

# The Power of PROC FORMAT

Jonas V. Bilenas  
JP Morgan Chase Bank

# The Power of Proc Format

Jonas V. Bilenas, CHASE Manhattan Bank

- Tutorial on how to build user defined formats using PROC FORMAT.
- Examples come from the Credit Industry but are applicable to other disciplines.
- All data is illustrative (hypothetical data either made up or generated with random variables).
- Source Code using SAS9.1
- From a new SAS/Book By User Book, Power of Proc Format. Release date is September 2004

## Using SAS Internal INFORMATS and FORMATS

You can view INFORMATS as instructions for reading data into SAS and FORMATS for formatting of output. We will look at a number of examples of how to use SAS provided INFORMATS and FORMATS. Some of the examples will point out some of the pitfalls to watch out for when reading and formatting data.

CONCEPT	APPLICATION	USAGE IN A DATA STEP	USAGE IN A PROC
<b>INFORMAT</b>	INPUT Data	Use with INPUT, ATTRIB, or INFORMAT statements. Use with INPUT and/or INPUTN Function.	Used mostly in DATA STEP, but can also use with ATTRIB or INFORMAT Statements in PROCs
<b>FORMAT</b>	OUTPUT Data or Format Data in reports	Use with PUT, ATTRIB, or FORMAT statements. Use with PUT and/or PUTN Function.	Use ATTRIB or FORMAT Statements

## USING SAS INFORMATS

- INFORMATS are typically used to read or INPUT data from external non-SAS files called flat files (text files, ASCII files, or sequential files). The INFORMAT instructs SAS on how to read data into SAS VARIABLES. SAS INFORMATS are typically grouped into 3 categories:
  - CHARACTER INFORMATS: **\$INFORMAT<sub>w</sub>**.
  - NUMERIC INFORMATS: **INFORMAT<sub>w.d</sub>**
  - DATE/TIME INFORMATS: **INFORMAT<sub>w</sub>**.
- The \$ is used to indicate a character informat.
- **INFORMAT** refers to the SAS INFORMAT name which, we will see, is sometimes optional.
- The *w* indicates the width (bytes or number of columns) of the variable.
- The *d* is used for numeric data to specify the number of digits to the right of the decimal place.
- All INFORMATS must contain the decimal point (.) so that SAS can differentiate an INFORMAT from a VARIABLE.

# USING SAS INFORMATS: examples

## USING THE INPUT STATEMENT:

DATA:

ID	Transaction Date	Transaction Amount
124325	08/10/2003	1250.03
7	08/11/2003	12500.02
114565	08/11/2003	5.11

```
filename transact 'C:\BBU FORMAT\DATA\TRANS1.DAT';

data transact;
  infile transact;
  input      @1      id          $6.
            @10     tran_date   mmddyy10.
            @25     amount      8.2
            ;
run;
```

## USING SAS INFORMATS: INPUT Function

- One can use INFORMATS in an INPUT function within a DATA step. As an example, we can convert the id variable used in the above example from a character variable to a numeric variable in a subsequent data step. The code is shown here:

```
data transact2;  
  set transact;  
  id_num = input(id,6.);
```

- Note that when using the INPUT function we do not have to specify the *d* component if the character variable contains embedded decimal values.
- Variable type (numeric or character) is determined by the INFORMAT used.

## USING SAS FORMATS: Application in a PROC

- Let us return to the first example use PROC PRINT to include a FORMAT statement that would return dates in standard mm/dd/yyyy format and list out transaction amounts using dollar signs and commas. Here is the code:

```
options center;
filename transact 'C:\BBU FORMAT\DATA\TRANS1.DAT';

data transact;
  infile transact;
  input @1 id          $6.
        @10 tran_date mmddy10.
        @25 amount    8.2
        ;
run;

proc print data=transact;
  format tran_date mmddy10.
         amount    dollar10.2;
run;
```

Obs	id	tran_date	amount
1	124325	08/10/2003	\$1,250.03
2	7	08/11/2003	\$12,500.02
3	114565	08/11/2003	\$5.11

## USING SAS FORMATS: Using a PUT Function

- Like the INPUT function, SAS also has the PUT function to use with sas variables and FORMATS to return character variables. The format applied to the source variable must be the same type as the source; numeric or character.
- For example, what if we have a data set with a 13 digit numeric variable called accn\_id and wish to generate a character variable called char\_accn\_id from the numeric variable with leading zero's? The following PUT Function can be applied in a DATA step:

```
char_accn_id = put(accn_id,z13.);
```

