Paper S1-20

Automagically Copying and Pasting Variable Names

Arthur S. Tabachneck, Ph. D.
Insurance Bureau of Canada
Toronto, Ontario (Canada)

Randy Herbison
Westat
Rockville, MD

Andrew Clapson
Ottawa, Ontario (Canada)

John King
Ouachita Clinical Data Services, Inc.
Mount Ida, AR

Roger DeAngelis
CompuCraft Inc.
Newbury Park CA

Tom Abernathy
Pfizer, Inc.
New York, NY

ABSTRACT

Have you ever wished that, with one click, you could copy any table's variable names (or labels) so that you could paste them into your editor or possibly a Word file, powerpoint or spreadsheet? You can and, with just base SAS®, there are some little known but easy to use methods available for automating many of your (or your user's) common tasks.

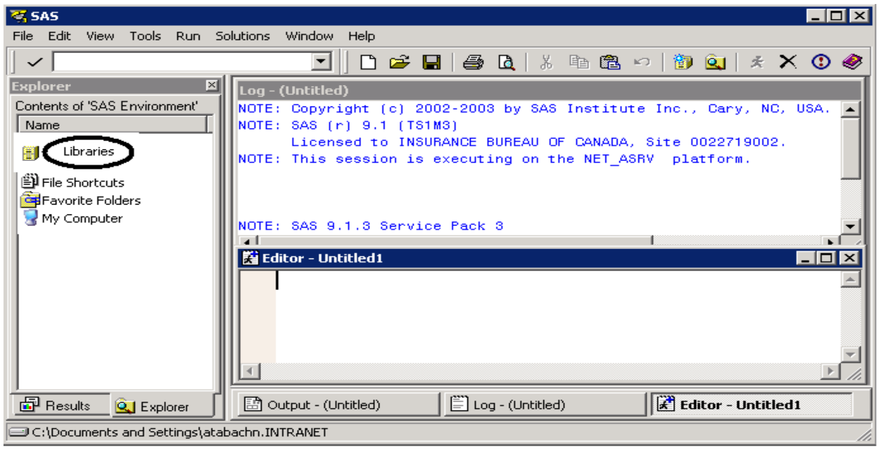
BACKGROUND

This paper describes two methods that capitalize on some powerful, but not well known base SAS tools that let you easily build applications to accomplish common tasks. One method lets users right-click on a file in a SAS explorer window and then select a task from a menu. The other lets you assign a key that, when pressed, gives you or your users a way to specify a one or two-level filename and then run a particular task on the file. Both methods give you an easy way to accomplish virtually any task, process or analysis. Each method was successfully tested on various operating systems, including Windows XP, Windows Server 2003, AIX and SUN.

The question of how to build an application that would let users copy and paste a SAS file's variable names was recently raised on SAS-L and, like many such questions, received some answers (from SAS-L and SAS) that the task might not be doable or could only be accomplished with the help of some external program or language.

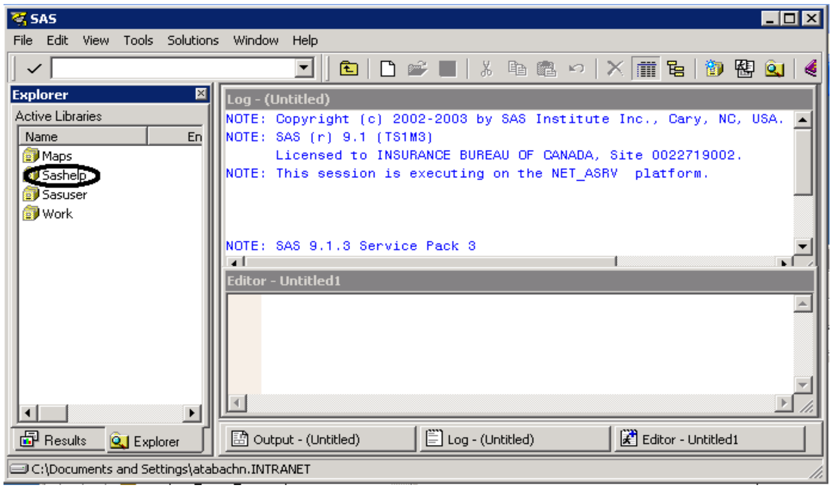
To make a long story short, no external programs, languages, or macros are needed. Our preferred solution is one that capitalizes on capabilities inherent in the SAS Explorer, and only requires base SAS. With SAS Explorer one can easily bring up any library, see all of the files in that library, and simply select the file they are interested in.

