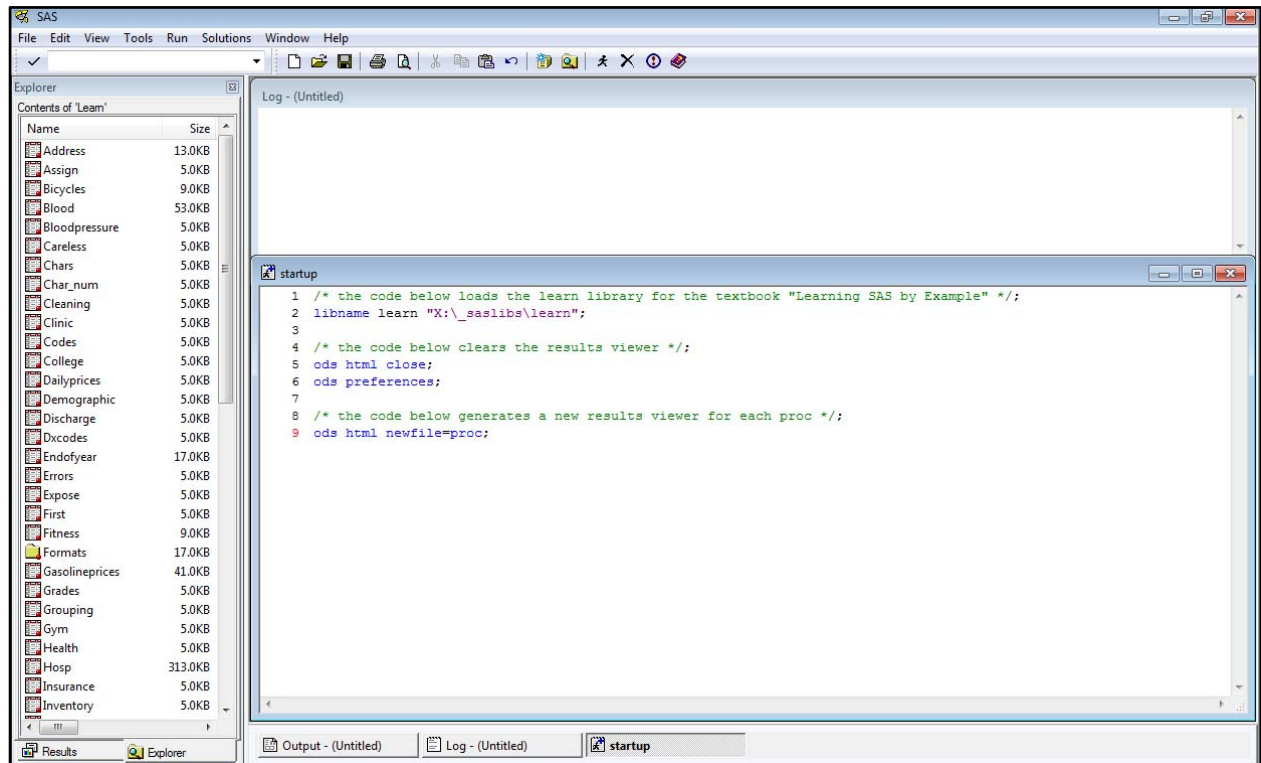


## Summary of SAS Usage Suggestions

### 1. SAS Editor Setup and Usage Suggestions



Setup the SAS Editor so that

- the left-hand panel shows the dataset library explorer in list view, with columns resized so that both dataset names and dataset sizes are readable;
- the right-hand side shows both the log window and the editor window;
- using the Tools -> Options -> Enhanced Editor menu command,
  - enable the "Show line numbers" option,
  - enable the "Insert spaces for tabs" option,
  - enable the "Replace tabs with spaces on file open" option, and
  - enable the "Collapsible code sections", "Show section lines in text", and "Show section brackets in margin" options;
- using the Tools -> Options -> Preferences menu command,
  - under the "General" tab, change the "Recently used file list" entries number to a useful number (e.g., 10);
  - under the "General" tab, enable the "Save settings on exit" option; and
  - under the "Results" tab, consider unchecking the "Use WORK folder" option, setting a replacement folder, and changing the "View result using" option to an internet browser (e.g., Firefox).

