Paper 203-2013

# Extending the Power of Base SAS® with Microsoft Excel

Shilpa Khambhati, Mathematica Policy Research

## ABSTRACT

The SAS Macro Language is an invaluable SAS tool that can be used for iterative SAS data processing, eliminating redundancy in SAS code. Using the SAS Macro Language with Microsoft Excel makes programming tasks even easier. This paper describes how to use the SAS Macro Language and Microsoft Excel to automatically generate customized reports. The proposed method uses Excel macros to drive SAS macros without having to open SAS programs and manually upgrade parameters for each site's data when the data become available. The process eliminates manually editing SAS programs and improves data quality by reducing programming error and program maintenance time.

## INTRODUCTION

Why would you want to run SAS from Excel?  You can create a one-click hands-off production job that imports data from different sources and locations, manipulates and formats the data, and dispatches SAS files and reports to specific network locations. All this can be done by non-SAS programmers and without touching the existing SAS programs. If you find yourself preparing the same reports repeatedly for different data sources, automation can expedite the workflow, whereas currently, users must navigate to specific folders, open them, and modify programs whenever new data are available.

This paper and the associated poster explain how to develop the Excel base front-end application, which incorporates Excel features such as data validation, look-up tables, and Excel macro with Base SAS to run SAS macros based on parameters selected in the Excel file. The result is a simple, user-friendly interface in Excel for running automated SAS routines.

To illustrate this process, we describe a project that consists of three main tasks:

1. Define and organize the output, input, and SAS program names and location folders.
2. Set up an Excel file with the corresponding SAS program macro variables as look-up values.
3. Develop the SAS Macro programs that inherit the parameters being passed from the Excel file.

## STEP 1:  IDENTIFY COMMON PARAMETERS ACROSS ALL SITES

### a. Identify common settings

When working with a large number of files, the first task is to organize and define all of the common variables. First, we identified common settings across all 120 sites:

1. There are four population types for each site: Student, Teacher, Parent, and Principal.
2. For each population, there are two surveys : Baseline and Follow-Up.
3. Generated reports have characteristic formats and functionality that remain consistent across all sites for a particular survey and population.
4. Reports for each site have consistent log and list outputs.

### b. Derive assumptions

Based on the common settings listed above, we derived the following assumptions for each  site:

1. The folder structures will be consistent.
2. Folder paths will be specific for each site.
3. Input programs for each population across sites will be the same for the Baseline and Follow-Up rounds.

4. Programs for each report's output will be the same across each site.
5. Names of the reports themselves and the output SAS data sets will be similar with the specific site IDs and site names embedded in them.
6. The date we receive the data will be different for each instrument type for a site.
7. The date reports are created will be different for each instrument type for a site.

After organizing the project information, we then laid the groundwork for designing the Excel application and developing SAS macros.

### c. Define folder structures for sites and SAS programs

For the folder structure, we developed the following hierarchy for each site:

The site folder is at the top of the tree with subfolders for the Baseline and Follow-Up data and each of these folders contains folders for the individual Student and Parent population data. Finally, each population has additional subfolders for Input, Output, Report, and SASData **(**see Figure 1**).**

Because the SAS programs are the same across all the sites for a particular instrument, we placed them in the general repository beneath the "SAS" folder **(**see Figure 2**).**

Figures 1 and 2 display examples of Student and Parent folders.