For example, given the initial view one sees upon starting a SAS session, one can readily click on a desired environment (e.g., libraries).



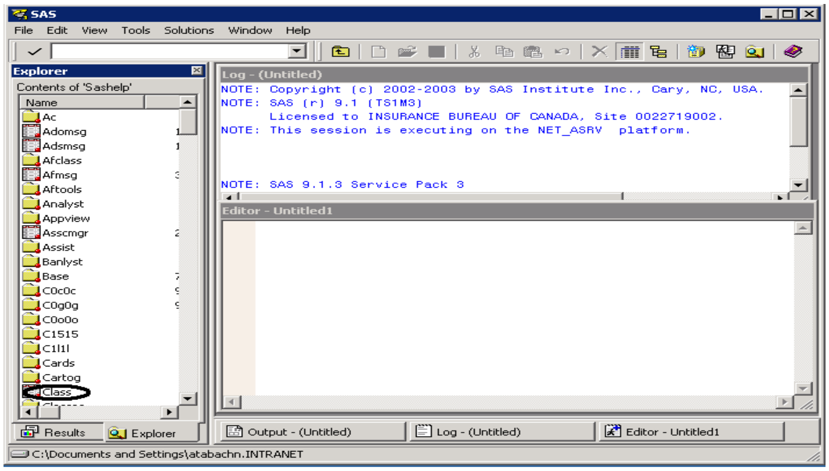
```
NOTE: Copyright (c) 2002-2003 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) 9.1 (TS1M3)
      Licensed to INSURANCE BUREAU OF CANADA, Site 0022719002.
NOTE: This session is executing on the NET_ASRV  platform.



NOTE: SAS 9.1.3 Service Pack 3
```

Upon clicking on ‘Libraries’, one sees the following screen, where you can click on a specific library (e.g., Sashelp):



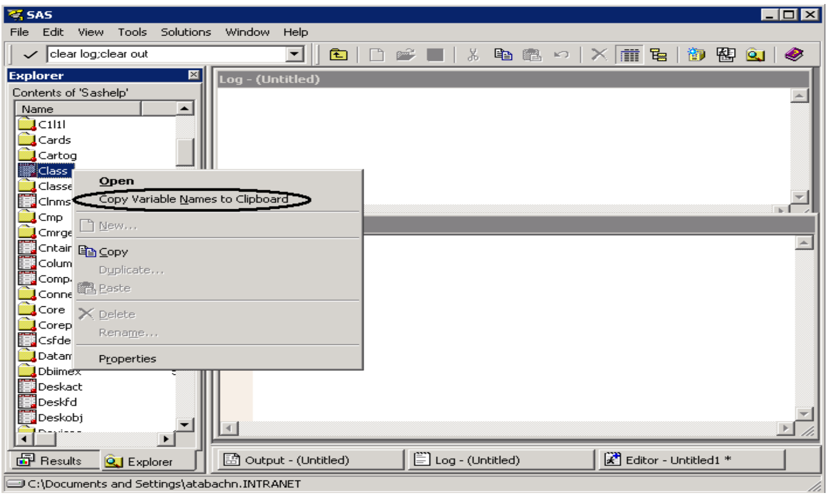
```
NOTE: Copyright (c) 2002-2003 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) 9.1 (TS1M3)
      Licensed to INSURANCE BUREAU OF CANADA, Site 0022719002.
NOTE: This session is executing on the NET_ASRV  platform.


NOTE: SAS 9.1.3 Service Pack 3
```

