PharmaSUG 2026 - Paper TT-130

My DIY Swiss Army Knife of SAS® Procedures
A Macro Approach of Forging with My Favorite PROCs

Jason Su, Daiichi Sankyo, Inc.

ABSTRACT

Here I take advantage of SAS® macro facility and forge these following four (4) extremely popular procedures into one (1) Swiss-Army knife (SAK)-styled macro %pfs (the acronyms): PROC PRINT, PROC CONTENTS (not in the acronym), PROC FREQ, and PROC SQL. Controlled by a mode-switch parameter (MODE), the macro can fashion out any one of the 4 procedures in a succinct version supporting popular options in various procedures, such as OBS, FIRSTOBS, WHERE, SHORT, VAR, etc. The new macro has the capacity to carry out my most-frequent jobs from such procedures, such as selectively printing some records from a dataset, displaying its data structure, quickly deriving certain variable frequency, counting certain variables, etc. Based on the spirit, fellow programmers are encouraged to create their own version of the macro %pfs. Upon being called with different modes, the SAK macro can perform any of the procedures, and immediately release the programmers from much of the repetitive syntax-typing work. Additionally, the functionalities of the tool can be expanded and many innovative capacities can be added such as performing fuzzy searching on ID variables, automatically saving the counting results into a macro variable for later use, so that macro can become smart and surprisingly powerful.

INTRODUCTION

SAS programming has always been mostly a pleasant experience for me but I got to admit: Sometimes there can be lots of syntax typing in my daily work. For example, recently I did a lot of data compare with a fellow programmer and found myself pulling these four (4) procedures continuously: PROC PRINT, PROC CONTENTS, PROC FREQ, and PROC SQL. The boredom was insensibly reaching hundreds per day. Such kind of repetition makes the regular copy-&-paste-&-modification overwhelming, almost endangering my programming delightfulness before the macro facility comes to rescue.

The new Swiss-Army knife (SAK)-styled macro %pfs is acronym-named for PROC PRINT, (PROC CONTENTS), PROC FREQ, and PROC SQL, and have the capacity to represent a concise version of any one of the 4 procedure, supporting the following functionalities:

- Printing some records from a dataset.
- Displaying its data structure.
- Calculating certain variable frequency.
- Selectively counting total records, etc.

THE MACRO PARAMETERS: A BRIEF DESCRIPTION

The complete program of the macro is included at the end of the paper. For ensure the minimal typing, we set up eight (8) parameters, all of which are optionally. Below is the layout of the macro:

- The “ds” is the only positional parameter used to indicate the input dataset name for processing. The default is the latest dataset created.
- The “mode” is used to indicate the procedure to invoke.
- The “firstobs” is a data set option & specifies the starting point record number for processing.
- The “obs” is a data set option & specifies the end point record number for processing. If FIRSTOBS is larger than OBS values, it specifies the number of records to report, starting from the starting point

record number.

- The “where” is used to specify specific conditions to use to select observations from a SAS data set.
- The “format” is used to associate formats with variables.
- The “var” is used to specify the list of variables for processing, space-delimited.
- The “keyword” is used to reduce contents output or indicate unique values, so far only supporting “short” & “distinct”.
- The “print_by” is as an example of expanded functionalities, being a BY-GROUP variable list for displaying the records after inherent sorting, for MODE=P only.

PROCEDURE-SWITCHING: MODE

The “mode” is the central parameter to drive the switching between different procedures and the rest are naturally designed around it. In this version, it can take on the following case-insensitive values as the following:

- P or print, for PROC PRINT. It is the default value. In other words, if none of the parameters are invoked like %pfs(), it will print the first 20 records of the last dataset created. It is designed to accommodate the fact that PROC PRINT might be the most often used procedure for me.
- C or contents, for PROC CONTENTS. There is one KEYWORD value SHORT applicable in this mode. Unapplicable parameters for PROC CONTENTS are to be simply ignored, such as OBS.
- F or freq, for PROC FREQ. Only in this mode, VAR is mandatory and there should be only one (1) var supported for the easy application of FORMAT parameter. The design is to reflect the fact that it is what is usually encountered in my daily work. Unapplicable parameters are to be simply ignored, such as OBS.
- S or sql, for PROC SQL. It is only used for counting VAR values. In this version there should be only one (1) var supported. If VAR is left null, all variables in the dataset are used and total records are counted. If desired, KEYWORD value “DISTINCT” can be provided in this mode. Unapplicable parameters are to be simply ignored, such as PRINT_BY.
- SV, is an extension mode from the above S mode, e.g. it saves the results into a global var &PFS_COUNT upon successful completion of the counting of a variable, such as PATEINTID, or all records if the VAR parameter unpopulated. By default, the result will also be displayed, but can be turned off with a value like “sv_np”, of which “np” means not printing. Either way, in the LOG, the value of the newly created macro var &PFS_COUNT is also displayed. Users can directly use this value as a BIGN or anything desirable.