- **Note that PUT function always returns a character variable while the INPUT function returns a type (numeric or character) dependent on the INFORMAT used in the argument.**

## Why use PROC FORMAT: Table Lookups

- Table Mappings (values to labels) possible with PROC FORMAT:
  - 1 to 1
    - example: ‘a’=‘approve’ ‘d’=‘decline’
  - many to 1
    - example: In the credit industry a credit grantor can request a credit score for an individual or small business. One such score provides scores from 370-870 with the higher the score, the less risk. Scores at or below 670 are considered sub-prime. Missing scores are scored as values very low (1, 2, 3, 4) or very high (9001, 9002, 9003, 9004) depending on the credit bureau. We wish to map scores into ranges
      - 370-670 = ‘LE 670’
      - 671-870 = ‘GT 670’

## Table Lookup - further discussion

- By default, 1 to many or many to many mappings are not supported in user defined formats. This is a good thing since you don't want values to take on more than one label. A DATA step will not flag values pointing to more than one label when using IF/THEN or IF/THEN/ELSE code.
  - I have seen many programs with values getting assigned multiple labels in IF/THEN code. These errors usually show up when converting IF/THEN code into PROC FORMAT. PROC FORMAT will return an error if one specifies 1 to many or many to many mappings.
- There is a MULTILABEL Option in PROC FORMAT that will allow one to generate one to many or many to many mappings.

## Example - without PROC FORMAT

- **Problem:** Need a count of names with SCORE<=670, SCORE>670 & unscored.

```
data sushi;  
  set trw14a.kp14ava;  
  if score <= 670 then scoreg='670-';  
  else if score <=900 then scoreg='671+';  
  else scoreg='unscored';
```

```
proc freq data=sushi; tables scoreg;  
run;
```

SCOREG	Frequency	Cumulative Percent	Cumulative Frequency	Percent
671+	623	10.7	623	10.7
670-	5170	89.2	5793	99.9
unsc	5	0.1	5798	100.0

## Example - with PROC FORMAT

```
proc format;
  value score low-670 = '670-'
             670<-900 = '671+'
             other    = 'unscored'
;
proc freq data= trwl4a.kp14ava;
  tables score;
  format score score.; * format assigned with a format statement;
run;
```

SCORE	Frequency	Percent	Cumulative Frequency	Cumulative Percent
670-	5170	89.2	5170	89.2
671+	623	10.7	5793	99.9
unscored	5	0.1	5798	100.0

## Application: Finding Unexpected Values

```
proc format;
  value looky 370-900 = '370-900'
  ;
proc freq data=stuff.seg131tr;
  tables score;
  format score looky.; * note: Name of format does not have to be same
                        as the name of the variable;
run;
```

FICOSC	Frequency	Percent	Cumulative Frequency	Cumulative Percent
370-900	30320	96.0	30320	96.0
9003	1264	4.0	31584	100.0

## Generating INFORMATS with PROC FORMAT

```
proc format;
  invalue $sx  'Fe' = 'Female'
              ' M' = 'Male'
;

data informat_test;
  input sex $sx.;
  datalines;
Fe
 M
Fe
Fe
 M
 M
;

proc print data=informat_test;
run;
```

Obs	sex
1	Female
2	Male
3	Female
4	Female
5	Male
6	Male

## Specifying Ranges of VALUES or INVALIDES

- Ranges can be constant values, or values separated by commas
  - ‘a’, ‘b’, ‘c’
  - 1,2,3
- Range can also include intervals.
  - lower-higher
    - Interval includes both endpoints
  - lower<-higher
    - Interval includes higher endpoint
  - lower-<higher
    - Interval includes lower endpoint
  - lower<-<higher
    - Interval does not include either endpoint
- Special Keywords
  - LOW, HIGH, OTHER, ., ‘ ‘
    - Note: LOW keyword does not include missing values.
    - OTHER includes missing unless specified by a mapping

