Paper 154-31

# Step-by-Step in Using SAS® DDE to Create an Excel Graph Based on N Observations from a SAS Data Set

Choon-Chern Lim, Mayo Clinic, Rochester, MN

## ABSTRACT

From the end user's perspective, the advantages of having an Excel graph include familiarity with Excel application and the ability to modify the graph by changing the underlying data in the Excel spreadsheet. Although the process of exporting the data from a SAS data set to an Excel file is relatively straightforward, someone has to manually create and format an Excel graph, which can be a time consuming process. It is possible to embed a SAS graph image into the Excel file, but this static graph cannot be altered once created.

This paper presents a detailed step by step approach to programmatically create an Excel graph based on N observations from a SAS data set. In most circumstances, we have a fixed number of variables with unknown number of observations for creating an Excel graph. Therefore, we will perform a minor tweak to the recorded Excel macro to satisfy this need.

This paper covers a little Visual Basic coding, but it is not necessary for you to know how to write it.

## INTRODUCTION

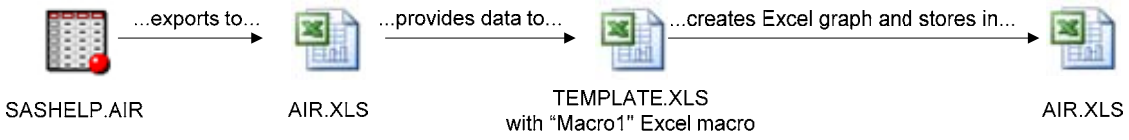
Why bother creating an Excel graph when we can have a beautiful looking SAS graph? I often ask myself that question when I receive these requests from my users. A little bit of my work experience… My job focuses more on web application development where I build online data entry screens and customized web reports for wide range of users, such as doctors, nurses etc… Most of the web reports consist of some graphs created using SAS/GRAPH. While my users are satisfied with the reports, many of them later tell me: "Excellent, now make them Excel graphs so that I can modify the graph myself".

I can always create these graphs as images (JPG or GIF files) from SAS and insert them in the Excel file, but these static images cannot be modified once created.

This is where DDE comes into play. Dynamic Data Exchange (DDE) allows two different applications, in this case SAS and Excel applications, to interact with one another. Once the communication bridge is set up, SAS uses X4ML commands to provide further instructions to the Excel application.

The goal of this paper is not to explain how DDE works or explore the X4ML commands, but to get you going right away in creating an Excel graph from SAS. There are many great past SUGI papers on DDE and X4ML topics, and it is highly recommended that you read them to learn more about the tips and tricks.

