Paper SM04

Your Wish is My COMMAND: Customising your session with SAS® Commands

Simon Purkess, Phastar, London, UK

ABSTRACT

SAS commands are a powerful feature of the SAS environment which go unnoticed by many programmers. The default SAS session comes with many of these commands at your fingertips – for example when you click on the floppy disk icon to save your work or press F8 to run a program, you are actually submitting pieces of command code to perform these actions.

Within this paper I will explain some of the different ways that you can submit commands (the command bar, command line, keyboard shortcuts and toolbar icons) and demonstrate how you can create shortcuts to automate commonly used processes. Finally, I will discuss how you can write macros which can be called with a command.

INTRODUCTION

While writing and running code in the SAS Windowing Environment, we often find ourselves repeating the same simple tasks again and again, such as closing all your open datasets, or clearing the log before submitting a program. Using SAS commands, you can create shortcuts to run these processes at the press of a button.

WHAT ARE SAS COMMANDS?

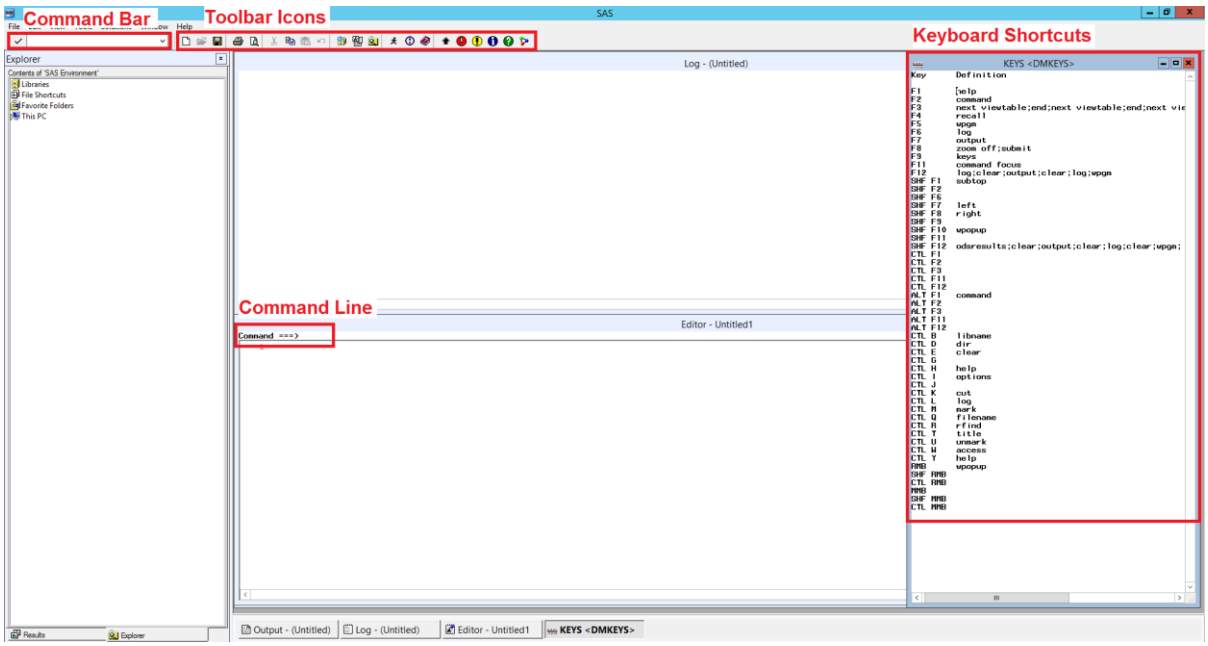
SAS commands are statements that can be submitted to perform a range of actions in the SAS environment, from opening files to searching through text. Whenever you use SAS you are probably issuing many of these commands without realising; for example, when you click the 'Open' icon at the top of the screen you are actually submitting a `FILEOPEN` command, and when you press the F8 to run your program you are actually submitting a `SUBMIT` command.