## Rules for FORMAT or INFORMAT NAMES

- All character NAMES and must begin with a '\$'. The choice for determining if a character or numeric format is required is dependent on the actual variable that the FORMAT/INFORMAT is being applied to.
- The name of the FORMAT/INFORMAT cannot begin or end in numbers.
- One cannot use names that are used by SAS INTERNAL FORMATS/INFORMATS.
- The FORMAT/INFORMAT name does not have to match the name of an existing variable. User defined FORMATS/INFORMATS are independent of the data variables. User defined FORMATS/INFORMATS are defined first and then applied to variables in subsequent PROCS or DATA steps.
- For SAS9, the length of an INFORMAT NAME has a maximum length of 31. If user defines a character INFORMAT, the \$ is included in the determination of the maximum length. The maximum length for the FMTNAME for a user defined FORMAT is 32 bytes which includes the \$ if required for character FORMATS.
- In earlier versions of SAS, the length of FORMAT/INFORMAT names was limited to 8 for FORMATS and 7 for INFORMATS (including the \$).

## PICTURE FORMATS

- Picture Formats provides a template for printing numbers. The template contains “DIGIT SELECTORS” that specify how the numbers will be displayed:
  - 0: Place holders for digits. If a digit exists, a number gets printed. If not, then nothing gets printed.
  - 9 (or any other digit other than 0): an absolute place holder that forces a number to print even if it is 0.
- Also used to embed characters within the label:
  - Decimals and comma placement
  - Embedding characters.
  - Prefixes.
- Multiply numbers by factors



# Picture formats - example 1: adding % in TABULATE Output

```
proc format;  
  value score low-<200 = 'LT 200'  
             200-high = 'GT 200'  
  ;
```

**FORMAT score:** Defines  
Score groups.

```
  picture pct low-high = '009.9%'  
  ;
```

**PICTURE FORMAT pct:**

- Includes a forced leading 0 to the left of the decimal point using the '9' selector.
- Specifies 1 digit to the right of the decimal point.
- Includes a '%' after digit selectors.

```
proc tabulate data=bbu.scores noseps;  
  class score;  
  format score score.;  
  keylabel n=' ' pctn=' ' sum=' '  
  table (score all)  
  ,  
  (N='Frequency'*f=comma10.  
  pctn='ROW %'*f=pct.  
  )/rts=10 misstext=' '  
run;
```

**Format score** assigned with  
FORMAT statement.

**Picture pct** assigned using  
the f= format option for the  
PCTN

## Picture formats - example 1: Output

```
„fffffffff...fffffffffffff...ffffff†  
,           ,Frequency ,ROW % ,  
‡fffffffff^fffffffffffff^ffffff%  
,score     ,           ,  
,LT 200    ,           5,976, 59.7%,  
,GT 200    ,           4,024, 40.2%,  
,All       ,           10,000,100.0%,  
Šfffffffff<fffffffffffff<ffffff&
```

← Is there an error in statistics not rounding?

PICTURE FORMATS truncate digits by default. In order to round results, you need to specify the **(ROUND)** option for the picture FMTNAME.

To round results, just change the picture statement to look like this:

```
picture pct (round) low-high = '009.9%'  
;
```

## Picture formats - PREFIX OPTION

- With picture formats, all text before the first the first digit selector is ignored. To add text before the first digit selector, use the (**PREFIX=**) label option. The following picture examples show how to add various prefix characters.

- Negative signs for percentages:

```
picture pct (round) low - <0 = '009.99%' (prefix='-')
                                0 - high = '001.23%'
;
```

- Negative and dollar signs:

```
picture dol (round) low - <0 = '000,009.99' (prefix='- $')
                                0 - high = '000,009.99' (prefix='$')
;
```

## Picture Formats - Test width of Format

- If the label of a PICTURE FORMAT is not wide enough, the FORMAT will truncate from the left.
- Test your FORMATS.

```
proc format;
  picture dol (round) low-<0 = '0,009'
                                (prefix='- $')
                                0-high = '0,009'
                                (prefix='$')
;
run;

data _null_;
  do test = -50000, -1000, -10, -1.5, -0.5, .4,
           .5, 10, 1000, 50000, 500000;
    put test= @20 'FORMATED Result: ' test dol.;
  end;
run;
```

```
test=-50000      FORMATED Result:  -$0
test=-1000       FORMATED Result:  1,000
test=-10         FORMATED Result:  -$10
test=-1.5       FORMATED Result:  -$2
test=-0.5       FORMATED Result:  -$1
test=0.4        FORMATED Result:   $0
test=0.5        FORMATED Result:   $1
test=10         FORMATED Result:   $10
test=1000       FORMATED Result:  1,000
test=50000     FORMATED Result:   $0
test=500000    FORMATED Result:   $0
```