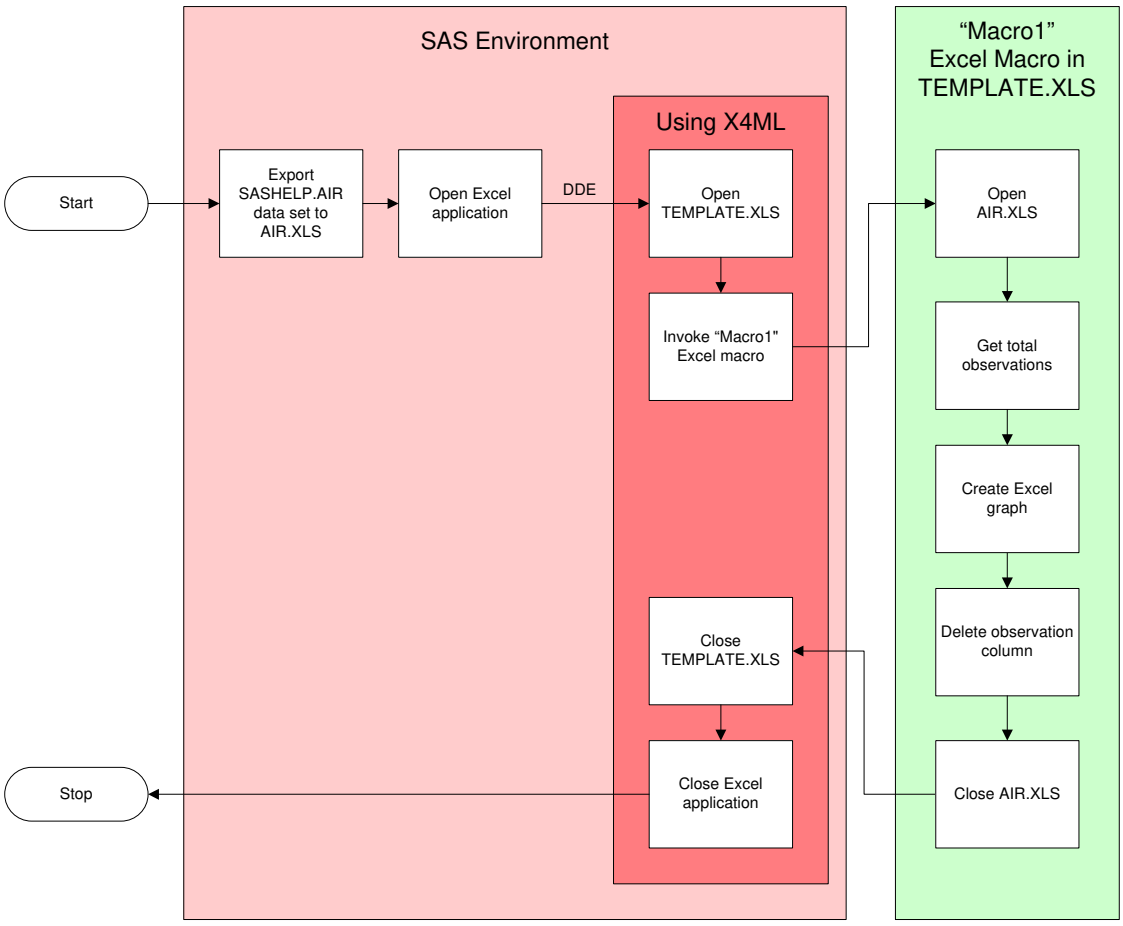
## GENERAL OVERVIEW




In this paper, we will do the following example together:-

1. SAS exports AIR data set from SASHELP library to an Excel file called AIR.XLS.
2. The "Macro1" Excel macro in TEMPLATE.XLS reads data and creates an Excel graph in AIR.XLS.

## UNDERSTANDING THE CONCEPT




First of all, the SASHELP.AIR data set has to be exported to an Excel file so that the Excel macro can read the data and create an Excel graph. Although there are many ways for SAS to export the data to Excel file, in this example, we will use PROC EXPORT to do so. Why? It's fast, hassle free, and requires less code to be written.

Once SAS exports the data to an Excel file (we called it AIR.XLS), SAS will communicate with the Excel application using DDE by passing X4ML commands. The X4ML commands will open TEMPLATE.XLS and execute "Macro1" Excel macro. This "Macro1" Excel macro will read data from AIR.XLS and create an Excel graph in it.