And, upon clicking on a library, users see a list of that library’s SAS files and catalogs:



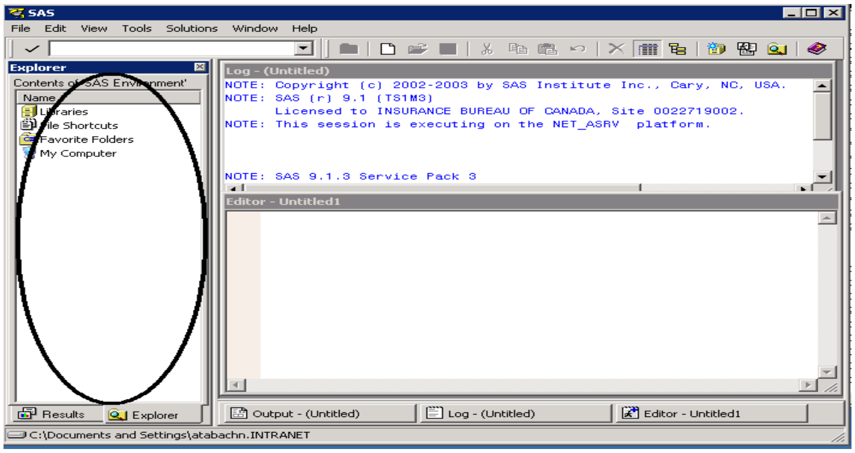
```
NOTE: Copyright (c) 2002-2003 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) 9.1 (TS1M3)
      Licensed to INSURANCE BUREAU OF CANADA, Site 0022719002.
NOTE: This session is executing on the NET_ASRV  platform.


NOTE: SAS 9.1.3 Service Pack 3
```

Wouldn’t it be nice if, from there, users could right-click on a file (e.g., Class), select ‘Copy Variable Names to Clipboard’ from a menu that would appear (or just press the letter ‘N’), and then be able to paste the table’s variable names into a keep or drop statement, a word file, a powerpoint presentation, or virtually any program?



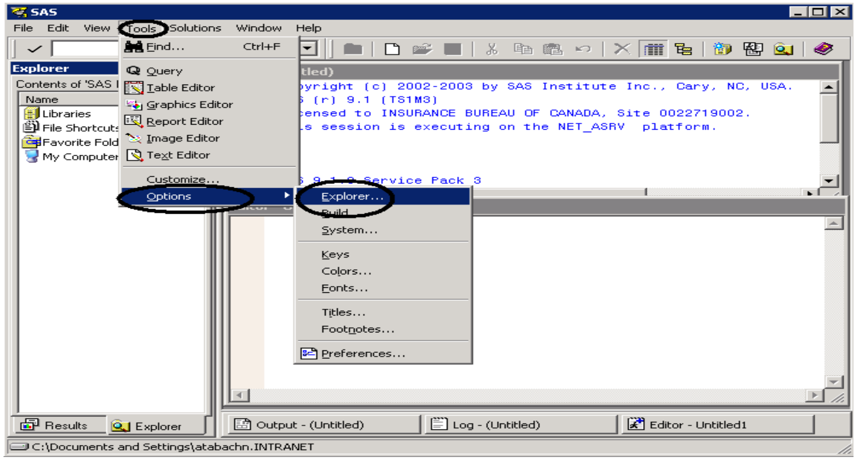
HOW TO DO IT: METHOD 1

Getting there is actually quite easy. First, simply click anywhere in the SAS Explorer side of the screen.

