



Experts in SAS Business Intelligence

Get funky with SYSFUNC

Elena Muriel

Amadeus Software Limited





- Introduction
- SYSFUNC
 - Check the existence of a SAS data set
 - Obtain attributes from a SAS data set
 - Obtain attributes from variables
 - Check the existence of external files
 - How to read the contents of an external directory
 - Formatting macro variables
- QSYSFUNC
 - Formatting macro variables
- Conclusion



- What are %SYSFUNC and %QSYSFUNC?
 - Two powerful macro functions
 - Enable use of most of Base SAS software and SAS Component Language (SCL) functions
 - Formatting macro variables

- %SYSFUNC vs Data step
 - SCL functions are also available through the data step environment
 - Data step has faster processing time than macro
 - Data step is limited by step boundaries



- %SYSFUNC is a function that executes SAS functions
 - Can reference either SAS base functions, SCL functions or custom made functions created with PROC FCMP
- Syntax:

`%SYSFUNC(function-name(function-arguments), <format>)`



■ All data step functions except:

All variable information functions	ALLCOMB	ALLPERM
DIF	DIM	HBOUND
IORCMMSG	INPUT	LAG
LBOUND	LEXCOMB	LEXCOMBI
LEXPERK	LEXPERM	MISSING
PUT	RESOLVE	SYMGET

Note: use INPUTC, INPUTN, PUTC and PUTN for character or numeric formats



■ Some SAS functions:

ATTRC	ATTRN	CEXIST	CLOSE	CUROBS
DCLOSE	DINFO	DNUM	DOPEN	DOPTNAME
DOPTNUM	DREAD	DROPNOTE	DSNAME	EXIST
FAPPEDN	FCLOSE	FCOL	FDELETE	FETCH
FETCHOBS	FEXIST	FGET	FILEEXIST	FILEREF
FINFO	FNOTE	FOPEN	FOPTNMAE	OPTNUM
FPOINT	FPOS	FPUT	FREAD	FREWIND
FRLLEN	FSEP	FWRITE	GETOPTION	GETVARC
GETVARN	LIBNAME	LIBREF	MOPEN	NOTE
OPEN	PATHNAME	POINT	REWIND	SPEDES
SYSGET	SYSMSG	SYSRC	VARFMT	VARINFMT
VARLABEL	VARLEN	VARNAME	VARNUM	VARTYPE

Note: functions as per SAS Version 9.2 online documentation



- Check the existence of a SAS data set

```
%let exist=%sysfunc(exist(sashelp.shoes));
```

- The exist function will return:
 - 1 if the SAS data set exists
 - 0 if it does not exist
- To retrieve the value of the exist variable use macro notation (&exist)



■ Checking the existence of a data set and writing to the log window

```
%macro dataset_exist;  
%let exist=%sysfunc(exist(sashelp.shoes));  
%if &exist=0 %then %put Data set does not exist;  
%else %if &exist=1 %then %put Data set exists!;  
%mend;
```

```
%dataset_exist;
```

A screenshot of a SAS Log window titled "Log - (Untitled)". The window contains the following text:

```
1 %macro dataset_exist;  
2 %let exist=%sysfunc(exist(sashelp.shoes));  
3 %if &exist=0 %then %put Data set does not exist;  
4 %if &exist=1 %then %put Data set exists!;  
5 %mend;  
6  
7 %dataset_exist;  
Data set exists!
```

The log window has a blue title bar with standard window controls (minimize, maximize, close) on the right. The text is displayed in a monospaced font on a white background.



- Obtain data set attributes
 - number of observations
 - number of variables

```
%let did=%sysfunc(open(sashelp.shoes,i));
```

A screenshot of a SAS Log window titled "Log - (Untitled)". The window contains the following code and output:

```
8 %let did=%sysfunc(open(sashelp.shoes,i));
9 %let nobs=%sysfunc(attrn(&did,nobs));
10 %let nvars=%sysfunc(attrn(&did,nvars));
11 %let rc=%sysfunc(close(&did));
12 %put Data set SASUSER.SHOES contains &nobs observations and &nvars variables;
```

Data set SASUSER.SHOES contains 395 observations and 7 variables

■ nesting property of SYSFUNC

```
%let nobs=%sysfunc(attrn(%sysfunc(open(sashelp.shoes,i)),nobs));
```



