VBA Microsoft Access 2007 Macros to Import Formats and Labels to SAS®

Maria S. Melguizo Castro, Jerry R Stalnaker, and Christopher J. Swearingen

Biostatistics Program, Department of Pediatrics
University of Arkansas for Medical Sciences, Little Rock, AR

ABSTRACT

It is a common practice for statisticians and data analysts to import data from Microsoft Access into SAS for analysis. Although it is relatively straightforward to transfer data between the two programs, currently there is not an option to fully transfer descriptions or data formats into SAS. This paper proposes a solution using both Visual Basic for Applications (VBA) and SAS macros to address this issue and create a more complete SAS dataset with descriptions as variable labels and with appropriate data formats.

KEYWORDS: Data integration, Macros

INTRODUCTION

Microsoft (MS) Access 2007® description of field names in the database tables are usually employed when collecting data. These descriptions inherently provide a statistician or data analyst with meaningful labels to be used for SAS dataset variables. However, SAS v9.2 does not have the capability of importing descriptions from MS Access. Additionally, lookup fields are sometimes employed to create formats for the variables (e.g., 0=Male, 1=Female); there is also not an option in SAS to import those.

To convert variable labels and value formats from MS Access to SAS, a Visual Basic for Applications (VBA) macro was created in MS Access that generates a metadata table consisting of field names, descriptions, and table names stored within the database. This new table is subsequently imported into SAS and used to create labels for variables in SAS datasets. In order to convert the variable formats, VBA macro converted MS Access variable formats stored in field lists into a complete PROC FORMAT statement exported as a text file for implementation into SAS. The process is illustrated with an example database.

ACCESS DATABASE

An example of a MS Access database is presented below to mimic a data collection instrument typical of a research study. The database contains three tables: Demographics, Test and Questionnaire. Figure 1 presents the relationships between the tables. In this example, table Demographics captures unique study participant data (i.e. one row represents one unique record); unique identifier is linked to both tables Test and Questionnaire in a “1-to-Many” relationship (i.e. multiple observations are collected on each unique study participant). Inside a Design View of any table, variable descriptions can be located for as well as field lists defined for variable formats (Figure 2).
ACCESS VBA FOR VARIABLE LABELS

In the Visual Basic window from MS Access, a new Module was inserted to create a VBA macro. The following code creates a new table in the MS Access database that contains all of the field names, descriptions and names for all the tables in the database. This code should be run first before invoking the SAS code.

Running the macro CreateLabels generates the metadata table of variable labels; the macro calls the CreateTable code as a sub routine to initialize the metadata table. This VBA code can be used in any database. Please note that any database that contains a table Descriptions may need to be renamed as the VBA macro would delete such table.

Figure 1. Example Database and Relationships
' Auxiliary sub routine CreateTable

Sub CreateTable()
  Dim wspDefault As Workspace, dbs As Database
  Dim tdf As TableDef, fld1 As Field, fld2 As Field
  Dim idx As Index, fldIndex As Field

  Set wspDefault = DBEngine.Workspaces(0)
  ' Open Current database.
  Set dbs = CurrentDb()

  ' Create new table with three fields.
  Set tdf = dbs.CreateTableDef("Descriptions")
  Set fld1 = tdf.CreateField("Variablename", dbText)
  Set fld2 = tdf.CreateField("VariableDescription", dbText, 50)
  Set fld3 = tdf.CreateField("tblNAME", dbText)

  ' Append fields.
  tdf.Fields.Append fld1
  tdf.Fields.Append fld2
  tdf.Fields.Append fld3

  ' Append TableDef object.
  dbs.TableDefs.Append tdf
  dbs.TableDefs.Refresh
  Set dbs = Nothing
End Sub

Figure 2. Example of Lookup fields
' Main VBA subroutine
Sub CreateLabels()
    Dim tdf As TableDef
    Dim db As Database
    Dim fld As Field
    Dim prp As Property
    Dim pos As Integer
    Dim repl, replwith As String
    Dim repldesc, repldescwith As String
    Dim tname As String
    Dim rst As Recordset
    Dim i As Integer
    Dim SQL As String
    Dim DescrAvailable As Boolean

    Set db = CurrentDb()

