PharmaSUG China 2014 - Paper PO08

SAS2VBA2SAS: Automated solution to string truncation in PROC IMPORT

Amarnath Vijayarangan, Genpact, India

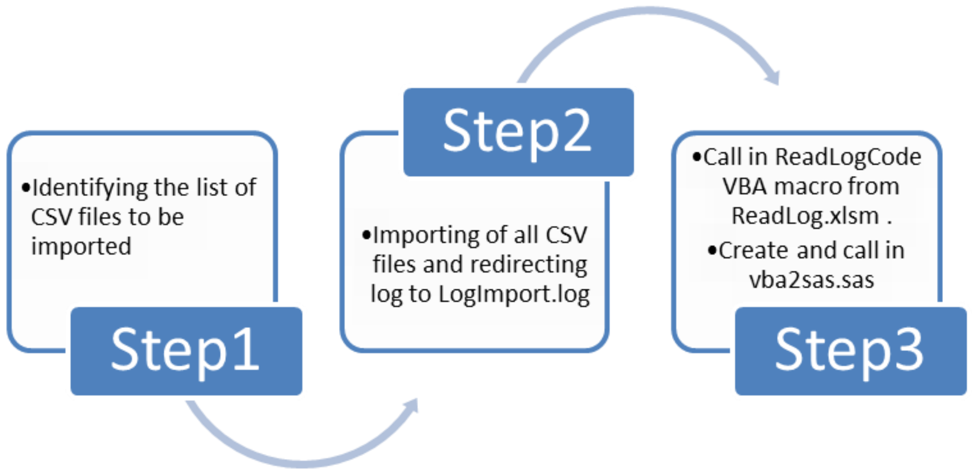
ABSTRACT

SAS PROC IMPORT is one of the most commonly and widely used approach to convert flat files that are native to other database engines into SAS datasets. Nevertheless, this procedure also has its own flaws. The most commonly encountered issue is that PROC IMPORT might not properly assign the length for character variables as it fixes the length of a particular character variable based on the initial few observations which leads to string truncation. The proposed approach delivers a solution that combines SAS and Excel VBA to resolve the problem by using the data step code generated in the log while importing flat files without any manual intervention.

INTRODUCTION

Use of flat files such as CSV and other text files are part of everyday task of a programmer. Nevertheless, these files cannot be used directly in SAS, one needs to convert them into a SAS dataset before processing and this task is tedious especially when the dataset has many character variables , each varying in length. Using PROC IMPORT either through the Import Wizard or by submitting a code is a quick fix for this, but this also comes with its own in built flaw. PROC IMPORT assigns the length for a character variable based on the first twenty observations that it encounters. The guessing row statement comes in handy to increase the number of rows that PROC IMPORT considers before assigning a length to the character variables. But, the use of guessing row is a universal statement for that dataset being processed and hence will be applicable for the entire variables in the dataset eventually leading to increase in time and disk space utilization. The current available approach is to write a manual data step with an infile statement or modifying the data step generated through the proc import, but this becomes tedious and time consuming task when one has too many variables. This paper suggests a solution taking into the current flaws in mind. The approach suggested here uses SAS along with Excel VBA to overcome these flaws.

APPROACH


•Identifying the list of CSV files to be imported
Step1
Step2
•Importing of all CSV files and redirecting log to LogImport.log
•Call in ReadLogCode VBA macro from ReadLog.xlsm .
•Create and call in vba2sas.sas
Step3

Step 1 Uses 'D' functions to identify the list of flat files that needs to be imported. The function used here access the path specified in the filepath parameter given by the user and makes a list of all the files that needs to accessed and hence imported.

Step2 imports all the files identified in step1 using D function. Once the import procedure is executed, this phase also ensures that the procedures' s log is redirected into a master log file named Logimport.log facilitates the VBA code to access the log even before the program execution is completed.

Step3 uses the Logimport.log (contains the log of the original PROC IMPORT) file generated in step 2 as an input to identify the optimal length for the character variables. In simple words the VBA code reads the flat files that is imported and finds out the maximum length for a character variable across observations. The VBA code combines the Logimport.log file (copies the actual data step program from the log file) along with optimal length found for a character variable across observations and creates a vba2sas program containing data steps with a modified format and informat attribute of the character variables. Then vba2sas program is executed through %include statement.

Overall the code is executed in two stages. First, PROC IMPORT is run and the log of the IMPORT procedure is copied into a file named Logimport.log. Then the VBA code scans the flat files being imported and finds out the maximum length for each character variable across observation. The VBA code uses the data step part of the program in the log and modifies the FORMAT and the INFORMAT length with the new length found and then the data step is executed again. In this way we ensure that all the character variables are properly imported and we overcome the issue of string truncation.