An important thing to know about commands is that they are **window-specific**. There are several different windows in the SAS environment (e.g. Log, Results, Output) and at any given moment only one of these is in focus, or 'active'. If you click between windows you will notice that the icons on the toolbar at the top of the screen change to give you commands relevant to the active window. Even icons which appear for multiple windows may have different commands attached; for example, the 'New' icon:

Window	Command	Action
Enhanced Editor	WHOSTEDIT	Opens a blank Enhanced Editor file
Output	CLEAR SAVE	Clears the log
Explorer	NEWOBJ	Opens menu to create a new folder, library or program

Like regular SAS code, commands are case-insensitive. Multiple commands can be submitted in the same line if they are separated by semicolons.

FIVE WAYS OF SUBMITTING COMMANDS



The **Command Bar** sits in the toolbar at the top left of the environment. Commands can be submitted by typing them in and pressing Enter, making sure to choose the desired active window first. One useful feature of the Command Bar is that it remembers recent commands, which can be accessed by clicking the arrow on its right end.

You can customise the Command Bar through the Tools > Customize… menu, including choosing how many previous commands to save, and whether to sort previous commands by most recent or most used.

The **Command Line** is turned off by default but can be turned on for the active window or all windows using the `COMMAND` or `PMENU` commands respectively. The fact that it appears at the top of each window helps to ensure that you are submitting commands to the correct window, but it lacks the drop-down list of recent commands.

The KEYS menu (accessed by pressing F9 or submitting the `KEYS` command) can be used to assign commands to **keyboard shortcuts**. Some shortcuts are set by default (e.g. F8 to run your current program) but these can be edited if desired. Note that the KEYS menu is partially but not fully window-specific; the <DMKEYS> menu covers most windows (including Enhanced Editor, Log, Output, Explorer and Results) but VIEWTABLES (i.e. open SAS datasets) have a separate <VT> menu. Changes to either KEYS menu will be saved to your SAS profile, so they are retained between sessions.

The easiest way to add a keyboard shortcut is to open the KEYS menu and type your command next to a key combination. Alternatively, you can use the `KEYDEF` command to set definitions programmatically, for example the command below would assign the help command to F2:

```
keydef F2 'HELP'
```

Commands can also be assigned to **toolbar icons** on the toolbar at the top of the environment. You can edit these icons through the Tools > Customize… menu, in the ‘Customize’ tab. Changes to the toolbar are also saved between sessions.

Finally, a **DM statement** can be used to submit commands through SAS code. You will need to specify the desired active window(s) in the statement, for example:

```
dm 'log; clear; output; clear; wpgm';
```

The command above will clear the log, clear the output, and then set the active window to the most recent editor window.

Commands will run identically regardless of which method is used to submit them, so the decision of which method to use is up to personal preference. The main thing to consider is how often you will be running the command; if it something specific to your current task then the Command Bar is the natural choice, while commands that you want to use in the long-term can be saved to a keyboard shortcut or toolbar icon.

Keyboard shortcuts are limited to 80 characters, so they are only suitable for shorter commands. The command bar and toolbar icons are limited to 500 characters, while DM statements have no size limit.

LEARNING THE LANGUAGE

Unfortunately, documentation for the SAS command language is fairly limited. SAS Help includes some lists of commands but be aware that many of the listed commands are now obsolete. One of the best ways of exploring how you can use commands is through papers like this one; there are some other similar papers listed at the end of this paper.

We can also learn from the commands behind existing SAS menus and shortcuts. In the Tools > Customize… menu we can see the commands behind each of the toolbar icons. Similarly, we can look at the KEYS menus to see the predefined shortcut commands.

A useful command to know is `PREVCMD` (which has the alias `?`). This displays the last command submitted to SAS, regardless of whether it was explicitly submitted by you, or implicitly through a SAS process. This can help uncover commands that SAS is submitting under the bonnet.

