The SAS System® on the Apple Macintosh™: What should the Display Manager look like?

Al M. Best
Biostatistics Department & Clinical Research Center for Periodontal Diseases,
Virginia Commonwealth University, Richmond, VA 23298

At last year's SUGI it was announced that the SAS System® would be available under IBM PC® DOS. That the SAS System should be available on personal computers is obvious. In addition, the choice of the IBM as the first PC for version 6 was also clear. As anyone who has used the system knows, one of the things that makes this new software so useful is the use of the Display Manager (DM) as an environment for one's session. As such, the DM is presumably designed to be the only thing that might change from system to system. That is, the code for all procedures and DATA step code would essentially remain the same whether one was running on a mainframe, mini or micro. What would change would be how information was passed down to the guts of the SAS system and how the output generated would be presented to the user. As a result, for any system to run the SAS System, one important question is, "what is the look and feel of the Display Manager?" What follows is thus a modest proposal from a long-time SAS user of what the interface should be like.

Disclaimer - Please notice that I am not speaking for SAS Institute. Except for the fact that I teach some of the Institute's courses and have been active in SUGI, I am not officially associated with the Institute in any way. As a result, what follows is simply hopeful speculation which is tempered by an in-depth understanding of both what the SAS and the Apple Macintosh Systems are all about.

Minimum requirements - For the SAS System to run on any machine three conditions must be met: 1) at least 512K of memory, 2) a hard disk of at least 10MB, and 3) a good C compiler comparable to the Lattice C compiler the Institute uses. All of these conditions are met in the Macintosh Plus. Its unmodified system includes 1MB of memory (upgradable to 4MB) and 128K of ROM where much of the system routines lie that would otherwise take up RAM. In addition the new version of the system has a sophisticated Hierarchial File System (HFS) with directories and subdirectories similar to those on the DEC under VMS, DG under AOS or PC under DOS. It also includes an enlarged keypad with a numeric or PF keypad and arrow keys. One may add 20 MB of disk storage by adding the Apple Hard Disk 20 (HD 20). With the SCSI interface on the Mac Plus, many other external, hard disks will also be available. One of the more attractive internal ones will be the internal Hyperdrive now available for the 512K Mac. There are many development systems and C compilers to choose from. The most likely candidates would be Aztec C and Consulair C. So, I know of no technical reasons to reject the Macintosh Plus as a host for the system.

Principals - There are two principals that must be followed for a successful implementation of the SAS System on the Mac: 1) The Macintosh "look and feel" should be fully supported through the Macintosh interface. 2) SAS statements should be identical across systems with few exceptions.

The Macintosh "look and feel" - Why not just port the PC version over to the Mac? The experience of Apple II and PC programmers is that you cannot just recompile a program on the Mac and let it go at that. It will simply not sell. Anyone operating any software on the Mac expects it to be "Mac like" and reviewers have consistently panned software vendors that have tried to take this short-cut. More importantly, it is clear that the Display Manager and the Macintosh were made for each other. Windowing is a prominent feature of Version 6 on the PC. However, a more unified structure would incorporate real windows, Menus and Dialog boxes to replace the awkward LCD display system (lowest common denominator) used on the PC with its command key structure. In my sample session below, it can be seen that the Macintosh interface is an attractive alternative for anyone adverse to learning a large number of command keys.

Identical SAS Statements across systems - Second only to the first principal above, code generated on the mainframe, mini or micro should run identically on any other system. That it is secondary should not be surprising. The most commonly known differences between Base SAS running in different environments has to do with the operating system, particular how files are referred to. Aside from the differences between JCL, CMS, VMS, AOS and DOS, there are comparable differences in FILENAME, LIBNAME, FILE, INFILE and other statements in SAS code. I am sure that anyone working in the different host groups at the Institute could generate a much longer list of operating environment exceptions. What is important is that these be kept to a minimum. In addition, newer versions of the SAS System should be improvements. Version 6 on the PC has the WINDOW statement analogous to the INPUT statement. Every operating environment will dictate certain improvements. The Macintosh is no exception. The goal is this: anyone with a knowledge of MacWrite and the SAS Introductory Guide should be able to run the SAS System on the Macintosh without difficulty.
Organization of the paper - After a large section on a sample session, the Windows, Menus, keys and other features will be discussed. The sample session will demonstrate many of the features documented in the remaining sessions. One will see how Windows operate and what they would look like. Menus demonstrate an important feature of the Macintosh interface. There are essentially no "Syntax Errors" on the Macintosh. The commands one can select from the Menus are the only ones available - you can not try to do something that can not be done. Commands often require additional information. This information is collected in Dialog Boxes that are windows which request information through two kinds of displays: edit fields and buttons. Edit fields are text entry fields where the user may fill in, for instance, a libname (first level name) for storing a permanent SAS dataset. There are 3 kinds of buttons: Radio buttons (which indicate which option of a set of options is to be selected), Check boxes (yes/no selection of an option), and Push buttons (which most commonly indicate that one has completed entries in the Dialog box or that some action is to be taken immediately). All of these features are mentioned here for those not familiar with the Mac.