- To remedy the FORMAT, increase the number of leading 0's in the LABEL. Here is the new code and the resulting test:

```
picture dol (round) low-<0 = '000,000,009' (prefix='- $')
                                0-high = '000,000,009' (prefix='$')
;
```

## Picture Formats - MULTIPLIER Option

- MULTIPLIER or MULT option multiplies value by a factor before applying the label.

```
proc format;
  picture dolm (round) low-<0  = '000,000,009M' (prefix='- $' mult=1e-3)
                                0-high = '000,000,009M' (prefix='$'   mult=1e-3)
;
run;

data _null_;
  do test = -55678, 10, 1000, 51230, 500132;
    put test= @20 'FORMATED Result: ' test dolm.;
  end;
run;
```

test=-55678	FORMATED Result:	-\$56M
test=10	FORMATED Result:	\$0M
test=1000	FORMATED Result:	\$1M
test=51230	FORMATED Result:	\$51M
test=500132	FORMATED Result:	\$500M

## Picture Formats - MULTIPLIER Option (continued)

- What if we want to add more precision on the output? Like adding a 100's place to the output.

```
proc format;
  picture dolm (round) low-<0  = '000,000,009.9M' (prefix='-$$' mult=1e-3)
                                0-high = '000,000,009.9M' (prefix='$'  mult=1e-3)
  ;
run;

data _null_;
  do test = -55678, 10, 1000, 51230, 500132;
    put test= @20 'FORMATED Result: ' test dolm.;
  end;
run;
```

test=-55678	FORMATED Result:	-\$5.6M
test=10	FORMATED Result:	\$0.0M
test=1000	FORMATED Result:	\$0.1M
test=51230	FORMATED Result:	\$5.1M
test=500132	FORMATED Result:	\$50.0M

- Why the unexpected results?.

## Picture Formats - MULTIPLIER Option (continued)

- Why the unexpected results?.
- The thing to remember is that when we add the decimal place in the LABEL and a MULT option, we really have 2 multipliers in effect.
  - There is one multiplier in the label ( $1e-1$  for the example) that accounts for how many digits to the left of the last digit the decimal value is placed
  - The second multiplier is in the MULT OPTION ( $1e-3$ ).
  - When we add the 2 multipliers, the overall effect is  $1e-4$ .
- So the result looks like each test value is multiplied by  $1e-4$ . To print results to the nearest \$1000 with a significant decimal to the right of the decimal, we have to decrease the MULT specified option to  $1e-2$ . The final result is to multiply the VALUE by  $1e-3$  (or move the decimal to the left 3 places.
- The documentation in SAS help for MULT= is a bit different, but I find that taking into account the total multiplier effect (LABEL effect plus MULT effect) is a useful way to pick the correct MULT value. Modifying the previous code shows the output as expected.

## Picture Formats - MULTIPLIER Option (continued)

```
proc format;
  picture dolm (round) low-<0 = '000,000,009.9M' (prefix='- $' mult=1e-2)
                                0-high = '000,000,009.9M' (prefix='$' mult=1e-2)
  ;
run;

data _null_;
  do test = -55678, 10, 1000, 51230, 500132;
    put test= @20 'FORMATED Result: ' test dolm.;
  end;
run;
```

LOG Output:

test=-55678	FORMATED Result:	-\$55.7M
test=10	FORMATED Result:	\$0.0M
test=1000	FORMATED Result:	\$1.0M
test=51230	FORMATED Result:	\$51.2M
test=500132	FORMATED Result:	\$500.1M

## Picture Formats - NOEDIT Option

- Use the (**noedit**) LABEL option when you just want to use the LABEL as is without any edits of the digits. Here is an example:

```
proc format;
  picture dolm (round) low-0    = '000,000,009.9M' (prefix='- $' mult=1e-2)
                               0-100000 = '000,000,009.9M' (prefix='$' mult=1e-2)
                               100000-high = 'GE $100,000'    (noedit)
;
run;

data _null_;
  do test = -55678, 10, 1000, 51230, 500132;
    put test= @20 'FORMATED Result: ' test dolm.;
  end;
run;
```

