Ottawa Area SAS Users Society

www.OASUS.ca

Technical Tips and Tricks

Presented By: Glenn Robbins

"Come out of the desert of ignorance to the OASUS of knowledge"

Ottawa Area SAS Users Society

www.OASUS.ca

Topics

- Readability – some basic programming hints
- Data Conversion / Translation Techniques
- SAS User Defined Formats – obvious and less obvious uses

Making Your Code Easier to Read and Debug

- Use lots of empty space
  - Use Tabs to indent logic groups
    - Default on Enhanced editor is 4 spaces but you can change this to your own preference.
  - Use blank lines to separate data steps and procedures
  - Use blank lines to separate different logic blocks within a data step or proc step

```
* Sample Code Demonstrating Spacing *;
* Create dataset with converted codes *;
Data convert;
   set test;
   attrib
        hc_sex  length=$1 label = 'Sex of Person'
   ;
   * Convert sex data into one character abbreviation ;
   if sex = 1 then do;
        hc_sex = 'M';
   end;
   else if sex = 2 then do;
        hc_sex = 'F';
   end;
   else do;
        hc_sex = 'E';
   end;
run;

* Produce Cross Tabulation *;
proc freq data=testing ;
   table hc_sex
        / missing      ;
   format hc_sex $sex_lbl.
   ;
run;
```

Making Your Code Easier to Read and Debug
continued…

- Maximum of one program line per physical line of
  - Split one SAS program line across several physical lines for readability. SAS continues to read until it reads a semi-colon ';'
- Don't be afraid to add 'extra' lines of code
  - More self-documenting
  - Easier to debug
- Use lots of comments / documentation
  - /* This is valid syntax for a comment */
  - * This is valid syntax for a comment ;

Use more physical lines not less

Version A – lots of spaces

```
proc format;
   value cvt_sexN
        1      = 'M'
        2      = 'F'
        other = 'E'
   ;
run;
```

Version B – packed on one line

```
proc format; value cvt_sexN 1= 'M' 2= 'F' other
   = 'E'; run;
```

More code rather than less

Version A

```
if sex = 1 then hc_sex = 'M';
else if sex = 2 then hc_sex = 'F';
else if sex = 3 then hc_sex = 'E';
```

Version B

```
if sex = 1 then do;
   hc_sex = 'M';
end;
else if sex = 2 then do;
   hc_sex = 'F';
end;
else do;
   hc_sex = 'E';
end;
```

Use comments and self –documenting features

- comments
  - /* This is valid syntax for a comment */
  - * This is valid syntax for a comment ;
- Meaningful names
  - Variable names can be 32 characters long
  - Datasets names can be 32 characters long
  - Use underscore to separate names
    - e.g. first_name =
- Use Labels
  - Variables
    - Label varname = ‘Variable Label’
  - Datasets (New in version 8)

Use comments and self –documenting features

```
* Sample Code Demonstrating Documenting and comments *;

* Create dataset with work phone list *;
Data Phone_Numbers
   (label='List of Health Canada Phone Numbers');

   * List of variables to be created;
   attrib  first_name   length=$1 label = 'First Name of Person'
           last_name    length=$1 label = 'Surname of Person'
           work_phone_number length=10 label = 'Work Phone No'
   ;
   infile input_data; /* This data comes from HC phone book */
run;
```

Data Quality / Conversion

- If -Then -Else
- Select / When
  (the case construct)
- The PUT function with a user defined format

If - Then - Else

```
if sex = 1 then do;
  hc_sex = 'M';
end;
else if sex = 2 then do;
  hc_sex = 'F';
end;
else do;
  hc_sex = 'E';
end;
```

Select / When

```
Select (sex);
    when(1) do;
          hc_sex = 'M';
    end;
    when(2) do;
          hc_sex = 'F';
    end;
    otherwise do;
          hc_sex = 'E';
    end;
end;
```

The PUT function with a user defined format

```
proc format;
   value cvt_sexN
        1      = 'M'
        2      = 'F'
        other = 'E'
   ;
run;

* Create dataset with converted codes *;
Data convert;
    set test;
   * Convert data using User Defined FORMAT;
   hc_sex = put(sex,cvt_sexN.);
run;
```

User Defined Formats

- **Uses**
  - **Labels**
  - **Grouping Data**
  - **Conversions / Data Quality Verification**
  - **Documentation (Data Value Labels)**

Labels

```
proc format;

    * Create labels for SEX variable;
   value $sex_lbl
        'M' = 'Male'
        'F' = 'Female'
        'E' = 'Unknown - Errors'
   ;
run;
```

Sample Output

Ottawa Area SAS Users Society

www.OASUS.ca

Using Formats for Labeling

Province / Territory NF

		Number of Persons
		Sum
Sex of Person	Age	
F	24	2500.00
	26	3000.00
	28	2000.00
M	24	2000.00
	26	5000.00
	28	1000.00

Province / Territory : Newfoundland

		Number of Persons
		Sum
Sex of Person	Age	
Female	20 - 24	2500.00
	25 - 29	5000.00
Male	20 - 24	2000.00
	25 - 29	6000.00

Conversion / Lookup Table

```
proc format;

* Create conversion table for SEX
  variable;
  value cvt_sexN
      1      = 'M'
      2      = 'F'
      other  = 'E'
  ;
run;
```

Grouping Data

```
proc format;
   * Create AGE groups;
   value age
       20 - 24 = '20 - 24'
       25 - 29 = '25 - 39'
       30 - 35 = '30 - 35'
   ;
run;
```

Grouping Data

- Use to group data in an output dataset : e.g. proc freq, summary

```
proc freq data=test;
   table age
              / out=data_agegrp;
      format age age.;
run;
```

Grouped Data

Original Data

1 NF 24 2000
2 NF 24 2500
1 NF 26 2000
2 NF 26 3000
1 NF 28 3000
2 NF 28 3000

Sample Output Grouped Using a Format

Obs	age	COUNT	PERCENT
1	24	4500	29.0323
2	26	11000	70.9677

Ottawa Area SAS Users Society

www.OASUS.ca

User Defined Formats

- **Proc FORMAT**
  - **Using information from formats to create data dictionary [cntlout=]**
  - **Using information in a dataset to create formats [cntlin=]**

Create Dataset from existing formats

```
proc format cntlout=fmtdata;
run;

proc print data=fmtdata;
   var fmtname start end label;
run;
```

Sample output from Proc Format Dataset

Obs	FMTNAME	START	END	LABEL	TYPE
1	SEX_LBL	E	E	Unknown - Errors	C
2	SEX_LBL	F	F	Female	C
3	SEX_LBL	M	M	Male	C

Create Dataset from pre-existing data - Step 1 -

```
Data DD;
   attrib
     var_name  length=$16 label = 'Variable Name'
     var_value length=$1  label = 'Variable Value'
     Label     length=$16 label = 'Label - English'
    ;
    input var_name var_value labele;

cards;
sex M Male
sex F Female
;
run;
```

Create Dataset from pre-existing data - Step 2 -

```
* Use dataset for creating formats;
* Change variable names to those that PROC FORMAT
   expects (SAS Procedure Guide - Chapter 19);
data format_labels;
   set dd;
   rename
        var_name     =      FMTNAME
        var_value    =      START
        LabelE       =      LABEL
   ;
     type = 'C';
run;

* Create User defined format from dataset ;
proc format cntlin=format_labels;
run;
```

Summary

- Make code easier to read (for humans)
- Variety of methods for data quality verification – use what makes sense to you
- User defined formats can be extremely useful.