**Disclaimer**

Although I have tested these macros, I would highly recommend understanding how they work before using any of them. That way you will understand how the output is generated and be able to sort through any issues that come up. If you have any questions feel free to contact dvanlunen. Now on to the fun stuff! I. shows how to set up the macro library, II. Explains how to use the premade library, and III. explains how to make your own macro library.

**Pre-Made Macro Library How To**

1. Set Up
   1. Load the Macro Library
      1. Make sure the enhanced editor is the active window (i.e., click in the editor window)
      2. Click Tools > Keyboard Macros > Macros…
      3. Click Import…
      4. Locate the DVSavedMacros.kmf file and open it in the dialog
      5. Click Close
   2. Initialize a program with all the macros defined at the top
      1. In a blank document type (case sensitive): initialize
      2. Press ENTER
      3. All of the variable definitions and a program template automatically appear. You can either run this program (WARNING THIS PROGRAM DELETES THE ENTIRE WORKING DIRECTORY, but that is the first line and can be easily edited out) and then work in a different .sas file (after all the macros are defined) OR you can use it as a template.
      4. You’ll note that the author is set to Dan VanLunen and the change directory line is for his user folder (this can be changed so it automatically shows your name for future calls– see section III. below)
2. Pre-made macro library Abbreviations and explanations
   1. The table below shows how to use each piece of the macro library to make your life easier
   2. Abbreviations work as follows:
      1. Type the abbreviation (case sensitive)
      2. Press ENTER
      3. Enjoy the code that has been generated from less typing. An example is the initialize abbreviation used above to set up the macro library.
   3. Note that for the macro calls, \* for the parameter indicates it is a required parameter, the others have default values.

