Improving Programming Efficiency for Statisticians and Programmers with Customization

Duong Tran, Independent Contractor, London, UK

ABSTRACT

A well customized programming environment can significantly boost a statistician or programmer's productivity by simplifying those repetitive tedious tasks that occurs often during a programming session. This paper discusses how to submit SAS® codes interactively from a Program Editor of your choice, define Display Manager (DM) commands that can accept options and introduces the scripting language in Ultra Edit (UE)[5] through an example of defining abbreviation.

Keywords: Gsubmit, Display Manager Command, Scripting, Enhance Editor, Ultra Edit

INTRODUCTION

The priority to search for improvements in the programming environment set up is perhaps less of a concern for some programmers. But with a little investment this quest can reap significant benefit. For example what if one could integrate an alternative Program Editor with powerful features into the DM environment and takes advantages of all its features? The first section of this paper discusses how this is done. That is how to customize and submit SAS codes from UE to an interactive SAS session. Next a discussion on how to define your own DM commands that can accept options. Imagine the flexibility that these options can bring to a command line command. Last a discussion on how to use the script language in UE to define abbreviation to shortcuts typing text, a welcome time saver.

SUBMITTING SAS CODES FROM UE TO AN INTERACTIVE SAS SESSION

Although SAS Institute (SI) has made considerable improvement with its Enhanced Editor (EE), many SAS programmers still prefer an alternative. This preference is obvious for those working with Operating Systems (OS) other than Windows, for example UNIX, where EE is not available. And as for which alternative, there are many specialized Editors in the market to choose from. Most are very cheap or even free, though not surprisingly some features are more powerful than the EE. This section goes through how to set up and submit codes interactively from UE as if UE were integrated into SAS.

The SAS system allows submission of SAS codes stored on the clipboard via the **gsubmit** command and therefore we can copy or cut codes from another application then submit them. To do this, first set up a function key to submit the codes in the clipboard as follow:

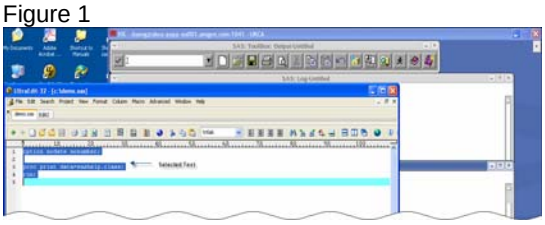
1. In an interactive SAS session - issue the **keys** command to bring up the keys window
2. Set CTRL S (or Q) to:
   output; clear; autoscroll 0; log; clear; autoscroll 0; gsubmit buffer=default;
3. Close the window to save the setting

Note the **autoscroll 0** setting implies maximum scrolling.

Now for copying SAS code from UE into the clipboard - Figures 1 and 2 shows a typical arrangement of panels for an interactive session. Note by clicking the mouse in each panel makes that the active panel.

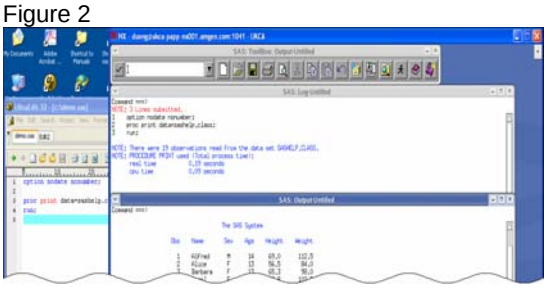
1. Select SAS code by dragging the mouse with left button held down or CTRL+A to select all
2. CTRL+C to store selected text on the clipboard

Figure 1



3. Click the SAS workspace (e.g. any of the SAS panel) to make SAS current
4. CTRL+S to submit the code stored on the clipboard

Figure 2



This method is also a convenient way for submitting sample programs from the SAS System Help or snippet of codes from the internet etc. Also the advantages with interactive mode are well known, you can highlight section and submit codes, test section by section and get immediate response etc.

DEFINING DISPLAY MANAGER (DM) COMMANDS THAT ACCEPTS OPTIONS

Having keystroke commands at your disposal is quite handy during an interactive session. For example instead of manually performing a series of time-consuming, repetitive actions you could customize this into a keystroke command and this can be done with the **%KEYDEF** statement.

/* refer to SAS documentation for more details */
%KEYDEF *key-name* | '*key-name*' | "*key-name*" <text>;

You execute this **%KEYDEF** statement within a macro definition or in open code, for example **%keydef F1 '%tidy';** will execute the **%tidy** macro when **F1** is pressed. But note this mechanism of defining an action command is limited. That is passing parameters to **%tidy** is '**not**' possible. In contrast imagine an issued command that accepts parameters (options) of the form:

/* if you want the string to be a single parameter then it must be in quote */
Command ===> **COMMAND** <<*parameter(n)*> | "<*parameter*>">

where **COMMAND** is your defined macro and <<*parameter(n)*> | "<*parameter*>"> are the parameter(s) that pass into it. For example this could be:

Command ===> **c** mydset ***p*** /* execute the command (i.e. the macro **c**) to **print** mydset */
OR
Command ===> **c** mydset ***f*** /* execute the command (i.e. the macro **c**) to **file** mydset */

This is just like executing **%c**(mydset, parm), where **mydset** and **parm** are the two parameters passed into the **%c** macro. The following example shows that this is possible and very simple to do.

```
/* cmd_mac.sas - when executed make the commands available */
Options cmdmac;
%macro c(dset) / cmd;
  gsubmit "%inc 'proc_contents.sas';";
  output;
%mend c;

/* proc_contents.sas - the action command */
%macro proc_contents;
   :
   data final;
     set &dset; /* &dset is the parameter passed from command line command */
     :
   run;
   :
   proc report data=final;
      etc;
   run;
%mend;
%proc_contents;
```

So here are some examples of usage.