CALL THE MACRO

Calling the macro is extremely simple, which is a goal of the design. Table 1 illustrates some sample calls of the macro and the codes typically created and executed.

Sample Call	Sample Codes Executed
%**pfs**()	proc print obs=20; run;
%**pfs**(**firstobs**=50,**obs**=5,**var**=patientid startdt)	proc print firstobs=50 obs=54; var patientid startdt; run;
%**pfs**(_tempds,**obs**=10,**where**=startdt)	proc print data=_tempds obs=10; where startdt is not missing; run;
%**pfs**(**mode**=contents)	proc contents; run;
%**pfs**(prod.axattr,**mode**=c,**keyword**=short)	proc contents data=prod.axattr short; run;
%**pfs**(**mode**=f,**var**=sex)	proc freq; tables sex; run;
%**pfs**(prod.adsl,**mode**=f,**var**=sex,**format**=$sex, ,**where**= state is missing)	proc freq data=prod.adsl; tables sex; format sex $sex.; where state is missing; run;
%**pfs**(**mode**=s)	proc sql; select count(*) from NAME_OF_LAST_DS; run;
%**pfs**(prod.axattr,**mode**=s,**where**=^missing(inc5))	proc sql; select count(*) from prod.axattr where ^missing(inc5); run;
%**pfs**(prod.axattr,**mode**=s,**where**=^missing(inc5) ,**var**=subjid,**keyword**=distinct)	proc sql; select count(distinct subjid) from prod.axattr where ^missing(inc5); run;

Sample Call	Sample Codes Executed
%**pfs**(**mode**=sv_np)	proc sql noprint; select count(*) into :pfs_count from NAME_OF_LAST_DS; run;

Table 1 Sample PFS Calls and Codes Created & Executed

THE EXPANDED FUNCTIONALITY OF THE MACRO

Besides cutting a lot of syntax typing and saving the SAS users from repetitive & tiring work, many innovative functionalities can be added, and I latterly have added such as follows,

SMART SEARCHING ON ID VARIABLES

In my daily work, I might use different databases and the patient id variables usually are differently named, such as PID, PATIENTID, SUBJID, USUBJID, CHAI_PATIENT_ID, etc. When I switch between these datasets and do the patient-id-based searching, I often forget these trivial differences and therefore bumped into the famous variable-is-not-on-file programming errors like this:

ERROR: Variable patientid is not on file *****.

The dataset does not contain this particular variable, i.e. patientid in this case. Although the debug for this error is quite simple for such instances, the new SAK tool can perfectly deal with the issue even without the user's help. Now it can perform searching on the ID variables in the example as follows,

```
data some;
    set sashelp.class;

    rename name=pid;
run;

%pfs(where=patientid='John')
```

Behind the curtain, our SAK tool first checks the id variables and if needed, automatically replaces it with the real id variables in the current data. Additionally, the tool gives a nudge to the user by leaving an obnoxious WARNING note in the LOG to make users aware of this situation.

```
WARNING: Real VAR name is used to replace the id var in WHERE parameter:
PID.
```

In this way users can do ID-based searching smoothly in the worry-free mood and are nicely relieved to some degree from the attention demand as in this.

AUTOMATICALLY SAVING THE COUNTING RESULTS INTO A MACRO VARIABLE FOR LATER USE

BIGN is a common variable to be used as the denominator in the percentage-related calculation, and often its value is saved into a macro variable so that it can be used everywhere. For example, with the previous SOME dataset created, it is quickly created via a standalone PROC-SQL step like this

```
proc sql noprint;
    select count(pid) into :total
    from some
    ;
quit;
```

Now with this tool, the likewise macro variable &PFS_COUNT can be created simply by this,

```
%pfs(some,var=pid,mode=sv)
```

Or, if the result printing is to be suppressed,

```
%pfs(some,var=pid,mode=sv_np)
```

, where the appendix “np” means no-print. Either way, a NOTE will show in the LOG to let users be aware of this situation like this,

```
NOTE: This mvar is successfully created: PFS_COUNT=19
```

In this way, the SAK tool can help users be less tangled in the routine works and divert their creativity elsewhere.

In all, the functionality expansion is literally without any limit. Users are encouraged to fully customize the codes and tailor it for their own specific situations.