### LOG OUTPUT:

```
test=-55678      FORMATED Result:      -$55.7M
test=10          FORMATED Result:         $0.0M
test=1000        FORMATED Result:         $1.0M
test=51230       FORMATED Result:         $51.2M
test=500132      FORMATED Result:      GE $100,000
```

## DATA Set Applications

- Variable Assignment
- Data Merge

# DATA Set Variable Assignment: Simple Credit Line Assignment Example

## TABLE LOOKUP USING INFORMAT

```
proc format;
  invalue STL low-<160 = 1000
              160-179 = 2500
              180-199 = 5000
              200-219 = 7500
              220-high = 9500
;

data scores;
  set bbu.scores;
  line = input(score,STL.);
run;
```

Creation of  
INFORMAT  
STL

TABLE LOOKUP:  
Mapping score  
invalues to line  
using STL  
INFORMAT

## TABLE LOOKUP USING FORMAT

```
proc format;
  value STL low-<160 = 1000
           160-179 = 2500
           180-199 = 5000
           200-219 = 7500
           220-high = 9500
;

data scores;
  set bbu.scores;
  line = input(put(score,STL.),best12.);
run;
```

Creation of  
FORMAT STL

TABLE LOOKUP:  
Mapping score  
invalues to line  
using STL  
INFORMAT

## DATA Set Variable Assignment: 2 Dimension Table Lookup Credit Line Assignment Example

One can use PROC FORMAT to generate a value based on values of 2 other variables. In this example, LINE assignment is dependent on SCORE and INCOME\_EST variables. We wish to assign LINE based on the following assignment specification table:

		<b>INCOME Estimate:</b>			
		<b>low-&lt;35,000</b>	<b>35,000-&lt;45,000</b>	<b>45,000-&lt;55,000</b>	<b>55,000+</b>
	<b>low-&lt;160</b>	500	750	1000	1250
	<b>160-179</b>	1500	2000	2500	3000
<b>SCORE</b>	<b>180-199</b>	4000	4500	5000	6000
	<b>200-219</b>	7000	7500	8000	8500
	<b>220-high</b>	9000	10000	15000	20000

## DATA Set Variable Assignment: 2 Dimension Table Lookup Credit Line Assignment Example

```
proc format;
  value score_f low-<160 = INCA
                160-179 = INCB
                180-199 = INCC
                200-219 = INCD
                220-high = INCE
  ;
  value INCA low-<35000= 500 35000-<45000= 750 45000-<55000=1000 55000-high=1250
  ;
  value INCB low-<35000=1500 35000-<45000=2000 45000-<55000=2500 55000-high=3000
  ;
  value INCC low-<35000=4000 35000-<45000=4500 45000-<55000=5000 55000-high=6000
  ;
  value INCD low-<35000=7000 35000-<45000=7500 45000-<55000=8000 55000-high=8500
  ;
  value INCE low-<35000=9000 35000-<45000=10000 45000-<55000=15000 55000-high=20000
  ;

data scores;
  set bbu.scores;
  fmtuse = put(score,score_f.);
  line = input(putn(income_est,fmtuse),best12.);
run;
```

## Large Data Extracts

Using Formats to match merge a large unsorted dataset.

### **Problem:**

- A large data set (up to 100,000 observations) has to be merged with a very large (millions of observations) unsorted data set.

## Large Data Extracts

### **Solution:**

- Build a format using the DATA step. The special data that will generate the FORMAT must not have dupes of matching key field. Data must have the following elements:
  - **FMTNAME**: name of format to be created.
  - **TYPE**: ‘C’ for Character, ‘N’ for numeric (optional).
  - **START**: the value you want to translate (If using range, this is lower end of range. Also require and **END** for the upper end.
  - **LABEL**: the label you want to assign
- Generate the format with a **CNTLIN=** option to point to the dataset in a PROC FORMAT.
- Apply that format to the large dataset to select records.

