



3 Comparing ~~6~~ Techniques To Do Data Driven Programming



Joris Derks, OCS Life Sciences, the Netherlands



Introduction

- What is data driven?
- Call a macro from values in a dataset
- Differences:
 - Complexity (timing, preferred language)
 - Maintainability (Extra statements needed, counter variable)
 - Features (Return values)
 - Flexibility (SAS Data Step / PROC SQL / Macro)
 - Availability (recently added)



Case

Example:

I have a dataset with study IDs in it and I want to count all the subjects in these studies.

work.overview

Study_name	Count_subjects
TEST_001	21365
TEST_002	20
TEST_003	2658

C:/TEST_001/

C:/TEST_002/

C:/TEST_003/

```
%macro count_nr_subjects(study_name);

/* Build path to the study */
%let path = C:/&study_name.;

libname study "&path";

/* Obtain the number of records and put it into macro variable
nr_subjects. */
%let nr_subjects = 0;
data _null_;
  set study.dm end = last;
  call symputx('nr_subjects',_n_);
run;

Libname study clear;

%mend count_nr_subjects;
```



PROC SQL with INTO clause

INTO :var1-varx

INTO :command separated by ‘ ‘

- PROC SQL only:
 - Summary functions available
 - Join / Order by
- Two methods:
 - Put values in macro variables
 - Create a command



PROC SQL with INTO clause

```
/* The PROC SQL with INTO clause to call  
/* Command is a macro variable which will  
PROC SQL NOPRINT;  
    SELECT '%COUNT_NR_SUBJECTS(' || strip(st  
    INTO :command separated by ' '  
    FROM work.overview  
;  
QUIT;
```

```
%COUNT_NR_SUBJECTS(TEST_001);  
%COUNT_NR_SUBJECTS(TEST_002);  
%COUNT_NR_SUBJECTS(TEST_003);
```

```
%put %superq(command);
```

```
/* EXECUTE the macros. */
```

```
&command
```

```
%macro count_nr_subjects(study_name);
```

```
/* Build path to the study */
```

```
%let path = C:/&study_name.;
```

```
libname study "&path";
```

```
/* Obtain the number of records and put it into macro variable  
nr_subjects. */
```

```
%let nr_subjects = 0;
```

```
data _null_;
```

```
    set study.dm end = last;
```

```
    call symputx('nr_subjects',_n_);
```

```
run;
```

```
Libname study clear;
```

```
/* Add result to the source dataset. */
```

```
data work.overview;
```

```
    length count_nr_subjects 8;
```

```
    set work.overview;
```

```
    if study_name = "&study_name" then
```

```
        count_nr_subjects = &nr_subjects;
```

```
run;
```

```
%mend count_nr_subjects;
```



PROC FCMP with run_macro

run_macro('<macro>',<variable1>,<variable2>)

- SAS Data Step & proc SQL
- Create a custom function using PROC FCMP, this function calls macro using RUN_MACRO
- Learning curve
- Cannot fill macro parameters (creates local macro parameters)
- Return values possible
- Executed during the iteration of the SAS Data Step / proc SQL



PROC FCMP with run_m

```
/* In this fcmp procedure a custom function is defined
** which will set the value of the parameter nr_subjects
** create empty macro variable nr_subjects, and
** when the macro count_nr_subjects is finished
```

```
proc fcmp outlib = work.func.functions;
  function count_nr_subjects(study $);
    rc = run_macro('count_nr_subjects', study, rc);
    return(nr_subjects);
  endsub;
run;
```

```
/* Add this to the cmplib option, otherwise the macro is not found
options cmplib=work.func;
```

```
/* Loop over all studies in work.overview and
** with study name as parameter.
```

```
** If there is negative value in the count_subjects
```

```
data work.overview_completed;
  length count_subjects 8;
  set work.overview;
  count_subjects = count_nr_subjects(study_name);
run;
```

```
%macro count_nr_subjects;
```

```
%let study_name = %sysfunc(dequote(&study));
```

```
/* Build path to the study */
```

```
%let path = C:/&study_name.;
```

```
libname study "&path";
```

```
/* Obtain the number of records and put it into macro variable
nr_subjects. */
```

```
%let nr_subjects = 0;
```

```
data _null_;
```

```
set study.dm end = last;
```

```
call symputx('nr_subjects', _n_);
```

```
run;
```

```
Libname study clear;
```

```
%mend count_nr_subjects;
```



dosubl('%mymacro(<variable1>);')

- SAS Data Step & proc SQL
- Return values possible
- Can fill macro parameter values
- Executed during the iteration of the SAS Data Step / proc SQL
- Available SAS9.3M2



DOSUBL

```
/* Loop over all studies in work.overview and
** call the macro count_nr_subjects with stud
** Then use symget to get the return value.
** If there is negative value in the count_su
data work.overview_completed(drop = rc);
  length count_subjects 8;
  set work.overview;
  rc = dosubl('%count_nr_subjects('||strip(st
  count_subjects = symgetn('nr_subjects');
  if count_subjects < 0 then put 'ER' 'ROR: s
                                'counting su
run;
```

```
%macro count_nr_subjects(study_name);
```

```
/* Build path to the study */
%let path = C:/&study_name.;
```

```
libname study "&path";
```

```
/* Obtain the number of records and put it into macro variable
nr_subjects. */
%let nr_subjects = 0;
data _null_;
  set study.dm end = last;
  call symputx('nr_subjects',_n_);
run;
```

```
Libname study clear;
```

```
%mend count_nr_subjects;
```

No modification!

Closing



- Three techniques presented: proc sql with an into clause, proc fcmp with run_macro, dosubl
- Three techniques not presented: fetchobs, call symputx, call execute
- Differences:
 - Complexity
 - Maintainability
 - Features
 - Flexibility
 - Availability
- Some situations, one technique more suitable
- Programmers Preference



3 Comparing ~~6~~ Techniques To Do Data Driven Programming



Joris Derks, OCS Life Sciences, the Netherlands