Example 1:
Command ===> **c sashelp.class**

Will display in the SAS output panel:

*** Proc Contents of sashelp.class ***

```
                          Type\
obs           Variable    Length       Label
-----------------------------------------------
1             AGE         N \ 8        Age
2             HEIGHT      N \ 8        Height
3             NAME        C \ 8        Name
```

Example 2:
Command ===> **cmd**

Will display in the SAS output panel:

*** List of Commands ***

```
command       command description
-------------------------------------------------------------
cmd           list available commands
man           display help for the given command
c             display contents of given dataset
v             compare and display variables between two datasets
fmt           display format definition for given format
etc...
```

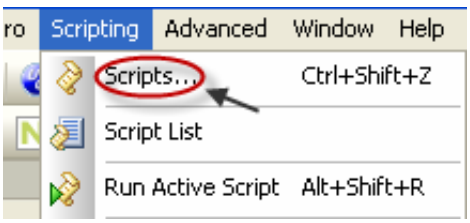
This technique is simple but a powerful way to define DM commands and there is no reason why you can't call other languages for example Perl, or any others for that matter with the SAS **x** statement within your command macro. So in addition to the commands **fsv, lib** etc provided by SAS, **you can define your own**. This concept is analogous to executing a command for example "**ls -la**" or "**dir /w**" at the UNIX or DOS prompt.

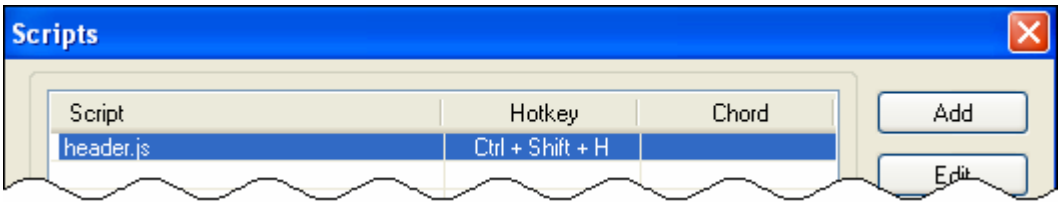
SCRIPTING
Most Editors have some level of macros facilities to record and replay keystrokes and a macro language though maybe limited to build macros. UE has an added similarity - a **scripting** language, which is more powerful than macro. This section introduces an example on how to define abbreviation with scripting. Imaging at a push of a button a defined block of text appears in the editor. This could be a program header, a standard macro call routine, a procedure definition etc.

Step 1: Create the Script
Use any text editor to create and save the script for example:

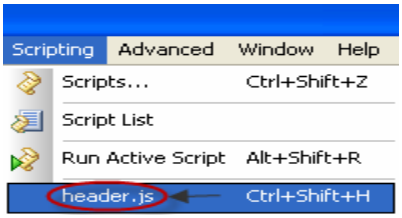
```
// header.js
function header() {
       UltraEdit.activeDocument.write("/*===============\r\n")
       UltraEdit.activeDocument.write("Program Name: \r\n")
       UltraEdit.activeDocument.write("Author: \r\n")
       UltraEdit.activeDocument.write("===============*/\r\n")
}header();
```

Step 2: Add the Script
Click Scripts…then the Add button and add the header.js script





Step 3: Execute the Script



CONCLUSION

Customizing your programming environment does not have to be complicated, but is an essential necessity to improve productivity in programming development and therefore deserves attention, particularly if one is involved in programming daily. This paper has demonstrated that some customizations can be simple yet prove invaluable.

REFERENCES

[1] SAS Explorer: Use and Customization (Richard A DeVenezia); NESUG 2005
[2] Easy SAS Session Customisation (Peter Crawford); SGF 2007
[3] GSUBMIT: Simple Customization of your SAS Application Toolbar in SAS for Windows (Rob Howard); Pharmasug 2004
[4] Lessons Learned: Valuable But Hidden SAS Details (D diTommaso, B Szilagyi); PhUSE 2006
[5] www.ultraedit.com
[6] A Novel Approach to Patient Profiling (Duong Tran) PSI-EFPSI 2008
[7] %RTFParser (Duong Tran); PhUSE 2008
[8] Excel 2 SAS (Duong Tran); PhUSE 2008
[9] %RiTEN (Duong Tran); PhUSE 2007
[10] Inline Formatting (Duong Tran); PhUSE 2007

ACKNOWLEDGMENTS

I would like to thank Gerry Downey for his comments on this paper.

CONTACT INFORMATION

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

Duong Tran
Phone: 07752326413
Email: trand000@aol.com
www.tranz.co.uk