Intro to Macros

Heide Jackson
heidej@umd.edu

University of Maryland Population Research Center

August 2019

High Level Things to Know

- Macros are objects or functions called within SAS to streamline coding and reduce redundancies.
- This presentation will introduce macro variables and functions and how they can be useful for data manipulation.

Macro Variables

- Macro variables are a way of representing an object that can later be called into a data set.
- Macro variables usually have a & preceeding their name.
- Macro variables are often concluded with a .
- Let is a common way of defining the macro.
- Put displays the content of a macro variable.
- Let's see this in action:

```
%let doors=HELLO;
%put &doors.;
```

Macro variables can hold multiple words numbers or other types of objects that we may want to store

When Might Macro Variables be Useful?

- When working with data that is repetitive and just one element might change every year (for example 2018data, 2019data etc.)–we could create a macro shorthand for referring to the year.
- When working with big data sets or lists of variables. Here's an example:

```
%let keep= my long list of various things;
%let drop= things i do not need anymore;
data savedata (keep=&keep.);
set loaddata (drop=&drop.);
run;
```

Macro Variable Types

- Macros can be defined globally (available throughout the sas session) or locally, available only in the context of a macro function.
- Additionally, sas has some system defined macro variables and users can define their own macro variables.
- To see all macros available in a session use %put _ALL_;

Macro Functions

- Local macro variables are often called in the context of macro functions.
- Functions are a way of calling code that may get used on multiple occasions.
- Let's say we want to recode all dummy variables to be 0 or 1. Here's a macro to do it:

```
%macro dummy(list);
%let count=%sysfunc(countw(&list.));
%do i=1 %to &count;
%let var=%scan(&list,&i);
if &var=2 then &var=0;
 %end;
%mend dummy;
```

Macro Functions

Unpacking the Function

- %macro begins the macro function
- "dummy" is the name of the function for when we call it later.
- (item) contains the local macro variables used within the function. Note: when the function is used they can have different names from whatever is specified when the macro is created.
- %sysfunc, and %scan are internal sas functions for looking within macro objects.
- %do %to and %end indicate we are looping in a macro environment.
- mend macro name indicates the macro is completed.

Calling the Function

- Once defined, this function can be used by using % function name.

```
data cars; set sashelp.cars;
Asia=1; if origin="Asia" then Asia=2;
Europe=1; if origin="Europe" then Europe=2;
USA=1; if origin="USA" then USA=2;
run;
%let mylist=Asia Europe USA;
data cars2; set cars;
%dummy(&mylist.);
run;
```

Debugging the Function

- SAS has two options, mlogic and mprint that can help unpack what a function does and how it interprets macro objects.
- These options can sometimes be quite verbose, so they should be mainly used for debugging purposes.

Defining Macro Variables in SQL

- SQL can be very useful for defining macro variables and macro lists.
- The into command of SQL can help us do this.

```
        proc sql;
select distinct origin into :orig separated by " "
from sashelp.cars;
quit;
```

Using Macros to Read in Other SAS Scripts

- The include command can be used to reference libnames or macros stored in other files.
- This can often be useful if there are a list of functions or libraries that you want to call without adding bulk and redundancy to your current sas script.

```
/*Reads in library locations stored in a
sas file and makes them available in session*/
%include "somelocations\libnames.sas";
```