| **Abbrev** | **Generated Code / Description / Example Code / Output** |
| --- | --- |
| data | **data** NEWDATANAME;  set INPUTDATA;  **run**;  Constructs a data set quickly. |
| means | **proc means data= nway noprint;**  **var ;**  **/\*class / missing;\*/**  **output out= (drop= \_type\_ \_freq\_) mean=**  **;**  **run;**  Constructs a proc means statement quickly (and you don’t have to remember syntax). |
| ulist | %***uniquelist***(uniquelistname=\*,indata=\*,variables=\*,where=/\*where var=8\*/ );  Creates a unique list of variables in indata. Can be used for pairs, triplets, etc by separating variables by spaces. The where condition can limit to a subset of your data. Uses a proc sort nodupkey.  %***uniquelist***(uniquelistname=test,  indata=sashelp.cars,  variables=Type DriveTrain,  where=where DriveTrain=”Rear” OR DriveTrain=”All”); |
| merge | %***mergematch***(mergeddatname=\*,datasets=\*,matchingvars=\*,where=/\*if INDAT1; other data statements;\*/ );  Merges datasets together by matchingvars. The where clause can be used to only take variables from a particular dataset. For example where=if INDAT1; would only include data that was from the first dataset in the list of datasets. Datasets are automatically sorted before the merge.  **data** EPA;  input Description $ Type $;  datalines;  Clunker SUV  ;  **data** CoalIndus;  input Description2 $ Type $;  datalines;  Fun SUV  ;  %***mergematch***(mergeddatname=test,  datasets=sashelp.cars EPA CoalIndus,  matchingvars=type,  where=if INDAT2; /\*this keeps only those lines that matched in the EPA dataset – i.e. the second one in the list\*/  keep Make Model Type Description Description2;); |
| merg | **data** NEWDATANAME;  merge DAT1(in=indat1) DAT2(in=indat2);  by BYVARS;  **run**;  If you prefer to write your own merge statements this abbreviation saves time. Make sure the datasets are sorted by BYVARS. |
| MtoM | %***mergemTOm***(manytomanyname=\*,data1=\*,data2=\*,where= /\*where D1.var=8\*/);  Does a many to many merge of data1 and data2. The where statement can be used for sql where statements about the variables in the data. Variables in data1 can be referenced using D1.varname; in data2 using D2.varname.Note this macro doesn’t allow more than two datasets. Multiple calls can always be run.  **data** first;input a b;datalines;  1 2  3 4  5 6  ;  **data** second;input c d;datalines;  55 90  100 12  24 8  ;  %***mergemTOm***(manytomanyname=test,data1=first,data2=second,where= where D1.a>**1** and D2.d<**90**); |
| sql | **proc** **sql**;  create table OUTNAME as  select \*  from IN1 as D1, IN2 as D2  where    ;  **run**;**quit**;  If you prefer to make your own many to many merge statement this abbreviation saves time. |
| sort | **proc** **sort** data=IN ; by BYVARS; **run**;  Simple dataset sort statement that gets typed quickly. |
| csort | %***sort***(outputname=\*,inputdata=\*,byvars=\*,where= /\*where = var=8\*/,dupoption=/\*nodupkey\*/ );  Sorts data. Just makes it a little quicker to type.  %***sort***(outputname=test(keep= type mpg\_city),inputdata=sashelp.cars,byvars=mpg\_city,where=where type="SUV" ,dupoption= ); |
| Xexcel | **proc** **export** data=DATA\_2\_EXPORT dbms=xlsx  outfile="OUTFILE\_LOCATION" replace;  sheet="SHEET\_NAME";  **run**;  Exports DATA\_2\_EXPORT to an excel file. |
| Xstata | PROC EXPORT DATA=DATA\_2\_EXPORT  DBMS=Stata  OUTFILE="FILE\_LOCATION"  DBMS=STATA REPLACE;  RUN;  Exports DATA\_2\_EXPORT to a Stata data file. |
| stats | %***stats***(outputname=\*,inputdata=\*,vars=\*,stats=mean,classesNopts=/\*classvar1 classvar2\*/ / missing ,where= /\* where variable=8 \*/, dataoptions= /\*nway\*/);  Runs proc means on INPUTDATA across VARS by classes with options in CLASSESNOPTS and organizes output in an easily copy and paste-able table.  **WARNING:** using class variables can be tricky if there are missing values of those class variables because . shows up in the table as the value for the class variable when (1) it is the stat across all values of that class OR (2) it is the stat across all missing values of that class variable. To know which, look at the number suffix on the variable in the Variable field of the output. This suffix is the \_type\_ produced by proc means (see <http://www2.sas.com/proceedings/sugi23/Coders/p73.pdf>). If you want to exclude (1) and only see the most granular level, dataoptions=nway will do that.  **data** test1;  input a b c d;  datalines;  1 2 3 .  9 3 5 2  9 11 1 .  . 2 32 3  . 1 1 .  ;  %***stats***(outputname=test2,inputdata=test1,vars=d,stats=mean min max,classesNopts=a b / missing ,where= /\*where variable=8\*/ , dataoptions= );    Above stats for d are shown across: all values of a and b in row 1, particular values of b and all values of a in rows 2-5, particular values of a and across all values of b for rows 6-8, and particular combinations of a and b for rows 9 to 13.  %***stats***(outputname=test3,inputdata=test1,vars=d,stats=mean min max,classesNopts=a b / missing,where= /\*where variable=8\*/ , dataoptions= nway);    Above Nway is specified, so we only get stats for d across particular combinations of a and b. |