## Large Data Extracts

```
proc sort data=acct8.acct8 (keep=acct8)
          out=temp nodupkey force;
  by acct8;

data fmt (rename=(acct8=start));
  retain fmtname '$acct' type 'C' label 'Y';
  set temp;

proc format cntlin=fmt;

data bigmatch;
  set bigfile /* LARGE UNSORTED FILE */;
  where put(acct,$acct.)='Y';
run;
```

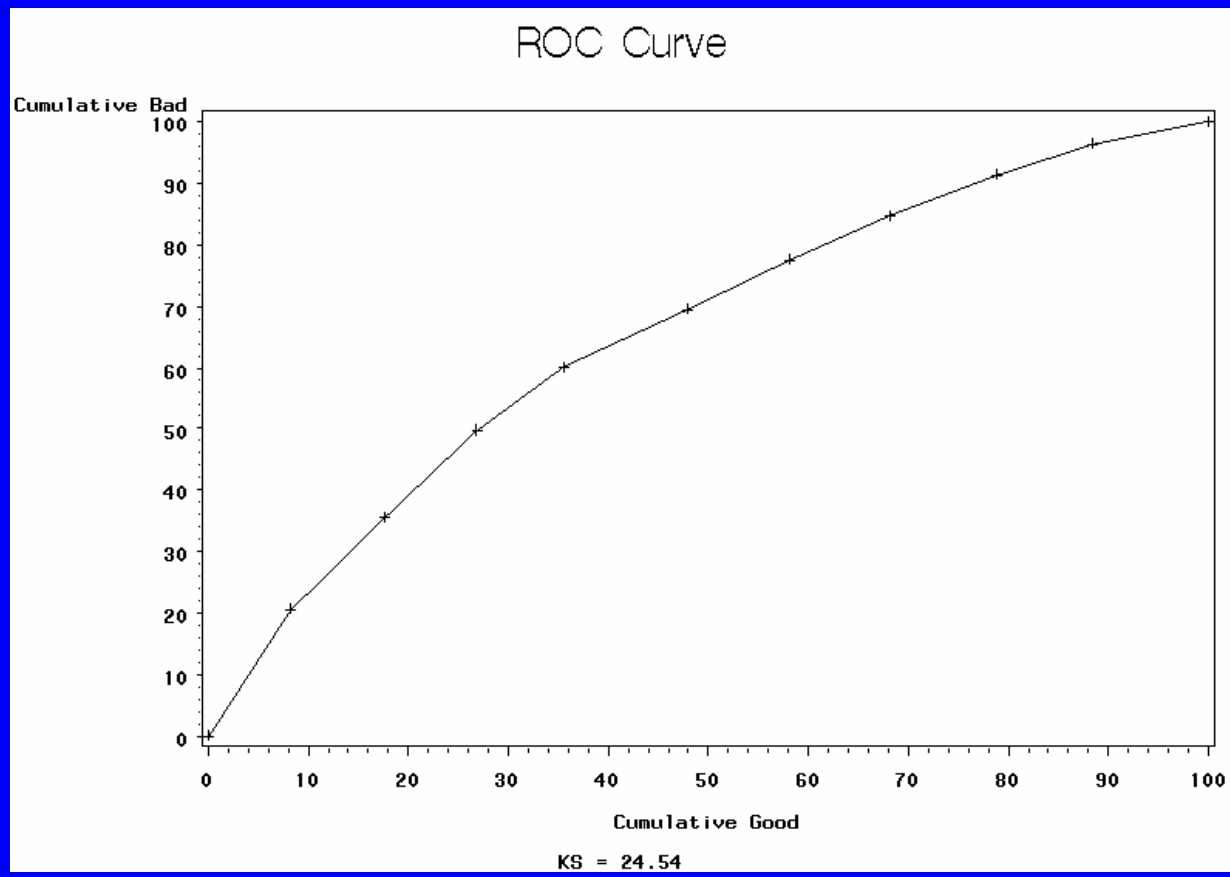
# MULTILABEL OPTION: Gains Charts

From: Charles Partridge (HASUG 8/15/2002), [www.sconsig.com](http://www.sconsig.com),  
TIP00341

```
/** Notice the Repeated USES of Values on left side of Equal Signs
**/
Proc Format;
Value $REGNFMT (multilabel) /** NEW OPTION for Proc Format **/
CT, MA, RI, VT, NH, ME = "1 New England" /** New England **/
CT = "1 CT" MA = "1 MA" RI = "1 RI" VT = "1 VT" NH = "1 NH"
ME = "1 ME"
NY, NJ, PA, DE, MD, DC, VA, WV = "2 Mid Atlantic"/**MidAltantic**/
NY = "2 NY" NJ = "2 NJ" PA = "2 PA" DE = "2 DE" MD = "2 MD"
DC = "2 DC" VA = "2 VA" WV = "2 WV"
...

/** Calculate Statistics using Proc Summary **/
Proc Summary data=Select nway missing completetypes;
class state/MLF preloadfmt;
/**
NOTICE new option COMPLETETYPES
NOTICE new option MLF (Multilabel Format)
STATE grouped multiple times based upon the MLF and format $REGNFMT
NOTICE new option PRELOADFMT for creating entries that don't exist
**/
...
```

# MULTILABEL OPTION: Gains Charts



## MULTILABEL OPTION: ROC curve and KS

```
proc format;
  value scr_rank (multilabel) 0      = ' 1'  0 - 1 = ' 2'  0 - 2 = ' 3'
                             0 - 3 = ' 4'  0 - 4 = ' 5'  0 - 5 = ' 6'
                             0 - 6 = ' 7'  0 - 7 = ' 8'  0 - 8 = ' 9' 0 - 9 = '10'
  ;
  picture pct (round) low-high = '009.99%'
  ;
  value bad  0 = 'GOOD'
            1 = 'BAD'