■ Obtain attributes from data set variables:

```
%let di
20 %let did=%sysfunc(open(sashe lp.shoes));
21
%let va
22 %let rc=%sysfunc(fetchobs(&did,1));
23 %let var type=%sysfunc(var type(&did,5));
%put va
24 %put var type=&var type;
var type=N
25
%let va
26 %let var fmt=%sysfunc(var fmt(&did,5));
%put va
27 %put var fmt=&var fmt;
var fmt=DOLLAR12.
28
%let va
29 %let var infmt=%sysfunc(var infmt(&did,5));
30 %put var infmt=&var infmt;
%put va
var infmt=DOLLAR12.
31
32 %let vname=%sysfunc(varname(&did,5));
%let vn
33 %let vlabel=%sysfunc(var label(&did,5));
34 %put Variable &vname has a label of &vlabel;
%let vl
Variable Sales has a label of Total Sales
35
%put Va
36 %let rc=%sysfunc(close(&did));

%let rc
```

ement in
a set



- How to extract the position of a variable in a data set

```
%let var1=%sysfunc(varnum(&did,sales));  
%let vartype=%sysfunc(vartype(&did,&var1));
```

- Or using the nesting property

```
%let vartype=%sysfunc(vartype(&did,%sysfunc(varnum(&did,sales))));
```



- Check the existence of an external file

```
%let rc=%sysfunc(fileexist(c:\elena\data\10Feb2004.csv));
```

- And the *fileexist* function will return:

- 1 if the CSV file exists
- 0 if it doesn't exist



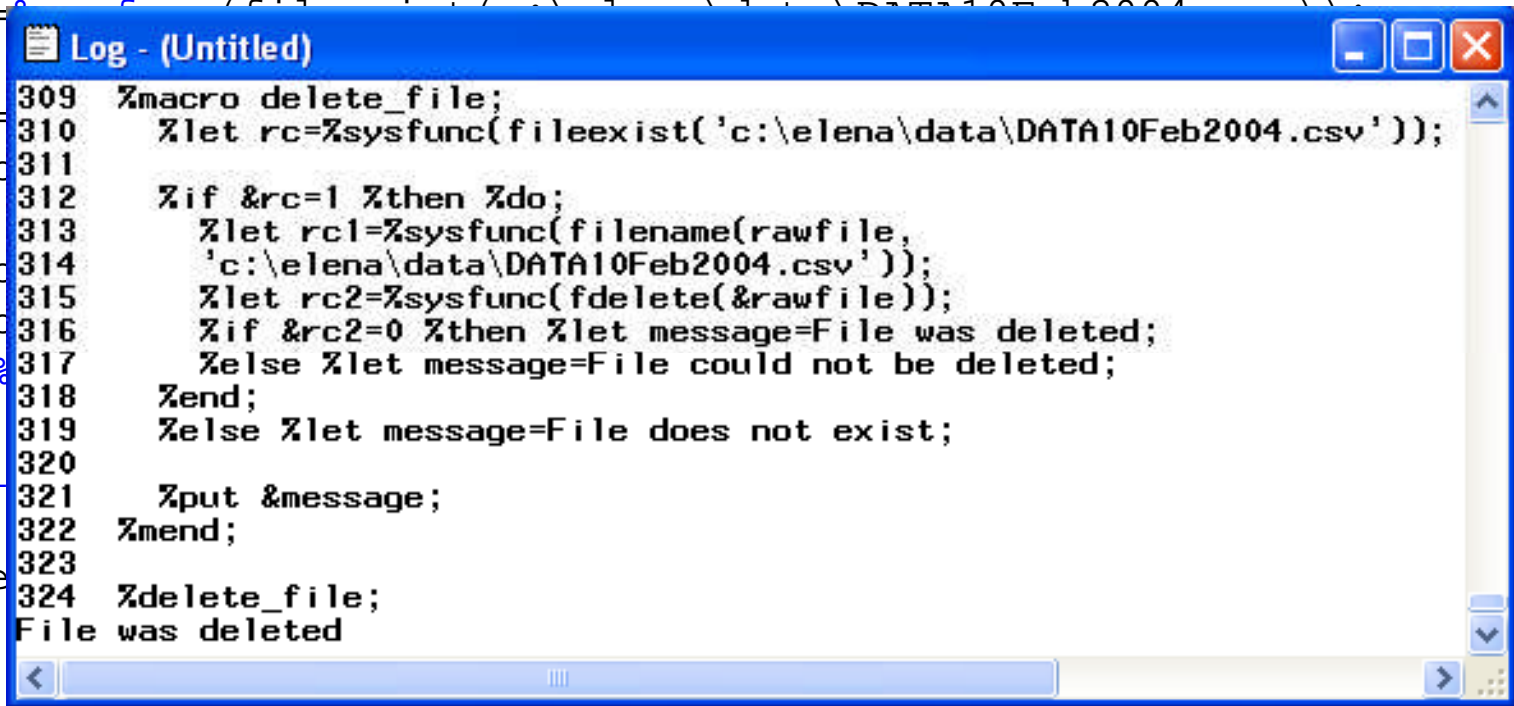
■ Checking the existence of a file before deleting

```
%macro delete_file;
%let rc=%sysfunc(fileexist('c:\elena\data\DATA10Feb2004.csv'));

%if &rc=1 %then %do;
  %let rc1=%sysfunc(filename(rawfile,
    'c:\elena\data\DATA10Feb2004.csv'));
  %let rc2=%sysfunc(fdelete(&rawfile));
  %if &rc2=0 %then %let message=File was deleted;
  %else %let message=File could not be deleted;
%end;
%else %let message=File does not exist;
%end;

%put &message;
%delete_file;
File was deleted

%delete_file;
```



