

Dynamically Setting Decimal Precision Using PUTN

Brandon Welch, Rho[®], Inc., Chapel Hill, NC

ABSTRACT

In the pharmaceutical industry, there are often rules for displaying the number of decimal places in a report. This is particularly important when reporting laboratory results in a table. For example, for a particular laboratory test, if the laboratory result in the data is carried to the tenths place, the displayed mean will be reported at the hundredths place. The common rule is displaying the mean result one place higher than what is in the data. There are similar rules for other statistics such as the median, standard deviation, minimum, and maximum. This paper will illustrate a simple technique using the PUTN function to dynamically set the decimal precision. The techniques presented offer a good overview of basic SAS[®] functions that will educate programmers at all levels.

INTRODUCTION

Summarizing statistics on a continuous variable is a common task in clinical trials computing. Below is a popular way to display these statistics:

Parameter (unit) Statistic	Treatment A N = XXX	Treatment B N = XXX	Placebo N = XXX
Temperature (°F)			
n	XXX	XXX	XXX
Mean	XXX.XX	XXX.XX	XXX.XX
SD	XXX.XX	XXX.XX	XXX.XX
Median	XXX.XX	XXX.XX	XXX.XX
Range (Min, Max)	(XXX, XXX)	(XXX, XXX)	(XXX, XXX)

Table 1. Vital signs shell

At a glance, the number of decimal places appears to be fixed across the different statistics within the shell. Programmers need to be attentive to the decimals for each parameter. Temperature is usually presented as XXX.X (98.6°F for example rather than XX F). Therefore it makes sense to display these statistics according to a set of rules. The remainder of this paper will only concentrate on the statistics: mean, standard deviation, median and min/max – clearly sample size (*n*) is always presented as a whole number.

RULES FOR SETTING PRECISION

The common rule is this:

1. Mean and median: the number of decimals is displayed as one more than what is shown in the data.
2. Standard deviation is displayed as two more than what is shown in the data.
3. Min/max are displayed exactly as what is shown in the data (zero places).

For example:

TEMP	TRT01PN
97.8	A
101.3	A
98.5	B
99.4	B
97.2	Placebo

Table 2. Dummy temperature data

The statistics:

Parameter (unit)	Treatment A	Treatment B	Placebo
Statistic	N = 2	N = 2	N = 1
Temperature (°F)			
n	2	2	1
Mean	99.55	98.95	97.20
SD	2.475	0.636	.
Median	99.55	98.95	97.20
Range (Min, Max)	(97.8, 101.3)	(98.5, 99.4)	(97.2, 97.2)

Table 3: Statistics on dummy temperature data

The dummy data on temperature goes out to one decimal place. So the mean and median are +1 decimal places equaling two decimal places (99.55 for example), standard deviation is +2 equaling three decimal places (2.475 for example) and min/max are +0 equaling one decimal place (97.8 and 101.3). Note for assigning precision, we can formulate a method with three rules – +0, +1 and +2 scenarios.

SAS CODE

The first step is to identify the number of decimal places shown in the data and obtain the statistics needed for reporting. Writing a program to return the precision from the data is ideal, but that is beyond the scope of this paper.

```
DATA ex1;
input paramcd $ aval trt01pn $;
cards;
TEMP 97.8 A
TEMP 101.3 A
TEMP 98.5 B
TEMP 99.4 B
TEMP 97.2 Placebo
PULSE 83 B
PULSE 85 Placebo
PULSE 88 A
PULSE 89 B
PULSE 80 Placebo
;
RUN;

PROC MEANS noprint nway;
class paramcd trt01pn;
var aval;
output out = stats
      n      = n
      mean   = mean
      std    = std
      median = median
      min    = min
      max    = max;
RUN;
```

Here are the raw values as output from PROC MEANS – assume we have merged on the precision values calculated a priori:

paramcd	trt01pn	n	mean	std	median	min	max	prec
PULSE	A	1	88.00	.	88.00	88.0	88.0	0
PULSE	B	2	86.00	4.24264	86.00	83.0	89.0	0
PULSE	Placebo	2	82.50	3.53553	82.50	80.0	85.0	0
TEMP	A	2	99.55	2.47487	99.55	97.8	101.3	1

paramcd	trt01pn	n	mean	std	median	min	max	prec
TEMP	B	2	98.95	0.63640	98.95	98.5	99.4	1
TEMP	Placebo	1	97.20	.	97.20	97.2	97.2	1

Table 4: Output on dummy vital signs data

The obvious solution is the use of IF/THEN statements. This is a perfectly valid technique and works as follows:

```
DATA precl;
  set stats;

  n_      = put(n,3.);

  if prec = 0 then do;
    mean_ = put(mean,8.1);                      /*+1*/
    if n > 0 then stddv_ = put(std,8.2);          /*+2*/
    med_   = put(median,8.1);                    /*+1*/
    mnmx_ = '('||strip(put(min,8.))||', '||strip(put(max,8.))||')'; /*+0*/
  end;

  else if prec = 1 then do;
    mean_ = put(mean,8.2);
    if n > 0 then stddv_ = put(std,8.3);
    med_   = put(median,8.2);
    mnmx_ = '('||strip(put(min,8.1))||', '||strip(put(max,8.1))||')';
  end;

RUN;
```

Note these data have a maximum precision of one. If the precision exceeded one, additional IF/THEN statements are needed. In clinical trials, we often analyze laboratory values. These values usually have a wider variance of precision values. Using PUTN, you can dynamically account for any precision value.

