Guidelines for Organizing SAS Code and Project Files

Nate Derby

Stakana Analytics
Seattle, WA

Club des Utilisateurs SAS de Québec
11/1/16

Outline

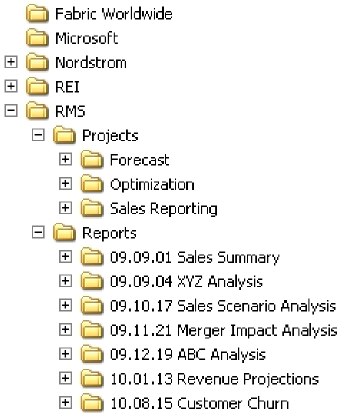
Basic Organizational Ideas

- Never throw anything away.
- Know where to find everything.
- Make the code reusable.
- Automate as much as possible.

File Organization

Files should be organized

- by Company
    - Projects
        - by Name
    - Reports
        - by Date Given, plus title or description



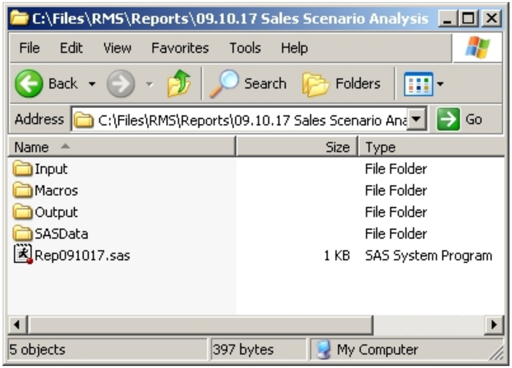
```
   Fabric Worldwide
   Microsoft
⊞ Nordstrom
⊞ REI
⊟ RMS
   ⊟ Projects
      ⊞ Forecast
      ⊞ Optimization
      ⊞ Sales Reporting
   ⊟ Reports
      ⊞ 09.09.01 Sales Summary
      ⊞ 09.09.04 XYZ Analysis
      ⊞ 09.10.17 Sales Scenario Analysis
      ⊞ 09.11.21 Merger Impact Analysis
      ⊞ 09.12.19 ABC Analysis
      ⊞ 10.01.13 Revenue Projections
      ⊞ 10.08.15 Customer Churn
```

Report vs. Project

- Report = relatively minor, one-time *ad hoc*.
- Project = repeatedly updated with new data.
- If it has a name, it's a project.
- Date rules:
  - The date given (not assigned).
  - If two turned in on same day, add a letter (*10.07.22a*, *10.07.22b*).
  - To keep chronological order: Use *YY.MM.DD*.

Within a Report Directory

- The *calling program*
- Subdirectories:
  - *Input*
  - *Macros*
  - *Output*
  - *SASData*
  - Various other subdirectories?

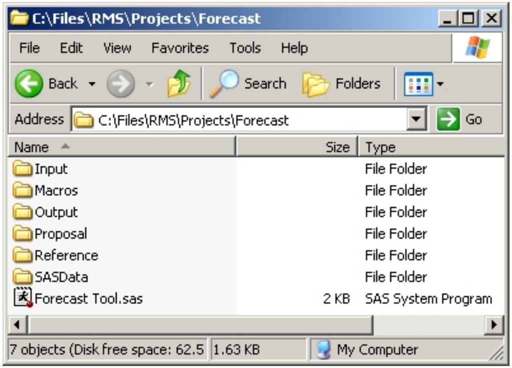

C:\Files\RMS\Reports\09.10.17 Sales Scenario Analysis
File
Edit
View
Favorites
Tools
Help
Back
Search
Folders
Address
C:\Files\RMS\Reports\09.10.17 Sales Scenario Ana
Go

Name	Size	Type
Input		File Folder
Macros		File Folder
Output		File Folder
SASData		File Folder
Rep091017.sas	1 KB	SAS System Program

5 objects | 397 bytes | My Computer

Within a Project Directory

- The *calling program*
- Subdirectories:
  - *Input*
  - *Macros*
  - *Output*
  - *SASData*
  - Various other subdirectories?


