Paper 083-2008

Guidelines for Organizing SAS® Project Files

Nathaniel Derby, Statis Pro Data Analytics, Seattle, WA

ABSTRACT

For a beginning SAS® user, there are several resources available for learning how to write efficient programs, but few for how to organize program files, data sets, and related files. This is very important for many reasons such as: knowing where to find everything, avoiding duplicating efforts, and maintaining version controls. This paper presents ideas for organizing these files, along with some short but useful pieces of code.

Keywords: Macros, code, organize.

This paper is an excerpt[1] from Derby (2007a), which may be updated and can be downloaded for more information.

INTRODUCTION

There are many resources available for learning how to write efficient code (e.g., Lafler (2000); Winn Jr. (2004), or many other entries from an internet search of "writing efficient SAS code"), but there appear to be few available for organizing code or project files. This paper is not a comprehensive guide to this subject – rather, it is a short overview of one idea. There are many ways to be organized, and the best way depends on many factors: Is the work done by one person or within a team? Does the project have specifications? Is the project ongoing, or does it have a specific deadline? The opinions contained in this paper are from the perspective of a statistician working in a fast-paced business setting with many changing aspects, within a very small team of managers or analysts without strong SAS or statistics skills.

In the business setting described here, many project aspects change constantly – deadlines get moved, one-time reports become projects (or vice-versa), focuses change, and occasionally an obscure report from the distant past suddenly becomes relevant. The ideas presented here are meant to make it easier to deal with these changes. This organizational style has been effective for the author for over three years and two employers. This is not intended to replace another functional system that works in a given setting – rather, it can serve as a starting point for such a system if none currently exists.

By "project files", we mean any files related to a project or report, such as SAS code, SAS data sets, project documentation, input and output data, or Excel files.

ORGANIZING FILES

The approach advocated here reflects four basic organizational ideas:

- *Never throw anything away.* A project that was cancelled prematurely could be restarted later. It would be inefficient to start completely over.
- *Know where to find everything.* The report submitted six months ago that was never mentioned again could suddenly come up tomorrow, with a request that it be done over again with newer data. The code should be readily available.
- *Make the code reusable.* If the analysis is for *X* and the boss wants the same for *Y*, why re-write the code? Instead, make the code useable for both *X* and *Y*.
- *Automate as much as possible.* Everything should be done by the program – including making new directories or making custom-formatted Excel output (see Derby (2007b)).

By the proposed scheme, files should be organized

- by Company
  - Projects
    - by Name
  - Reports

[1] Reprinted with permission of the author.

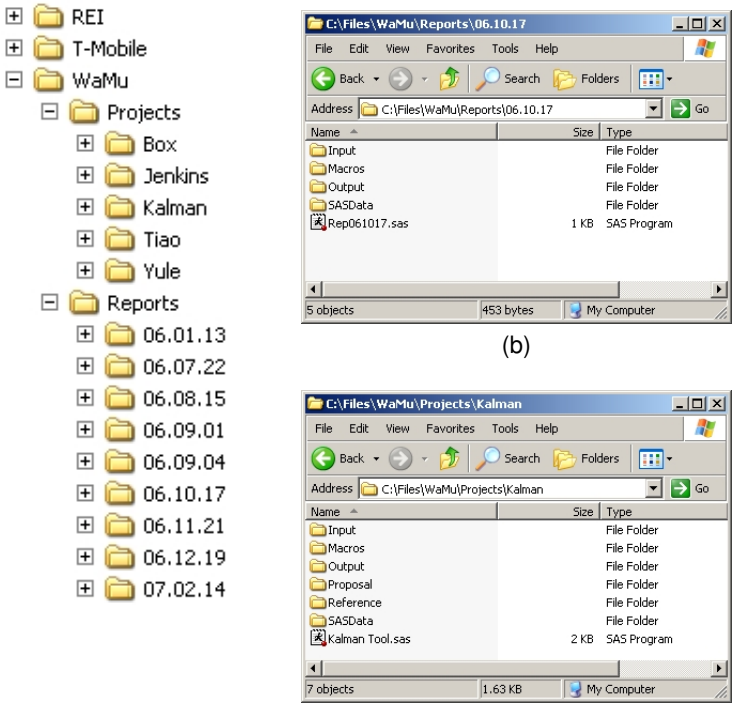

Figure 1: Examples of directory structure in general (a), for a report (b), and for a project (c).

- by Date Given

as shown in Figure 1(a). Generally, a *report* is something relatively minor we are asked to do one time, whereas a *project* is something major or repeatedly updated with new data. There are no exact rules for differentiating the two, and often a report can turn into a project. As a general rule, if it has a name, it is a project. For a report, some notes are worthy of mention:

- For dates, we mean *the date given*, whether to the boss or to other coworkers. If any date pertaining to the report will be remembered by the boss, it will probably be this one (rather than, say, the date it was assigned).
- If two reports are turned in on the same day, we can just assign the them one day apart. The main point is not to get the dates correct, but to keep the relevant files together in one place. The exact date is not important.
- To keep all the files in chronological order, it is best to follow a date format of the year first, then the month, then the day (like *YY-MM-DD*, *YY.MM.DD*, or *YYMMDD*).