Note: also consider operating system security



■ Import all CSV files found under a network directory

```
%macro read_files;
```

```
%let rc=%sysfun
```

```
%let did=%sysfu
```

```
%let dnum=%sysf
```

```
%do i=1 %to &dn
```

```
  %let name_fil
```

```
  %put file=&na
```

```
  %let csv=%sys
```

```
  %if &csv=csv
```

```
    %let name_f
```

```
    %put file=
```

```
  PROC IMPOR
```

```
  DBMS=C
```

```
  RUN;
```

```
  %end;
```

```
%end;
```

```
%mend;
```

```
%read_files;
```

```
Log - (Untitled)
260:44
NOTE: The infile 'C:\Elena\data\DATA14Feb2004.csv' is:
      File Name=C:\Elena\data\DATA14Feb2004.csv,
      RECFM=V,LRECL=32767
NOTE: 3 records were read from the infile 'C:\Elena\data\DATA14Feb2004.csv'.
      The minimum record length was 5.
      The maximum record length was 8.
NOTE: The data set WORK.DATA14FEB2004 has 3 observations and 3 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.01 seconds
3 rows created in WORK.DATA14FEB2004 from
C:\Elena\data\DATA14Feb2004.csv.
NOTE: WORK.DATA14FEB2004 was successfully created.
NOTE: PROCEDURE IMPORT used:
      real time          0.10 seconds
      cpu time           0.03 seconds
file=DATA15Feb2004.xls
```



- When formatting macro variables SYSFUNC can reduce the number of steps required
- Include the date at the start of a title using data step processing:

```
data _null_;  
    formatdate=put(date(),worddate.);  
    call symput('wdate',formatdate);  
run;  
title "&wdate Report title";
```





■ To SYSFUNC:

```
title "%sysfunc(date()),worddate.) Report title";
```



■ Include the date at the end of a title

```
title "Report title %sysfunc(date()),worddate.)";
```





- Remove the leading blank spaces using the left function

```
title "Report title %sysfunc(left(%sysfunc(date()),worddate.))";
```

- But if we execute the code we obtain:

```
ERROR: The function LEFT referenced by the %SYSFUNC or %QSYSFUNC macro function has too many arguments.
```

- Step by step processing:

```
title "Report title %sysfunc(left(%sysfunc(date()),worddate.))";
```

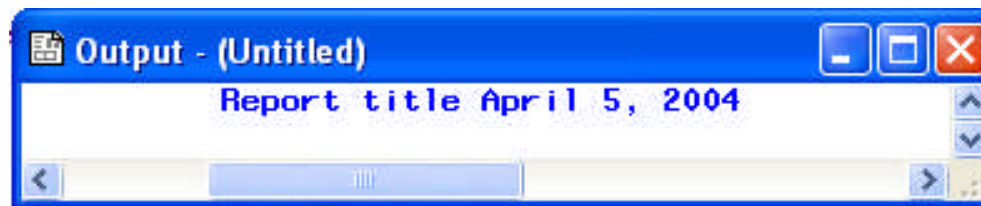
April 5, 2004

First argument of the left function Second argument



- Solution: Use QSYSFUNC quoting function
- Same as SYSFUNC but it can mask special characters, including & and %, and mnemonic operators

```
title "Report title  
      %sysfunc(left(%qsysfunc(date()),worddate.))";
```





- **SYSFUNC – Funky or not?**
 - + Alternative to data step processing with SCL functions
 - + SYSFUNC and SCL can increase the number of functions available to a SAS programmer in the macro environment
 - + Very powerful formatting tool
 - Processing time can be longer compared to data step



Thank you for your attention!



Elena Muriel

Amadeus Software Ltd.

Tel: +44 (0)1993 878287

E-Mail: elena.muriel@amadeus.co.uk

Web: www.amadeus.co.uk