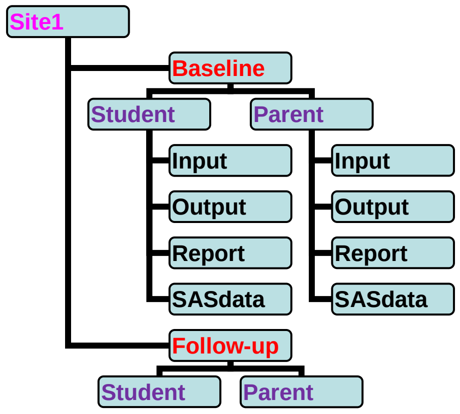


**Figure 1. Reports and Output Folder Structure**

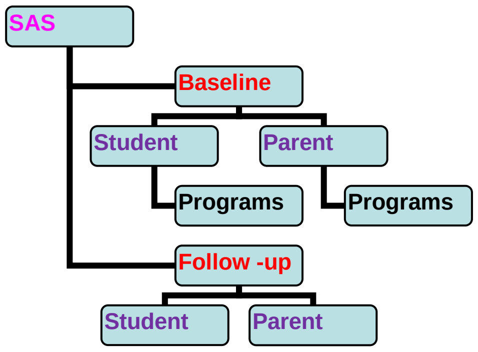
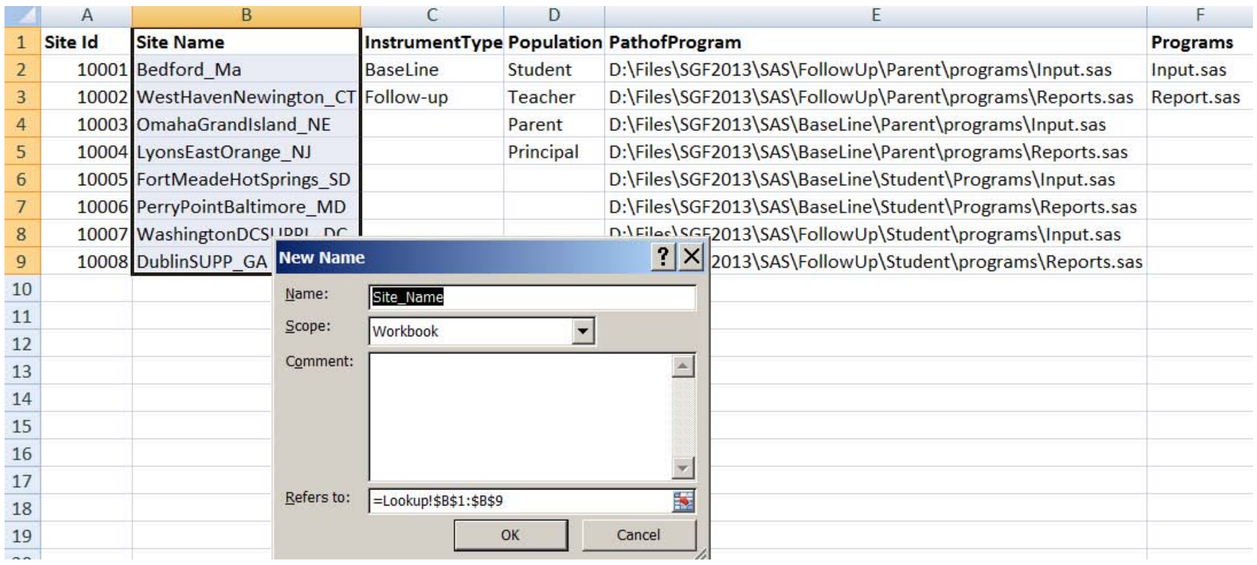


**Figure 2. SAS Programs Folders for Each Instrument Type**

## STEP 2: SET UP EXCEL FILE

### a. Create look-up tables

For the SAS macro parameters, we created look-up tables. For the example illustrated in this paper, we first created a list containing Site ID, Site Name, and Program names in the look-up sheet. We gave a name to each range of these look-up lists. To do this, select or highlight the entire Site Name list, right click, and select "Name a range" from the pop-up menu. This step will open a window where you can enter the name for the corresponding list (see Figure 3). For example, we gave the name "ID" to the Site ID list. We repeated these steps for the Site Name, Instrument Type, Population, Program Names, and Folder Paths lists. We will use the Range Name later to reference the list elsewhere in the Excel file; that is, we will refer to it by this name when we populate the drop-down box with the appropriate values from the list.



| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Site Id | Site Name | InstrumentType | Population | PathofProgram | Programs |
| 2 | 10001 | Bedford_Ma | BaseLine | Student | D:\Files\SGF2013\SAS\FollowUp\Parent\programs\Input.sas | Input.sas |
| 3 | 10002 | WestHavenNewington_CT | Follow-up | Teacher | D:\Files\SGF2013\SAS\FollowUp\Parent\programs\Reports.sas | Report.sas |
| 4 | 10003 | OmahaGrandIsland_NE | | Parent | D:\Files\SGF2013\SAS\BaseLine\Parent\programs\Input.sas | |
| 5 | 10004 | LyonsEastOrange_NJ | | Principal | D:\Files\SGF2013\SAS\BaseLine\Parent\programs\Reports.sas | |
| 6 | 10005 | FortMeadeHotSprings_SD | | | D:\Files\SGF2013\SAS\BaseLine\Student\Programs\Input.sas | |
| 7 | 10006 | PerryPointBaltimore_MD | | | D:\Files\SGF2013\SAS\BaseLine\Student\Programs\Reports.sas | |
| 8 | 10007 | WashingtonDCSUPPL_DC | | | D:\Files\SGF2013\SAS\FollowUp\Student\programs\Input.sas | |
| 9 | 10008 | DublinSUPP_GA | | | 2013\SAS\FollowUp\Student\programs\Reports.sas | |

New Name

Name: Site_Name

Scope: Workbook

Comment:

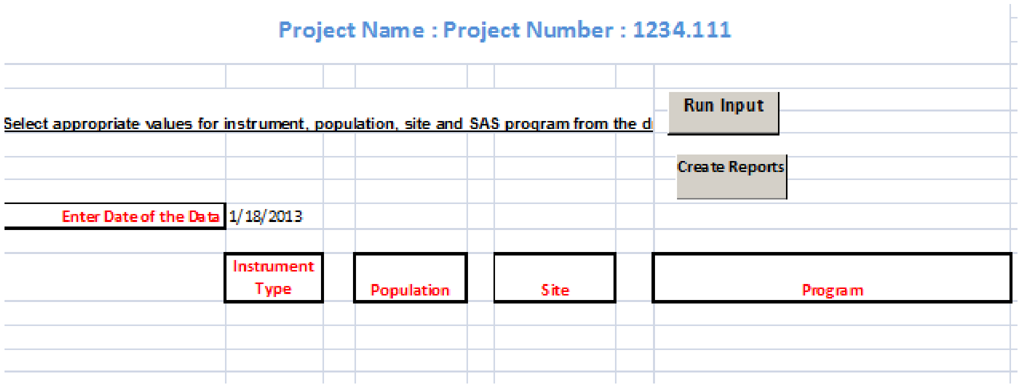
Refers to: =Lookup!$B$1:$B$9

OK Cancel

**Figure 3. Look-Up Sheet with Name Range for Site Name Column**

## b. Design the Main sheet for the users or programmers

The “Main” sheet is designed to encompass all the functionality needed for the users to run the SAS programs. In this sheet, we positioned drop-down cells for users to select the values and buttons to carry out particular actions using the SAS programs. The drop-down cells are populated by creating Data Validation, which points to the particular look-up list from the look-up sheet.



Project Name : Project Number : 1234.111

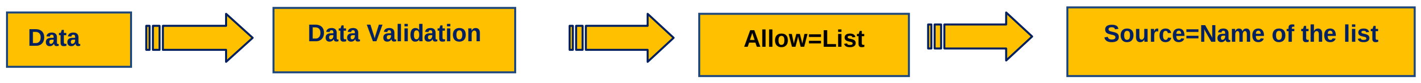
Select appropriate values for instrument, population, site and SAS program from the d