Sample Session

After turning on your Macintosh Plus with HD-20, the following familiar desktop will appear:

The window (directory) for the HD 20 is open with 3 of its 14 items visible. Double-Clicking or selecting Open from the File Menu with "SAS Folder" selected will show the contents of this sub-directory.

Note that in addition to the Close box in the top left-hand corner of the Title bar on a window, there is also a new feature of Mac windows. This is called a "Zoom Box". Those of you who have used the PC version of the SAS System now know how easy it will be to implement the ZOOM command available in Version 6. But that is getting ahead of the story. The arrow is positioned above the familiar diamond shaped icon of an application (program) on the Mac and double-clicked. The program runs and up comes its windows and menus.

The Editor window is the front-most, active window. One may make the Log the active window by clicking anywhere on it. Making a window larger or smaller is as simple as dragging on the Size-box in the lower, right-hand corner.

Notice that the Mac will always give you immediate visual feedback. The dotted-lines show where the window will be when you release the mouse button.

Moving the window is also as easy as putting the arrow anywhere in the Title bar of a window and dragging it. But let's click on the editor window and zoom it full screen to begin our program.
We will be reading in an external file so let's create the link to the external file with the Fileref statement.

One clicks on the Window Menu to display the selections and, while holding the mouse button down, moves down until the Filename... window is selected. When the mouse is released, the Dialog box for Filename appears.

We key in the name of the fileref we are defining in the edit field and then select the document labeled "Example 1". Notice how the pull-down menu-like directory entry tells you exactly where in the directory this file is located (HD 20 (disk), SAS Folder (directory), Example 1 (filename)).

Like all good SAS programmers, we will need some titles. We could type them in directly or call up the Titles window.

We key in our first title line and press the Enter key or press the Save button to build the Title statement in the Editor. Then we key in a DATA step familiar to anyone who has ever taken a SAS Basics course or read the Introductory Guide. Notice that there are no statement numbers. In version 6 of PC-SAS it is assumed that an SPF or XEDIT like editor would be most familiar to users of that system. That may be true, but MacWrite is the word processor most familiar to Mac users and editing features used in MacWrite are used here. So, we then unzoom the Editor.

We make the log active and use the scroll arrows on the right to scroll up. Where we see that an I was punched instead of a U in the INPUT statement. Now, we could retype the statements in again but Version 6 allows us to bring back statements previously submitted. But, we forgot how to do this and need some help.
The SAS help dialog box appears with its selections for different kinds of help. We select Display Manager and press the Help button (or enter key). Note that one may use up and down arrow keys on the Mac plus to select other entries (as well as using the mouse).

The beginning of the Display Manager help appears. One reads it and sees that what we want is just below the bottom so one scrolls down to read how to do a recall. Note that in addition to the Recall command, there is a special control on the vertical scroll-bar of the editor window. As documented here, this does an automatic recall. There are many ways this function could have been implemented. One reasonable way would have been to have a command key. Another way that would also have its advantages would be to implement the black line control as a split screen control. The idea here would be to scroll up into the previously submitted code area of the Editor window. There are real advantages to this method as it allows you to copy code out of previously submitted code without recalling it. One can then paste it into the to-be-submitted part of the Editor window. It would also allow you to view previously submitted code without recalling it. In any event, the recall command is issued and the statements return to the Editor window.

We could fix it by moving the I beam after the I in INPT, backspacing to delete it, and typing U but we will do a Change command instead.

Select it from the Command Menu and up comes the change dialog box.

Notice that this Dialog (and that resulting from the similar Find... command) has a Title bar. It can therefore be moved anywhere you want if it is hiding something you need to see. We key in the "from" and "to" strings and press the Change All button to make the change. The editor finds the string searched for, highlights it and changes it. I'll add a Proc Print and Resubmit the code. We see that all seems well in the Log and up comes the List Output window with our output.
These data should be familiar to many of you. You may notice that the font style for the title line is different from that used for the rest of the output. The system has defaults for these and thus they may be changed. To see how to do this we can select Preferences... from the File Menu.

Notice that it makes no sense to Open or create a new List Output window, so these commands are dimmed and not eligible for selection. Selecting Preferences... brings up the preferences dialog box for the Output (active) window.

