SAS Toolkit for running cleaner statistical analyses: Using macros to declutter your code

Luke W. Johnston

2014-08-22

Contents

Purpose **1**

Installation and usage **2**

Version numbering and meaning **2**

Documentations **3**

HTML 3

PDF 3

Contents **3**

`doc` folder: 3

`src` folder: 4

Style guide for writing comments and code **4**

Purpose

I developed this toolkit to consolidate macros that I use in my research and also to share my efforts with anyone interested, but in particular to share with my fellow labmates and graduate students. As a disclaimer, not all macros are finished and some are very specific and situational, so I encourage anyone to update the files for readability, user-friendliness, flexibility, and parsimony.

Installation and usage

In order to use this repo, run the following shell and git commands (this should run if using Git Bash or other git shell on Windows):

```
cd ~
git clone git://github.com/lwjohnst86/sasToolkit.git
```

Then, include the following code into the top of your SAS file/script and run it:

```
filename macrolib '~/sasToolkit/src/';
options mautosource sasautos=(sasautos macrolib);
```

At this time, the `macro.sas` file will need to be sourced in the SAS file/script, using this command:

```
%inc '$HOME/SAS/src/macros.sas';
```

Version numbering and meaning

Version numbers are specified as tags in git. They are in the format as #.#.#.# (for example, 1.0.4.8 or 1.8.13.11).

- The first number represents a version that has undergone major changes and is likely incompatible with past versions.
- The second number represents the state the macros were in when I published a paper. For example, if I published a paper with the previous version being 1.0.6.0, after the paper has been accepted, the version will change to version 1.1.6.0. Therefore, if anyone needs to go back to check my macros used at that time, they do so using the version number.
- The third number represents that I have added `n` number of new macros. For example, if I added 4 new macros from the previous version (eg. 1.0.3.4), the new version would be 4 numbers ahead (eg. 1.0.7.4).
- The fourth number represents that I have made a change to an existing macro that would change the functioning of that macro or added an arguments. For example, if I added a new argument plus edited an existing argument to function differently, the version number would go from, eg., 1.0.3.4 to 1.0.3.6 (+2).

Table 1: Description of what each version number (in order) means in the versioning system.

Number	Meaning
First (eg. 1.0.0.0)	Major changes. Incompatible

Number	Meaning
	with previous versions.
Second (eg. 0.1.0.0)	State macros were in after one of my manuscripts was accepted for publication.
Third (eg. 0.0.1.0)	Addition of a new macro
Forth (eg. 0.0.0.1)	Changes to an existing macro that affects it's functioning

Documentations

HTML

Contains the documentations generated by DocItOut. Use the `index.html` file in the `doc/html/` or the `documentation.html` file link in the parent folder to view the documentation.

PDF

All the html documentation files have been merged together and sent into pandoc to be converted into a single file manual for those that like pdf and printable versions of documentations.

Contents

There are two primary folders in this toolkit.

`doc` folder:

The `doc` folder contains the documemtation for the macros. So far, I haven't found many documentation generators for SAS, one of which is called DocItOut that is pretty good at generating html pages for looking at the comments and parameters of the custom macros. Open the `documentation.html` file (in the parent directory) in a web browser (for example, Firefox) to view the documentation.

There is also a single file manual in the `doc/pdf` folder. This was generated using code in the Makefile.

`src` folder:

This folder contains all macro files. The format to how the comments are structured are important and need to be consistently adhered to in order for DocItOut to work. Please, if you plan to add to the macros and submit a pull request, be consistent with the way the code is structured (see below).

Style guide for writing comments and code

For writing comments, I based much of what I will be describing below on `DocItOut.pdf` (found in the `doc/` folder). In addition, I also used the website "How to write comments for the Javadoc tool", which provides a more comprehensive documentation on how to write comments that are similar to Java (SAS has similar syntax to Java). For more reference on things to keep in mind when writing code/documentations, see the files in the `doc/` folder. The file `WritingCodeThatLasts.pdf` by Sandra Minjoe is an excellent resource on practices and techniques to make code have greater longevity. The file `WritingMacros.pdf` (slides) by Patrick Breheny is a great resource to learning how to learning how to write macros. Also, I have a few blog posts on my website that go through step-by-step on learning to do basic SAS macros, at least to get started.

When writing up SAS macros, always include documentation. To make it quicker to type of formatted documentation, use the table below for using "quick tags" that can be converted later.

Table 2: Quick tags to use when writing up SAS comment documentation, which can be converted to the html tags by using the `sasQuickTags.sh` in my git `bin` repo. These html tags are used by DocItOut when creating the documentation.

Quick tag	Converted html tag	Detail
{{{		Start a code block
}}}		End code block
{{word}}	<code>word</code>	Code font
[word]	word	Bolds word
\\ (end of line)	<p>	Paragraph break
'&' (space, then & at the end of line)	 	Line break

Code within the SAS macro should always be indented properly and a new line should be started after the SAS semicolon ";" or if the code is longer than 80 characters, whichever comes first.