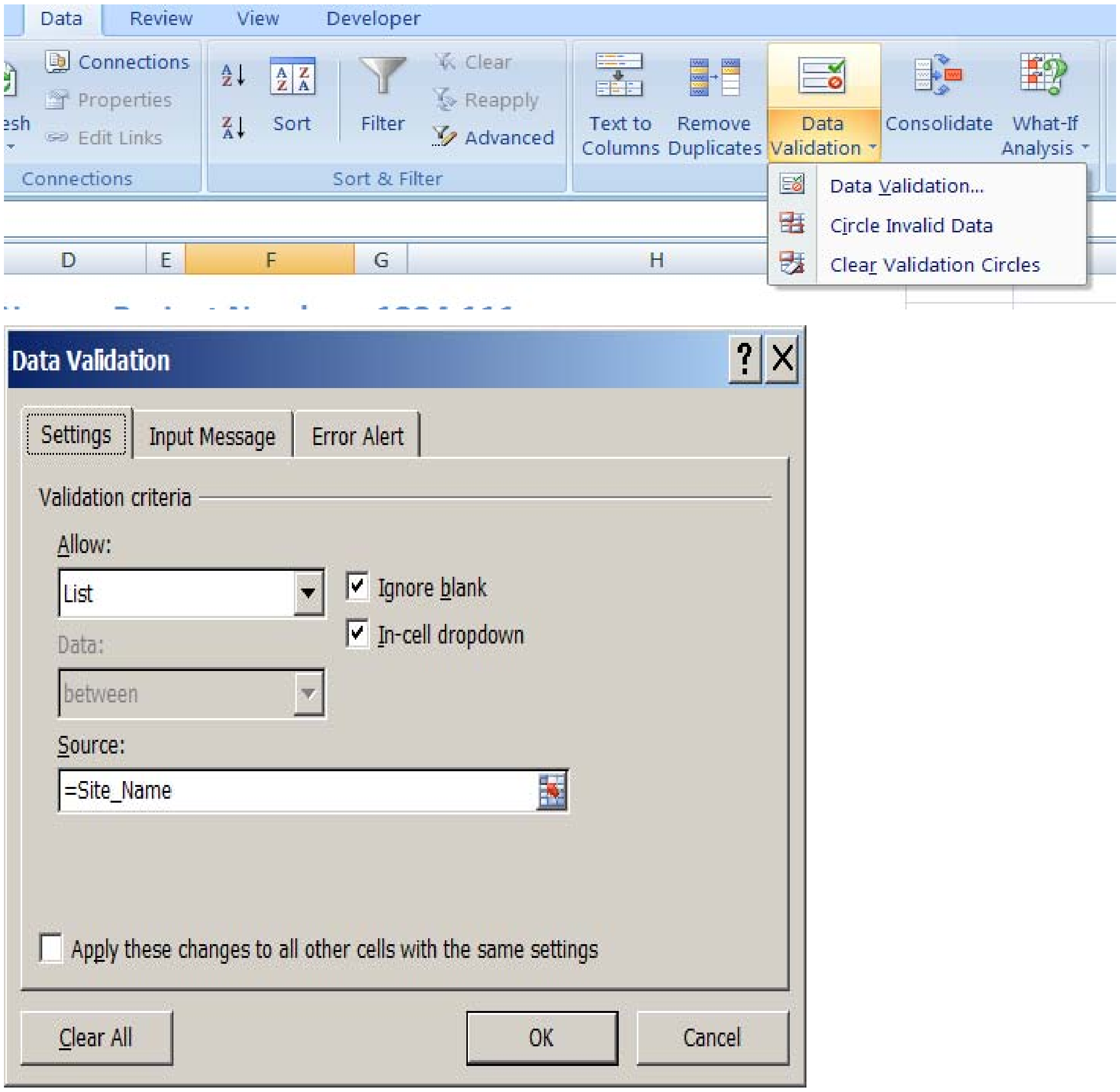
Enter Date of the Data 1/18/2013

**Figure 4. Main Screen Layout Design**

## c. Populate drop-down cells

To populate Site Name in the drop-down cells, select the Site textbox in the “Main” sheet and then select Data Validation from the Data menu. This will open the Data Validation window; then, select “list” from the Allow box and type “=*site_name*” in the source box. “Site_name” is the name range we defined for the site list in the look-up sheet. Repeat these steps for all the other drop-down cells for the *Population, Instrument Type,* and *Program* cells (Figure 5).





