Paper #1495-2014

Jazz it up a Little with Formats

Brian Bee, The Knowledge Warehouse Ltd

Abstract

Formats are an often under-valued tool in the SAS® toolbox. They can be used in just about all domains to improve the readability of a report, or be used as a look-up table to re-code your data.

SAS, out of the box, includes a multitude of ready-defined formats that can be applied without modification to address most recode/re-display requirements; and if that's not enough, there is also a FORMAT procedure for defining your own custom formats.

This paper will look at using some of the SAS-supplied formats in some innovative ways, but will primarily focus on the techniques we can apply in creating our own custom formats.

Introduction

The simplest way to improve the appearance of a report is to apply formats to the variables. The important thing to remember about applying a format is that the data value stored in the data set does not change. A format modifies the APPEARANCE of a variables VALUE.

SAS-supplied formats

The first group we should look at is the SAS-supplied formats. There are a multitude of them , but there is a small group that gets more use than most.

The DOLLAR format is the first of this group – used extensively across all business and industry sectors. Its purpose is to print a money amount with a dollar sign on the front, and commas separating the millions, thousands etc.

Like many formats, modifying the overall width can produce quite different results

If my variable contains this value…	and I apply this format…	this is the result
12345.67	Dollar12.2	`  $12,345.67`
12345.67	Dollar10.2	`$12,345.67`
12345.67	Dollar9.2	`$12345.67`
12345.67	Dollar8.2	`12345.67`
12345.67	Dollar7.2	`12345.7`
12345.67	Dollar5.2	`12346`
12345.67	Dollar4.2	`12E3`

Table1: The DOLLAR format

Most of the date and datetime formats will display a similar behavior when the overall width is modified, giving different representations of dates and times eg 2- vs 4-digit years, full vs abbreviated monthname etc.

There are also several lesser known formats that can be brought into more common usage. The Z. format will print a numeric value without suppressing leading zeros – which is the default with the DOLLAR and COMMA formats. This is used often for printing account numbers or id numbers of various sorts.

The ROMAN format will print a numeric value in roman numerals. An interesting way of printing page numbers! Note it does not print decimals.

One that is used less frequently these days is the WORDS format. This prints a numeric value in words rather than digits. A common usage was writing cheques!

If my variable contains this value…..	**And I apply this format….**	**this is the result**
1234.5	**Z8.1**	`001234.5`
1234.5	**Roman12.**	`MCCXXXIV`
1234.5	**Words60.**	`one thousand two hundred thirty-four and fifty hundredths`

Table 2: Selected formats

A new date format DTDATE has proved to be very useful. This will print just the date portion of a date-time value. Prior to the introduction of this format, the DATEPART function would need to be used to separate off the date, and then format the result.

If my variable contains this value…..	**And I apply this format….**	**this is the result**
`'23jul1999:12:34:56'dt`	**Dtdate5.**	**23JUL**
`'23jul1999:12:34:56'dt`	**Dtdate7.**	**23JUL99**
`'23jul1999:12:34:56'dt`	**Dtdate9.**	**23JUL1999**
`'23jul1999:12:34:56'dt`	**Dtdate.**	**23JUL99**

Table 3: the DTDATE format

The FORMAT procedure

When we are faced with the situation of not being able to find a format that does the job exactly as we want, then we have the format procedure to help us out. There are three important statements that we can use in the format procedure : the value, invalue and picture statements.

VALUE statement.

This one is reasonably well known, but not fully utilized. The simplest form is a series of single values, formatted to labels. Eg ‘F’=’Female’   ‘M’=’Male’   or 1=’One’   2=’More’.

Next we can specify ranges of values eg low-10=’Young’   11-13=’Mid’   14-high=’Older’. LOW and HIGH are special values we can use to represent ‘The lowest possible value’ and ‘The highest possible value’ respectively.

We can also have lists of values eg 2,3,4,5,6=’Weekday’   1,7=’Weekend’.

And lastly we can have multiple formats!   Eg .=’missing’ other=[ages.].

For example:

```
proc format;
value $gender       'M'='Male'
                    'F'='Female';

value ages       low-10='Young'
                  11-13='Mid'
                 14-high='Older';

value weeks  2,3,4,5,6='Weekday'
                    1,7='Weekend';

value multi           .='Missing'
                  other=[ages5.];
run;
```

In the last value statement, a missing value returns the word ‘Missing’ and all other values have the AGES5. format applied.