  ;
run;

proc rank data=book.scores groups=10 ties=low out=ranks;
  var score;
  ranks score_rank;
run;
```

# MULTILABEL OPTION: Gains Charts

```
ods output table=report;
proc tabulate data=ranks noseps formchar=' ' ;
  class bad score_rank / mlf;
  var score;
  format score_rank scr_rank. bad bad.;
  keylabel sum=' ' mean=' ' pctsum=' ' pctn=' ' max=' ';
  table (score_rank)
        /
        (bad)*(n*f=comma6. pctn<score_rank>='%'*f=pct.)
        score='Score Cut Off'*(max)*f=5.
        /rts=20 misstext=' ' row=float;
run;
ods output close;
```

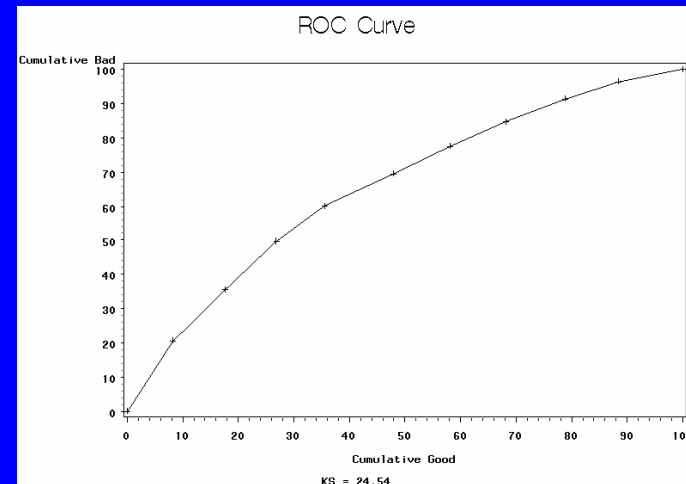
Rank for Variable score	bad				Score Cut Off
	BAD		GOOD		
	N	%	N	%	
1	416	20.59%	658	8.25%	175
2	717	35.50%	1,405	17.61%	182
3	1,003	49.65%	2,140	26.82%	187
4	1,213	60.05%	2,834	35.51%	191
5	1,404	69.50%	3,821	47.88%	196
6	1,566	77.52%	4,640	58.15%	200
7	1,710	84.65%	5,441	68.18%	204
8	1,845	91.34%	6,288	78.80%	209
9	1,946	96.34%	7,057	88.43%	215
10	2,020	100.00%	7,980	100.00%	259

# MULTILABEL OPTION: Gains Charts

```
data report2;
  set report;
  if score_rank=' ' or bad=' ' then delete;

data plot;
  set report2 end=eof;
  if _n_=1 then do;      c_bad=0;      c_good=0;      ks=0;      output;      end;
  by score_rank;
  retain c_bad c_good ks;
  if first.score_rank then c_bad=PctN_10;
  if last.score_rank then do;
    c_good = PctN_10;
    KS = KS <> abs(c_bad-c_good);
    output;
  end;
  if eof then call symputx('KS',put(KS,6.2));
run;

symbol1 i=join value=plus;
axis1 order=(0 to 100 by 10);
proc gplot data=plot;
  plot c_bad*c_good/haxis=axis1 vaxis=axis1;
  label c_bad = 'Cumulative Bad'
        c_good = 'Cumulative Good'
  ;
  footnote "KS = &KS";
  title ROC Curve;
run;
```



