

HUG (SAS Health Users Group)

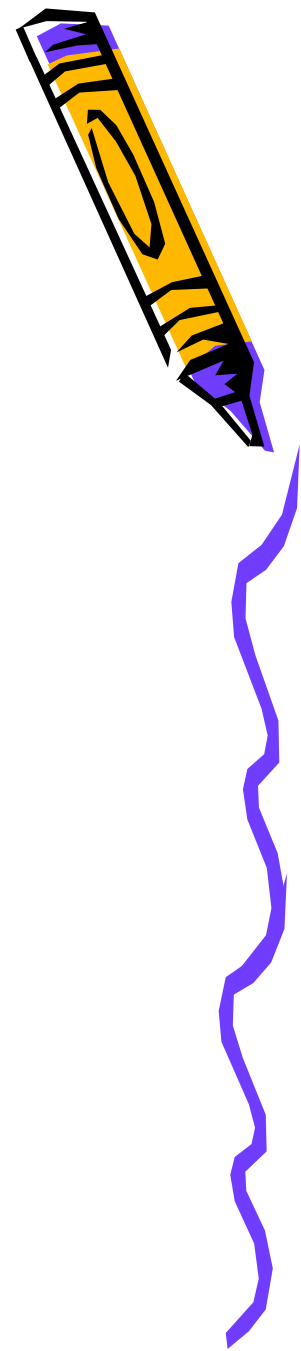
SAS TIPS & TRICKS

Jun Guan, Helen Guo

Programming and Biostatistics

Institute for Clinical Evaluative Sciences

April 1, 2011



When save input / output

- Use the **length** statement to override the default length of a variable. The shortest length for a numeric variable is 3.
- Use '**drop**' and '**keep**' statements to minimize the number of variables that are read in.
- Use a **where** statement rather than subsetting the dataset after it is read in. But...,
- Create more than one dataset during a single data step (where possible).

```
Data males females;  
  length yesno $3 drug 3;  
  set everyone;  
  drug=(Beta_blocker='Yes');  
  if sex='F' then output females;  
  else output males;
```

```
Run;
```

When save input / output (Cont.)

- Sort dataset only when necessary.

```
Proc means data=mydata;  
    var age height weight;  
    class sex;
```

```
run;
```

```
Proc sort data=mydata;  
    by id index_date;
```

```
Run;
```

```
Data mydata_unique;  
    set mydata;  
    by id;  
    if first.id=1;  
  
run;
```

```
proc sort data=mydata out=mydata_unique  
    nodupkey;  
    by id;  
  
run;
```

Finalize outputs in SAS rather than Excel

LHIN

01

MHI

\$67,000 (\$28,183)

```
Proc means data=income2003 mean std noprint nway;  
  class lhin;  
  var median_hh_income;  
  output out=_table(drop = _freq_ _type_) mean=m_mhi  
  std=std_mhi;
```

Run;

```
Data _table;  
  set _table;  
  MHI = put (m_mhi, dollar8.0)||' ('|| put(std_mhi, dollar8.0)||)';  
  drop m_mhi std_mhi;
```

Run;

Appending files efficiently

- With a SET statement in a data step:

```
Data work.step1;  
  set work.step1 work.step2;  
Run;
```

- With the Datasets procedure:

```
Proc datasets;  
Append base=work.step1 data=work.step2;  
Run;
```

Modifying variables

- Using a data step:

```
Data my.data1(rename=(sex=gender));  
  set my.data1;  
  format index_date mmddyy10.;  
Run;
```

- Using Proc Datasets;

```
Proc datasets lib=my nolist;  
  modify data1;  
  rename sex=gender;  
  format index_date=mmddyy10.;  
Quit;
```

More about PROC DATASETS:

- Remove all labels or formats in my.data1:

```
Proc datasets lib=my nolist;
```

```
  modify data1;
```

```
  attrib _all_ label=' ';
```

```
  format _all_ ;
```

```
Quit;
```

- Rename my.data1 to my.data2;

```
Proc datasets library=my nolist;
```

```
  change data1=data2;
```

```
Quit;
```


Use “Index” to speed up:

- Count how many physician claims each patient had?

```
Proc datasets library=work nolist;
```

```
  modify ohipclaim;
```

```
  index create patient_id;
```

```
Run;
```

```
Quit;
```

```
Data count_claims;
```

```
  set ohipclaim;
```

```
  by patient_id;
```

```
  if first.patient_id then num_claims=0;
```

```
    num_claims + 1;
```

```
  if last.patient_id then output;
```

```
Run;
```

Listing variables:

- Keep all numeric or character columns when creating a new dataset without listing each one in your KEEP statement:

```
Data test1 (keep= _numeric_ );  
    set source;
```

```
Run;
```

```
Proc format;  
    value missing . = 'Missing'  
           other = 'Nonmiss';  
    value $missing ' ' = 'Missing'  
           other = 'Nonmiss';
```

```
Run;
```

```
data test1 (keep= _character_);  
    set source;
```

```
Run;
```

```
proc freq data=source;  
    tables _all_ /missing;  
    format _numeric_ missing.;  
    format _character_ $missing.;
```

```
run;
```

Dealing with missing values:

- Initialise values: `a=.`; `b=0`; `c=-7`; `d=99`;

True or False:

1. `a<0`
2. `Add=a+b+c+d`; `add=92`;
3. `Sum=sum(a,b,c,d)`; `sum=92`;
4. `Sumzero=sum(0,a)`; `sumzero=0`;

- Many procedures (summary, tabulate, freq) ignore missing values, you should override the default if you want them included.