If my variable contains this value…..	And I apply this format….	this is the result
F	$gender	Female
27	Ages.	Older
4	Weeks.	Weekday
12	Multi.	Mid
.	Multi.	Missing

Table 4: Using the VALUE statement

INVALUE statement

This one is invaluable when we are validating data – either from a dataset or a raw data file. It allows us to preserve values if they are valid, or change them if they are not.

For example:

```
proc format;
invalue group      1-100=_same_
                   other=.;

invalue num    'A' - 'M' = 1
               'N' - 'Z' = 2
               1 - 100   = 3;

invalue nume    1-10=_same_
                  99=.
               other=_error_;
run;
```

In the last invalue statement , values 1-10 remain the same, 99 returns a missing value, and all other values cause the automatic variable _ERROR_ to be set to 1. The invalue statement is effectively creating informats, and they are applied in the usual manner

eg input smallnums nume. etc;

If my variable contains this value…..	And I apply this informat….	this is the result
77	Group.	77
326	Group.	.
J	Num.	1
12	Num.	3
99	Nume.	.
45	Nume.	_ERROR_ set to 1

Table 5: Using the INVALUE statement

PICTURE Statement

This is where we get maximum utility from PROC FORMAT. If you are required to print a numeric value a certain way, and you just can't find a SAS-supplied format to do it, and value and invalue statements don't help, then the PICTURE statement comes to the rescue. The PICTURE statement allows us to define a print template for printing numeric values, including text. A common requirement here is where we want a report with a column heading 'Total (in Millions)' and the values taken from our dataset displayed simply as, say, 16, representing 16million. So using the PICTURE statement we specify how we want the number represented, and include a multiplier of .000001 in this example. The value from the dataset is multiplied by the multiplier, and then the print template is applied to the result. A prefix can also be included, for example a $ sign, and there is a NOEDIT option where the label includes numeric values which should not be considered print directives. Another use is where there is a requirement to print a date in an unusual way that is not covered by any of the date/datetime formats. There is a series of directives that can be used to represent all the different elements of dates and times. For example the string %d(%A)*%B*%Y(%H:%M:%S) would cause a datetime value to be printed like 01(Wednesday)*January*2012(01:11:03).

When specifying a print template the directives are used to identify what characters should be returned.

0 - when leading zeros are included in the value, suppress them

1-9 - when leading zeros are included in the value, print them.

EG a picture of ‘0009.99’ will suppress leading zeroes in the first 3 positions and will print a value for the 4th digit even if it is a zero. Directives 1 thru 9 can be used. Common usage is to use 0’s and 9’s.

For example:

```
proc format;
picture mills low-high='0009.99M' (mult=.000001);

picture millsp   other='0009.99M' (mult=.000001 prefix='$');

picture mydate   other='0%d(%A)*%B*%Y(%H:%M:%S)' (datatype=datetime);

picture big    1-1000 = '0009'
             1000<-high = '>1000km' (noedit);
run;
```

If my variable contains this value…..	**And I apply this format….**	**this is the result**
23945123	**Mills.**	**23.95M**
614339	**Mills.**	**0.61M**
23945123	**Millsp**	**$23.95**
05jun2001:12:34:56	**Mydate.**	05(Tuesday)*June*2001(12:34:56)
633	**Big.**	**633**
1874	**Big.**	**>1000km**

Table 6: Using the PICTURE statement

CNTLIN option

The CNTLIN option allows PROC FORMAT to read data from a dataset and use it to build a format. This becomes very useful when the information in the format changes on a regular basis – it saves you having to modify a program each time it has to be updated.The dataset being read must conform to certain rules in that it must contain the following variables…

- A character variable called FMTNAME that contains the name that should be assigned to the format
- A variable called START that contains the range
- A character variable called LABEL that contains the label for the range

For example:

A dataset to create the GENDER format illustrated earlier should look like..

FMTNAME	START	LABEL
$GENDER	F	Female
$GENDER	M	Male

Table 7: Single values in the format

A dataset to create the num format illustrated above should include an extra column named END and look like

FMTNAME	START	END	LABEL
NUMS	A	M	1
NUMS	N	Z	2
NUMS	1	100	3

Table 8: Ranges in the format

The dataset can also be created from an existing dataset.