## EXECUTING EXCEL MACRO FROM SAS

In this step, SASHELP.AIR data set is exported into an Excel file with an additional column called “Total”, which basically keeps track of the total number of observations in the data set. Once the Excel file is created, the X4ML commands will invoke the “Macro1” Excel macro in TEMPLATE.XLS to generate the Excel graph.

```
/*************************************************************
 *** Clear SAS log and output.
 *************************************************************/
dm 'clear log; clear output';

options mrecall mprint mlogic symbolgen;

%macro createExcelGraph;
       %local lib dsn xlsCmdPath outputPath xlsTemplatePath;

       /*************************************************************
        *** Excel application path.
        *************************************************************/
       %let xlsCmdPath = C:\Program Files\Microsoft Office\OFFICE11\excel.exe;

       /*************************************************************
        *** Data library and data set name. In this example, we will
        *** use "AIR" data set  from "SASHELP" library.
        *************************************************************/
       %let lib = sashelp;
       %let dsn = air;

       /*************************************************************
        *** Excel template file path. This file should contain the Excel
        *** macro called "Macro1".
        *************************************************************/
       %let xlsTemplatePath = C:\sugi\template.xls;

       /*************************************************************
        *** Excel output file that will have the generated Excel graph.
        *************************************************************/
       %let outputPath = C:\sugi\air.xls;


       proc sql noprint;
              /*************************************************************
               *** Get number of rows in data set and store the count in
               *** macro variable "total".
               ***
               *** The dictionary.tables are special read-only tables that
               *** contains plenty information about SAS data sets.
               *************************************************************/
              select nlobs into :total
              from dictionary.tables
              where libname = upcase("&lib.") and memname = upcase("&dsn.");


              /*************************************************************
               *** Add a "total" variable and store the computed total in it.
               *** Since the Excel macro will be reading the third column
               *** (column C) to retrieve the total row count, it is
               *** important to make sure that the "total" variable is stored
               *** at the right location.
               ***
               *** The reason we choose the data set name to be "sheet1" is
               *** because when we do a PROC EXPORT, the Excel spreadsheet
```

```
         *** is automatically named after the SAS data set name.
         *** To simplify the process, we choose "sheet1", which is the
         *** default name for the first Excel spreadsheet.
         ************************************************************/
        create table sheet1 as
        select date, air, &total. as total
        from &lib..&dsn.;
quit;

/*****************************************************************
 *** Export the data set to excel file.
 *****************************************************************/
proc export data=sheet1 outfile="&outputPath." dbms=excel replace;
run;

options noxsync noxwait;

/*****************************************************************
 *** Open Excel application.
 *****************************************************************/
x "'&xlsCmdPath.'";

/*****************************************************************
 *** Suspend SAS for 5 seconds to allow Excel to be fully started.
 *****************************************************************/
data _null_;
       rc = sleep(5);
run;

/*****************************************************************
 *** Create a file reference to the excel sheet.
 *****************************************************************/
filename sas2xl dde 'excel|system';

/*****************************************************************
 *** Use X4ML commands to communicate with Excel.
 *** Although you can use X4ML commands to format the Excel
 *** spreadsheet's look and feel (colors, borders etc...), it is
 *** preferable to keep it to minimum use, and have the Excel
 *** macro to do those formatting. This makes your SAS code look
 *** clean and readable.
 ***
 *** So, in the below steps, we will open the Excel template file
 *** and execute "Macro1" Excel macro.
 *****************************************************************/
data _null_;
       file sas2xl;

       /*************************************************************
        *** Open Excel template file in read-only mode. This file
        *** should have the Excel macro that you have created.
        *************************************************************/
       put "[open(""&xlsTemplatePath."", 0 , true)]";

       /*************************************************************
        *** Execute the Excel macro. By default, the macro name is
        *** called "Macro1".
        *************************************************************/
       put "[run(""Macro1"")]";

       /*************************************************************
        *** Close the Excel template file.
        *************************************************************/
       put '[file.close(false)]';
```

```
            /*************************************************************
             *** Close Excel application.
             *************************************************************/
            put '[quit()]';
      run;

      /*****************************************************************
       *** Suspend SAS for 5 seconds to allow Excel to be fully closed.
       *****************************************************************/
      data _null_;
            rc = sleep(5);
      run;

      /*****************************************************************
       *** Delete sheet1 data set.
       *****************************************************************/
      proc datasets library=work nodetails nolist;
            delete sheet1;
      quit;
%mend createExcelGraph;

%createExcelGraph;
```

## CREATING SAMPLE DATA IN EXCEL (SAMPLE.XLS)

Before we start recording an Excel macro to create an Excel graph, we need a sample data that reflects the actual data.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | DATE | AIR | TOTAL | |
| 2 | 1/1/1949 | 112 | 4 | |
| 3 | 2/1/1949 | 118 | 4 | |
| 4 | 3/1/1949 | 132 | 4 | |
| 5 | 4/1/1949 | 129 | 4 | |
| 6 | | | | |

When we run a PROC EXPORT to export SASHELP.AIR data set to AIR.XLS, we will get the similar spreadsheet layout as shown above. Columns A and B are part of SASHELP.AIR data set and we added Column C to keep track of the total row count. We will call this Excel spreadsheet (with 4 data rows) SAMPLE.XLS.