Allow: List

[✓] Ignore blank

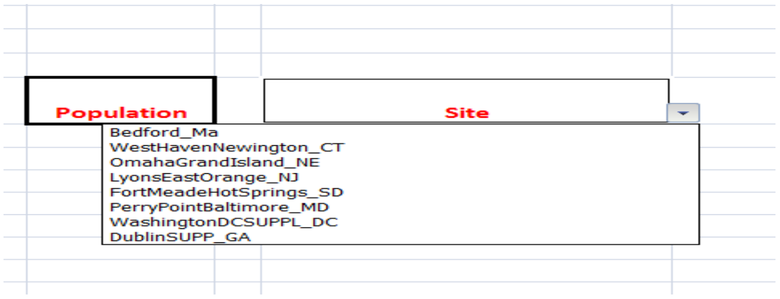
[✓] In-cell dropdown

Data: between

Source: =Site_Name

[ ] Apply these changes to all other cells with the same settings

Figure 5. Data Validation Pop-Up Window



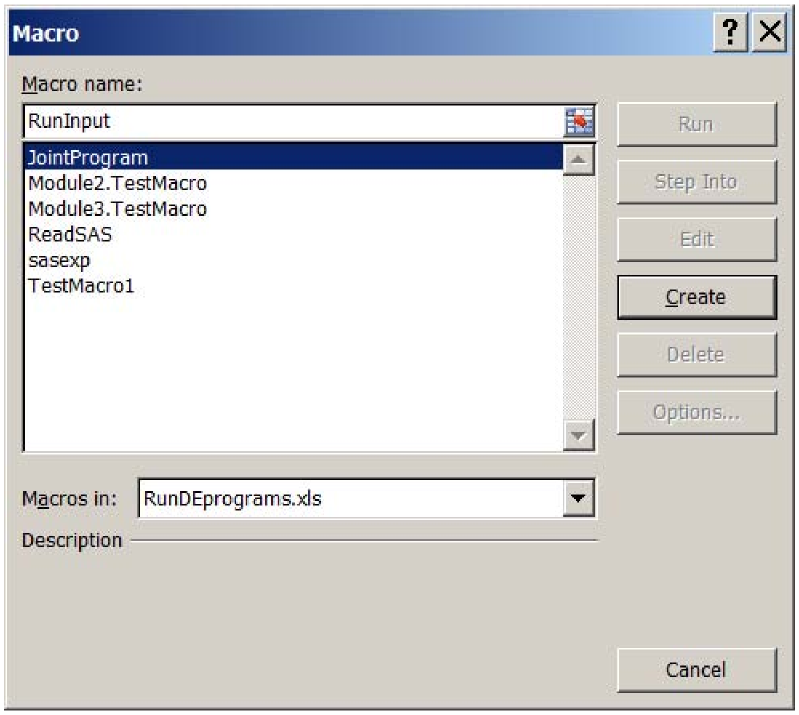
Bedford_Ma
WestHavenNewington_CT
OmahaGrandIsland_NE
LyonsEastOrange_NJ
FortMeadeHotSprings_SD
PerryPointBaltimore_MD
WashingtonDCSUPPL_DC
DublinSUPP_GA

Figure 6. Populated Site Drop-Down

### d. Record Excel Macros

The next step is to write the VBA macro that runs the SAS programs. The macro is recorded for the command button click action. From the Developer Menu, select Macro. In the pop-up window, type the Macro name (for our example, we used RunInput), and then click Create (see Figure 7). This opens the VBA screen where you will insert the code to invoke SAS and run your programs.

$$\begin{tikzcd}
\text{Developer Menu} \arrow[r] & \text{Macros} \arrow[r] & \text{Macro Name in Popup} \arrow[r] & \text{Click Create Button}
\end{tikzcd}$$