    ' If the table descriptions exist then delete it
    For i = 0 To db.TableDefs.Count - 1
        If db.TableDefs(i).Name = "Descriptions" Then
            db.TableDefs.Delete "Descriptions"
            db.TableDefs.Refresh
            Exit For
        End If
    Next

    ' the macro CreateTable will create a table with the Descriptions
    Call CreateTable
    Set rst = db.OpenRecordset("Descriptions")

    For Each tdf In db.TableDefs
        For Each fld In tdf.Fields
            DescrAvailable = False
            For Each prp In fld.Properties
                If prp.Name = "Description" Then
                    With rst
                        .AddNew
                        !Variablename = fld.Name
                        !VariableDescription = prp.Value
                        !tblNAME = tdf.Name
                        .Update
                        DescrAvailable = True
                    End With
                End If
            Next
            If DescrAvailable = False Then
                With rst
                    .AddNew
                    !Variablename = fld.Name
                    !tblNAME = tdf.Name
                    .Update
                    DescrAvailable = True
                End With
            End If
        Next
        DescrAvailable = False
    Next
    db.Close
End Sub
SAS V9.2 CODE FOR VARIABLE LABELS

The first part of the SAS code was created to import the three data tables available in the database, as well as the Description table. The second part of the code takes the Description table and use it to create labels into the three newly created SAS data sets.

To make this code useful for any specific data analysis, the analyst would need to modify the database path, and create as many %ImportTables(), and %Labels() macro invocations as tables in the database.
ACCESS VBA FOR VARIABLE FORMATS

Capturing the field values for variables within the database is completed using the VBA macro `createFields`, which in turn calls two sub-functions `CreateTableFormats` and `CreateFormats`. This macro creates a table `Format` in the database containing all of the field labels and corresponding values for each variable described with a field list. The final VBA macro `CreateSASFormat` converts the data within table `Format` into a SAS PROC FORMAT chunk and exports the code chunk as a text file. This text file can then be utilized by an analyst in SAS. Please note that the current location for the text file is a simple directory listing to a user's local hard drive (i.e. C:\).

```vba
'---------------------------------------------------------------------
' Code to create formats
'---------------------------------------------------------------------
Sub CreateTableFormats()
    Dim wspDefault As Workspace, dbs As Database
    Dim tdf As TableDef, fld1 As Field, fld2 As Field, fld3 As Field, fld4 As Field
    Dim idx As Index, fldIndex As Field

    Set wspDefault = DBEngine.Workspaces(0)
    ' Open Current database.
    Set dbs = CurrentDb()

    ' Create new table with four fields.
    Set tdf = dbs.CreateTableDef("Format")
    Set fld1 = tdf.CreateField("Variablename", dbText)
    Set fld2 = tdf.CreateField("VariableNumericValue", dbText, 150)
    Set fld3 = tdf.CreateField("VariableFormat", dbText)
    Set fld4 = tdf.CreateField("tblNAME", dbText)

    ' Append fields.
    tdf.Fields.Append fld1
    tdf.Fields.Append fld2
    tdf.Fields.Append fld3
    tdf.Fields.Append fld4

    ' Append TableDef object.
    dbs.TableDefs.Append tdf
    dbs.TableDefs.Refresh
    Set dbs = Nothing
End Sub
```
Sub CreateFormats()
    Dim tdf As TableDef
    Dim db As Database
    Dim fld As Field
    Dim prp As Property
    Dim pos As Integer
    Dim repl, replwith As String
    Dim repldesc, repldescwith As String
    Dim tname As String
    Dim rst As Recordset
    Dim i As Integer
    Dim SQL As String
    Dim DescrAvailable As Boolean

    Dim strnname

    Set db = CurrentDb()

    ' If the Format exist then delete it
    For i = 0 To db.TableDefs.Count - 1
        If db.TableDefs(i).Name = "Format" Then
            db.TableDefs.Delete "Format"
            db.TableDefs.Refresh
            Exit For
        End If
    Next

    ' the macro CreateTable will create a table with the Formats
    Call CreateTableFormats
    Set rst = db.OpenRecordset("Format")