VBA SET UP

Below are the specifications one needs to set before executing the program.

1. Create workbook with the name **"ReadLog.xlsm"** with the VBA macro provided in the appendix

2. Ensure the **"ReadLog.xlsm"** has the tab named "**Output"**

3. Save the file in the location specified in the filepath parameter

SAS2VBA2SAS MACRO

```
%let filepath=D:\;

%macro sas2vba2sas;

filename fpth "&filepath";

data FileList;
       length FileName $ 50;
       dval=dopen('fpth');
       if dval> 0 then do;
          count=dnum(dval);
          do i= 1 to count;
               FileName=dread(dval,i);
               if upcase(scan(FileName,2,'.'))='CSV' then output;
          end;
       end;
       keep FileName;
run;

proc printto log="&filepath/LogImport.log" new;
run;

libname datapath "&filepath";

%let crb=%sysfunc(open(FileList));
%let i=1;
%do %while(not %sysfunc(fetch(&crb)));
       %let Fname=%sysfunc(getvarc(&crb, %sysfunc(varnum(&crb, FileName))));
       proc import datafile= "&filepath\&FName" out=datapath.data&i dbms=csv replace;
```

```
              getnames=yes; datarow=2;
run;
%let i=%eval(&i+1);
%end;
%let crb=%sysfunc(close(&crb));
proc printto;
run;

options noxwait noxsync;
data _null_;
       rc=system('START EXCEL');
       rc=sleep(5);
run;

filename cmds dde 'EXCEL|SYSTEM';

data _null_;
       file cmds;
       put "[open(""&filepath\ReadLog.xlsm"")]";
       put '[run("ReadLogCode")]';
       put "[quit()]";
run;
quit;

data _null_;
       rc=sleep(5);
run;
%include "&filepath/vba2sas.sas";

%mend;

%sas2vba2sas
```

CONCLUSION

The proposed approach does not restrict itself to a limited number of observations to specify the length of a particular character variable. In this way the solution overcomes the problem of string truncation associated with the proc import even after using guessing rows which again restrict only to the exact user specified number of observations. Thus, this approach ensures the quality of the data is not compromised.

CONTACT INFORMATION

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

Name: Amarnath Vijayarangan
Enterprise: Genpact
Address: Block B Salarpuria Soft Zone Outer Ring Road, Bellandur
City, State ZIP: Bangalore, Karnataka, 560103
E-mail: amarnath7@gmail.com