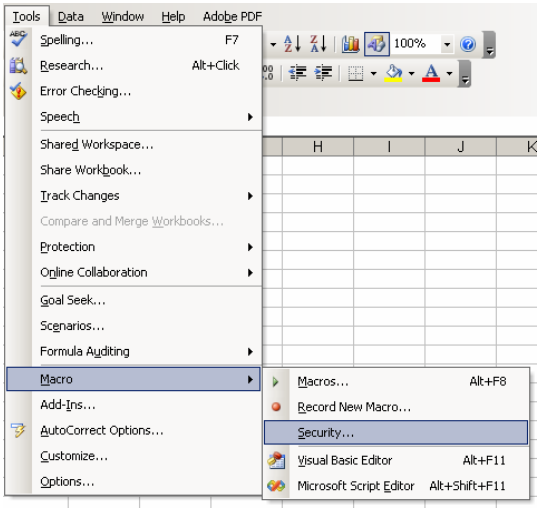
Once we have successfully recorded an Excel macro, we can safely delete SAMPLE.XLS and use the actual Excel file, which is AIR.XLS.

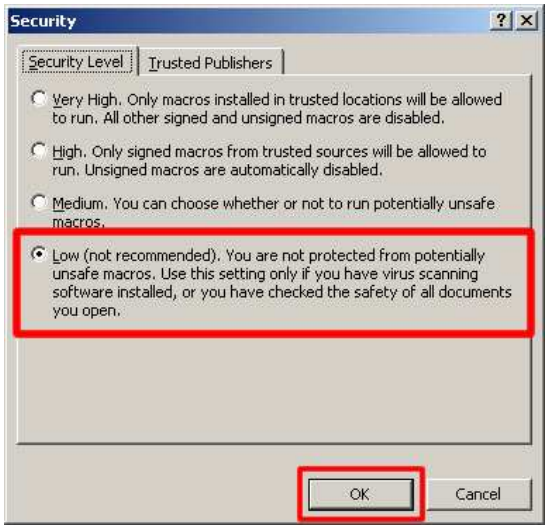
## RECORDING EXCEL MACRO (TEMPLATE.XLS)

In this step, we will record an Excel macro that creates a simple Excel graph based on the data in SAMPLE.XLS. This Excel macro will be called “Macro1” and it will be saved in TEMPLATE.XLS.

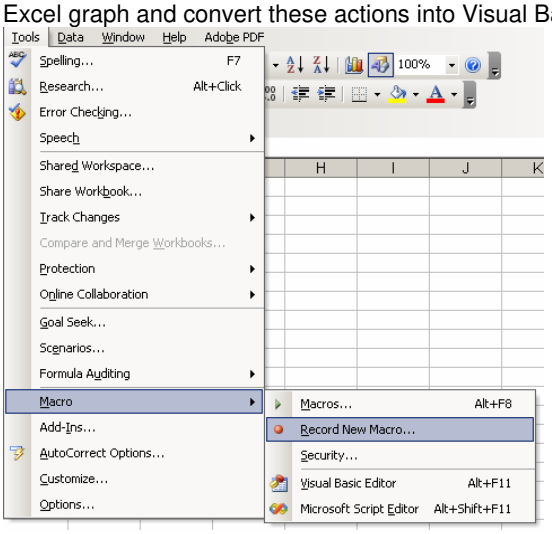
1. Open Excel application.

2. Click “Tools” → “Macro” → “Security”. Set Security Level from Medium (by default) to Low. This suppresses the Excel macro warning prompt when SAS opens the Excel file that contains a macro. Then, click “OK”.

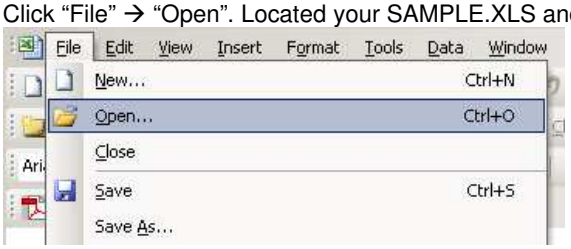



3. Click “Tools” → “Macro” → “Record New Macro”. By default, the macro name is “Macro1”. We will use this macro name for this example. Click “OK”. It will start recording all the actions you do for creating the Excel graph and convert these actions into Visual Basic code.



