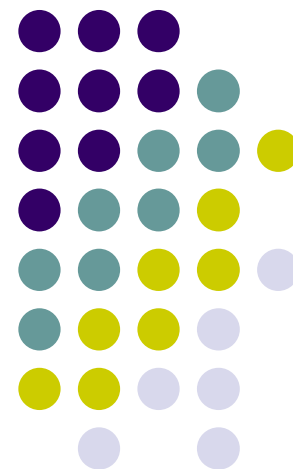
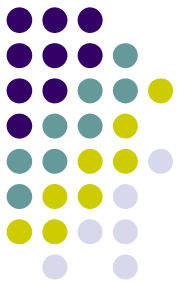


Which SASAUTOS Macros Are Available to My SAS® Session?

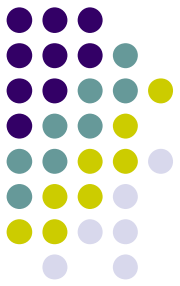
TASS 2009-12-11
Harry Droogendyk
Stratia Consulting Inc.





Introduction

- Briefly, what is macro ?
- SAS macro libraries
- What is AUTOCALL / SASAUTOS ?
- Stored / compiled macros
- System options involving macro
- Macro search order
- %list_sasautos
 - see my website for the latest version
 - fixed a bug that surfaced when used in Unix ;-(



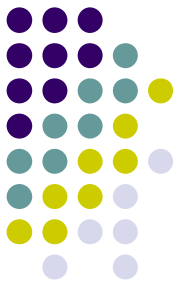
What is a SAS Macro ?

- simplistically – a text generator
 - conditional / iterative processing
 - typically generates SAS code
 - entire steps, statements or snippets
- function style macros
 - return a value
- examples



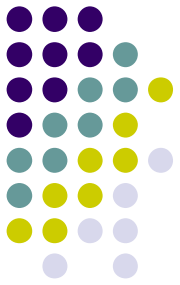
What is a SAS Macro ?

```
%macro print(start=,end=);  
  
    %do year = &start %to &end;  
        title "Sales for &year";  
        proc print data = mylib.mydata;  
            where year = &year;  
        run;  
    %end;  
  
%mend print;  
  
%print(start=2006, end=2007)
```



What is a SAS Macro ?

```
%macro attrn(ds=,attrib=);  
  %let dsid = %sysfunc(open(&ds,is));  
  %if &dsid = 0 %then  
    %put ERROR: (attrn) Opening &ds:  
                                     %sysfunc(sysmsg());  
  %else %do;  
    %sysfunc(attrn(&dsid,&attrib))  
    %let rc=%sysfunc(close(&dsid));  
  %end;  
%mend;  
%put Observations: %attrn(ds=mydata,attrib=nobs);  
Observations: 12
```



SAS Macro Libraries

- Why use SAS Macro Libraries?
 - reuse
 - maintenance
 - consistent, single answer
 - QC



SAS Macro Libraries

- Three types of SAS macro libraries
 - %include
 - compiled stored macros
 - AUTOCALL or SASAUTOS
- can be used independently or in conjunction
 - advantages, but increases complexity



%INCLUDE

```
%macro customer_profit_calc(term=);  
    ... SAS code ...  
%mend customer_profit_calc;
```

- stored in f:\sas\include\customer_profit_calc.sas

```
%include "f:\sas\include\customer_profit_calc.sas"  
        /source2;
```

or

```
filename inc "f:\sas\include";  
%include inc(customer_profit_calc) /source2;
```

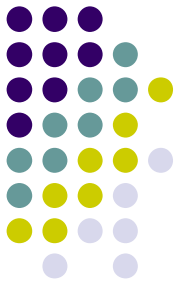
```
%customer_profit_calc(term=12)
```



Compiled Stored Macros

- macros must be compiled
 - by default to WORK.SASMACR catalog
 - overhead
- large, complex macros
 - pre-compiled
 - stored in a catalog in a permanent library
 - entries have TYPE = MACRO
 - MSTORED, SASMSTORE=
 - /store