THE MACRO CODES

Below are all the sas codes for the macro pfs.

```
%macro pfs(ds
            ,mode      = print
            ,firstobs  =
            ,obs       = 20
            ,where     =
            ,format    =
            ,var       =
            ,keyword   =
            ,print_by  =
            ,freq_out  =)/minoperator;

        %local msgtype first_letter realvar orig_obs _lib _mem real_lib
this_idvar;

        %let syscc=0;

        %if ^%length(&ds.) %then %let ds=%sysfunc(getoption(_last_));

        %if %nrbquote(&ds.)= ? %then
            %do;
                %syntax:
                   %put;
                   %put &msgtype: &sysmacroname. documentation:;
                   %put &msgtype- Purpose: Quickly switch to one of the 4
procedures: ;
                   %put &msgtype-          print, contents, freq, sql, and
dictionary w/sql;
                   %put;
                   %put &msgtype- Syntax:
%nrstr(%%)&sysmacroname.(<ds><,mode=><,firstobs=><,obs=><,where=><,format=>
<,var=><,keyword=><,print_by=><,freq_out=>);
                   %put;
                   %put &msgtype- ds:       (Optional) Source dataset to be
processed. Default being the last dataset created.;
                   %put &msgtype- mode:     (Optional) Which procedure
invoked, values: p(rint)/c(ontents)/f(req)/s(sql)/sv. Default being Print.;
                   %put ;
                   %put &msgtype- firstobs: (Optional) First obs to print,
only for MODE=P ;
```

```
                    %put &msgtype- obs:      (Optional) OBS to print, only
for MODE=P/S. Value changed to FIRSTOBS+OBS ;
                    %put &msgtype-                        if the value is
smaller than FIRSTOBS. Default being 20.;
                    %put &msgtype- where:    (Optional) select which records
for processing. Required for MODE=D.;
                    %put ;
                    %put &msgtype- format:   (Optional) space-separated
pairs of variable and its format for processing, only for MODE=P/F without
print_by;
                    %put &msgtype- var:      (Optional) select variables for
processing, only for MODE=P/F/S. Required for MODE=F.;
                    %put &msgtype-                        Must only be 1 var
for MODE=S;
                    %put &msgtype- keyword:  (Optional) keyword, values
being SHORT/DISTINCT for MODE=C/S/P, respectively;
                    %put &msgtype- print_by: (Optional) var list for sorting
& displaying the records, for MODE=P only;

                    %put ;
                    %put &msgtype- Use ? to print documentation to the SAS
log.;
                    %put;
                    %return;
            %end;

        %let mode=%lowcase(&mode.);
        %let ds=%lowcase(&ds.);

        %let _lib=%sysfunc(ifc(%index(&ds.,.),%scan(&ds.,1,.),none));
        %let _mem=%sysfunc(ifc(%index(&ds.,.),%scan(&ds.,2,.),&ds.));
        %if %index(&mode.,sv) %then
            %do;
                %global pfs_count;
                %let pfs_count=0;
            %end;

        %if &_lib.=none %then
            %do;
                proc sql noprint;
                    select libname into :real_lib separated by ' '
                    from dictionary.tables
                    where libname %sysfunc(ifc(&_lib.=none,in ('USER'
'WORK'),=upcase("&_lib.")))
                                & memname=upcase("&_mem.")
                    ;
                quit;

                %if &real_lib.=WORK %then %let ds=work.&ds.;
            %end;

        %if %index(&ds.,adsl) %then %let
var=%sysfunc(tranwrd(&var.,patientid,subjid));

        %let msgtype=NOTE;

        %if &syscc.>4 %then %return;
```

```
        %let first_letter=%substr(&mode.,1,1);

        %if %length(&first_letter.) & ^(&first_letter. in p c f d s) %then
            %do;
                %put WAR%str(NING:) Wrong mode values.;
                %goto syntax;
            %end;

        %*CHECKING &DS.;
        %let special_char_pos=%sysfunc(notalnum(&ds.));
        %if &special_char_pos. %then %if
^%sysfunc(anypunct(%substr(&ds.,&special_char_pos.,1))) %then
            %do;
                %put WARNING: The DS value is not a dataset: %upcase(&ds.);
                %goto syntax;
            %end;

        %*AVOIDING DICTIONARY TABLES NOT EXISTING ISSUES;
        %if &first_letter.=d %then
                %do;
                    %let ds=dictionary.tables;
                    %put WAR%str(NING:) It is assumed DICTIONARY.;
                %end;
            %else %if ^%index(&ds.,dictionary) & (^%sysfunc(exist(&ds.)) &
^%sysfunc(exist(&ds.,view))
                        %sysfunc(ifc(&_lib.=none, &
^%sysfunc(exist(work.&ds.)) & ^%sysfunc(exist(work.&ds.,view)),))
                        | ^%length(&ds.)
                    ) %then
                %do;
                    %put ERR%str(OR:) The dataset does not exist:
%upcase(&ds.);
                    %abort;
                %end;