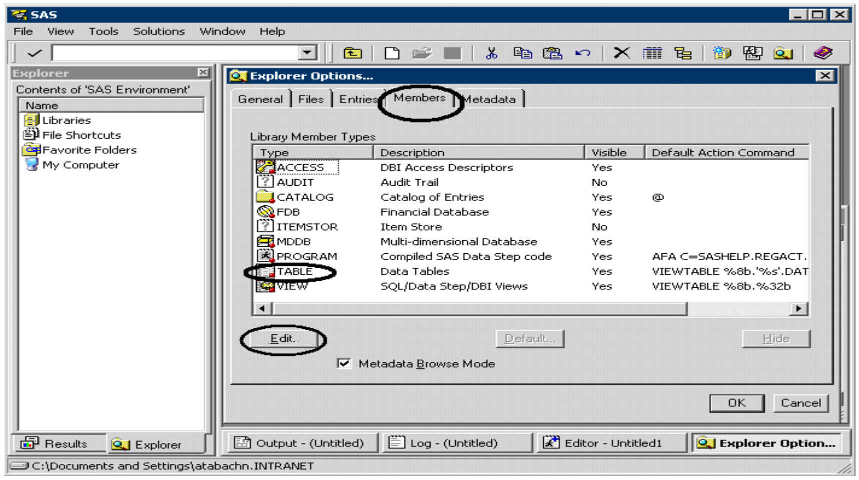


```
NOTE: Copyright (c) 2002-2003 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) 9.1 (TS1M3)
      Licensed to INSURANCE BUREAU OF CANADA, Site 0022719002.
NOTE: This session is executing on the NET_ASRV  platform.


NOTE: SAS 9.1.3 Service Pack 3
```

Then click on ‘Tools→Options→Explorer’:



Click on the ‘Members’ tab and, from there, click on the ‘Table’ Library Member Type and, then, click on the ‘Edit’ button:



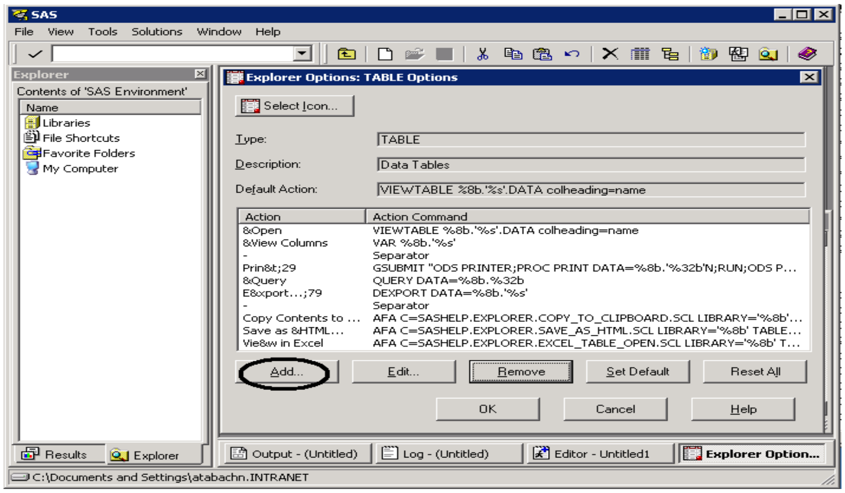
Explorer Options...

Library Member Types

Type	Description	Visible	Default Action Command
ACCESS	DBI Access Descriptors	Yes	
AUDIT	Audit Trail	No	
CATALOG	Catalog of Entries	Yes	@
FDB	Financial Database	Yes	
ITEMSTOR	Item Store	No	
MDDB	Multi-dimensional Database	Yes	
PROGRAM	Compiled SAS Data Step code	Yes	AFA C=SASHELP.REGACT.
TABLE	Data Tables	Yes	VIEWTABLE %8b.'%s'.DAT
VIEW	SQL/Data Step/DBI Views	Yes	VIEWTABLE %8b.%32b

☑ Metadata Browse Mode

From there, click on Add:



Explorer Options: TABLE Options

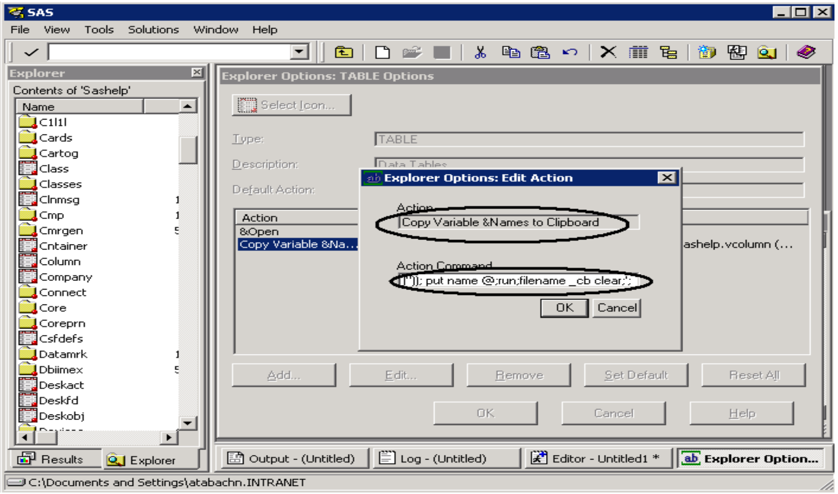
Type: TABLE

Description: Data Tables

Default Action: VIEWTABLE %8b.'%s'.DATA colheading=name

Action	Action Command
&Open	VIEWTABLE %8b.'%s'.DATA colheading=name
&View Columns	VAR %8b.'%s'
-	Separator
Prin&t;29	GSUBMIT "ODS PRINTER;PROC PRINT DATA=%8b.'%32b'N;RUN;ODS P...
&Query	QUERY DATA=%8b.%32b
E&xport...;79	DEXPORT DATA=%8b.'%s'
-	Separator
Copy Contents to ...	AFA C=SASHELP.EXPLORER.COPY_TO_CLIPBOARD.SCL LIBRARY='%8b'...
Save as &HTML...	AFA C=SASHELP.EXPLORER.SAVE_AS_HTML.SCL LIBRARY='%8b' TABLE...
Vie&w in Excel	AFA C=SASHELP.EXPLORER.EXCEL_TABLE_OPEN.SCL LIBRARY='%8b' T...

Then, in the Explorer Options' Add Action screen type the words or phrase you want to include on the menu that users see whenever they right-click on a filename (i.e., the 'Action' users want to perform; e.g., 'Copy Variable &Names to Clipboard'), and the desired Action Command (i.e., the code you want SAS to run when users select the action). In typing the words or phrase you want to use, preceding one of the characters with an ampersand will result in making that letter a shortcut key. Thus, in the present example, right clicking on a file and typing the letter 'N' will have the same effect as right clicking on a file and selecting 'Copy Variable Names to Clipboard'.



Explorer Options: TABLE Options

Type: TABLE

Description: Data Tables

Default Action:

Action	
&Open	
Copy Variable &Na...	ashelp.vcolumn (...

Explorer Options: Edit Action

Action: Copy Variable &Names to Clipboard

Action Command: |"']); put name @;run;filename _cb clear;';

The action command we want, in this case, is:

gsubmit "filename _cb clipbrd;data _null_;file _cb; dsn='%8b'||'.'||'%32b';length name $32;do dsid = open(dsn,'I') while(dsid ne 0);do i = 1 to attrn(dsid,'NVARS');name = varname(dsid,i);put name @;end;dsid = close(dsid);end;run;filename _cb clear;";

Then, exit the window by clicking on OK in response to all of the remaining questions.

Why does it work? Gsubmit runs all of the code between the two double quotation marks. The code itself is simply two filename statements and a data step which accesses the desired variable names and puts the names into your system's clipboard.

You would probably want to enter such instructions as one long line of code like we did, avoiding unnecessary spacing wherever possible, as the action command can only contain up to 255 characters.