Compiled Stored Macros



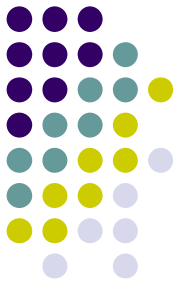
```
libname macrocmp "f:\sas\macros\compiled";
options mstored
        sasmstore = macrocmp;

%macro compiled_macro /store source
        des = 'Meaningful macro description';

        %put Now executing compiled_macro ;

%mend;
```

Compiled Stored Macros



```
proc catalog catalog=macrocmp.sasmacr;  
  contents;  
run;
```

Contents of Catalog **MACROCMP.SASMACR**

#	Name	Type	Create Date
---	------	------	-------------

1	COMPILED_MACRO	MACRO	12JUL2008:14:53:35
---	-----------------------	--------------	--------------------

Modified Date	Description
---------------	-------------

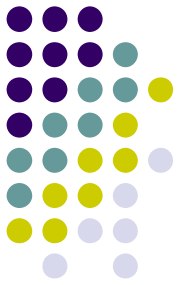
12JUL2008:14:53:35	Meaningful macro description
--------------------	-------------------------------------

AUTOCALL Facility



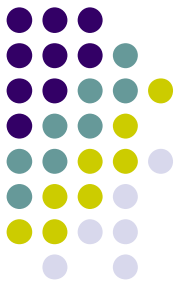
- makes macro source available
 - MAUTOSOURCE must be on
- AUTOCALL macros may be in:
 - directories
 - source catalogs
- search paths defined:
 - in sasv9.cfg
 - via SASAUTOS system option
- SAS supplied examples:
 - %LEFT, %LOWCASE

AUTOCALL Facility



```
/* Setup autocall library definition */  
  
-SET SASAUTOS (   
    "!sasroot\core\sasmacro"  
    "!sasext0\dmine\sasmacro"  
        <snip>  
    "!sasext0\share\sasmacro"  
    "!sasext0\stat\sasmacro"  
)
```

AUTOCALL Facility

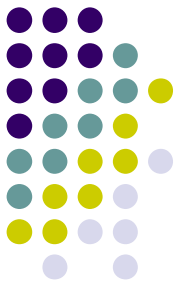


```
libname macros 'f:\sas\macros\sasautos\catalogs';  
  
    * location of source catalog;  
filename catmacro catalog 'macros.macrocat';  
    * specific macro source catalog;  
  
filename othmacro 'f:\sas\macros\sasautos\project';  
    * dir containing macro source;  
  
options mautosource  
        sasautos = ( catmacro othmacro  
                    "f:\sas\macros\sasautos"  
                    SASAUTOS );
```

AUTOCALL Facility

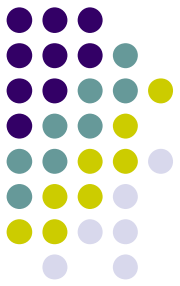


- .sas and SOURCE entries
 - each file / entry must contain only one macro
 - file / SOURCE entry must have same name as macro
 - %customer_profit macro :
 - %customer_profit.sas
 - SOURCE entry – customer_profit



Macro Search Order

- how does SAS decide?
 - three distinctly different methods
 - %include ⇨ defaults to WORK.SASMACR
 - compiled stored macros
 - AUTOCALL locations



Macro Search Order

- order ?
 - WORK.SASMACR
 - compiled stored macros
 - AUTOCALL locations
- gotcha #1
 - compiled macro called %npv_calc
 - in the SAS code of the program we're running:

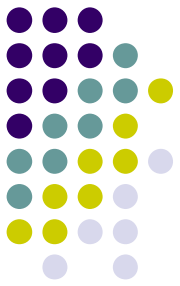
```
%macro npv_calc;  
    ... SAS code ...  
%mend npv_calc;
```

```
%npv_calc
```



Macro Search Order

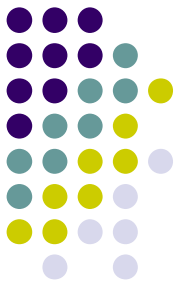
- gotcha #2
 - SASAUTOS location contains customer_profit.sas macro definition
 - invoke %customer_profit, searches:
 - WORK.SASMACR
 - compiled stored macros
 - AUTOCALL concatenation
 - %customer_profit from AUTOCALL location is compiled to WORK.SASMACR
 - *oops*, **error**, update customer_profit.sas
 - invoke %customer_profile again.....