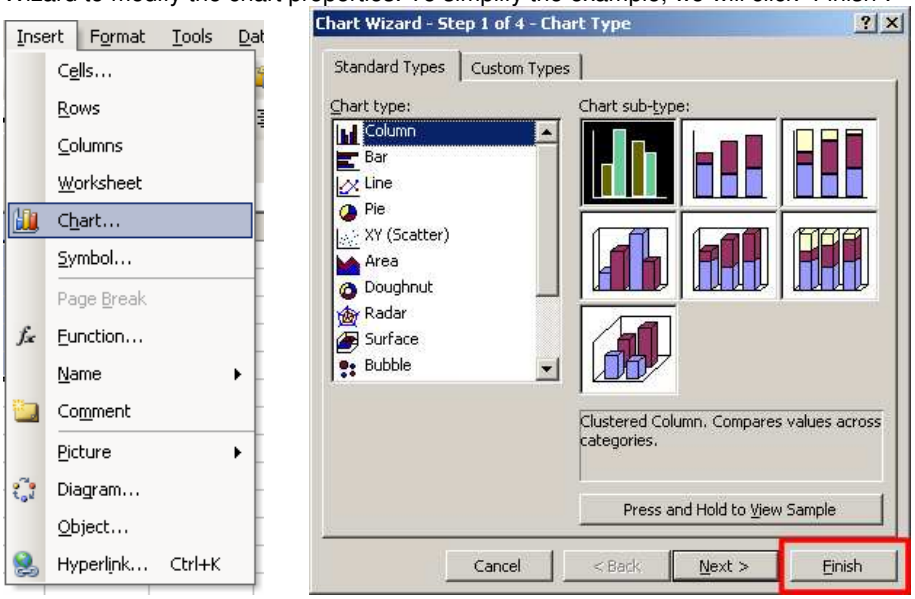

4. Click “File” → “Open”. Located your SAMPLE.XLS and open it.




5. Highlight the sample data.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | DATE | AIR | TOTAL | |
| 2 | 1/1/1949 | 112 | 4 | |
| 3 | 2/1/1949 | 118 | 4 | |
| 4 | 3/1/1949 | 132 | 4 | |
| 5 | 4/1/1949 | 129 | 4 | |
| 6 | | | | |

6. Click “Insert” → “Chart…”. Select the chart you want to create. Here, you can step through the Chart Wizard to modify the chart properties. To simplify the example, we will click “Finish”.



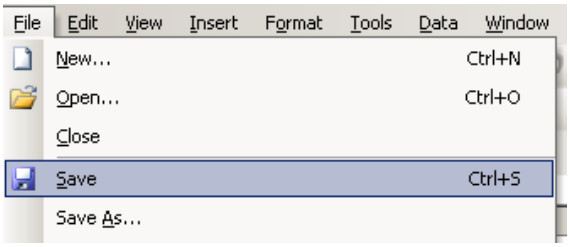

7. Highlight whole Column C by clicking the letter C once.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | DATE | AIR | TOTAL | |
| 2 | 1/1/1949 | 112 | 4 | |
| 3 | 2/1/1949 | 118 | 4 | |
| 4 | 3/1/1949 | 132 | 4 | |
| 5 | 4/1/1949 | 129 | 4 | |
| 6 | | | | |

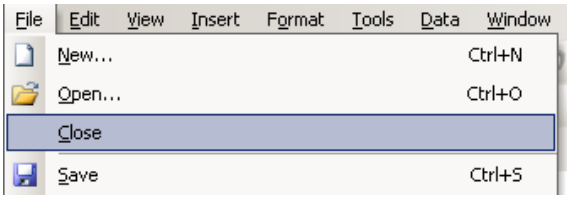
8. Click “Delete” key to delete the whole column. Although we are deleting this column, we will need to tweak the Visual Basic source code so that it picks up the value first before deleting it.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | DATE | AIR | | |
| 2 | 1/1/1949 | 112 | | |
| 3 | 2/1/1949 | 118 | | |
| 4 | 3/1/1949 | 132 | | |
| 5 | 4/1/1949 | 129 | | |
| 6 | | | | |

