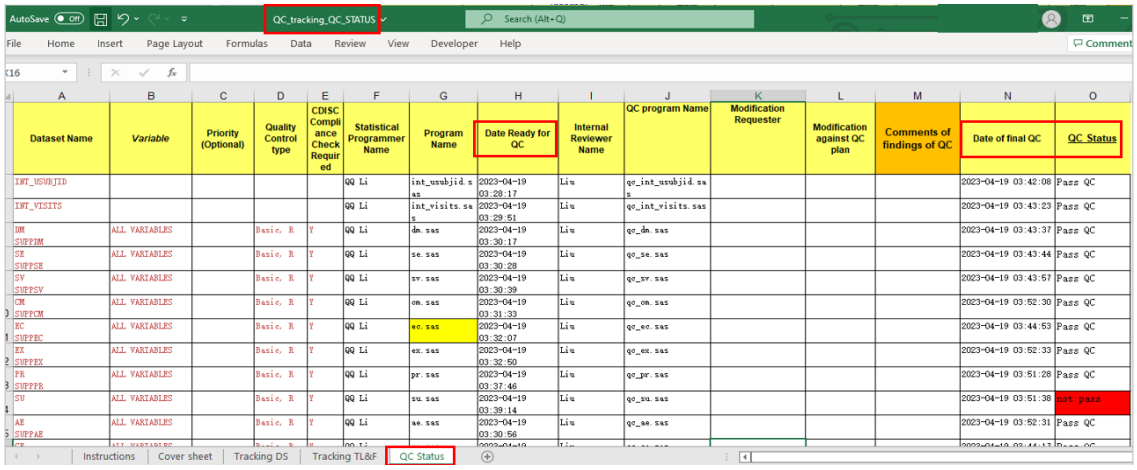
LOG SHOW:



EXCEL FILES WITH SUFFIX" _QC_STATUS":



	A	B	C	D	E	F	G	H
1	esub_pgm	pgmname	filename	sect_app	nb_app	Date Ready for QC	QC Status	Date of QC output
2	Y	ran_batch_l	ran_batch_l	16.1.6	16.1.6.0001	2023-05-04 05:19:58	Pass QC	2023-05-11 04:14:42
3	Y	dis_dispo_ori_s_t	dis_dispo_ori_s_t	16.2.1	16.2.1.0002	2023-04-10 11:59:51	Pass QC	2023-04-27 09:21:28
4	Y	dis_dispo_rev_s_t	dis_dispo_rev_s_t	16.2.1	16.2.1.0003	2023-04-10 11:59:24	Pass QC	2023-04-27 09:21:28
5	Y	dis_hold_2017_s_l	dis_hold_2017_s_l	16.2.1	16.2.1.0004	2023-05-09 12:27:36	Pass QC	2023-05-11 04:14:35
6	Y	dis_hold_2020_s_l	dis_hold_2020_s_l	16.2.1	16.2.1.0005	2023-05-05 08:23:17	Pass QC	2023-05-11 04:14:35
7	Y	dis_screened_cntrysite_a_t	dis_screened_cntrysite_a	16.2.1	16.2.1.0006	2022-12-19 03:29:08	Pass QC	2023-05-11 04:14:35
8	Y	dis_endtrt_imp_ori_s_l	dis_endtrt_imp_ori_s_l	16.2.1	16.2.1.0007	2023-04-25 11:30:59	Pass QC	2023-05-11 04:14:35
9	Y	dis_endtrt_imp_rev_s_l	dis_endtrt_imp_rev_s_l	16.2.1	16.2.1.0008	2023-04-27 04:44:20	Pass QC	2023-05-11 04:14:35
10	Y	dev_critmaj_a_t	dev_critmaj_ori_a_t	16.2.2	16.2.2.0009	2023-04-13 10:23:12	Pass QC	2023-04-27 09:21:28
11	Y	dev_critmaj_a_t	dev_critmaj_rev_a_t	16.2.2	16.2.2.0010	2023-04-13 10:23:12	Pass QC	2023-04-27 09:21:28
12	Y	dev_critmaj_a_l	dev_critmaj_ori_a_l	16.2.2	16.2.2.0011	2023-04-25 11:49:22	Pass QC	2023-05-11 04:14:35
13	Y	dev_critmaj_a_l	dev_critmaj_rev_a_l	16.2.2	16.2.2.0012	2023-04-25 11:49:22	Pass QC	2023-05-11 04:14:35
14	N	dev_enrol_nottrt_p_l	dev_enrol_nottrt_p_l	16.2.2	16.2.2.0013	2023-03-15 10:49:51	Pass QC	2023-04-27 09:21:28