EXAMPLES OF USE

Below are a few examples of ways that you can use commands in your programming. Of course, your needs and preferences will be different to mine, but hopefully you will be able to adapt some of the ideas into something that is useful for you.

Example 1: Using a where statement to subset multiple datasets by the same condition

When programming a CDISC dataset I often find myself needing to look at records for a single subject across several work datasets. While this can be done using the right-click ‘Where’ menu for each dataset, it is slow to set up the same where condition for several datasets and there is a chance of clicking on the wrong USUBJID without realising.

A quicker alternative is to open the first dataset and then type `where usubjid='ABC123/01001'` (or alternatively `where usubjid contains('01001')`) into the Command Bar. Then you can open the next dataset, select the saved command from the Command Bar memory and press Enter to submit it.

As with the right-click ‘Where’ menu, the subset is only being applied to the View of the dataset, and no data is removed from the actual dataset.

Example 2: Finding errors and warnings in the log using toolbar icons

When running a long program, it can be time-consuming and inaccurate to check through the whole log visually. The `FIND` command can be used in much the same way as the Ctrl+F 'Find' dialog box to scan through the log and find specific text.

As the list of terms of interest is usually not going to change, this is a good case for creating shortcut icons. In this example we are going to create buttons to look for the following issues:

- ERROR
- WARNING
- uninitialized
- missing values
- converted to

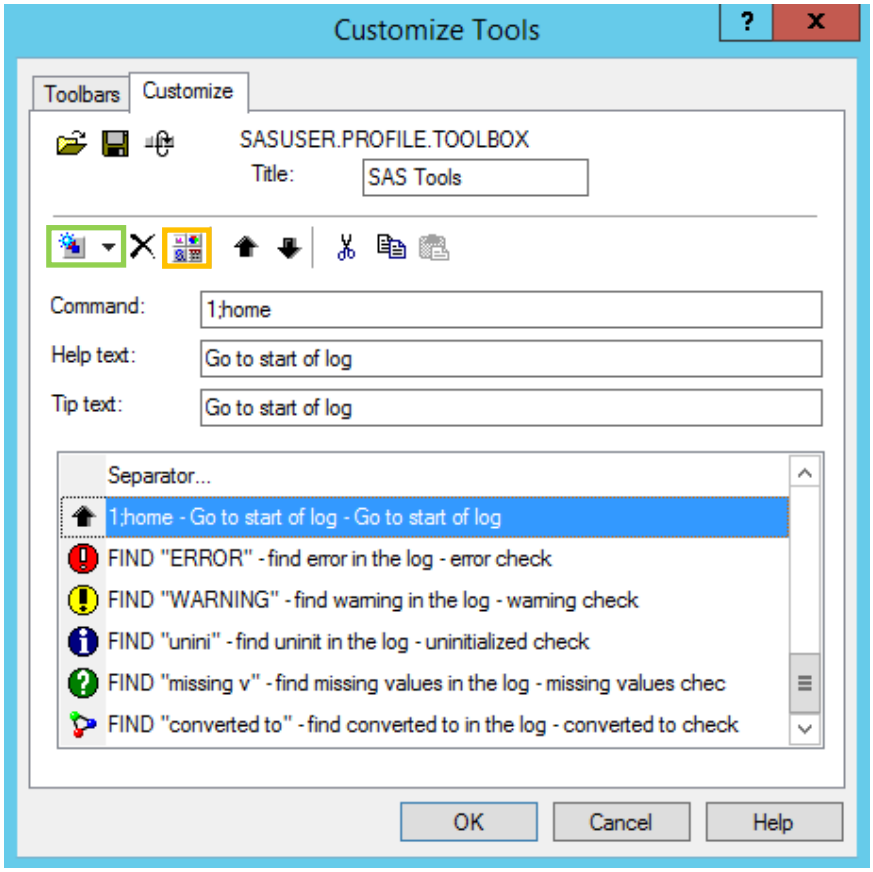
The default syntax for the `FIND` command is simply `FIND "character-string"`. You can add an ICASE option to the end to make the search case-insensitive, but that is not needed for this example.