C:\Files\RMS\Projects\Forecast
File
Edit
View
Favorites
Tools
Help
Back
Search
Folders
Address
C:\Files\RMS\Projects\Forecast
Go

Name	Size	Type
Input		File Folder
Macros		File Folder
Output		File Folder
Proposal		File Folder
Reference		File Folder
SASData		File Folder
Forecast Tool.sas	2 KB	SAS System Program

7 objects (Disk free space: 62.5 | 1.63 KB | My Computer

Code Organization

Calling Program for a Report (Figure 2, page 4)

```
DM 'output' clear;
DM 'log' clear;

%LET root = C:\Files\RMS\Reports\09.10.17 Sales Forecasts;
  *where the root directory is located;

OPTIONS SASAUTOS=( "&root\Macros" ) MAUTOSOURCE;

%makeSetup;
  *makes the setup structures.  FURTHER FUNCTIONALITY WILL NOT WORK IF COMMENTED OUT;

*%readData;

%analyzeData;

*%exportOutput;
```

Code Organization

Calling Program for a Project (Figure 2, page 4)

```
DM 'output' clear;
DM 'log' clear;

%LET root = C:\Files\RMS\Projects\Forecasts;
%LET exroot = C:\SAS\ExportToXL

%LET orig = SEA;
%LET dest = YQB;
%LET datecut = 7/25/16; *cutoff date for the data;
%LET hzn = 15; *forecast horizon;

OPTIONS SASAUTOS=( "&root\Macros", "&exroot" ) MAUTOSOURCE;

%makeSetup;
  *makes the setup structures.  FURTHER FUNCTIONALITY WILL NOT WORK IF COMMENTED OUT;

*%readFormatData;
  *reads and segments the input data set;

%makeForecasts( fnumber=1542 );
  *produces the forecasts, cycling through various forecasting methods;

*%exportForecasts;
  *exports the forecasts onto Excel spreadsheets;
```

Code Organization

The %makeSetup Macro (Figure 3, page 5)

```
%MACRO makeSetup;

  OPTIONS ORIENTATION=landscape LINESIZE=150 PAGESIZE=60 NODATE NONUMBER
    MCOMPILENOTE=NONE NOTES SOURCE;

  %LOCAL dir1 dir2; %*these macro variables are only used here;
  %GLOBAL outputroot enddate; %*these macro variables ar used in other macros;

  %LET outputroot = &root\Output;

  DATA _NULL_;
    dir1 = "'"||'mkdir'||' "'||"&outputroot"||'"'||"'"; %*creates directory commands;
    dir2 = "'"||'mkdir'||' "'||"&root\Data"||'"'||"'";
    CALL SYMPUTX( 'dir1', dir1 );
    CALL SYMPUTX( 'dir2', dir2 );
    CALL SYMPUTX( 'enddate', INPUT( "&datecut", mmddyy8. ) );
  RUN;

  DATA _NULL_; %*makes the directories for the graphics;
    SYSTASK COMMAND &dir1 WAIT;
    SYSTASK COMMAND &dir2 WAIT;
  RUN;

  LIBNAME rms "&root\Data";
```

What’s Really Happening?

The %makeSetup Macro (Figure 3, page 5)

```
%MACRO makeSetup;

  OPTIONS ORIENTATION=landscape LINESIZE=150 PAGESIZE=60 NODATE NONUMBER
    MCOMPILENOTE=NONE NOTES SOURCE;

  %LOCAL dir1 dir2; %*these macro variables are only used here;
  %GLOBAL outputroot enddate; %*these macro variables ar used in other macros;

  %LET outputroot = &root\Output;

  DATA _NULL_;
    dir1 = "'"||'mkdir'||' "'||"&outputroot"||'"'||"'"; %*creates directory commands;
    dir2 = "'"||'mkdir'||' "'||"&root\Data"||'"'||"'";
    CALL SYMPUTX( 'dir1', dir1 );
    CALL SYMPUTX( 'dir2', dir2 );
    CALL SYMPUTX( 'enddate', INPUT( "&datecut", mmddyy8. ) );
  RUN;

  %PUT dir1=&dir1;
  %PUT dir2=&dir2;
  %PUT enddate=&enddate;

  DATA _NULL_; %*makes the directories for the graphics;
    SYSTASK COMMAND &dir1 WAIT;
    SYSTASK COMMAND &dir2 WAIT;
```

What’s Really Happening?