| freq | %***freq***(outputname=\*,inputdata=\*,freqvars=\*,byvars= ,where=/\*where var=8\*/ , tableopts=missprint outcum);  Runs proc freq on INPUTDATA across FREQVARS by BYVARS and organizes output in an easily copy and paste-able table. Also converts percents to decimal values so after pasted they can be easily formatted in excel.  %***freq***(outputname=test,inputdata=sashelp.cars,  freqvars=type origin,byvars=DriveTrain ,  where=where DriveTrain in ("All","Front") ,  tableopts=missprint outcum); |
| percentiles | %***percentiles***(outputname=\*,inputdata=\*,vars=\*,pctls=\*,where=/\*where var=8\*/ ,by=/\*by var\*/ );  Creates a table of custom percentiles. Used for percentiles that proc means can’t come up with. Uses proc stdize to make the output nicely formatted. See  <http://blogs.sas.com/content/iml/2013/10/23/percentiles-in-a-tabular-format.html>  %***percentiles***(outputname=test,inputdata=sashelp.cars,  vars=mpg\_city mpg\_highway,  pctls=**0** to **1** by **.1**,  where=where type="SUV",  by=by Make); |
| Idelim | proc import out=IMPORTED\_DATASET  datafile="FILE\_LOCATION" dbms=dlm replace;  delimiter=",";  guessingrows=**100000**;  run;  Imports a delimited file. **If guessingrows<number of observations, there are potential truncation issues.** To fix these issues, run the above code with a small number of guessingrows. Then in a new editor window, press F4. This will copy the data step infile statement from the log into your code so you can adjust the length of the variables as needed. |
| (r  (w | (rename=());  (where=());  Used in data steps to speed typing. Automatically moves the cursor inside the () so you can continue typing. |
| Iexcel | proc import out=IMPORTED\_DATASET datafile="FILE\_LOCATION" dbms=xlsx replace;  sheet="SHEET\_NAME";  run;  Imports an xlsx file. There shouldn’t be any truncation issues UNLESS a field should be imported as text and has leading zeros. This can be solved by formatting that field in the xlsx file to be “Text”  To import an xls file, change dbms=xls and add guessingrows=65536; as a new line. |
| Istata | **proc** **import** out=IMPORTED\_DATASET file="FILE\_LOCATION" dbms=STATA replace;  **run**;  Imports a .dta STATA file. Note requires newer stata dta file. If you get the error message “Import Unsuccessful” where it says “For Stata, Release flags of 103 to 115 are supported”, open the .dta file in STATA, save the .dta as a stata 12 dta instead. Then it should work. |
| LLdist | geodist(lat1,long1,lat2,long2,'M'); /\*M for miles, default km, add R if radian inputs\*/  Calculates the crow fly distance between two latitude-longitude pairs. |
| ZIPdist | ZIPCITYDISTANCE(strip(zip1),strip(zip2));  Calculates the crow fly distance between two zip codes’ centroids. |
| NtoC | CHARVAR=put(NUMVAR,numformat);  Converts Numeric to Character  CHARVAR=put(NUMVAR,8.);  Makes a character variable CHARVAR out of a number variable NUMVAR with length 8. |
| CtoN | NUMVAR=input(CHARVAR,informat);  Converts Character to Numeric  NUMVAR=input(CHARVAR,8.);  Makes a number variable NUMVAR out of a character variable CHARVAR with length 8.  Can also be used to create dates if the informat is set to a date informat. |

1. Editing the macro library yourself.
   1. How to add your own abbreviations
      1. Make sure the editor window is active (click in the editor window)
      2. Click Tools -> Add Abbreviation…
      3. Enter the short abbreviation in the Abbreviation box (e.g. ulist)
      4. Enter the text you want to appear when you press enter in the big Text to insert for abbreviation box. Format it has you would like with spaces and tabs, they will be made the same way when you use the abbreviation later.
   2. How to modify existing abbreviations
      1. Make sure the enhanced editor is the active window (i.e., click in the editor window)
      2. Click Tools > Keyboard Macros > Macros…
      3. Click on the abbreviation you want to modify
      4. click Edit…
      5. click on the Insert the string … line in Keyboard macro contents: and click modify
      6. modify as you would like
      7. As an example, change the initialize abbreviation and modify the first Insert the string line at the bottom so the author is your name instead of Dan Vanlunen
   3. How to export your macro library
      1. Make sure the enhanced editor is the active window (i.e., click in the editor window)
      2. Click Tools > Keyboard Macros > Macros…
      3. Highlight all of the macros in the list
      4. Click Export…