Alberta

Freedom To Create. Spirit To Achieve.

Iterating Data Using SAS

Alberta Health

Health Analytics Branch

Sophie Zhang

Government of Alberta

1.1 OVERVIEW

Why do we iterate?

REASON in (‘to simplify …’, ‘to satisfy …’, ‘to challenge …’, ‘to easy up …’, ‘to speed up …’ <, >)

1.2 OVERVIEW

How do I iterate?

METHOD in (Mtd_1, Mtd_2, <,>)

It depends …

1.3 OVERVIEW

Outlines

- Where does iteration take place in my codes?
- My ways to get value lists that are to be iterated through
- Options to me to conduct iteration
- Hard coding – once in a while
- Bonus

2.1 Where do I iterate?

Dataset variables vs. Macro variables?

Timing – dataset variables can be taken care of during Data step, while Macro variables have to be assigned before that

E.g.

```
data tbl_02;
set tbl_01;
array dx $9. DX_1-DX_25;
flag=0;
do i=1 to 25;
if missing(dx(i))=0 then do;
                if substr(compbl(upcase(dx(i))),1,3) in (‘F29') then flag=flag+1; end;
end;
if flag>0 then output;
run;
```

2.2 Where do I iterate?

Dataset variables vs. Macro variables?

Timing – dataset variables can be taken care of during Data step, while Macro variables have to be assigned before that

E.g.

```
proc sql;
select max(year(pers_dob)), min(year(pers_dob)) into :max_dob, :min_dob
from mst_tbl; quit;

%macro ChkName;
%do i=&min_dob %to &max_dob %by 10;
        proc sql; create table temp_01 as select distinct * from mst_tbl
        where &i<= year(pers_dob) < %eval(&i+10)
                and missing(pers_dob) = 0; quit;
        ...
```

2.3 Where do I iterate?

Dataset variables vs. Macro variables?

Clarifying -- Macros dealing with passed on or internally generated Macro variables

E.g.

```
%Macro Pnt(f_year=);
…
%Mend Pnt;
```

2.4 Where do I iterate?

Dataset variables vs. Macro variables?

Clarifying -- Macros dealing with passed on or internally generated Macro variables

E.g.

```
%Macro Pnt;
        %do i=  %to ;
        …
        %end;
%Mend Pnt;
```

3.1 Where do I get the list?

Generalized values vs. not?

Difference – generalized values can be created in SAS by following certain logic

E.g.

```
%macro SumUp(start, end);
        %do i = %eval(&start.) %to %eval(&end.);
        table_&i._%eval(&i.+1)
        %end;
%mend SumUp;

data project.clm_cmn; set %SumUp(2008, 2011); run;
```

MPRINT(SUMUP_CLM): **table_2008_2009**
table_2009_2010
table_2010_2011

3.2 Where do I get the list?

Generalized values vs. not?

Difference – non-generalized values can be input from outside

E.g.

```
proc import out= vbl_name
datafile = “c:\Data for Annual Report.xlsx"
dbms = excelcs replace;
sheet = "sheet1$";
run;
```

3.3 Where do I get the list?

Generalized values vs. not?

Difference – or, non-generalized values can be queried from the SASHELP library

E.g.

```
proc sql;
create table vbl_list_in_tbl_1 as
select          name
,               type
,               length
,               format
from sashelp.vcolumn
where upcase(libname) in ('PROD')
and upcase(memname) in  ('TBL_1')
;
quit;
```

4.1 How do I use the list?

Distinguish the members?

– no, then use it as a table

E.g.

```
proc sql;
create table comb as
select *
from (select * from sub_table a join list b on a.sid=b.sid) c
join (select * from mst_table
                where m_id in (select distinct m_id from list)) d
on c.m_id = d.m_id
;
quit;
```

4.2 How do I use the list?

Distinguish the members?

– no, then use it as a table

E.g.

```
data reg_new (drop=rc);
attrib name_ length=$20
                var length=$500
                n_v length=3;
if _n_=1 then do;
                declare hash name_v(dataset: 'list');
                                name_v.definekey('name_');
                                name_v.definedata('name_','var','n_v');
                                name_v.definedone();
                                call missing(name_, var, n_v);
end;
set reg;
rc=name_v.find(key: scan(pers_name, 1));
run;
```

4.3 How do I use the list?

Distinguish the members?

– yes, then iterate among them

E.g.

```
proc sql;
select distinct quote(id), count(*) into :vbl_list separated by ' ', :vbl_num
from v_list; quit;

%macro Gen(f_year);
%do i=1 %to &vbl_num;
        %let _vbl_=%scan(&vbl_list.,&i,' ');
        ...
```

4.4 How do I use the list?

Distinguish the members?

– yes, then rotate among them

E.g.

```
proc sql; create table lst_table as
                select distinct quote(strip(nm_t)) as nm_tbl, nm_t
from lst_vbl; quit;

%macro lst_var(tbl_n, nm_t);
                proc sql; select distinct nm_v into :v_&nm_t separated by ' '
                from lst_vbl where nm_t = &tbl_n; quit;
%mend lst_var;

data _null_; set lst_table;
                call execute('%lst_var('||nm_tbl||','||nm_t||');');
run;

v_table_1=id vbl_1 vbl_2
```

4.4 How do I use the list?

Distinguish the members?

– yes, then iterate among them

E.g.

```
[cont'd]
proc sql;
        create table result_table_1 as
        select distinct &v_table_1
        from table_1
        ;
quit;
```

5.1 Hard code a list?

Yes

Concerns – easy to maintain?

E.g.

```
%let year=2014;
%let year=2015;


%pnt(table_1);
%pnt(table_2);
```

5.2 Hard code a list?

Yes

Concerns – easy to maintain?

E.g.

```
data _null_ ;
do i = 2006 to 2010 by 2, 2014, 2012 to 2005 by -3 while( i > 2005 ) ;
put i= ;
end ;
run ;
```

```
i=2006
i=2008
i=2010
i=2014
i=2012
i=2009
i=2006
```

5.3 Hard code a list?

Yes

Concerns – easy to maintain?

E.g.

```
%macro SendToExcel(source);
PROC EXPORT DATA= project.&source
   DBMS=EXCELCS REPLACE OUTFILE= “c:\example.xls";
   SHEET="&source"; RUN;
%mend SendToExcel;

data _null_;
input src $10.;
call execute('%sendtoexcel('||src||');');
cards;
table_1
table_2
;
run;
```

6.1 Tips?

Which way is better?

Concerns – in terms of coding time or of machine time?

E.g.

```
proc sql;
create table temp_1 as
select distinct &v_list
from temp_0;
quit;
```

vs.

```
proc sql;
create table temp_1 as
select distinct *
from temp_0 (keep=&v_list);
quit;
```

6.2 Tips?

Which way is better?

Concerns – in terms of coding time or of machine time?

E.g.

```
proc sql;
create table comb as
select * from list a join table b
on id_list=id_table;
quit;
```

vs.

```
proc sql;
create table comb as
select * from list a join
(select * from table where id_table in (select id_list from list)) b
on id_list=id_table;
quit;
```

6.3 Tips?

What are your tips to me?

Topics – anything that is SAS coding related,

to sophie.zhang@gov.ab.ca

谢谢！