9. Click “File” → “Save”.



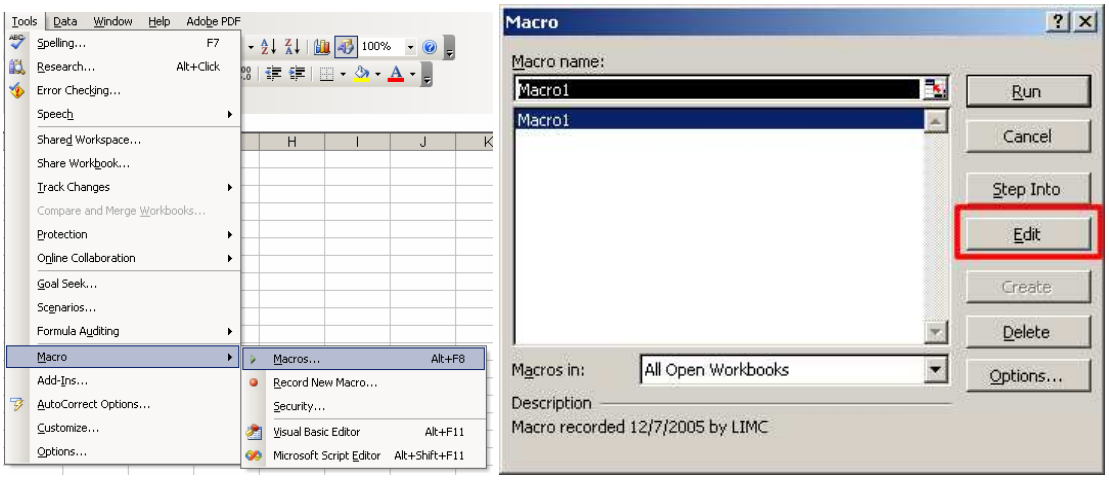

10. Click “File” → “Close”.




11. Click Stop Recording button (blue square).


12. Delete SAMPLE.XLS since it is not needed anymore.

13. Click “Tools” → “Macro” → “Macros…”. All the recorded steps for creating the Excel graph are stored in “Macro1”. Click “Edit” to view the Visual Basic source code.



14. We have to make a little modification to the recorded source code so that it reads from AIR.XLS instead of SAMPLE.XLS. We also want to make sure that the Excel graph is created based on N rows of data instead of just 4 data rows.

```
Sub Macro1()
    Workbooks.Open Filename:="C:\sugi\excel\xls\sample.xls"
    Range("A1:B5").Select
    Charts.Add
    ActiveChart.ChartType = xlColumnClustered
    ActiveChart.SetSourceData Source:=Sheets("SHEET1").Range("A1:B5")
    ActiveChart.Location Where:=xlLocationAsObject, Name:="SHEET1"
    ActiveWindow.Visible = False
    Windows("sample.xls").Activate
    Columns("C:C").Select
    Selection.ClearContents
    ActiveWorkbook.Save
    ActiveWorkbook.Close
End Sub
```