Our example code, if entered as a typical data step, might appear as:

```
Line                                    Code
-----    -------------------------------------------------------
   1     FILENAME _cb CLIPBRD;
   2
   3     DATA _NULL_;
   4        FILE _cb;
   5        dsn='%8b'||'.'||'%32b';
   6        length name $32;
   7        do dsid = open(dsn,'I') while(dsid ne 0);
   8          do i = 1 to attrn(dsid,'NVARS');
   9            name = varname(dsid,i);
  10            put name @;
  11          end;
  12          dsid = close(dsid);
  13        end;
  14     RUN;
  15
  16     FILENAME _cb CLEAR;
```

In the code's first line, the filename '_cb' is assigned using the clipboard access method. The clipboard access method, according to the documentation, 'enables you to read text data from and write text data to the clipboard on the host computer.'

Line 3 initiates a _null_ data step and its first command, in line 4, assigns file _cb for writing any text that is later defined by 'put' statements.

When users right-click on a filename in SAS Explorer, the file's libname is automatically written to the first eight bytes of memory, while the filename itself is written to the next 32 bytes. Line 5 takes advantage of that and creates a variable called 'dsn' with a value equal to the desired two-level filename. It does that by concatenating the libname, a period, and the filename.

In line 6, a variable called 'name' is simply declared as a character variable with a length of 32 characters.

In lines 7 thru 13, two nested loops are used to open the selected file, extract the desired information and write the results to the clipboard. The first loop is only necessary to ensure that the data step won't fail in the event that the selected file isn't available for whatever reason. The second loop goes through all of the variables in the selected file, obtains the variable names, and writes the names to the clipboard. By design, a put statement with an @ will write the names on the same line, separated by spaces.

Line 14 ends the data step and line 16 is included to release the filename _cb as it is no longer needed.

As soon as the action is submitted, a user can paste (i.e., type ctrl-v) the names into any keep or drop statement, word file, powerpoint, or anywhere else they choose.

HOW TO DO IT: METHOD 2

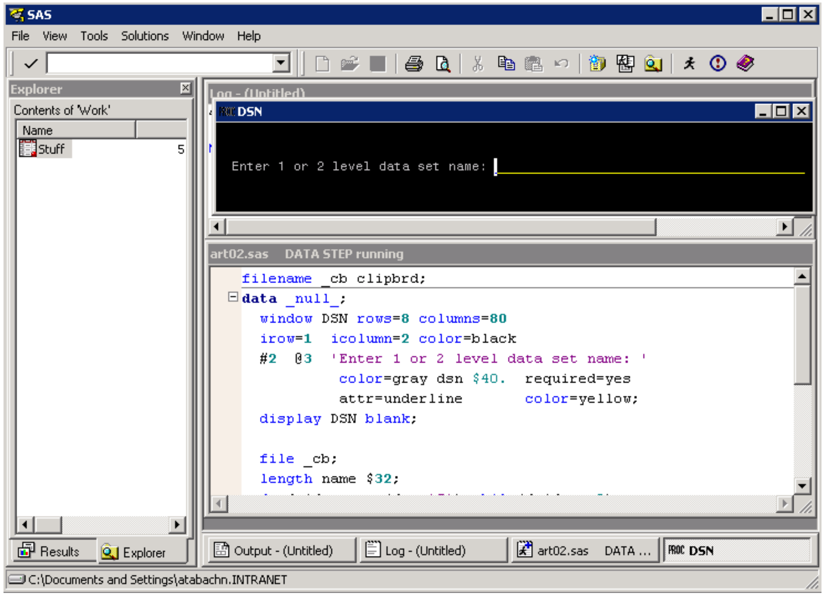
An alternative approach is to use the keydef command to assign a key that, when pressed, will run some SAS code. The code, in this case, is a single data step that (1) uses the Window statement to prompt users to enter the one or two level file name that represents the file from which they want to copy and paste variable names (returning their response as a macro variable) and (2) opens the selected file and copies the variable names directly to the user's clipboard.

