

Macros for Beginners

- Gitte Churlish
- Churlish Consulting
- Victoria, BC

Overview

- Build a macro – for reading MS/Access tables
 - Illustrate %let , parameter list
 - %macro %mend
 - Nesting of macros
- System Macro variables
- Macro Libraries

Why set up macros

- Provides easy access to repetitive code
- Allows easy re-use of code
- Macro libraries can provide consistency in an organization

Read a MS/Access table

Obtain Code from “import menu”

```
PROC IMPORT OUT= WORK.POLYGON  
    DATATABLE= "POLYGON"  
    DBMS=ACCESS REPLACE;  
    DATABASE="C:\Dawson_mdb\0931066_VE  
G_DDC.MDB";  
    SCANMEMO=YES;  
    USEDATE=NO;  
    SCANTIME=YES;  
RUN;
```

Macro variable

- **Challenge: Need to import many tables**
- *Macro variables -an efficient way of replacing text strings in SAS code.*
- *The simplest way to define a macro variable is to use the %LET statement to assign the macro variable a name*
- *Variable name- subject to standard SAS naming conventions*
- **%let table = Polygon;**

Use %Let

& denotes Macro variable

```
%let table = Polygon;
```

```
PROC IMPORT OUT= WORK.&table
```

```
    DATATABLE= "&table"
```

```
    DBMS=ACCESS REPLACE;
```

```
    DATABASE="C:\Dawson_mdb\0931066_VEG_  
    DDC.MDB"; RUN;
```

- **Always use “ with macro variables**

Changing variable names

```
*%let table = Polygon;
```

```
%let table = Layer;
```

```
*%let table = species;
```

```
PROC IMPORT OUT= WORK.&table
```

```
    DATATABLE= "&table"
```

```
    DBMS=ACCESS REPLACE;
```

```
DATABASE="C:\Dawson_mdb\093I066_VEG  
_DDC.MDB"; RUN;
```

%macro %mend

A macro can be a data step, many data steps, procedure, or a few lines of code

%macro name;

- ▣ Starts a macro
- ▣ Follow SAS naming conventions

%mend;

or

%mend name;

-ends a macro

There are reserved words - such as SCAN that can not be used

%macro -%mend

```
%let table = Polygon;
```

```
%macro import_access;
```

```
    PROC IMPORT OUT= WORK.&table
```

```
        DATATABLE= "&table"
```

```
        DBMS=ACCESS REPLACE;
```

```
        DATABASE="C:\Dawson_mdb\0931066_VEG_DDC.MDB";
```

```
    RUN;
```

```
%mend;
```

```
%import_access
```

```
%let table = species;
```

```
%import_access;
```

Parameter List

- parameter-list –names one or more local macro variables whose values you specify when you invoke the macro.
- Parameters - local to the macro that defines them
- You must supply each parameter name;
- you cannot use a text expression to generate it.
- A parameter list can contain any number of macro parameters separated by commas.
- The macro variables in the parameter list are usually referenced in the macro.

Parameter List

□ Positional

- `%macro import_access(table, lib);`
- `%import_access (polygon,work)`

□ Keyword Parameters

- `%macro import_access(table = , lib =);`
- `%import_access(table = polygon,lib = work);`
- `%import_access(lib = new, table = species);`

Parameter List

```
%macro import_access (table = Polygon);  
  PROC IMPORT OUT= WORK.&table  
    DATATABLE= "&table"  
    DBMS=ACCESS REPLACE;  
    DATABASE="C:\Dawson_mdb\093I066_VEG_DDC.MDB";  
  RUN;  
%mend;  
%import_access (table = Polygon);  
*%let table = species;  
%import_access (table = species);  
%import_access(table = layer);  
%import_access;
```

Parameter list

```
%macro import_access (table =, lib = work,  
Mdb = C:\Dawson_mdb\093I066_VEG_DDC.mdb);  
    PROC IMPORT OUT= &lib..&table  
        DATATABLE= "&table"  
        DBMS=ACCESS REPLACE;  
        DATABASE="&mdb"; RUN;  
%mend;  
%import_access (table = Polygon, lib = old);  
%import_access (table = species, mdb =  
    C:\Dawson_mdb\093I050_VEG_DDC.mdb);  
%import_access(table = layer);
```

Use of . In macro variables

PROC IMPORT OUT= &lib..&table

Will generate **Old.polygon**

PROC IMPORT OUT= &lib.&table

Will generate **oldpolygon**

Use of parameter lists

```
Libname old 'C:\sample';  
%macro import_access (table =, lib = work,  
dir =, Mdb =);  
    PROC IMPORT OUT= &lib.&table  
        DATATABLE= "&table"  
        DBMS=ACCESS REPLACE;  
        DATABASE="&dir.&mdb..mdb"; RUN;%mend;  
%import_access (table = species, lib = old,  
mdb = 093I050_VEG_DDC,  
dir = c:\dawson_mdb);
```

Nesting macros

```
%macro import_all (dir = , mdb =, lib = );  
  %import_access(table = polygon, lib = &lib, mdb =  
    &mdb);  
  %import_access(table = species, lib = &lib, mdb =  
    &mdb);  
  %import_access(table = layer, lib = &lib, mdb = &mdb);  
  proc contents data = &lib.._all_;  
%mend;  
%import_all (dir = c:\dawson_mdb,  
Mdb = 093A050_veg_ddc, lib = a50);  
%import_all (dir = c:\dawson_mdb,  
Mdb = 093A060_veg_ddc, lib = a60);
```

Or a few lines of code

```
%macro site_class5 (site_index =, si = site_cl5);  
• *Creates site class with mid point values ;  
  *   7.5 – 12.5 = SITE CLASS 10;  
  &si = int((&site_index + 2.5)/5)* 5;  
%mend;
```

Example of Use

Data verify;

 set old.species;

 %site_class5 (site_index = sindex, si= si_cl5);

Run;

Proc means data = verify;

 class si_cl5;

 var sindex;

Run;

System Macro variables

- SYSDATE current date in DATE6. or DATE7. format
- SYSDAY current day of the week
- FOOTNOTE "THIS REPORT WAS RUN ON &SYSDAY, &SYSDATE";
THIS REPORT WAS RUN ON Friday, 11May07


Macro library

```
OPTION SASAUTOS = ( 'd:\xy_roads'  
sasautos);
```

In this location, name of file **MUST** match macro name

i.e.

import_access – stored as import_access.sas in D:\xy_roads.

- 
- Try them
 - Read about them
 - Look at others macros