Where *is* that macro ?

- gotchas have demonstrated potential confusion
- addition of new source catalogs into existing AUTOCALL locations
- new compiled macros

- %list_sasautos



`%list_sasautos`

- macro definition

```
%macro list_sasautos(help,work=Y);
```

- positional parameter, help

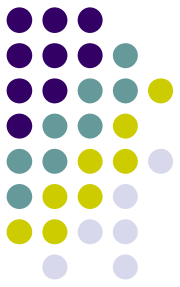
- ? or HELP

```
%list_sasautos(?)
```

- keyword parameter, work

- Y = include WORK.SASMACR in search

`%list_sasautos`



```
%list_sasautos(work=Y);
```

Lists the .sas files and catalog source/macro objects found within directories surfaced by:

- `getoption('sasautos')` - SASAUTOS altered by options statement
- `pathname('sasautos')` - config file SASAUTOS definition
- filerefs / catalogs found within SASAUTOS definitions
- `pathname(getoption('sasmstore'))` - compiled macros.

If `&work=Y` (default), compiled macros from the WORK library will be included as well.

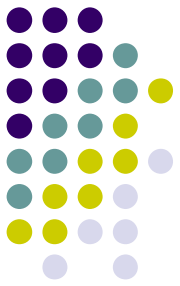
NOTE: Not every .sas / source module found within these directories is

NOTE: NECESSARILY a macro definition. `%list_sasautos` does NOT open up

NOTE: objects to verify the presence of the required `%macro` statement.

In addition to the OUTPUT report (use ODS for fancier report formats), results are also available in the `WORK._LIST_SASAUTOS` dataset.

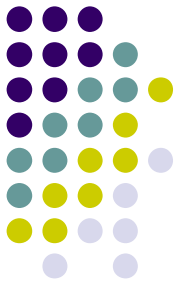
%list_sasautos



```
/*
   work.sasmacr and the sasmacr catalog found at the SASMSTORE location
   are the first searched.  If &WORK=Y, grab the path info for the work
   directory.  If the SASMSTORE option is set, surface the path information
   for that library.  We'll include at the front of the concatenation since
   reflects the search order SAS uses.
*/
%if %upcase(&work) = Y and %sysfunc(cexist(work.sasmacr))
%then %let work_lib =
        %unquote(%str('%')%sysfunc(pathname(work))%str('%'));
%else %local work_lib;

%let sasmstore = %sysfunc(getoption(sasmstore));
%if &sasmstore ne %then %do;
    %let sasmstore = %sysfunc(pathname(&sasmstore));
    %if &sasmstore ne %then
        %let sasmstore = %unquote(%str('%')&sasmstore%str('%'));
%end;
```

%list_sasautos



```
data _list_sasautos
  ( keep = path member order type catalog );

  if substr(upcase("&sysscp"),1,3) = 'WIN' then
    s = '\';
  else
    s = '/';
```



%list_sasautos

- SASMSTORE is simple
- SASAUTOS is not
 - contains directory paths
 - filerefs to paths
 - including SASAUTOS itself
 - filerefs to catalogs
- require:
 - complete directory paths
 - fully qualified catalog names including paths

%list_sasautos



```
/* get SASAUTOS option value */
option = compress(getoption('sasautos'),' ( ) ');

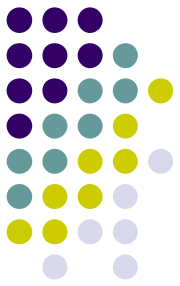
/* Grab space-delimited SASAUTOS definitions.  Entries that
   are file system paths will be captured and file references,
   SASAUTOS and catalog references will be expanded.  */

do I = 1 by 1 until ( scanq(option,I,' ') = ' ' );
    chunk = compress(scanq(option,I,' '),"'");

    /* if path delimiters found, pass straight in */
    if indexc(chunk,':/\') then do;
        sasautos = catx(' ', sasautos, chunk );
    end; else do;

/* no path delimiters found, expand entry to path level */
```