For example

```
Data control;
      Set incoming;
      retain fmtname 'nums';
      Rename gender=start description=label;
      *an assignment statement or an expression can be used instead;
      run;
```

The dataset can now be read into the FORMAT procedure thus…

```
Proc format cntlin=control;
      run;
```

Conditional highlighting

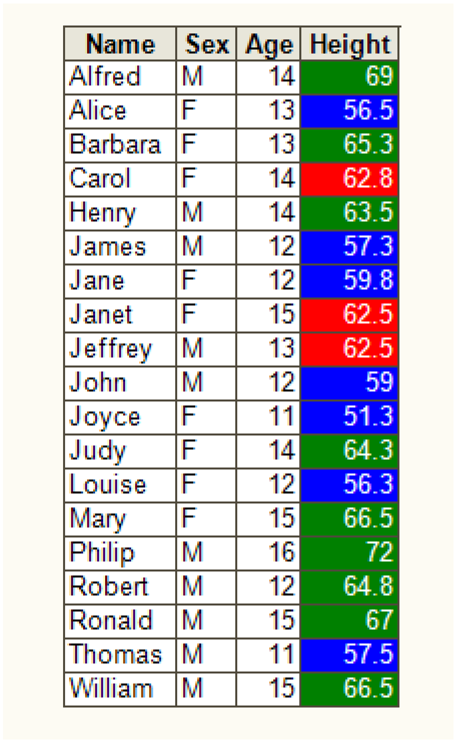
Conditional highlighting – otherwise known as Traffic lighting - involves using colour to highlight data values. This is something that has in the past often been accomplished by exporting the data out to other reporting software and then going through the report manually modifying the appearance cell by cell. So step one of the SAS solution is to create a format that defines the ranges or values you want highlighted, and assign a label that specifies a colour to each range of values.

```
Proc format;
      Value ranges 0 - 62 = 'Blue'
                  62< - 63 = 'Red'
                    Other  = 'Green';
Run;
```

Step two is to generate your report. PROC REPORT lends itself very nicely to this application and is very simple to modify. Once the basic report is created and fine tuned, you can modify the code to implement the traffic-lighting. The changes that need to be made involving adding a style statement to the line that defines the variable you want to apply your traffic-lighting to.

```
proc report data=sashelp.class nowd;
      column name sex age height;
     define name / group 'Name' missing;
     compute name;
          if name ne ' ' then hold1=name;
          if name eq ' ' then name=hold1;
     endcomp;
     define sex / group 'Sex' missing;
     compute sex;
          if sex ne ' ' then hold2=sex;
          if sex eq ' ' then sex=hold2;
     endcomp;
     define age / analysis SUM 'Age' missing;
     define height / analysis SUM 'Height' missing style(column)
           = [background=ranges. foreground=white];
Run;
```

The key to the whole process is expressing the ranges FORMAT for the background, and white for the foreground to give white text. The obvious alternative is to use the format for the foreground and leave the background unchanged. This will return coloured text on a white background.



Name	Sex	Age	Height
Alfred	M	14	69
Alice	F	13	56.5
Barbara	F	13	65.3
Carol	F	14	62.8
Henry	M	14	63.5
James	M	12	57.3
Jane	F	12	59.8
Janet	F	15	62.5
Jeffrey	M	13	62.5
John	M	12	59
Joyce	F	11	51.3
Judy	F	14	64.3
Louise	F	12	56.3
Mary	F	15	66.5
Philip	M	16	72
Robert	M	12	64.8
Ronald	M	15	67
Thomas	M	11	57.5
William	M	15	66.5

Traffic-lighting is a far simpler process than most people imagine. For a fuller description of this subject, see Paper 273-2009 Traffic-Lighting Your Reports the Easy Way with PROC REPORT and ODS. by Andrew Karp. This excellent paper was presented at SAS Global Forum 2009, and explores several different options for traffic-lighting.

Summary

In using formats you are restricted only by your imagination. You can take any value and apply a format to it where the format is re-presenting the value as absolutely anything you can type on your keyboard (and more). It is generally a very simple method of ‘jazzing up’ your report to take it from ordinary to a work of art!

Author contact

Brian Bee

The Knowledge Warehouse Limited

P.O. Box 10-541

The Terrace

Wellington

New Zealand

+64 21 630075

Brian.Bee@knoware.co.nz

Copyright