Calling Program for a Project (Figure 2, page 4)

```
DM 'output' clear;
DM 'log' clear;

%LET root = C:\Files\RMS\Projects\Forecasts;
%LET exroot = C:\SAS\ExportToXL

%LET orig = SEA;
%LET dest = YQB;
%LET datecut = 7/25/16; *cutoff date for the data;
%LET hzn = 15; *forecast horizon;

OPTIONS SASAUTOS=( "&root\Macros", "&exroot" ) MAUTOSOURCE;

%makeSetup;
  *makes the setup structures.  FURTHER FUNCTIONALITY WILL NOT WORK IF COMMENTED OUT;
```

Log Output

```
dir1='mkdir "C:\Files\RMS\Projects\Forecast\Output"'
dir2='mkdir "C:\Files\RMS\Projects\Forecast\Data"'
enddate=20660
```

What's Really Happening?

SAS Code

```
%LET root = C:\Files\RMS\Projects\Forecasts;
%LET datecut = 7/25/16;
%LET outputroot = &root\Output;

DATA _NULL_;
  dir1 = "'"||'mkdir'||' "'||"&outputroot"||'"'||"'";
  dir2 = "'"||'mkdir'||' "'||"&root\Data"||'"'||"'";
  CALL SYMPUTX( 'dir1', dir1 );
  CALL SYMPUTX( 'dir2', dir2 );
  CALL SYMPUTX( 'enddate', INPUT( "&datecut", mmddyy8. ) );
RUN;

DATA _NULL_; %*makes the directories for the graphics;
  SYSTASK COMMAND &dir1 WAIT;
  SYSTASK COMMAND &dir2 WAIT;
RUN;
```

Log Output

```
dir1='mkdir "C:\Files\RMS\Projects\Forecast\Output"'
dir2='mkdir "C:\Files\RMS\Projects\Forecast\Data"'
enddate=20660
```

Basic Idea

Suppose we want to make forecasts for flight 1542:

```
%makeForecasts( fnumber=1542 );
```

Now we want to make forecasts for all flights:

```
%makeForecasts( fnumber=1542 );
%makeForecasts( fnumber=1543 );
%makeForecasts( fnumber=1544 );
%makeForecasts( fnumber=1545 );
%makeForecasts( fnumber=1546 );
```

Problems with above:

- We have to list them out individually.
- We have to find the right flight numbers.

Better Idea

To make forecasts for flight 1542:

```
%makeForecasts( fnumber=1542 );
```

To make forecasts for all flights:

```
%makeForecasts;
```

Flight numbers are determined, listed automatically.

- Easy for testing (test for one before performing for all)
- Easy for drill-down (perform for all, then in depth for one)

Step One

Define the macro for parameter:

%makeForecasts

```
%MACRO makeForecasts( fnumber );

  [Code for making forecasts]

%MEND makeForecasts;
```

Step One

Really easy example:

%makeForecasts

```
%MACRO makeForecasts( fnumber );

  %PUT fnumber=&fnumber;

%MEND makeForecasts;
```

Step Two

Step 2: Create auxiliary macro for getting flight numbers:

%getFlightNumbers

```
%MACRO getFlightNumbers;

  PROC SQL NOPRINT;
    SELECT DISTINCT flightnumber INTO :fnumbers
    SEPARATED BY ''
    FROM datasource
    WHERE orig="&orig" AND dest="&dest"
    ORDER BY by flightnumber;
  QUIT;

%MEND getFlightNumbers;
```

Step Three: Putting It All Together

%makeForecasts

```
%MACRO makeForecasts( fnumber=all );

  %LOCAL i n fnumbers;
  %IF &fnumber = all %THEN %DO;

    %getFlightNumbers;

    %LET i = 1;
    %DO %WHILE( %LENGTH( %SCAN( &fnumbers, &i ) ) > 0 );
     %LOCAL fnumber&i;
     %LET fnumber&i = %SCAN( &fnumbers, &i );
     %LET i = %EVAL( &i + 1 );
     %END;
    %LET n = %EVAL( &i - 1 );

    %DO i=1 %TO &n;
     %makeForecasts( fnumber=&&fnumber&i );
     %END;

    %GOTO theend;
    %END;

   %PUT fnumber=&fnumber;

  %theend:
```

Does It Work?