    For Each tdf In db.TableDefs
        For Each fld In tdf.Fields
            For Each prp In fld.Properties
                If prp.Name = "RowSource" Then
                    strname = prp.Value
                    fldname = fld.Name
                    tdfname = tdf.Name
                    Call createFields(strname, tdfname, fldname)
                End If
            Next
        Next
    Next
    db.Close
End Sub
Public Function createFields(strname, tdfname, fldname)
    Dim rst As Recordset
    Dim db As Database

    Set db = CurrentDb()
    Set rst = db.OpenRecordset("Format")

    pos1 = 1
    i = 1
    Do While (InStr(i, strname, ";") <> 0)
        For j = 1 To 2
            i = InStr(i, strname, ";")
            pos2 = i
            If pos1 < pos2 Then
                If j = 1 Then
                    newstr1 = Mid(strname, pos1, (pos2 - pos1))
                    With rst
                        .AddNew
                        !Variablename = fldname
                        !VariableNumericValue = newstr1
                        !VariableFormat = newstr2
                        !Tblname = tdfname
                        .Update
                    End With
                Else
                    If j = 2 And pos1 < pos2 Then
                        newstr2 = Mid(strname, pos1, (pos2 - pos1))
                        With rst
                            .AddNew
                            !Variablename = fldname
                            !VariableNumericValue = newstr1
                            !VariableFormat = newstr2
                            !Tblname = tdfname
                            .Update
                        End With
                    End If
                End If
            Else
                newstr2 = Mid(strname, pos1, Len(strname) - 1)
                With rst
                    .AddNew
                    !Variablename = fldname
                    !VariableNumericValue = newstr1
                    !VariableFormat = newstr2
                    !Tblname = tdfname
                    .Update
                End With
            End If
        Next j
    Loop

    Exit Function
End Function
Public Sub CreateSASFormat()
    Dim rst As Recordset
    Dim db As Database

    Set db = CurrentDb()
    Set rst = db.OpenRecordset("Format")

    Dim countVariables As Integer
    Dim SQL As String
    Dim VarNames() As String
    Dim VarNumeric() As Integer
    Dim VarFormat() As String
    Dim fs, a As Object
    n = rst.RecordCount
    ReDim VarNumericCount As Integer
    ReDim VarFormatCount As Integer
    ReDim VarNamesCount As Integer

    rst.MoveFirst
    Varname = rst.Fields("Variablename")
    VarNames(1) = rst.Fields("Variablename")
    VarNumeric(1) = rst.Fields("VariableNumericValue")
    VarFormat(1) = rst.Fields("VariableFormat")
    countVariables = 1
    rst.MoveNext
    For i = 2 To rst.RecordCount
        If rst.Fields("Variablename") <> Varname Then
            countVariables = countVariables + 1
            VarNames(countVariables) = rst.Fields("Variablename")
        End If
        Varname = rst.Fields("Variablename")
        VarNumeric(i) = rst.Fields("VariableNumericValue")
        VarFormat(i) = rst.Fields("VariableFormat")
        rst.MoveNext
    Next
    rst.MoveFirst
    Varname = rst.Fields("Variablename")
    SQL = "Proc Format "
    jini = 1
    For i = 1 To countVariables
        SQL = SQL + ";" + Chr$(10) + "    value " + VarNames(i) + " "
        For j = jini To rst.RecordCount
            If rst.Fields("Variablename") = Varname Then
                SQL = SQL & rst.Fields("VariableNumericValue") & " = " & " & rst.Fields("VariableFormat") & " "
                Varname = rst.Fields("Variablename")
                rst.MoveNext
            Else
                jini = j
                Varname = rst.Fields("Variablename")
                j = rst.RecordCount
            End If
        Next
    Debug.Print SQL
Next
CONCLUSION

It is a common practice between statisticians and data managers to import data collected in Access into SAS for analysis. The current paper shows that the capabilities of SAS can be expanded by the use of VBA macros.

REFERENCES


CONTACT INFORMATION

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

Name: Maria S. Melguizo Castro  
Enterprise: University of Arkansas For Medical Sciences UAMS - Department of Pediatrics  
Address: 1 Childrens Way, Slot 512-43  
City, State ZIP: Little Rock, AR 72202-3500  
Work Phone: (501) 364-6619  
Fax: (501)364-1431  
E-mail: msmelguizo@uams.edu  
Web: [http://www.arpediatrics.org/research/biostatistics](http://www.arpediatrics.org/research/biostatistics)

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.