The Vscroll and Hscroll selections allow the user to specify how far the window will scroll when clicking in the grey area of the scroll box. One can specify the screen coordinates of the window or can simply choose the Selection Rectangle Tool to simply draw how you want it to look. In the bottom of the window one sees that there are 5 areas of an output screen. The characteristics of the Title lines are displayed in an example text. The font (presently Geneva, the default application font) and font size (presently 10 point) and style (the default, plain, setting) may all be changed and their effects viewed. But lets leave things alone and press the Cancel button.

Now let move the List Output window out of the way by dragging on its Title bar. To do SAS/AF we select the Procedures... command from the Windows Menu.

Up comes a dialog box to ask us which procedure we wish to run.

One can display any of the 5 categories of procedures but we are only displaying Basic procedures at the moment. Selecting the Means procedure yields its AF screen.

All of the defaults and options are laid out before you. We can change any of these we wish to. Notice also that options not available are dimmed (NWAY and DESCENDING are unavailable without CLASS and ID variables). In any event, I wish to enter specific variables in the VAR statement. Assume for the moment that I can not remember the Variable names, How can I find this information? In Version 6 there is a Libname... window

Which will list the Librefs active

and allow us to select the Files... in Work by selecting the right items. This brings us to the Files... dialog (analogous to the DIR window in the PC)
The Files dialog box tells us that there is only one SAS file in Work, our Class dataset. We can select it and press the Variables button to bring up an abbreviated Proc Contents like screen

```
Libref: Work
Order: Alphabetical Orientation

Name Type Label
Age Num Name
Sex Num Sex
```

It may seem like a long way to get to this screen but with a default Libref of Work and a default File of _LAST_, this screen would be shown if one simply selected the Variables... item in the Window Menu. One can shift-click on the variables one wants, and Copy (either from the Edit menu or Command C). This puts them on the clipboard where they can be pasted into the appropriate entry in the Means Screen.

This data transfer between windows is a feature that can be easily implemented on the Mac. Pressing the Edit button would return us to the editor and allow changing the statements before they are submitted. The Submit button will also save the statements so that they can be recalled later but submits them directly. The Log scrolls with the new statements and notes and the List Output's new screen appears.

A similar set of commands will lead to a VBAR dialog box for PROC GCHART

These graphics can, of course be saved to be modified in MacPaint (as bitmaps) or in MacDraw (as QuickDraw objects). Note that the controls are a little different. The intent is to allow resizing the window (should it clip the image, hide parts of it or redraw the image to the new size?) and scrolling one page (image) at a time. Continuous scrolling could get complicated with half of one image on the top and half on the bottom. Anyway, we are done so let us Quit.

Unlike version 6 on the PC, the Mac prompts you to have a second chance at printing or saving your work before it returns you to the operating system.

This is the end of the sample session. Details as to the Menus, Windows and keypad follows.

Details

Windows - There are two kinds of windows in this proposal for SAS on the Macintosh. The Log, Editor, List Output, and Graphics Output windows are "modeless" in Macintosh terms. They can be moved about at will, resized, scrolled and closed at the users whim. They are called modeless since they do not put the user in a mode of responding to that particular window. The user can ignore the window and make another window the active, front-most window. All of these windows have an associated Preference dialog window in which all appropriate display options may be viewed and changed. These preferences should be immediately evident and all changes should generate Display
Manager statements or SAS statements so that saving the recalled contents of the Editor window and reexecuting it will have identical results.

The remaining windows are, in contrast, "modal" since they put the user in a response mode. These windows (Filename, Libname, Files, Variables, Command Keys, Titles, Footnotes, and Procedures (AF)) are implemented as dialog boxes. The computer is in some measure directing the dialog from the user. This is an unusual feature for the Mac and one which should be avoided because it to some degree is not in the spirit of the Macintosh interface. The user should be in control, not the computer; the computer should respond to the user's actions (whatever they may be), not be directed by the computer as to what to do (i.e., A>).

The only remaining window not demonstrated in the above sample session is the Command Keys window. This is a Key Caps-like accessory, perhaps somewhat akin the MouseTracks desk accessory, which would allow the display and editing of all keys (except X, C, and V, to maintain compatibility with desk accessories). The command keys displayed in the menus could be changed and are only my minimal set - one suggested by a user more comfortable with menus than "PF keys". For those not familiar with the Mac Plus keyboard, it looks much like the VT-100 keyboard. It has a PF keypad on the right with 15 keys and an Enter key. There are arrow keys and Command (Control) and Option (Alt) keys.

In Version 6 there are a number of Window Position, Display Manager global commands. They are made obsolete by the Macintosh windowing interface (except for an analog to WDEF which would allow Autoexec resizing of windows and WSAVE to save a particular setting). The Save entry in the File Menu should be enabled when the preferences dialog is visible to save window settings. The Version 6 Display Manager Global Scrolling Command Keys could be implemented for Autoexec purposes.