For example, consider the following program, that contains a single data step and which we happened to save as "c:\copy.sas":

```
FILENAME _cb CLIPBRD;
```

```
DATA _NULL_;
   WINDOW DSN rows=8 columns=80
   irow=1  icolumn=2 color=black
   #2  @3  'Enter 1 or 2 level data set name: '
            color=gray dsn $41.   required=yes
            attr=underline        color=yellow;
   DISPLAY DSN blank;

   FILE _cb;
   length name $32;
   do dsid = open(dsn,'I') while(dsid ne 0);
     do i = 1 to attrn(dsid,'NVARS');
       name = varname(dsid,i);
       put name @;
     end;
     dsid = close(dsid);
   end;
RUN;

FILENAME _cb CLEAR;
```

The first line assigns the filename _cb as a clipbrd type. Then, the next set of lines causes one's system to prompt the user to enter a one or two level data set name.



```
Enter 1 or 2 level data set name:
```

```
filename _cb clipbrd;
data _null_;
  window DSN rows=8 columns=80
  irow=1  icolumn=2 color=black
  #2  @3  'Enter 1 or 2 level data set name: '
           color=gray dsn $40.   required=yes
           attr=underline        color=yellow;
  display DSN blank;

  file _cb;
  length name $32;
```

If a user enters a two-level name, like sashelp.class, the system will use that file. However, if a user only enters a one-level data set name, the system will find the file (if it exists) in the work directory. The next set of lines opens the selected file, retrieves all of the variable names, and writes them onto the clipboard (i.e., file _cb).

The last line of the code simply releases the assigned filename, as it is no longer needed.

Since the original goal was to accomplish all of the above with a single click, we have to assign the entire process to a key (say, e.g., ctrl-f11).

That can be done with the following command, which (if we had saved the needed code to a file labeled 'c:\copy.sas') would only require submitting one statement to the command line, namely:

```
keydef "CTL F11" "gsubmit '%inc ""c:\copy.sas"";' "
```

You may have to change CTL to CTRL, in the keydef, dependent upon your particular operating system. Of course, if you choose to use this method, after submitting the keydef, you would probably want to enter *keys* (in the command line to bring up the keys window), followed by entering *save* (in the command line) and, when prompted, confirm that you want the key assignment saved for future use.

Then, whenever a user presses the key you defined, they can enter the desired filename and then paste the variable names (i.e., enter a ctrl-v) anywhere in SAS, a word file, a powerpoint presentation, or virtually any program.

ADDITIONAL USES

While some of this paper's authors openly admit to only stumbling on the described methods in a collaborative attempt to find a solution to a SAS-L post, the methods are generalizable to almost any task that has to be repeated. Thus, one could use the methods to build menu items for running proc insight, proc contents, proc descriptive, proc means, or any custom macro one might typically use to gain an understanding of any dataset.

DISCLAIMER

The contents of this paper are the work of the authors and do not necessarily represent the opinions, recommendations, or practices of their respective organizations.

CONTACT INFORMATION

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

Arthur Tabachneck, Ph.D.
Director, Data Management
Insurance Bureau of Canada
2235 Sheppard Ave. East
Toronto, ON L3T 5K9 Canada
E-mail: atabachneck@ibc.ca

John King
Ouachita Clinical Data Services, Inc.
Mount Ida, AR
ouachitaclinicaldataservices@gmail.com

Roger DeAngelis
CompuCraft Inc
1770 Via Petirrojo Apt A
Newbury Park CA 91320
E-mail: xlr82sas@aol.com

Randy Herbison,
Senior Systems Analyst
Westat
1650 Research Boulevard
Rockville, MD 20850
E-mail: RandyHerbison@westat.com

Andrew Clapson
Ottawa, ON Canada
E-mail: andy_clapson@hotmail.com

Tom Abernathy
Pfizer, Inc.
235 E. 42nd Street
New York, NY 1001
E-mail: tom.abernathy@pfizer.com