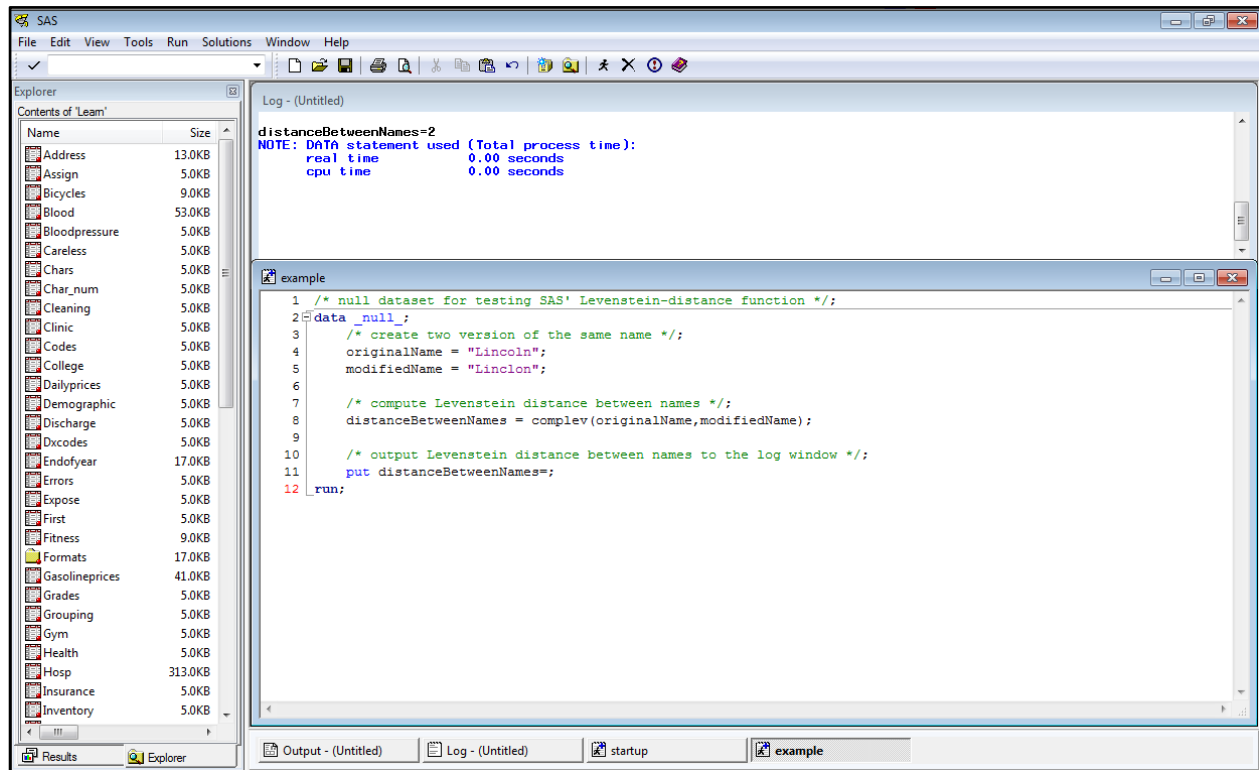
Category	Command	Keyboard Shortcut
Code Folding	Collapse all folding blocks	Alt + Ctrl + Number pad -
	Expand all folding blocks	Alt + Ctrl + Number pad +
Help	Get Help for a SAS procedure	Place the cursor within a procedure name and press F1
	Context Help	F1
Navigation	Move cursor to matching brace/parentheses	Ctrl + [ Ctrl + ]
	Move cursor to matching DO/END keyword	Alt + [ Alt + ]
	Move cursor to next case change	Alt + Right
	Move cursor to previous case change	Alt + Left
Selection Operations	Clean up white space	Ctrl + Shift + W
	Comment the selection with line comments	Ctrl + /
	Undo the Comment	Ctrl + Shift + /
	Convert the selected text to lowercase	Ctrl + Shift + L
	Convert the selected text to uppercase	Ctrl + Shift + U