%list_sasautos



```
pathname = compress(pathname(chunk), ' ( ) ');
if pathname > ' ' then do;
  /* catalog libname starts with .. */
  if pathname =: '..' then do;
    cat_pathname = substr(pathname,3); /* skip over */

    /* Since we have catalog name, enclose with
       single quotes and insert in path */

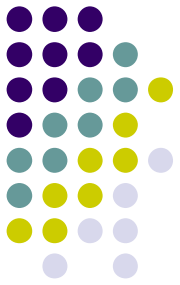
    path = pathname(scan(trim(cat_pathname),1,'.')) ||
            s||scan(trim(cat_pathname),2,'.')||'.SAS7BCAT';
    sasautos = catx(' ', sasautos, "' "||trim(path)||" ' ");
  end; else do;
```

%list_sasautos

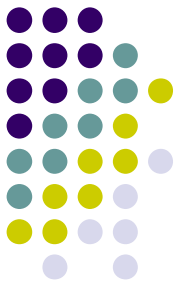


```
if left(pathname) =: "'" then
  /* sasautos */
  sasautos = catx(' ', sasautos,
                  compress(pathname, '()'));
else
  /* fileref */
  sasautos = catx(' ', sasautos,
                  "'" || trim(pathname) || "'");
```

%list_sasautos



```
/*  
    If we're going after WORK and COMPILED STORED macros,  
    add their paths at the front since that's the search  
    order SAS uses.  
*/  
  
sasautos = left("&work_lib &sasmstore " ||  
               translate(sasautos, "'", "''));
```



`%list_sasautos`

- now have all paths
 - fully qualified
 - including paths for catalogs
- for each path
 - find the .sas modules and SAS catalogs
 - catalogs will be examined later
- for each catalog
 - again, examined later

%list_sasautos

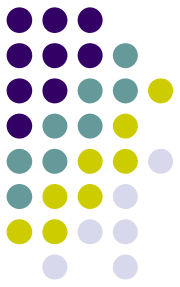


```
do i = 1 by 1 until ( scanq(sasautos,i,' ')=' ');  
  path = compress(scanq(sasautos,i,' '),"'");
```

```
/* Where the fileref pointed to a SAS catalog, we have  
specified the SAS7BCAT suffix to ensure we pick up only the  
specified catalog at this path. Since we're processing  
catalogs in more than one spot, we're using a common  
routine */
```

```
if scan(path,-1,'.') = 'SAS7BCAT' then do;  
  member = scan(path,-1,s);  
  path = substr(path,1,length(path)-length(member)-1);  
  link do_cat; /* catalog identified via fileref */
```

%list_sasautos



```
/* create a fileref pointing to dir */
problem = filename('dir',path);
if not(problem) then do;
    d = dopen('dir'); /* open the directory */
    if d then do; /* directory successfully open? */
        num = dnum(d); /* no. of files in directory */

        do _i = 1 to num; /* loop thru files in dir */
            /* get next filename in directory */
            member          = dread(d,_i);
```

%list_sasautos

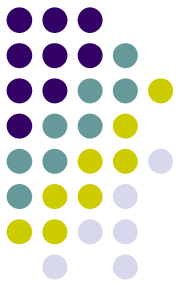


```
dir_test_file      =
  filename('dir_test', cats(path,s,member));

* cannot assign filename, iterate loop ;
if dir_test_file then continue;

* if open succeeds, it is a sub-directory, skip ;
dtf = dopen('dir_test');
if dtf then do;
  rc = dclose(dtf);
  continue;
end;
```

%list_sasautos



```
if upcase(scan(member,-1, '.')) = 'SAS7BCAT' then do;
```

```
    /* found a catalog in the directory */
```

```
    link do_cat;
```

```
end; else do;
```

```
    /* .sas member ? */
```

```
if upcase(scan(member,-1, '.')) = 'SAS' then do;
```

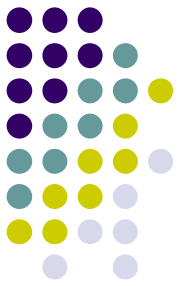
```
    order + 1;
```

```
    type = '.sas';
```

```
    output;
```

```
end;
```