The `FIND` command will start searching from the current cursor position and won't wrap around to the beginning, so generally you will want to scroll to the top of the log before searching. To aid this you can use the following commands:

```
1;home
```

The `1` scrolls the log window so line 1 is at the top, and `home` moves the cursor to the top left position in the window.

To add the icons, select the log as the active window and then go to Tools > Customize… to open the below menu:



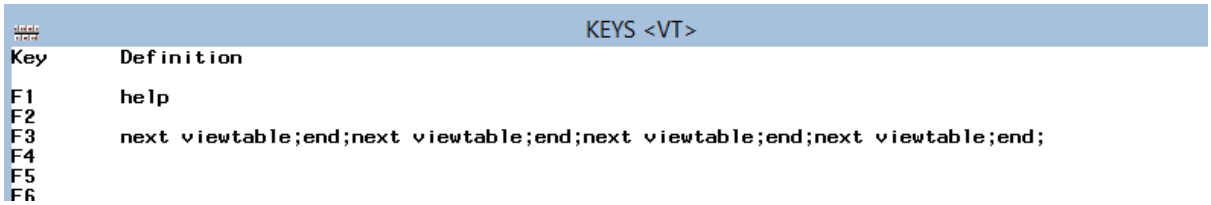
A new shortcut can be created by clicking the button highlighted in green. You can then type the desired command in the 'Command' field and optionally fill out the other fields to describe what the shortcut does. The button highlighted in orange lets you select a bitmap icon for your shortcut. You can also reorder shortcuts and add separators from this menu. Once you have confirmed your changes, you should find your new icons in the toolbar (remembering that they will only be visible while the log is the active window).

Example 3: Closing all open datasets with a keyboard shortcut

When looking through data in SAS, I often end up with several open datasets, which can be time-consuming to click and close individually. The command to close a dataset doesn't change based on its name, so this is a good candidate for a permanent shortcut. You could create a toolbar icon to do this if you preferred, but for this example I'll create a keyboard shortcut.

The first thing we need is a command that closes an open dataset, which turns out to be the `end` command. This is a good start, but it still leaves us having to select each dataset manually before closing them. The `next viewtable` command switches focus to an open dataset, so if we run it ahead of `end` then SAS will select an open dataset before closing it.

Closing multiple datasets is then just a case a running these two commands multiple times. The command language doesn't have a loop function, but we can just type the commands out multiple times:



The character limit only allows for four repeats, but if you have more datasets open then you can press the key multiple times. I would recommend assigning the shortcut to both KEYS menus (<DMKEYS> and <VT>) so you can close datasets regardless of whether you start with a dataset active.

```
CTL RMB
MMB      end
SHF MMB
```

Assigning just the `end` command to the middle mouse button allows you to close a single dataset by middle-clicking on it.

Example 4: Open a saved SAS program

Say you have a SAS program or text file which you reference often, you may wish to create a shortcut so you can quickly open it. This can be done with the `FILEOPEN` command:

```
FILEOPEN "C:\PHUSE\2020\progname.sas"
```

You can then save this into a toolbar icon or a keyboard shortcut, or run it as a dm command or through the command bar.

GSUBMIT

GSUBMIT is a particularly versatile command that allows you to submit SAS code as a command. This means that you can create shortcuts for things like setting options or even running data steps and procedures. The syntax is simply:

```
GSUBMIT "<SAS code>"
```

Example 5: Creating a command to run a recurring summary

Suppose a SAS dataset listing subjects randomized onto a study is being regularly updated as the study continues, and you want a quick way of seeing how many subjects have been randomized to each arm so you can regularly check the study progress. A simple piece of SAS code to do that might look like this:

```
libname rand 'C:\PHUSE\2020' access=readonly;

proc freq data=rand.subjects noprint;
  table arm / out=counts;
run;
```

This code can then be put into a GSUBMIT command:

```
gsubmit "libname rand 'C:\PhUSE\2020' access=readonly; proc freq
data=rand.subjects noprint; table arm / out=counts; run;"
```

You can then create a toolbar icon to run this command, and then you will be able to quickly run the report whenever you want to.

