Statistical Analysis System: Class 27

Dated: 02/06/2018

Proc Formats: creating your own format

Proc format is used to create user defined formats. This procedure starts with the statement PROC FORMAT and continues with 1 or more VALUE statements ("Value" is the keyword used alongwith Proc Format). Formats aren't stored in meta-data.

Syntax: PROC FORMAT LIB = (library name); /* library name where you want the format to be applicable */

VALUE name; /* name, here is the name of the format we are creating */

Range-1= 'formatted text '

Range-2= 'formatted text'

Range-n= 'formatted text'

1. If the format is for character data, the name must start with a $ - sign.
2. While calling a format, format name ends with a period (.) always.
3. Name can't be longer than 32 character and the rules for naming are same as are applicable in SAS.
4. Each RANGE here, signifies a value of a variable to which a text is assigned, enclosed in quotation marks.
5. Character values must be enclosed in ' '.
6. If there is more than value in the range, then separate the values with a comma or use hyphen (-) for continuous range.
7. Multiple spaces are considered as 1 space and it concatenates two strings if not comma separated.
8. **Keywords "LOW" & "HIGH"** can be used to indicate the lowest and the highest value of any variable.
9. **"Other" keyword** can be used to assign a format to any values not listed in the VALUE statement.

CODE	Explanation

`Proc format lib=work;`
`Value ag`
`0-<30='young'`
`30-<61='old';`
`Run;`
 	
`Proc print data=sasuser.admit;`
`Format age ag.;`
	Numeric Format **ag'** is created, in the work library (is a temporary format) with the specified ranges:
	0 to less than 30 coded as 'young'
	30 to less than 61 coded as 'old'
	
	Format '**ag'** is called & applied with 'age'

Run;	variable on the 'admit' dataset.
Data a; Set sasuser.admit; Status=put(age,ag.); Run;	This creates a new variable "Status" in the admit dataset with **'ag'** format applied to it
Proc format lib=work; Value agefmt low-<30='young' 30-<high='old'; Run;	This creates a numeric format named **'agefmt' in reference to point 8 listed above.**
Proc format lib=work; Value ag 0-<30='young' 30-<61='old'; Value he Low-<70='average height' 70-high='tall'; Run; Proc print data= sasuser.admit; Format age ag. height he.; Run;	Numeric Format **'ag'** created. Numeric Format **'he'** created considering point 8 above Both formats **'ag'** & **'he'** are called at once applied on different variables within a dataset
Proc format lib=work; Value $ act 'high'=1 'low'=2 'mod'=3 Run; Proc print data= sasuser.admit; Format actlevel $act.; Run;	Character format **'act'** is created here. **'act'** is called and applied within admit dataset on the values of variable actlevel.
Proc format lib=work; Value ag 0-<30='young' 30-<61='old' Other="NA"; Run; Proc print data=sasuser.admit; Format age ag.; Run;	Numeric format **'ag'** is created here, **in reference to point 9 above** (i.e for all values other than 0 to 60 coded value is "NA") **'ag'** format is applied to values of variable **age** in the admit dataset.
Proc format lib=work; Value $ act 'High'="High Glucose" 'Low'="low Glucose" 'Mod'="Moderate Glucose"; Run; Proc print data=sasuser.admit; Format actlevel $act.;	Character format **'act'** created. **'act'** applied on values of variable **actlevel.**

Run;	

CODE	Explanation
Data grade; Input grade$; Datalines; A B C D E F ; Run; Proc format; Value $grade 'A' = 'Excellent' 'B'-'D' = 'Pass' 'E', 'F'= 'Fail'; Run; Proc print data=grade; Format grade $grade.; Run;	Dataset grade created. Character format 'Grade' created with the specified range: 'A' coded as 'Excellent' 'B'to'D' coded as 'Pass' 'E', 'F' coded as 'Fail' **Note**: Point 6 applicable from above for comma separated and continuous ranges . Character format 'Grade' called applicable on values of variable 'grade' within dataset 'grade'.
Proc format; Value $grade 'A' = 'Excellent' 'B'-'D' = 'Pass' " student" 'E', 'F'= 'Fail'; Run; Proc print data=grade; Format grade $grade.; Run;	Here, continuous range 'B'to'D' will be coded as 'pass student'. **Note**: Point 7 applicable from above for non-comma separated with multiple spaces, formatted text.
Proc format lib=work fmtlib; Run; Proc format lib=work fmtlib; Select ag; Run;	Everything is available from format catalogue here. This format selects only the 'ag' format from the format catalogue.
Proc format lib=sasuser Value $ kaka 'High'=1 'Low'=2 'Mod'=3; Run;	This format creates character format **'kaka'** in the **permanent library "sasuser".**