Proc freq;

 tables a*b/missing;

Run;

Count missing percentage for a dataset:

- MISSING function can take either a numeric or a character argument.

```
%macro misspct (data=, output=);
```

```
Proc contents data=&data noprint out=varlist(keep=name); run;
```

```
Proc sql; select count(name) into: varn from varlist; quit;
```

```
%macro loop;
```

```
  %do i=1 %to &varn;
```

```
    data _null_;
```

```
      set varlist;
```

```
      if _n_ = &i;
```

```
      call symput ("varname", name);
```

```
run;
```

```
Data tmp;
```

```
  set &data end=eof nobs=total;
```

```
  if missing(&varname)
```

```
    then missing_n + 1;
```

```
  if eof;
```

```
  missing_p=round(missing_n /  
                  total*100, .01);
```

```
  drop total;
```

```
run;
```

Count missing percentage for a dataset (Count.):

```
% if &l =1 %then %do;                                %end;
  data &output;                                       %mend loop;
  length varname $32.;                               %loop;
  set tmp(drop=&varname);                             %mend misspct;
  varname="varname";
  run;
%end;
```

```
%if &i>1 %then %do;
  data &output;
  set &output tmp(in=c drop=&varname);
  if c then varname="varname";
  run;
%end;
```

Operators: in, like & contains

- 'in' operator looks for a list of values (exactly).
- 'like' operator is used to test whether a character variable contains a specified pattern.
- 'contains' operator looks for a string in the middle of a longer string.
- An underscore (_) matches any single character.
- a percent sign (%) matches any number of characters.
- A colon used at the end of a comparison operator matches any longer string as long as it starts with the designated shorter string.

Operators: in, like & contains (Cont.):

drugname	Data try1;
VITAMIN D2 & CALCIUM & MAGNESIUM	set drugname;
VITAMIN D2 & CALCIUM	where drugname in: ('CALCIUM');
CALCIUM & VITAMIN C	Run;
CALCIUM & VITAMIN A & VITAMIN D	Data try2;
CALCITRIOL	set drugname;
CALCITONIN SALMON SYNTHETIC	where drugname contains ('CALCIUM');
ALGINATE CALCIUM(FIBRED)	Run;
BENZOCAINE & CALAMINE & ZINC OXIDE & THYMOL IODIDE	Data try3;
	set drugname;
	Where drugname like ('%CAL%');
	Run;

Strip & CATT functions:

Strip (var) = Trim (Left (var))

CATT: To remove trailing blanks, and returns a concatenated character string

Example:

```
a="Hello ";  
b="SAS ";  
c=" Users";  
d=" Group ";
```

```
catt=CATT(a,b,c,d);
```

HelloSAS Users Group

```
strip=strip(a)||strip(b)||strip(c)||strip(d);
```

HelloSASUsersGroup

Trim & CATT functions:

```
data _null_ ;  
  length codes $100.;  
  do i=90 to 97;  
    codes=trim(codes)||" "||'C'||put(i,z2.)||" " ;  
  end;  
  call symput ("codes",codes);  
run;
```

```
data _null_ ;  
  length codes $100.;  
  do i=90 to 97;  
    codes=catt(codes," ','C'||put(i,z2.)," " );  
  end;  
  call symput ("codes",codes);  
run;  
%put &codes;  
'C90' 'C91' 'C92' 'C93' 'C94' 'C95' 'C96' 'C97'
```

Unbalanced quotes and interruption:

- Log error message: 'CHARACTER STRING WAS MORE THAN 200 CHARACTERS'

Submit: *'; *"; */; run;

If use macros, submit: *); */; /*'*/ /*"*/; %mend;

- Use the WAUTOSAVE command if you want SAS to automatically save your work more often or less often than the default interval of every 10 minutes. SAS saves the Program Editor contents to 'pgm.asv' in the current working folder or in the folder specified by the AUTOSAVELOC system option.

WAUTOSAVE <<ON | OFF> INTERVAL=*minutes*>

Commenting out code containing comments:

- Use `/* */` to comment out an section of code.
- Make a whole section of code into a macro;

Example;

```
%macro junk; *this statement used to comment out following code;  
*do the next step;  
Data this; /*this is a nice dataset*/  
%mend;
```

- Use `NOSOURCE` option to stop your commented out code from printing to the log
- The `NOSOURCE` option causes only source lines that contain errors to be printed. (option `source----return to default`).

Acknowledgment:

- Ruth Croxford
- SAS

Questions?