COMMAND MACROS

While GSUBMIT gives you the flexibility to submit SAS code into commands, the character limit significantly restricts what you can fit into a single shortcut. However, you can create SAS macros that can be called as commands, allowing for the same level of freedom that you get with regular SAS macros. These are known as command-style macros, or simply **command macros**.

HOW TO WRITE COMMAND MACROS

There are two things that you need to do when working with command macros. Firstly, you must set the global cmdmac option in your session:

```
option cmdmac;
```

Secondly, any macro that you want to run as a command must have a cmd option in its %macro statement:

```
%macro example / cmd;
  <SAS code>
%mend example;
```

The cmdmac option instructs SAS to scan each word submitted as a command to look for any command macros that have been defined. Note that you can still call command macros as if they were regular SAS macros, but if you define too many command macros then this could affect performance, so it's recommended to only use the cmd option for macros that you intend to call as commands.

Once you have defined your macro, you can call it as a command. This is done by writing the name of the macro (without a % sign) followed by any parameters in order, separated by spaces rather than commas.

Example 6: Creating a list of unique values of a variable in a dataset

Suppose you want to create a list of unique subjects in a dataset. This can be achieved with a short piece of SAS code, creating a macro variable called `uni_vals`:

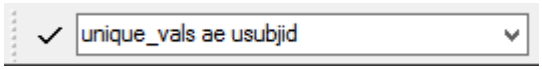
```
proc sql noprint;
  select unique usubjid into :uni_vals separated by ', '
  from dset;
quit;
```

You can then put this into a command macro, allowing us to list unique variables for any given variable and dataset:

```
%macro unique_vals(dset,var) / cmd;
gsubmit "
proc sql noprint;
  select unique &var. into :uni_vals separated by ', '
  from &dset.;
quit;
"

%mend unique_vals;
```

This can then be called as a command:



CONCLUSION

Commands are a little-known part of the SAS environment, but they are a versatile tool to aid your programming. Hopefully this paper has given you a few ideas of how you can utilise commands in your own programming.

REFERENCES / RECOMMENDED READING

1. SAS Help Center: Commands Syntax Listed by Category [https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.5&docsetId=allprodslang&docsetTarget=syntaxByCategory-command.htm&locale=en]
2. SAS Help Center: Commands Syntax Listed Alphabetically [https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.5&docsetId=allprodslang&docsetTarget=syntaxByType-command.htm&locale=en]
3. COMMAND Your Session: SAS® Commands and Command-Style Macros (Benjamin Neely) [https://support.sas.com/resources/papers/proceedings11/099-2011.pdf]
4. Practical Tips to Customize a SAS® Session (Beilei Xu, Xiaohui Wang) [https://support.sas.com/resources/papers/proceedings/proceedings/sugi28/240-28.pdf]
5. Using GSUBMIT command to customize the interface in SAS® (Xin Wang) [https://www.lexjansen.com/pharmasug-cn/2015/PT/PharmaSUG-China-2015-PT71.pdf]
6. Define your Own SAS® Command Line Commands (Duong Tran) [https://www.lexjansen.com/phuse/2011/pp/PP02.pdf]
7. Automagically Copying and Pasting Variable Names (Tabachneck, Herbison, Clapson, King, DeAngelis, Abernathy) [https://support.sas.com/resources/papers/proceedings10/046-2010.pdf]
8. Doing More with the SAS® Display Manager: From Editor to ViewTable - Options and Tools You Should Know (Arthur L. Carpenter) [https://support.sas.com/resources/papers/proceedings12/151-2012.pdf]

CONTACT INFORMATION

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

Simon Purkess
PHASTAR
Unit 2, 2A Bollo Lane
London, W4 5LE
Email: simon.purkess@phastar.com
Web: https://phastar.com/