APPENDIX

```
Public LogFolderName As String
Public Sub ReadLogCode()
Dim LogFileName As String
Dim FileNum As Integer, tLine As String, lngIndex As Long, lngLp As Long, splReadLog
Dim wbkSrc As Workbook, intLp As Integer
     Sheets("Output").Activate

    Sheets("output").Range("A1").Value =
"=LEFT(CELL(""filename"",A1),FIND(""["",CELL(""filename"",A1),1)-1)"

    LogFileName = Sheets("output").Range("A1").Value & "LogImport.log"
    LogFolderName = Sheets("output").Range("A1").Value

    ActiveSheet.Cells.Clear
    lngIndex = 1
    FileNum = FreeFile
    Open LogFileName For Input Access Read Shared As #FileNum
    Do While Not EOF(FileNum)
        Line Input #FileNum, tLine
        ActiveSheet.Range("A1").Value = ActiveSheet.Range("A1").Value & tLine &
Chr(10)
    Loop
    Close #FileNum

    If ActiveSheet.Range("A1").Value <> "" Then
            lngIndex = 1
            splReadLog = Split(ActiveSheet.Range("A1").Value, Chr(10))


            For lngLp = 0 To WorksheetFunction.CountA(splReadLog) - 1

                If (IsNumeric(Trim(Left(splReadLog(lngLp), 8))) = True) And InStr(1,
Trim(splReadLog(lngLp)), "SAS") = 0 Then
                    ActiveSheet.Range("B" & lngIndex).Value =
Trim(Mid(splReadLog(lngLp), 6))
                    lngIndex = lngIndex + 1


                ElseIf InStr(1, Trim(UCase(splReadLog(lngLp))), "LIBNAME") > 0 Then
                    ActiveSheet.Range("B" & lngIndex).Value =
Trim(Mid(splReadLog(lngLp), 6))
                    lngIndex = lngIndex + 1

                ElseIf InStr(1, Trim(Left(splReadLog(lngLp), 8)), "!") > 0 And
IsNumeric(Replace(Trim(Left(splReadLog(lngLp), 5)), "!", "")) = True Then
                    ActiveSheet.Range("B" & lngIndex).Value =
Trim(Mid(splReadLog(lngLp), 6))
                    lngIndex = lngIndex + 1

                End If

                Debug.Print Trim(Left(splReadLog(lngLp), 8))

            Next
    End If
    ActiveSheet.Range("A1").Value = ""
    Call editcode
    Call CreateSASfile
End Sub
Sub editcode()
Dim strpos As Long, path As String, wb As Workbook, lrow As Long
Dim CharNum As Integer, header As String, newnum As Long
    With ThisWorkbook.Sheets("Output")
    lrow = .Range("B100000").End(xlUp).Row
    For i = 1 To lrow
        If LCase(Left(.Cells(i, 2).Value, 6)) = "infile" Then
            strpos = InStr(7, .Cells(i, 2).Value, "delimiter", vbTextCompare)
            path = Right(.Cells(i, 2).Value, Len(.Cells(i, 2).Value) - 8)
            If strpos > 0 Then
            path = Left(path, strpos - 11)
            End If
            path = Replace(path, "'", "")
            Set wb = Workbooks.Open(path)

            ThisWorkbook.Activate
            j = i
            Do Until LCase(.Cells(j, 2).Value) = "run;"
                If LCase(Left(.Cells(j, 2).Value, 8)) = "informat" Then

                    CharNum = InStr(1, .Cells(j, 2).Value, "$", vbTextCompare)
                    If CharNum <> 0 Then
                        header = Left(.Cells(j, 2).Value, CharNum - 2)
                        header = Right(header, Len(header) - 9)
                        Set rngfound = wb.ActiveSheet.Rows("1:1").Find(What:=header,
LookIn:=xlValues, LookAt:=xlPart)
                        If Not rngfound Is Nothing Then
                            wb.ActiveSheet.Range("R1").FormulaR1C1 = "=MAX(LEN(C" &
rngfound.Column & "))"
                            wb.ActiveSheet.Range("R1").FormulaArray =
wb.ActiveSheet.Range("R1").Formula
                            newnum = wb.ActiveSheet.Range("R1").Value
                            .Cells(j, 2).Value = "informat " & header & " $" & newnum
& ". ;"
                            wb.ActiveSheet.Range("R1").ClearContents
                        End If
                    End If
                End If
                If LCase(Left(.Cells(j, 2).Value, 6)) = "format" Then
                    CharNum = InStr(1, .Cells(j, 2).Value, "$", vbTextCompare)
                    If CharNum <> 0 Then
                        header = Left(.Cells(j, 2).Value, CharNum - 2)
                        header = Right(header, Len(header) - 7)
                        Set rngfound = wb.ActiveSheet.Rows("1:1").Find(What:=header,
LookIn:=xlValues, LookAt:=xlPart)
                        If Not rngfound Is Nothing Then
                            wb.ActiveSheet.Range("R1").FormulaR1C1 = "=MAX(LEN(C" &
rngfound.Column & "))"
                            wb.ActiveSheet.Range("R1").FormulaArray =
wb.ActiveSheet.Range("R1").Formula
                            newnum = wb.ActiveSheet.Range("R1").Value
                            .Cells(j, 2).Value = "format " & header & " $" & newnum &
". ;"
                            wb.ActiveSheet.Range("R1").ClearContents
                        End If
                    End If

                End If
                j = j + 1
            Loop
            wb.Close False
            i = j
        End If

    Next i
    End With
End Sub
Sub CreateSASfile()
Dim pth As String
pth = ThisWorkbook.path
Dim fs As Object
Set fs = CreateObject("Scripting.FileSystemObject")
Dim a As Object
Set a = fs.CreateTextFile(LogFolderName & "vba2sas.sas", True)
Dim sh As Worksheet
Dim strReadText As String
Set sh = ThisWorkbook.Sheets("Output")
Dim rng As Range
Set rng = sh.UsedRange
strReadText = GetTextFromRangeText(rng)
a.WriteLine (strReadText)
a.Close
End Sub
Function GetTextFromRangeText(ByVal poRange As Range) As String
    Dim vRange As Variant
    Dim sRet As String
    Dim i As Integer
    Dim j As Integer
    If Not poRange Is Nothing Then
        vRange = poRange
        For i = LBound(vRange) To UBound(vRange)
            For j = LBound(vRange, 2) To UBound(vRange, 2)
                sRet = sRet & vRange(i, j)
            Next j
            sRet = sRet & vbCrLf
        Next i
    End If
    GetTextFromRangeText = sRet
End Function
```