(source: <http://www.sas.com/offices/europe/uk/support/sas-hints-tips/shortcut.html>)

#### When using the SAS Editor

- place libname statements and useful commands in a file named “startup.sas” and open the file each time you start a new SAS session;
- make frequent use of keyboard shortcuts, especially for folding/unfolding code blocks, navigating between the beginning and end of code blocks, and simultaneously commenting/uncommenting multiple lines of code;
- select a line of code by clicking its line number;
- hold down the shift key and click two line numbers to select a contiguous block of code;
- submit code by first selecting it (e.g., using the techniques above) and then right-clicking in order to use the “Submit Selection” contextual-menu command; and
- right-click within the log window between major programming/analytic steps and select the Edit -> Clear All command to clear the log of contents.

## 2. SAS Coding Convention Suggestions

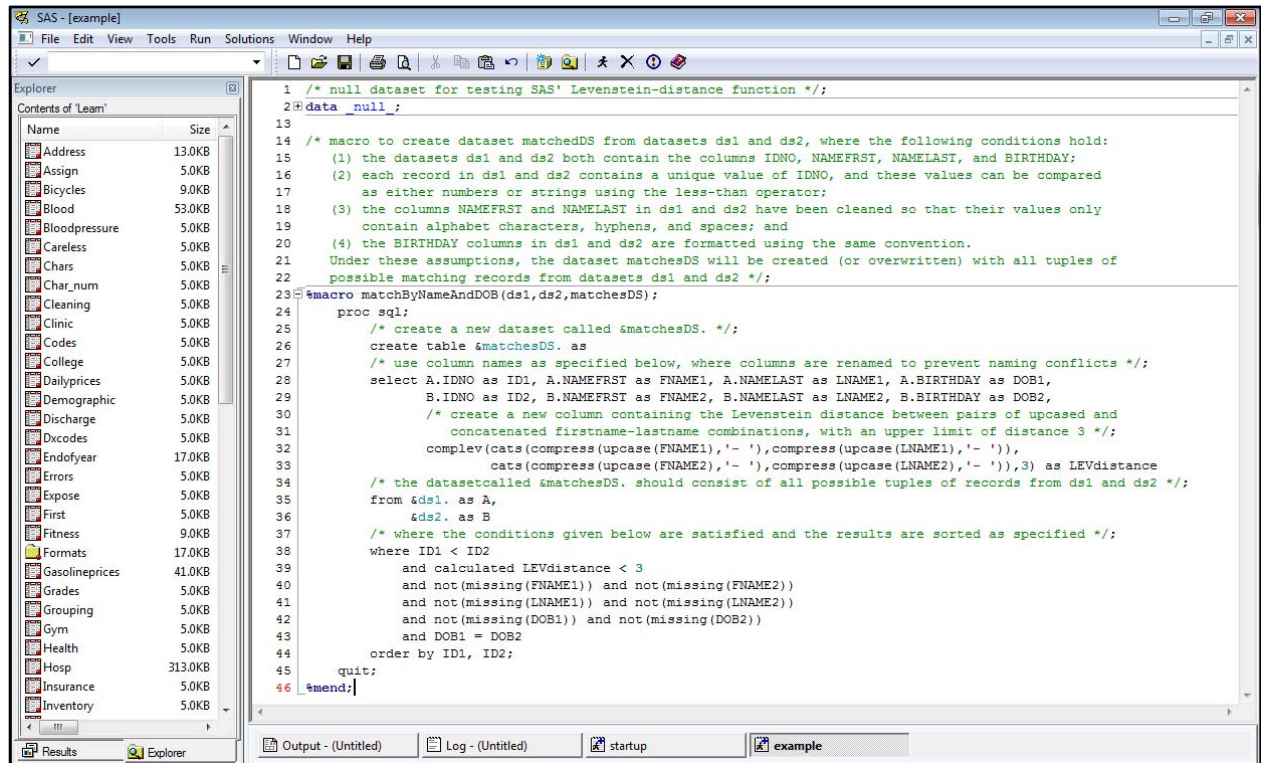


When working with unfamiliar commands/techniques,

- retype example code (rather than copying/pasting) and tinker/modify it until it's clear what each command and option accomplishes;
- test dataset programming commands/options using a `_null_` dataset together with put commands that write useful output to the log window;
- test procs using a small dataset with known or easily predicted results; and
- use previously written code for reference when writing something new but continue to retype code as much as is practical to do so.