User defined Windows should appear as additional entries in the Windows Menu. In a turn-key system using the SAS System, one might want to only display programmer defined windows (and Menus?)

**Menus** - All of the most common commands should be in menus with additional information requested, if necessary, in dialogs or by additional keystrokes.

**The Apple menu** traditionally has an About... entry and most applications have a Help included. I have proposed that Help be included in its own entry, as some other applications do. The About... dialog would include information on memory usage, License information, a phone number and name of the local SAS support person, and a dialog box which would generate Proc Options code to change system options. Additional entries in the Apple Menu are user choosable desk accessories (mini-applications that can run on top of applications currently executing or, as more and more do, run concurrently).

The **File Menu** - also has traditional entries. The New entry will clear all windows (after an appropriate dialog that allows Saving) and essentially begin again. The Open... entry is only useful in the Editor window where it can act as Include, to add SAS statements to the bottom of the editor. Close will clear and close any of the primary windows (after a dialog). Save or Save as will save the contents of any window into text files (where style and fonts will be lost) or perhaps to Microsoft Word and/or MacWrite compatible documents. Graphics output should be saved as either Bitmaps (MacPaint compatible), or QuickDraw items (MacDraw compatible). They could be saved to be GREPLAYed, I suppose? Perhaps the Preferences menu item should appear on the Bottom of the Window Menu (here is where the analog to the COLOR command in version 6 could be best implemented). The Print item should print any window (ImageWriter and LaserWriter support) and Quit would terminate the SAS System and return to the Finder (an identical result would obtain from use of the BYE or ENDSAS statements).

**The Edit Menu** - is the only one virtually required by the Macintosh interface. Many desk accessories use Cut, Copy, and Paste. These can be extensively used in the Editor window. Copy should be enabled for the other three primary menus also. Clear should clear out the contents of the entire window (after a dialog). If the > and < commands now in Version 6 are useful (moving whole blocks of text right and left, respectively), they can be implemented as edit commands available only for the Editor window if at least one line of text is selected. They could be implemented as two-key command keys. That is, by holding down the command key and pressing > then 5 the user could move over the selected text 5 spaces. There could be a default set (perhaps 3 spaces) so the user could simply press "command >". The Mac could wait a short while and then, hearing no numeric keys, implement the default. Actually, I would not recommend this. The Tab and Shift-Tab for moving text right and left would be more Mac-like.

**Absolutely all other editing commands** as implemented in Version 6 on the PC should not be implemented on the Mac (commands such as A, B, C, D, I, M, O and their block commands). Line numbers are not used so Locate is not required (use Find instead). This will thus not allow %Include n:m syntax.
The Windows Menu - has been extensively dealt with in the sample session and in the above section.

The Command Menu - is different for each of the primary windows and should also be user definable (use SELECT and CASE statements in the DATA step?) for user defined windows. The Find and Change items should generate standard dialogs to change (in the Editor only) and find (in Log, Editor, and List Output) text as needed. In the Editor window, Submit, Submit one, and Recall have functions identical to that in Version 6. I am unable to come up with Graphics Commands at the present. Perhaps Greplay could interact with the Commands Menu best.

Final Thoughts

The Mac as a Graphics device - There is no doubt that the Macintosh is a graphics machine powerful enough to warrant the attention of SAS/Graph device driver authors. Apple has recently announced a development system that allows remote host-based applications to control the Mac's user interface. MacWorkStation software would be invaluable for writing mainframe and mini device drivers capable of getting the most out of the Mac. Its software is available for Unix V and VMS.

Buttons and User Defined Windows - The Window statement as implemented in version 6 in Base SAS is an extremely powerful analog to the Input statement. Implementing it on the Mac should be a high priority. Edit fields and static text will be easy to implement but buttons pose a bit of a problem. My suggestion is that buttons be supported as Informats. "Radio." or "CheckBox." could put up a radio button and display the Label or Variable name next to the button. It could have missing=disabled, 0=off, 1=on attributes. "PushButt." could be similarly drawn with the Label text inside. Since there is no "command line" or "message line" in Mac windows, these could be best implemented as items in the Commands menu and dialogs, respectively. User programming of good, interactive windows in version 6 will not be trivial on either the PC or the Mac.

What to do now- The Institute is not full of people just waiting for things to do. Like any business, they respond to the market and/or try to precede it. If users want to see the Institute put resources toward developing the Macintosh as a host for Base SAS and/or other products they should contact anyone they know at the Institute and "agitute" for what they want. At the very least, users should write to Marketing at SAS Institute, Inc., Box 8000, Cary, N.C. 27511-8000.

©1986 Al M. Best
All Rights Reserved