        %*CHECKING PARAMETER CONFLICT;
        %if %length(&freq_out.) & &first_letter.^=f %then %put WARNING:
This parameter only applies to MODE=Freq: FREQ_OUT.;
        %if %length(&print_by.) & %length(&firstobs.) %then %put WARNING:
This parameter only applies to MODE=Print w/o PRINT_BY: FIRSTOBS.;

        %if &syscc.>4 %then %return;

        %*THIS CHECKING ID VARS IN WHERE, THEN FINDING THE CORRECT IDVAR,;
        %let pos_idvar=%sysfunc(prxmatch(/(p|subj).*id\b/i,&where.));
        %if &pos_idvar. %then
            %do;
                %*MAKING SURE IT EXIST;
                proc sql noprint;
                    select name into :realvar trimmed
                    from dictionary.columns
                    where libname %sysfunc(ifc(&_lib.=none,in ('USER'
'WORK'),=upcase("&_lib.")))
                            & memname=upcase("&_mem.")
                            & prxmatch('/(p|subj).*id\b/i',name)
                    ;
                quit;
```

```
                %if ^%index(%qupcase(&where.),%upcase(&realvar.)) %then
                    %do;
                        %let
this_idvar=%scan(%qsubstr(&where.,1,&pos_idvar.-1),-1)id;
                        %let
where=%sysfunc(prxchange(s/\b&this_idvar.\b/&realvar./i,1,&where.));
                        %put WARNING: Real VAR name is used to replace the
id var in WHERE parameter: %upcase(&realvar.).;
                    %end;
            %end;

        %if &syscc.>4 %then %return;

        %if &first_letter.=c %then
                %do;
                    %*THIS IS FOR: PROC CONTENTS;
                    proc contents data=&ds. &keyword.;
                        title "******** PROC CONTENTS OF DATASET:
%UPCASE(&DS.) ********";
                %end;
            %else %if &first_letter.=f %then
                %do;
                    %*THIS IS FOR: PROC FREQ;
                    %if ^%length(&var.) %then
                        %do;
                            %put WARNING: The parameter is required for
this mode: VAR.;
                            %goto syntax;
                        %end;

                    proc freq data=&ds. ;
                        title "*** PROC FREQ OF DATASET: %UPCASE(&DS.) **";
                        table &var. %sysfunc(ifc(%length(&freq_out.),/
out=&freq_out.,));

                        %if %length(%bquote(&format.)) %then
                            %do;
                                format &format.;
                            %end;

                        %if %superq(where)^= %then
                            %do;
                                title2 %tslit(WHERE: %UPCASE(&where.));
                                where &where.;
                            %end;
                %end;
            %else %if &first_letter.=p %then
                %do;
                    %*THIS IS FOR: PROC PRINT;
                    %if %bquote(&firstobs.)>=&obs. & &obs.>0 %then
                        %do;
                            %let orig_obs   = &obs.;
                            %let obs        = %eval(&firstobs.+&obs.-1);
                        %end;
```

```
                    %if %index(&ds.,dictionary) |
%length(%bquote(&print_by.)) | %index(%lowcase(&keyword.),distinct) %then
                        %do;
                            title "*** PROC SQL &obs. RECORDS OF DATASET:
%UPCASE(&DS.) *****";
                            %if %superq(where)^= %then %str(title2
%tslit(WHERE: %UPCASE(&where.)) ;);

                            proc sql
%sysfunc(ifc(%length(&obs.),inobs=&obs.
%sysfunc(ifc(%length(&firstobs.),outobs=&orig_obs.,)),));
                                select
%sysfunc(ifc(%length(&keyword.),&keyword.,))

%sysfunc(ifc(%length(&var.),%qsysfunc(tranwrd(%cmpres(&var.),%str(
),%str(,))),*))
                                from &ds.
                                %if %length(%bquote(&where.)) |
%length(&firstobs.) %then where
                                %if %length(%bquote(&where.)) %then
&where. %sysfunc(ifc(%length(&firstobs.),& monotonic() >= &firstobs.,));
                                    %else
%sysfunc(ifc(%length(&firstobs.),monotonic() >= &firstobs.,));
                                %if %length(%bquote(&print_by.)) %then
order by %sysfunc(transtrn(%cmpres(&print_by.),%str( ),%str(,)));
                                    ;
                            quit;
                            title;

                            %return;
                        %end;

                    proc print
data=&ds.(%sysfunc(ifc(%length(&firstobs.),firstobs=&firstobs.,))
obs=&obs.);
                        title "******** PROC PRINT &obs. RECORDS OF
DATASET: %UPCASE(&DS.)
%sysfunc(ifc(%length(&firstobs.),firstobs=&firstobs.,))********";
                        %if %superq(where)^= %then %str(title2
%tslit(WHERE: %UPCASE(&where.)) ;);

                        %if %length(%bquote(&format.)) %then