Dataset Name	Variable	Priority (Optional)	Quality Control type	CDISC Compliance Check Required	Statistical Programmer Name	Program Name	Date Ready for QC	Internal Reviewer Name	QC program Name	Modification Requester	Modification against QC plan	Comments of findings of QC	Date of final QC	QC Status
INT_USUBJID					QQ Li	int_usubjid.sas	2023-04-19 03:28:17	Liu	qc_int_usubjid.sas				2023-04-19 03:42:08	Pass QC
INT_VISITS					QQ Li	int_visits.sas	2023-04-19 03:29:51	Liu	qc_int_visits.sas				2023-04-19 03:43:23	Pass QC
DM SUPPDM	ALL VARIABLES		Basic, R	Y	QQ Li	dm.sas	2023-04-19 03:30:17	Liu	qc_dm.sas				2023-04-19 03:43:37	Pass QC
SE SUPPSE	ALL VARIABLES		Basic, R	Y	QQ Li	se.sas	2023-04-19 03:30:28	Liu	qc_se.sas				2023-04-19 03:43:44	Pass QC
SV SUPPSV	ALL VARIABLES		Basic, R	Y	QQ Li	sv.sas	2023-04-19 03:30:39	Liu	qc_sv.sas				2023-04-19 03:43:57	Pass QC
CM SUPPCM	ALL VARIABLES		Basic, R	Y	QQ Li	cm.sas	2023-04-19 03:31:33	Liu	qc_cm.sas				2023-04-19 03:52:30	Pass QC
EC SUPPEC	ALL VARIABLES		Basic, R	Y	QQ Li	ec.sas	2023-04-19 03:32:07	Liu	qc_ec.sas				2023-04-19 03:44:53	Pass QC
EX SUPPEX	ALL VARIABLES		Basic, R	Y	QQ Li	ex.sas	2023-04-19 03:32:50	Liu	qc_ex.sas				2023-04-19 03:52:33	Pass QC
PR SUPPPR	ALL VARIABLES		Basic, R	Y	QQ Li	pr.sas	2023-04-19 03:37:46	Liu	qc_pr.sas				2023-04-19 03:51:28	Pass QC
SU	ALL VARIABLES		Basic, R	Y	QQ Li	su.sas	2023-04-19 03:39:14	Liu	qc_su.sas				2023-04-19 03:51:38	not pass
AE SUPPAE	ALL VARIABLES		Basic, R	Y	QQ Li	ae.sas	2023-04-19 03:30:56	Liu	qc_ae.sas				2023-04-19 03:52:31	Pass QC

CONCLUSION

While a SAS® programmer's activity is to develop SAS program and delivery clinical trial report, using Python could bring facilitation to our validation task and further enable the quality of report. A new method of automatic checking QC validation results by a dedicated python program was introduced in this article. This tool has glorious features, first it could gather information from cross-format files which include RTF, PDF file, sas7bdat data set. Second, it is able to create a new excel to keep the modification date of lead and qc file, the QC validation result, and remains the original VBA macro. Third, this python code is compatible with the various platform, and this code could be modified by abundant python packages to satisfy specific requirements of different studies in the future.

This python tool help SAS® programmer deliver high qualified QC validation results efficiently, and bring conveniences to programmer daily work.

REFERENCES

1. Zhao C. How to read RTF files into SAS® datasets?2017; Paper 64. Available from: https://www.lexjansen.com/pharmasug-cn/2017/CC/PharmaSUG-China-2017-CC07.pdf.
2. Wooding N. EXTRACTING DATA FROM PDF FILES2005; Paper SER10_05. Available from: http://www8.sas.com/scholars/05/SESUG_05/Proceedings/2005/Serendipity/SER10_05.pdf.
3. Yonghong Hao CJW, China Yulin Li,China Jingfang Wu. An Automatic QC Status Tracking and

Management System 2016; Paper 47. Available from: https://www.lexjansen.com/pharmasug-cn/2016/CR/PharmaSUG-China-2016-CR02.pdf.
4. Hassan RHR. How to Extract Specific Data from PDF to Excel Using VBA [updated Mar 22, 2023 10. Available from: https://www.exceldemy.com/extract-specific-data-from-PDF-to-excel-using-vba/.
5. Sandeep Bhandari FC. Difference Between Adobe Reader and Adobe Acrobat [updated April 28, 2023; cited 2023 15MAY]. Available from: https://www.bing.com/ck/a?!&&p=9c03f528d385e8bcJmltdHM9MTY4MzUwNDAwMCZpZ3VpZD0xOWVmNmVhNC01YzkzLTZhNTYtMzI1OC03YzMyNWRmNTZiZWQmaW5zaWQ9NTQ1OQ&ptn=3&hsh=3&fclid=19ef6ea4-5c93-6a56-3258-7c325df56bed&psq=Adobe+Acrobat+and+Adobe+reader+difference&u=a1aHR0cHM6Ly9hc2thbnlkaWZmZXJlbmNlLmNvbS9kaWZmZXJlbmNlLWJldHdlZW4tYWRvYmUtcmVhZGVyLWFuZC1hZG9iZS1hY3JvYmF0Lw&ntb=1.
6. VBA Adobe Acrobat [Available from: https://vbaplanet.com/appacrobat.php.
7. Kremer KH. Adobe Acrobat and VBA – An Introduction [updated March 4, 2009. Available from: http://khkonsulting.com/2009/03/adobe-acrobat-and-vba-an-introduction/

CONTACT INFORMATION

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

Qianqian Li
Research and development (R&D) operations, Sanofi (Beijing) Pharmaceutical Co. Ltd.
Qianqian1.Li@sanofi.com
Chaoyang District, Beijing 100022, China