FILE ORGANIZATION

For either a project or a report, within the appropriate directory (a date or a name), the contents are the following:

- The *calling program*: The file `RepYYMMDD.sas` for a report or `NAME Tool.sas` for a project. This will be the only file in this directory – everything else will be in a subdirectory.
- The subdirectory *Input*: Contains the raw input data files are.
- The subdirectory *Output*: Contains the final output.
- The subdirectory *SASData*: Contains permanent SAS data sets not in the *Input* or *Output* subdirectories. Clearly this is not needed if no such permanent data sets are used.
- The subdirectory *Macros*: Contains SAS macro definitions[2], which are accessed by the calling program `RepMMDDYY.sas`. This can have a further subdirectory, *Auxiliary* – see Derby (2007a) for details.
- Various other subdirectories, such as *Reference* (for background information) or *Proposal* (for the project proposal). Generally, however, the calling program will only access files from the above four subdirectories.

Of the subdirectories listed above, only *SASData* is restricted to one data type – the others can have PDF, Excel, PowerPoint, Word, or other files. Examples of this organizational structure are shown in Figures 1(b) (for a report) and 1(c) (for a project).

[2]For the reader unfamiliar with macros, we can define a *macro* as simply SAS code written into a separate component, which is called by the calling program. For an introduction to macros, an internet search of "intro to SAS macros" turns up many hits. Burlew (2007) can be a good comprehensive resource as well.

```
DM 'log' clear;
DM 'odsresults' clear;

%LET mainroot = c:\Companies\WaMu\Reports\06.10.17;
  *where the root directory is located;

OPTIONS SASAUTOS=( "&mainroot\Macros" ) MAUTOSOURCE MCOMPILENOTE=all NOTES SOURCE SOURCE2;


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

*%readData;

%analyzeData;

*%exportOutput;
```

Figure 2: Calling program for a report. For an example of a calling program for a major project, see Derby (2007a). Note that some macros are commented out.

CODE ORGANIZATION

For the SAS code itself, we run all parts of the report/project from the calling program (defined above). This program can be partitioned into a preamble, followed by a number of macros.

PREAMBLE

The preamble includes the following:

1. `DM 'log' clear; Dm 'odsresults' clear;` – clears the log and output files. This is very handy for avoiding having to clear them manually after every execution or two. The log and results windows thus pertain only to the most recent run of the calling program.
2. `%LET mainroot = ...;` – the root directory for the report or project. `libname` statements, macro definitions, and input/output directories are derived from this (usually in `%makeSetup` – see below). Other roots may be included here as well – such as `exroot`, for the export macro `%exportToXL`, explained in Derby (2007b). For program portability, it is best to define these in the preamble, so that they can easily be modified for use in another directory.
3. The definition of any global macro variables used later in the program, such as `skills` in Figure 2(b) of Derby (2007a).
4. `OPTIONS SASAUTOS=( "&mainroot\Macros" ) MAUTOSOURCE MCOMPILENOTE=all NOTES SOURCE SOURCE2;` – tells SAS to look for macro definitions in the `&mainroot\Macros` directory, and to show log notes for all macros used. Macros used in other directories as well (such as `&mainroot\Macros\Auxiliary`) must be listed here as well.
5. `%makeSetup` – a macro which serves as a repository for all setup statements, such as defining an output root or `libname` statements. An example is shown in Derby (2007a).

After this preamble, the rest of the program consists of either SAS statements or macros. A general strategy is to start with a basic set of SAS statements, but turn it into a macro as soon as it gets "large". However, it can also be a good idea to group code into macros from the onset, even if only including one procedure or `DATA` statement.

MACROS

The macro calls after the preamble are named after their respective functions, of the form `%verbNoun`. These calls are generally in a necessary order (e.g., `%readData` would precede `%analyzeData`). Furthermore, a semicolon is placed after each macro call, even though this is not necessary – this is done to allow for commenting out a given macro simply by adding an asterisk before it, as shown in Figure 2. Macro definitions typically use macro variables in them, but they do not need to – we can use them purely to organize the code, and add macro variables later for generalization.

An example of a calling program is shown in Figure 2. For another example, an example of `%makeSetup`, or some information on writing macros within this framework, see Derby (2007a). For further examples, see the examples in the download of the `%exportToXL` macro, at `http://exporttoxl.sourceforge.net`.

CONCLUSIONS

This is not meant to be a comprehensive document about file organization – rather, it is just a suggestion for one organizational structure which has worked in a certain setting for a few years. We hope it can be helpful.

REFERENCES

Burlew, M. (2007), *SAS Macro Programming Made Easy*, second edn, SAS Institute, Inc., Cary, NC.

Derby, N. (2007a), Suggestions for organizing SAS code and project files.
`http://www.nderby.org/docs/DerbyN-SASOrg-cur.pdf`

Derby, N. (2007b), User's guide to `%exportToXL`, version 1.0.
`http://exporttoxl.sourceforge.net/docs/exporttoxlv1.0-ug-cur.pdf`

Lafler, K. P. (2000), Efficient SAS programming techniques, *Proceedings of the Twenty-Fifth SAS Users Group International Conference*, paper 146-25.
`http://www2.sas.com/proceedings/sugi25/25/hands/25p146.pdf`

Winn Jr., T. J. (2004), Guidelines for coding of SAS programs, *Proceedings of the Twenty-Ninth SAS Users Group International Conference*, paper 258-29.
`http://www2.sas.com/proceedings/sugi29/258-29.pdf`

ACKNOWLEDGMENTS

I thank my present and past employers for giving me such a high degree of flexibility in organizing my projects. I very much thank SAS technical support for helping me learn the basics of macro programming. I also thank Ron Fehd for providing me with a LaTeX template for SAS conference papers (used here).

Lastly, and most importantly, I thank Charles for his patience and support.

CONTACT INFORMATION

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

Nathaniel Derby
Statis Pro LLC
815 First Ave., Suite 287
Seattle, WA 98104-1404
206-973-2403
`nderby@sprodata.com`