                            %do;
                                format &format.;
                            %end;

                        %if %superq(where)^= %then
                            %do;
                                title2 %tslit(WHERE: %UPCASE(&where.)) ;
                                where &where.;
                            %end;
                        %if %length(&var.) %then %str(var &var.;);
                %end;
            %else %if &first_letter.=s %then
                %do;
                    %*THIS IS FOR QUICK COUNTING: SQL;
                    %if ^%length(&var.) %then %let var=%str(*);
```

```
                        %else %if %sysfunc(countw(&var.))^=1 %then
                                %do;
                                    %put ERR%str(OR): ONLY 1 VAR FOR pro
sql COUNTING. &=VAR.;
                                    %goto syntax;
                                %end;

                    %*proc sql does not support colon;
                    %if ^%index(&where.,=:) %then
                            %do;
                                proc sql
%sysfunc(ifc(%index(&mode.,np),noprint,));
                                title "******** PROC SQL COUNT IN
DATASET: %UPCASE(&DS.) ********";
                                %if %superq(where)^= %then %str(title2
%tslit(WHERE: %UPCASE(&where.)) ;);
                                select count(&keyword. &var.)
%sysfunc(ifc(%index(&mode.,sv),into :pfs_count trimmed,))
                                from &ds.
                                %if %length(%bquote(&where.)) %then
%str(where &where.;);
                                    ;
                                quit;
                            %end;
                        %else
                            %do;
                                %deltable(tables=_temp);

                                data _temp;
                                    do until (eof);
                                        set &ds. end=eof;

                                        %if %length(%bquote(&where.)) %then
%str(where &where.;);
                                        total+1;
                                    end;

                                    call symputx('pfs_total',total);
                                    keep total;
                                run;

                                title "*** COUNTING TOTAL OBS IN DATASET:
%UPCASE(&DS.) ********";
                                %if %superq(where)^= %then %str(title2
%tslit(WHERE: %UPCASE(&where.)) ;);

                                %if &syscc.<5 & ^%index(&mode.,np)  %then
%do; proc print data=_temp noobs; run; %end;

                                title;

                            %end;

                    %if &syscc.<5 & %index(&mode.,sv) %then %put NOTE: This
mvar is successfully created: &=pfs_count.;
                %end;
```

```
            %else %if &first_letter.=d %then
                %do;
                    %if %superq(where)= %then
                    %do;
                        %put ERR%str(OR:) Aborted due to parameter
WHERE is empty;
                        %goto syntax;
                    %end;

                    %*THIS IS FOR: DIRECTORY FILE DISPLAY;
                    proc sql;
                        title "**** FILES UNDER DIR: %qUPCASE(&where.) *";
                        select memname
                        from &ds.
                        where &where.
                        ;
                    quit;
                %end;

            %if ^(&first_letter. in s d) %then %str(run;);
            title;
    %mend pfs;
```

CONCLUSION

The macro is not intended to unify all functionalities of these invoked procedures under one jumbo umbrella. Rather, it is only for quickly calling the most popular procedures with minimal key stroke. For that reason, it is a great time-savor for the SAS programmers. However, for other users whose daily tasks might be different, the functions supported by this version of the tool might be rather limited. For that, each user is encouraged to modify the current implementation and personalize it to everyone's own needs. For example, for F/freq MODE, multiple vars can be supported but accordingly FORMAT syntax needs to be thought well to avoid any confusion. Either way, the paper provides the fellow programmers a grain of salt so they can be slightly relieved from much of the repetitive syntax-typing work, and with a little expanded functionality, the tool can become creatively smart and surprisingly powerful.

ACKNOWLEDGMENTS

I appreciate my managers Antonio Lovatin, Yongin Kim & Ajay Gupta for their valuable encouragement and support. This paper would not be possible if in the absence of their tremendous support & encouragement.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Jason Su
Daiichi Sankyo Incorporation
919-260-5649
Jason.su@daiichisankyo.com