Chapters 7 & 8: Introducing Macro Variables
Outline

- Automatic Macro Variables
- User-defined Macro Variables
- Processing Macro Variables
- Displaying Macro Variables
- Masking Special Characters
- Manipulating Character Strings
- SAS Functions and Macro Variables
Macro variables allow the user

- to substitute text—particularly repetitive text
- to obtain session information
- to obtain information on text strings
SAS programs often include a single variable used and defined in multiple locations.

%LET allows the user to define a macro variable, often at the start of the program, and substitute the macro variable throughout the program.
Macro Variables-%LET

- **Original code**
  
  title "Citibase Data for 1991";
  data citiday1991;
  set citiday;
  if year(collection_date)=1991;
  run;

- **Modified code**

  %let year=1991;
  title "Citibase Data for &year";
  data citiday&year;
  set citiday;
  if year(collection_date)=&year;
Macro Variables

- SAS’s *macro facility* allows text to be saved as macro variables
- Macro variables are independent of SAS data sets
- Two types of macro variables
  - *automatic*
  - *user-defined*
Macro Variables

- The value of a macro variable is stored in a symbol table
- Automatic macro variables are always available in the global symbol table
- As you saw from the earlier example, macro variables are referenced by preceding their name with a &
Macro Variables

- The macro processor searches symbol tables for a referenced macro variable.
- A reference cannot be identified if it is placed within single quotes; double quotes must be used instead.
- A message will be printed in the SAS log when macro variable references cannot be resolved.
%let year=1991;
title "Citibase Data for &year";
data citiday&year;
set citiday;
if year(collection_date)=&year;
proc print data=citiday&year
   (obs=50);
run;
Automatic Macro Variables

- Automatic Macro Variables are created when a new SAS session starts.
- As mentioned before, they are global and typically assigned values by SAS.
- Users may be able to re-assign values in some cases.
Automatic Macro Variables

- The most common automatic variables reference the current date, day, or time, the current version of SAS or the current SAS data set.
Automatic Macro Variables

title "Yacht Rentals";
title2 "Data from &SYSLAST";
footnote "Created &systime &sysday, &sysdate9";
footnote2 "on &syssscp system using Release &sysver";
footnote3 "by User &sysuserid";
Automatic Macro Variables

proc tabulate data=boats
    format=dollar9.2;
class locomotion type;
var price;
table type,
mean=type*price;
run;
User-defined macro variables

- `%LET` is the most common method to assign a `value` (right side of statement) to your own macro `variable` (left side of statement)
  - Values are stored as character strings
  - Quotation marks are stored as part of the value
%let month=JAN;
title "Citibase Data for &month";
data citiday&month;
set citiday;
cdate=put(collection_date,date9.);
cmonth=substr(cdate,3,3);
if cmonth="&month";
proc print data=citiday&month
   (obs=50);
run;
Processing Macro Variables

- Processing macro variables takes place within SAS’s general text processing:
  - Program is sent to the *input stack*
  - Code is sent to compiler until the end of a step
  - Compiler executes the code
Processing Macro Variables

- SAS *parses* (or tokenizes) the code in the input stack and passes the tokens to the compiler a statement at a time
- Useful in understanding difficulties that arise in resolving macro references
Processing Macro Variables

- Tokens are
  - Quoted strings
  - Numbers
  - Names (SAS commands, infiles, variables, ..)
  - Special characters (*, &, ;, ..)
Example:

\[ sx = \text{sum (of } x1 - x4) ; \]

The 10 tokens are:
\[ sx = \text{sum (of } x1 - x4) ; \]
Processing Macro Variables

- Code is sent to the macro processor when particular token sequences occur
- The *macro triggers* are what you would expect
  - % immediately followed by a name token
  - & immediately followed by a name token
- Macro variables are created/updated in the symbol table then sent to the input stack and tokenized
You can display macro variables in the Log window using either `options symbolgen;` or `%put`. `%put` allows you to print text to the log, as well as macro variables.
Masking Special Characters

- SAS has several characters that can make complex macro variables difficult to print
- There are a couple different ways to handle these difficulties
  - `%STR` and `%NRSTR`
  - `%BQUOTE`
Masking Special Characters

Two methods to print a macro variable that is a sequence of SAS steps:

```sas
options symbolgen;
%let demo=%str(data a; set b; run;);
%let demo=data a%str(;) set b%str(;) run(%str);
```
Masking Special Characters

The % sign can be used within the %str argument to print single quotes embedded in a title.

%options symbolgen;
%let text=%str(Today%'s Weather);
%nrstr() works in the same way as %str(), but can also mask macro characters % and &

options symbolgen;

%let cite=%nrstr( (Grego, Li, Lynch & Sethuraman, 2012));

%put cite is interpreted as &cite;
Masking Special Characters

- \%bquote () ignores special characters during macro compilation and resolves them during execution
- It’s more user-friendly than \%str
Manipulating Character Strings

- Macro character functions are obvious analogs to SAS character functions, but designed to work with macro variables as character strings.
- Some of these work with:
  - `%upcase, %substr, %index, %scan, %cmpres
  - `%qupcase, etc works similarly to `%bquote
SAS Functions and Macro Variables

- `%SYSFUNC` is a powerful command that allows you to introduce standard SAS functions in the macro environment
- Only a limited number of SAS functions are unavailable for use
We have already seen several instances of macro variables combined with text

E.g:

```
data citiday&month&year;
```

SAS may have difficulty resolving some references, but these can be resolved by adding a delimiter to the end of a macro variable name