`Options fmtsearch = (sasuser);`
 	
`Proc print data=sasuser.admit;`
`Format age kaka.;`
`Run;`
	This statement conveys sas to look for the applicable format in the sasuser library
	
	Format **'kaka'** is called and is applied on the values of variable **'age'.**

`Proc format lib=sasuser lib=work;`
`Value $ khoka`
`'High'=1`
`'Low'=2`
`'Mod'=3;`
`Run;`
 	
`Options fmtsearch = (sasuser);`
	This format is an example to understand that the format **'KHOKA'** is created in the work library and not in the sasuser library, although defined with both.
	
	SAS ignores 1st one (i.e: lib=sasuser) and works with 2nd one ((i.e: lib=work)

Proc contents: listing contents of a SAS Dataset

SAS stores the information about the dataset, (also called the descriptor portion) along with data. To get a description of a sas data set we use Proc Contents.

Syntax: Proc contents data = (dataset name);

Output for Proc contents is like a table of contents for your data set. Output starts with information about the data set and then describes each variable, like :-

For the data set	**For each variable**
Data set name	Type (Numeric / character)
No of observations	Length (size)
No of variables	Format / Informat / Label (if any) etc.
Date created etc.	

CODE	**Explanation**

`Proc format lib=work;`
`Value ag`
`0-<30='young'`
`30-<61='old'`
`Other="NA";`
`Run;`
 	
`Data a;`
`Set sasuser.admit;`
`Status=put (age,ag.);`
`Run;`
 	
`Proc contents data = a;`
	Numeric format **'ag'** is created.
	
	Format called & applied on values of variable age resulting into a new variable **"Status"** in the dataset **'a'**.

`Run;`	Proc Contents called for dataset **'a'** will display all the information about its variables and the data within dataset **'a'.**

Proc Freq:

Proc Freq is used to get the statistics of categorical data.
Frequency gives the number of occurrences in a fixed interval of time.
It can also be used for separating duplicates (for a specific variable, any count > 1, is a duplicate)
Proc Freq is applied on a variable where its values are repeating.

Syntax: Proc Freq data = library.dataset;

Tables variables-combinations / options;

CODE	Explanation

Examples for 1-way frequency: is applied only on 1 variable
 	
`Proc freq data=sasuser.admit;`
`Tables actlevel;`
`Run;`
	Displays statistics from admit dataset based on variable "actlevel".

`Proc freq data=sasuser.admit;`
`Tables actlevel;`
`Where age gt 40;`
`Run;`
	Displays statistics from admit dataset based on variable "actlevel" but only where age is greater than 40.

`Proc format lib=work;`
`Value $ r`
`'Route1'-'Route3'="USA"`
`Other="Europe";`
`Run;`
 	
`Proc freq data= sasuser.cargorev;`
`Tables route;`
`Format route $r.;`
`Run;`
	Creates a format " r " in "work" with continuous ranges and other keywords used as per point 6 & 9.
	
	Displays statistics from "Cargorev" dataset based on variable "route" as per the character format "r" .

`Proc freq data=sasuser.admit;`
`Tables sex actlevel;`
`Run;`
	Displays statistics sex-wise , actlevel wise separately.

Cross-tabulation (2 way frequency)
.....
`Proc freq data=sasuser.admit;`
`Tables sex*actlevel;`
`Run;`
	Displays sex-wise , actlevel wise statistics altogether.

`Proc freq data=sasuser.admit;`
`Tables sex*actlevel/list;`
`Run;`
	Produces List wise display

`Proc freq data=sasuser.admit;`
`Tables sex*actlevel/ out=a;`
`Run;`
	Output in a new dataset "a"