Macro name: RunInput

JointProgram
Module2.TestMacro
Module3.TestMacro
ReadSAS
sasexp
TestMacro1

Macros in: RunDEprograms.xls

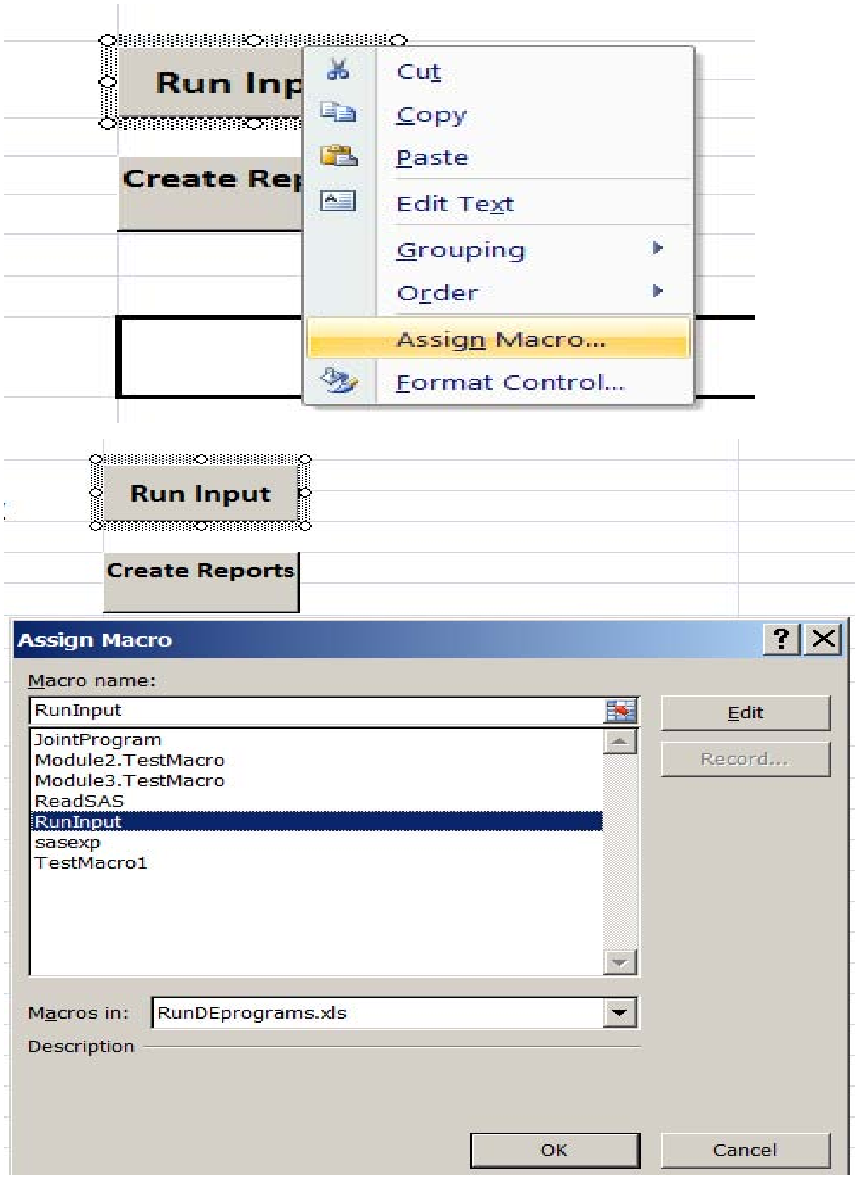
Figure 7. Create Macro from Excel Developer Menu

**Macro RunInput code is written in the Microsoft Visual Basic Code window:**

```
Sub RunInput()
Dim OleSAS As Object
'Invoke SAS
    Set OleSAS = CreateObject("SAS.Application")
    OleSAS.Visible = True
'Define %let used in the SAS programs
    OleSAS.Submit ("%Let Site=" & Sheets("Main").Range("F12").Value & ";")
    OleSAS.Submit ("%Let Instrument=" & Sheets("Main").Range("B12").Value & ";")
    OleSAS.Submit ("%let Population=" & Sheets("Main").Range("D12").Value & ";")
    OleSAS.Submit ("%let InProgram=" & Sheets("Main").Range("H12").Value & ";")
    OleSAS.Submit ("%let InDate=" & Sheets("Main").Range("B10").Value & ";")
'Include SAS program
    OleSAS.Submit ("data; %inc '&InProgram.'; run;")
    OleSAS.Quit
End Sub
```