Macro call		Log output

`%makeForecasts;`	⇒	`fnumber=1542`
		`fnumber=1543`
		`fnumber=1544`
		`fnumber=1545`
		`fnumber=1546`

`%makeForecasts( fnumber=1542 );`	⇒	`fnumber=1542`

Two Parameters

%makeForecasts

```
%MACRO makeForecasts( fnumber=all, method=all );

  %LOCAL i n1 n2 fnumbers methods;
  %IF &fnumber = all %THEN %DO;

    %getFlightNumbers;

    %LET i = 1;
    %DO %WHILE( %LENGTH( %SCAN( &fnumbers, &i ) ) > 0 );
     %LOCAL fnumber&i;
     %LET fnumber&i = %SCAN( &fnumbers, &i );
     %LET i = %EVAL( &i + 1 );
     %END;
    %LET n1 = %EVAL( &i - 1 );

    %DO i=1 %TO &n1;
     %makeForecasts( fnumber=&&fnumber&i, method=&method );
     %END;

    %GOTO theend;
    %END;

    ...
```

Two Parameters

%makeForecasts

```
    ...

  %IF &method = all %THEN %DO;

    %LET methods = AddPick ExSm ARIMA;

    %LET i = 1;
    %DO %WHILE( %LENGTH( %SCAN( &methods, &i ) ) > 0 );
     %LOCAL method&i;
     %LET method&i = %SCAN( &methods, &i );
     %LET i = %EVAL( &i + 1 );
     %END;
    %LET n2 = %EVAL( &i - 1 );

    %DO i=1 %TO &n2;
     %makeForecasts( fnumber=&fnumber, method=&&method&i );
     %END;

    %GOTO theend;
    %END;

   %PUT fnumber=&fnumber;

  %theend:
```

Two Parameters

Macro call   Log output

```
                                 fnumber=1542, method=AddPick
                                 fnumber=1542, method=ExSm
                                 fnumber=1542, method=ARIMA
                                 fnumber=1543, method=AddPick
                                 fnumber=1543, method=ExSm
                                 fnumber=1543, method=ARIMA
                                 fnumber=1544, method=AddPick
%makeForecasts;    ⇒             fnumber=1544, method=ExSm
                                 fnumber=1544, method=ARIMA
                                 fnumber=1545, method=AddPick
                                 fnumber=1545, method=ExSm
                                 fnumber=1545, method=ARIMA
                                 fnumber=1546, method=AddPick
                                 fnumber=1546, method=ExSm
                                 fnumber=1546, method=ARIMA
```

Two Parameters

Macro call		Log output

`%makeForecasts( method=ARIMA );`	⇒	`fnumber=1542, method=ARIMA`
		`fnumber=1543, method=ARIMA`
		`fnumber=1544, method=ARIMA`
		`fnumber=1545, method=ARIMA`
		`fnumber=1546, method=ARIMA`

`%makeForecasts( fnumber=1542 );`	⇒	`fnumber=1542, method=AddPick`
		`fnumber=1542, method=ExSm`
		`fnumber=1542, method=ARIMA`

`%makeForecasts( fnumber=1542, method=ARIMA );`	⇒	`fnumber=1542, method=ARIMA`

Conclusions

- Effective code organization incorporates reusability and automation.
- A recursive definition can make a macro more convenient.
- Recursion can be applied to one or more parameters.
- It can work especially well within a larger framework.

Further Resources

Kirk Paul Lafler.
Efficient SAS Programming Techniques.
Proceedings of the 25th SUGI Conference, 146-25, 2000.

Thomas J. Winn Jr.
Guidelines for Coding of SAS Programs.
Proceedings of the 29th SUGI Conference, 258-29, 2004.

Art Carpenter.
Carpenter's Complete Guide to the SAS Macro Language, 3rd Edition.
SAS Press, 2016.

nderby@stakana.com