15. Make the highlighted changes to the source code.

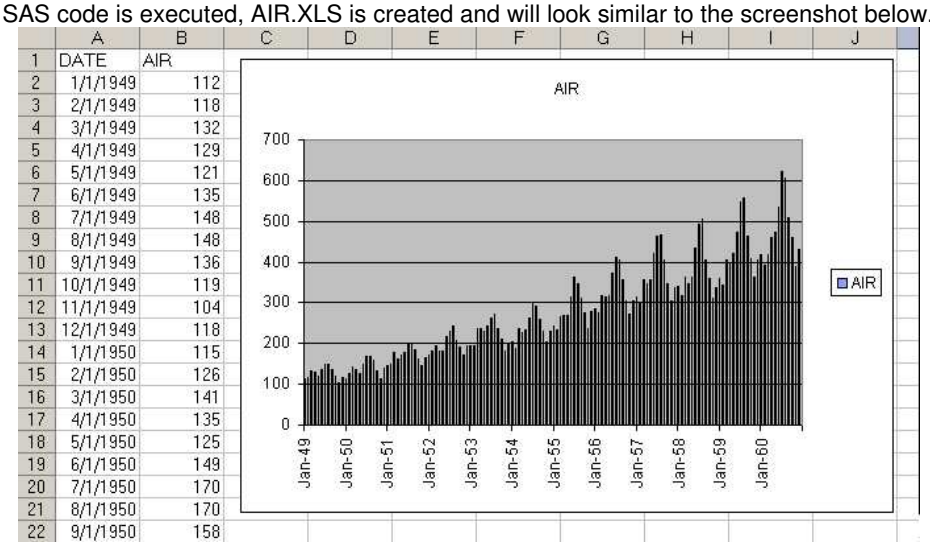
```
Sub Macro1()
    'Initialize variables
    Dim lastRow As Integer
    Dim xlsPath As String
    Dim xlsFile As String

    'Excel file path and file name
    xlsPath = "C:\sugi\excel\xls\"
    xlsFile = "air.xls"

    'Replace "C:\sugi\excel\xls\sample.xls" with xlsPath & xlsFile
    Workbooks.Open Filename:=xlsPath & xlsFile

    'Column C contains the total data count, so we need to
    'store the value first before this column is deleted.
    'We need to add 1 to the total count to include the title row
    lastRow = Sheets(1).Range("C2") + 1

    'Replace ("A1:B5") with ("A1:B" & lastRow)
    Range("A1:B" & lastRow).Select
    Charts.Add
    ActiveChart.ChartType = xlColumnClustered

    'Replace ("A1:B5") with ("A1:B" & lastRow)
    ActiveChart.SetSourceData Source:=Sheets("Sheet1").Range("A1:B" & lastRow)

    ActiveChart.Location Where:=xlLocationAsObject, Name:="Sheet1"
    ActiveWindow.Visible = False

    'Replace sample.xls with xlsFile
    Windows(xlsFile).Activate

    Columns("C:C").Select
    Selection.ClearContents
    ActiveWorkbook.Save
    ActiveWorkbook.Close
End Sub
```

16. Finally, save this Excel file to be TEMPLATE.XLS.

17. Close Visual Basic and Excel applications.

## TESTS AND RESULTS

When the SAS code is executed, AIR.XLS is created and will look similar to the screenshot below.



| | A | B |
|---|---|---|
| 1 | DATE | AIR |
| 2 | 1/1/1949 | 112 |
| 3 | 2/1/1949 | 118 |
| 4 | 3/1/1949 | 132 |
| 5 | 4/1/1949 | 129 |
| 6 | 5/1/1949 | 121 |
| 7 | 6/1/1949 | 135 |
| 8 | 7/1/1949 | 148 |
| 9 | 8/1/1949 | 148 |
| 10 | 9/1/1949 | 136 |
| 11 | 10/1/1949 | 119 |
| 12 | 11/1/1949 | 104 |
| 13 | 12/1/1949 | 118 |
| 14 | 1/1/1950 | 115 |
| 15 | 2/1/1950 | 126 |
| 16 | 3/1/1950 | 141 |
| 17 | 4/1/1950 | 135 |
| 18 | 5/1/1950 | 125 |
| 19 | 6/1/1950 | 149 |
| 20 | 7/1/1950 | 170 |
| 21 | 8/1/1950 | 170 |
| 22 | 9/1/1950 | 158 |



## HELPFUL TIPS

1. Practice and take note of the Excel graph settings before recording the actual Excel macro. This will prevent you from recording redundant steps in the Excel macro.

2. If you plan to create a complex Excel graph that requires many steps, it is highly recommended to record these steps in multiple small Excel macros. These small Excel macros can then be combined into one big Excel macro. They also act as “checkpoints” so that you can easily recognize the functionalities of the recorded Visual Basic source code.

## CONCLUSION

This paper provides a detailed step-by-step approach and helpful tips for creating an Excel graph based on N observations from a SAS data set. Although there are many ways to create an Excel graph from SAS, this paper presents one proven technique that has been working well in the production environment.

## REFERENCES

SAS Institute Inc. 2002. "DDE Example 3 : Sending Commands to Excel via DDE". http://ftp.sas.com/techsup/download/sample/samp_lib/hostsampDDE_Example_3__Sending_Commands_.html.

## ACKNOWLEDGEMENTS

Special thanks to Jon Kosanke for taking his precious time to review my paper.

## CONTACT INFORMATION

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

Choon-Chern Lim
Mayo Clinic
200 First Street SW
Rochester, MN 55905
Email: limc@mayo.edu