## e. Assign Excel macros to the command button

Now that we have inserted the parameter lists, created drop-down cells, and written the macro to invoke SAS in Excel, we are ready to bind the Excel macro *RunInput* to the *RunInput command button*. To assign the Macro to the RunInput command button, right click on the selected command button and click the “Assign Macro” option from the pop-up menu. This will open the “Assign Macro” window showing all existing macros in the Excel file; then, select the one you want to bind to the Run Input command button. For our example, we will select the macro “RunInput” to bind to the command button and then click the “OK” button.



Macro name:
RunInput

JointProgram
Module2.TestMacro
Module3.TestMacro
ReadSAS
RunInput
sasexp
TestMacro1

Macros in: RunDEprograms.xls
Description

**Figure 8. Assign RunInput to the Command Button "Run Input"**

## STEP 3: CREATE SAS PROGRAMS TO IMPORT AND MANIPULATE THE SITE DATA

We defined all our macro variables in the Excel macro RunInput. We will reference these macro variables in our SAS programs. The Indate, Site, Instrument, and Population macro variables are used in each of our input, data cleaning, and reporting programs.

**Here is the example of the input program for the Parent Baseline instrument:**

```
ODS Listing;

Options ps=60 ls=200 nodate nocenter;

**Define directory path as discussed in step 1**;

%Let dirpath=D:\Files\SGF2013\&Site.\&Instrument\&Population.;

**SASDATA folder to place final data set **;

Libname dsn “&dirpath.\sasdata”;

Proc PrintTo Log   ="&Dirpath.\output\&site.&Instrument.&population..log"  New;  Run;
Proc PrintTo Print  ="&Dirpath.\output\&site.&Instrument.&population..lst"  New;  Run;

%Let outfile1 = &site.&Instrument.&population.
%Let infile1   = &site.&Instrument.&population.&Indate..ds4;
%Let tdate     = %sysfunc(today(), mmddyy6.);

Title "&Site. - &population. &Instrument. : &tdate. ";

 Proc Format;
        Invalue DK (UPCASE JUST)
           'R'=.R     'D'=.D     'M'=.M    'N'=.N   'X'=.X;

 Data dsn.&outfile1.;

    Infile  “&dirpath.\input\&infile1." missover lrecl=705;

    Input

       ***your code***;

 Run;

 Proc Frequency Data=dsn.&outfile1.;
  Table _ALL_ /list missing;
Run;

%Include  “Clean_&Instrument.&population..sas”;

%Include  “CrReport_&Instrument.&population..sas”;
```

## CONCLUSION

This application is an excellent example of how collaboration between SAS and Excel can result in an automated process for a complex task. We generated reports and SAS files for each site on demand and in turn-key mode by just having Excel and Base SAS on our machine. In order for this application to work, it is important to identify similarities and differences across all desired outputs and data sources. For any iterative data processing that does not require logic modifications to the SAS programs, this type of setup can be cost-effective and efficient.

## REFERENCES

Website:
http://support.sas.com/documentation/cdl/en/hostwin/63047/HTML/default/viewer.htm#p11fjwtd3p7v9wn1mw6u7l08cnzu.htm

Book: *SAS 9.2 Companion for Windows,* Second Edition. Cary, NC: SAS Institute Inc., 2010 (p. 272).

## ACKNOWLEDGMENTS

The author would like to thank Ronald Palanca and Barbara Kolln of Mathematica Policy Research for their valuable contributions to this paper.

## CONTACT INFORMATION

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

Shilpa Khambhati
Mathematica Policy Research
Email: SKhambhati@Mathematica-mpr.com