When writing code,

- use meaningful identifier names (e.g., use a convention like "camelCase" for variable names and a convention like "all lowercase with underscore delimiters" for dataset and library names);
- properly indent code (e.g., to indent/unindent multiple lines of code simultaneously, highlight the lines and use tab/shift-tab to indent/unindent);
- use `/* ... */` comment blocks with trailing semicolons before each logical programming step; and
- check the log after each code submission for notes, warning, and errors.



```
1 /* null dataset for testing SAS' Levenstein-distance function */;
2 data _null_;
13
14 /* macro to create dataset matchedDS from datasets ds1 and ds2, where the following conditions hold:
15 (1) the datasets ds1 and ds2 both contain the columns IDNO, NAMEFRST, NAMELAST, and BIRTHDAY;
16 (2) each record in ds1 and ds2 contains a unique value of IDNO, and these values can be compared
17 as either numbers or strings using the less-than operator;
18 (3) the columns NAMEFRST and NAMELAST in ds1 and ds2 have been cleaned so that their values only
19 contain alphabet characters, hyphens, and spaces; and
20 (4) the BIRTHDAY columns in ds1 and ds2 are formatted using the same convention.
21 Under these assumptions, the dataset matchesDS will be created (or overwritten) with all tuples of
22 possible matching records from datasets ds1 and ds2 */;
23 %macro matchByNameAndDOB(ds1,ds2,matchesDS);
24 proc sql;
25 /* create a new dataset called &matchesDS. */;
26 create table &matchesDS. as
27 /* use column names as specified below, where columns are renamed to prevent naming conflicts */;
28 select A.IDNO as ID1, A.NAMEFRST as FNAME1, A.NAMELAST as LNAME1, A.BIRTHDAY as DOB1,
29 B.IDNO as ID2, B.NAMEFRST as FNAME2, B.NAMELAST as LNAME2, B.BIRTHDAY as DOB2,
30 /* create a new column containing the Levenstein distance between pairs of upcased and
31 concatenated first-name-lastname combinations, with an upper limit of distance 3 */;
32 complev(cats(compress(upcase(FNAME1),'- '),compress(upcase(LNAME1),'- ')),
33 cats(compress(upcase(FNAME2),'- '),compress(upcase(LNAME2),'- ')),3) as LEVdistance
34 /* the dataset called &matchesDS. should consist of all possible tuples of records from ds1 and ds2 */;
35 from &ds1. as A,
36 &ds2. as B
37 /* where the conditions given below are satisfied and the results are sorted as specified */;
38 where ID1 < ID2
39 and calculated LEVdistance < 3
40 and not(missing(FNAME1)) and not(missing(FNAME2))
41 and not(missing(LNAME1)) and not(missing(LNAME2))
42 and not(missing(DOB1)) and not(missing(DOB2))
43 and DOB1 = DOB2
44 order by ID1, ID2;
45 quit;
46 %mend;
```

When developing code,

- use an iterative "wire framing" approach in which main algorithmic steps are first written as a series of /\* ... \*/ comment blocks (this can be thought of as writing "pseudocode"), the algorithmic steps are checked by "playing computer" (meaning that the steps are followed by hand and program flow is tracked using pencil and paper in order to determine whether the steps will produce the intended results), the series of steps is refined until it accomplishes its intended goal, and, once the pseudocode seems sound, SAS code is written below each comment block (with the comment blocks serving as documentation);
- when performing dataset programming, use "protective" if-commands to check for exceptional cases (like a string being empty) and place code that depends upon particular conditions (like a string not being empty) in an else clause; and
- when writing macros, include datasets and procs for testing the results of the macros within /\* ... \*/ comment blocks.