## Working with FORMAT Catalogs

- Often a FORMAT is stored in the SAS work space. There are occasions when you wish to permanently store the FORMAT. Reasons for doing so are:
  - If the FORMAT/INFORMAT is large, storing the FORMAT will make them easier to manage.
- Stored FORMATS do not have to be re-specified in the source code.
- You can share FORMATS with colleagues just by letting them know the location of the FORMAT LIBRARY.
- The simplest way to store a FORMAT is to store under a LIBNAME named LIBRARY. By default SAS will look for FORMATS in WORK space, followed by LIBRARY, and then followed by other LIBNAMES specified by SAS system OPTION FMTSEARCH.

```
libname library 'c:\temp';

proc format library=library;
  value $sex 'M' = Male
            'F' = Female
            ;
run;
```

## Working with FORMAT Catalogs

- To use the saved FORMATS in another code, include the libname library at the top of your source code.
- The FORMAT catalog is stored under the CATALOG LIBRARY.FORMATS. If you look under the directory you saved your formats, you will see a the stored format catalog as formats.sas7bcat.
- You can add as many FORMATS, INFORMATS, and PICTURES as you want. They will all be stored in the entry formats.sas7bcat in the directory you specified with the LIBNAME statement.
- You can store the FORMATS under a LIBNAME other than LIBRARY, but the LIBNAME must be specified in SAS system OPTION FMTSEARCH. If one stores FORMATS under a LIBNAME called FMLIB, then apply the following OPTION statement:
  - `OPTIONS FMTSEARCH = (FMLIB);`
- The FMTSEARCH OPTION can even list multiple LIBNAMES, indicating the order in which to search for FORMATS. Example is:
  - `OPTIONS FMTSEARCH = (FMTEST FMLIB LIBRARY WORK);`

# Viewing Stored FORMATS

- Use the FMMLIB option

```
proc format library=library.fmttest fmmlib;  
run;
```

```
„fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff†  
,          FORMAT NAME: $SX          LENGTH:      6  NUMBER OF VALUES:    3      ,  
,  MIN LENGTH:    1  MAX LENGTH:   40  DEFAULT LENGTH  6  FUZZ:          0      ,  
‡fffffffffffffffffff...fffffffffffffffffff...fffffffffffffffffffffffffffffffffff%  
,START          ,END          ,LABEL (VER. V7|V8  15DEC2003:15:58:13),  
‡fffffffffffffffffff^fffffffffffffffffff^fffffffffffffffffffffffffffffffffff%  
,F          ,F          ,FEMALE          ,  
,M          ,M          ,MALE          ,  
,**OTHER**          ,**OTHER**          ,OTHER          ,  
Šfffffffffffffffffff<fffffffffffffffffff<fffffffffffffffffffffffffffffffffff&
```

# Viewing and Modifying FORMAT Catalogs

```
proc catalog c=library.fmtest;
  contents;
run;
```

Contents of Catalog LIBRARY.FMTEST

#	Name	Type	Create Date	Modified Date	Description
1	SX	FORMATC	29DEC2003:12:48:34	29DEC2003:12:48:34	

```
options ls=137;
proc catalog c=library.fmtest;
  contents stat;
run;
```

Contents of Catalog LIBRARY.FMTEST

#	Name	Type	Create Date	Modified Date	Description	Page Size	Block Size	Num of Blocks	Last Block Bytes	Last Block Size	Pages
1	SX	FORMATC	29DEC2003:12:48:34	29DEC2003:12:48:34		4096	4096	1	214	255	1

- To add a description, just use the following PROC CATALOG code:

```
proc catalog c=library.fmtest;
  modify sx.formatc (description='Sex Code Format');
  contents;
run;
```

Contents of Catalog LIBRARY.FMTEST

#	Name	Type	Create Date	Modified Date	Description
1	SX	FORMATC	29DEC2003:12:48:34	29DEC2003:13:12:57	Sex Code Format