%list_sasautos



```
do_cat:
  cat          + 1;
  order       + 1;

  call symput ('path' || put(cat, 3.-1), trim(path));

  * take off .sas7bcat;
  call symput ('cat' || put(cat, 3.-1),
              substr(member, 1, length(member) - 9));
  call symputx('order' || put(cat, 3.-1), order);

return;
```

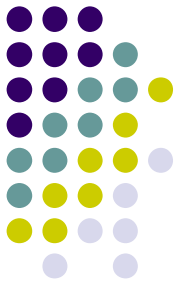
%list_sasautos



```
/* Now unpack the contents of the macro/source catalogs,  
working our way through the macro variables created in  
the previous step. Routing source/macro contents of  
the catalog to an OUT dataset. */
```

```
%do i = 1 %to &no_of_cats;  
  libname _c &&path&i;  
  
  proc catalog catalog = _c.&&cat&i;  
    contents out = _cat_contents  
      ( keep = memname name type  
        where = ( type in ( 'SOURCE', 'MACRO' ) ));  
  quit;
```

%list_sasautos



```
/* It's possible that duplicate paths are in the
   SASAUTOS definition, weed out dups here */
```

```
proc sort data = _list_sasautos;
  by path member order;
run;
```

```
/* Sorted by "order" so we'll keep the earliest one
   found in concatenation */
```

```
proc sort data = _list_sasautos nodupkey ;
  by path member;
run;
```

%list_sasautos

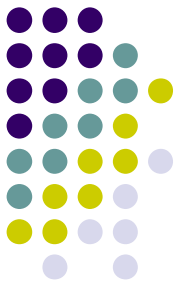


```
/* Sort the list by member name and the order  
   the macro was found */
```

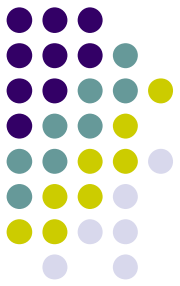
```
proc sort data = _list_sasautos;  
  by member order;  
run;
```

```
proc print data=_list_sasautos noobs ;  
  var order member path type catalog ;  
  format _character_ ;  
run;
```

%list_sasautos



	O/S Path	Macro / Filename	Catalog Name	Object Type	Resolution Order
1	C:\DOCUME~1\Hary\LOCALS~1\Temp\SAS Temporary Files_TD5748	ATTRN	sasmacr	macro	1
2	C:\DOCUME~1\Hary\LOCALS~1\Temp\SAS Temporary Files_TD5748	LIST_SASAUTOS	sasmacr	macro	1
3	c:\sas\913\SAS 9.1\eis\sasmacro	acatalog.sas		.sas	219
4	c:\sas\913\SAS 9.1\graph\sasmacro	addfeat.sas		.sas	275
5	c:\sas\913\SAS 9.1\qc\sasmacro	adxcc.sas		.sas	304
6	c:\sas\913\SAS 9.1\qc\sasmacro	adxff.sas		.sas	305
7	c:\sas\913\SAS 9.1\qc\sasmacro	adxgen.sas		.sas	306
8	c:\sas\913\SAS 9.1\qc\sasmacro	adxmix.sas		.sas	307



Wrap

- macro libraries are good!
 - reuse of quality code
- configuration files or autoexec.sas programs define SASAUTOS

- %list_sasautos
 - shows all that's available to you
 - identifies duplicate definitions

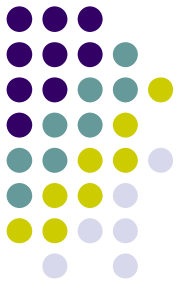


References

- Arthur L. Carpenter,
“Building and Using Macro Libraries” SUGI27
Proceedings, <http://www2.sas.com/proceedings/sugi27/p017-27.pdf>.
2008-07-14.
- Ronald Fehd,
“A SASAutos Companion: Reusing Macros” SUGI30
Proceedings,
<http://www2.sas.com/proceedings/sugi30/267-30.pdf>, 2008-07-14.

Acknowledgements

- Marje Fecht
- Laura House
- Dianne Piaskoski



Author

Harry Droogendyk – SAS Consultant

harry@stratia.ca

www.stratia.ca