### 3. Additional Suggestions

- Store hand-editable datasets (e.g., string-heavy datasets that should be spellchecked) in Excel files and access the files in SAS using libname statements;
- embedded R code within SAS files (see <http://www.r-bloggers.com/sas-macro-simplifies-sas-and-r-integration/>) to use R's array processing or graphics packages (e.g., ggplot2);
- use an online search engine to "ask" questions (many questions about SAS usage can be found on websites like <http://stackoverflow.com/questions/tagged/sas>); and
- look for SAS listservs and blogs to follow (e.g., <http://blogs.sas.com/content/>).

## Appendix A. Source code for file startup.sas

```
/* the code below loads the learn library for the textbook "Learning SAS  
by Example" */;  
libname learn "X:\_saslibs\learn";  
  
/* the code below clears the results viewer */;  
ods html close;  
ods preferences;  
  
/* the code below generates a new results viewer for each proc */;  
ods html newfile=proc;
```

## Appendix B. Expanded source code for file example.sas

```
/* null dataset for testing SAS' Levenstein-distance function */;  
data _null_;  
  /* create two version of the same name */;  
  originalName = "Lincoln";  
  modifiedName = "Linclon";  
  
  /* compute Levenstein distance between names */;  
  distanceBetweenNames = complev(originalName,modifiedName);  
  
  /* output Levenstein distance between names to the log window */;  
  put distanceBetweenNames=;  
run;  
  
/* macro to create dataset matchedDS from datasets ds1 and ds2, where the  
following conditions hold:  
(1) the datasets ds1 and ds2 both contain the columns IDNO, NAMEFRST,  
NAMELAST, and BIRTHDAY;  
(2) each record in ds1 and ds2 contains a unique value of IDNO, and  
these values can be compared as either numbers or strings using  
the less-than operator;  
(3) the columns NAMEFRST and NAMELAST in ds1 and ds2 have been cleaned  
so that their values only contain alphabet characters, hyphens,  
and spaces; and  
(4) the BIRTHDAY columns in ds1 and ds2 are formatted using the same  
convention.  
  
Under these assumptions, the dataset matchesDS will be created (or  
overwritten) with all tuples of possible matching records from  
datasets ds1 and ds2 */;  
%macro matchByNameAndDOB(ds1,ds2,matchesDS);  
  proc sql;  
    /* create a new dataset called &matchesDS. */;  
    create table &matchesDS. as
```

Appendix B (cont). Expanded source code for file example.sas

```
/* use column names as specified below, where columns are renamed
   to prevent naming conflicts */;
select A.IDNO as ID1, A.NAMEFRST as FNAME1, A.NAMELAST as LNAME1,
       A.BIRTHDAY as DOB1,
       B.IDNO as ID2, B.NAMEFRST as FNAME2, B.NAMELAST as LNAME2,
       B.BIRTHDAY as DOB2,

       /* also create a new column containing the Levenstein distance
          between pairs of upcased and first-name-last-name
          combinations, with an upper limit of distance 3 */;
       complev(cats(compress(upcase(FNAME1), '- '),
                    compress(upcase(LNAME1), '- ')),
              cats(compress(upcase(FNAME2), '- '),
                    compress(upcase(LNAME2), '- '))), 3) as LEVdistance

/* the dataset called &matchesDS. should consist of all possible
   tuples of records from ds1 and ds2 */;
from &ds1. as A,
     &ds2. as B

/* where the following conditions are satisfied:
   (1) Levenstein distance between concatenated, upcased name columns
       should be at most two (thus allowing for names to vary by at
       most missing/extra letters or to be at most one transposition
       of adjacent letters);
   (2) no matching information should be missing, and
   (3) birthdays should match exactly */;
where ID1 < ID2
      and calculated LEVdistance < 3
      and not(missing(FNAME1))
      and not(missing(FNAME2))
      and not(missing(LNAME1))
      and not(missing(LNAME2))
      and not(missing(DOB1))
      and not(missing(DOB2))
      and DOB1 = DOB2

/* and the results are sorted first by ID1 and then by ID2 */;
order by ID1,
         ID2

;
quit;

&mend;
```