USING PUTN

The first step is to create three variables for each +1, +2 and +0 scenario.

```
DATA precl;
  set stats;

  p0 = put(sum(prec,0),8.);
  p1 = put(sum(prec,1),8.);
  p2 = put(sum(prec,2),8.);

RUN;
```

paramcd	trt01pn	n	mean	std	median	min	max	prec	p0	p1	p2
PULSE	A	1	88.00	.	88.00	88.0	88.0	0	0	1	2
PULSE	B	2	86.00	4.24264	86.00	83.0	89.0	0	0	1	2
PULSE	Placebo	2	82.50	3.53553	82.50	80.0	85.0	0	0	1	2
TEMP	A	2	99.55	2.47487	99.55	97.8	101.3	1	1	2	3
TEMP	B	2	98.95	0.63640	98.95	98.5	99.4	1	1	2	3
TEMP	Placebo	1	97.20	.	97.20	97.2	97.2	1	1	2	3

Table 5: Output on dummy vital signs data with initial variables

Now we have the ingredients to use the PUTN function. The PUTN function allows programmers to pass a variable in place of a format. The next step is to create a variable that has a format value. – e.g., `var = 8.1`. Adding a little more code to our data step:

```
DATA prec1;
  length prec0-prec2 $8;
  set stats;

  p0 = put(sum(prec,0),8.);
  p1 = put(sum(prec,1),8.);
  p2 = put(sum(prec,2),8.);

  prec0 = cats("8.",p0);
  prec1 = cats("8.",p1);
  prec2 = cats("8.",p2);

RUN;
```

Now we have:

paramcd	trt01pn	n	mean	std	median	min	max	prec	p0	p1	p2	prec0	prec1	prec2
PULSE	A	1	88.00	.	88.00	88.0	88.0	0	0	1	2	8.0	8.1	8.2
PULSE	B	2	86.00	4.24264	86.00	83.0	89.0	0	0	1	2	8.0	8.1	8.2
PULSE	Placebo	2	82.50	3.53553	82.50	80.0	85.0	0	0	1	2	8.0	8.1	8.2
TEMP	A	2	99.55	2.47487	99.55	97.8	101.3	1	1	2	3	8.1	8.2	8.3
TEMP	B	2	98.95	0.63640	98.95	98.5	99.4	1	1	2	3	8.1	8.2	8.3
TEMP	Placebo	1	97.20	.	97.20	97.2	97.2	1	1	2	3	8.1	8.2	8.3

Table 6: Output on dummy vital signs data with precision variables

Finally we are in position to use the PUTN function.

```
DATA prec1;
  length prec0-prec2 $8;
  set stats;

  p0 = put(sum(prec,0),8.);
  p1 = put(sum(prec,1),8.);
  p2 = put(sum(prec,2),8.);

  prec0 = cats("8.",p0);
  prec1 = cats("8.",p1);
  prec2 = cats("8.",p2);

  n_ = put(n,3.);
  mean_ = putn(mean,prec1); /*+1*/
  if n > 0 then stddv_ = putn(std,prec2); /*+2*/
  med_ = putn(median,prec1); /*+1*/
  minmx_ = '('||strip(putn(min,prec0))||', '||strip(putn(max,prec0))||')'; /*+0*/

RUN;
```

We can also create the variables in arrays – an even more dynamic approach:

```
DATA prec1;
  length prc0-prc2 $8;
  set stats;

  array p{3} $ p0-p2;
  array prc{3} prc0-prc2;
  do i = 1 to 3;
```

```

p{i} = put(sum(prec,i - 1),8.);
prc{i} = cats("8.",p{i});

n_      = put(n,3.);
mean_   = putn(mean,prc1);
if n > 0 then stddv_ = putn(std, prc2);
med_    = putn(median, prc1);
mnmx_   = '('||strip(putn(min,prc0))||', '||strip(putn(max,prc0))||')';

end;

RUN;

```

paramcd	trt01pn	prec	n_	mean_	stdv_	med_	mnmx_
PULSE	A	0	1	88.0	.	88.0	(88, 88)
PULSE	B	0	2	86.0	4.24	86.0	(83, 89)
PULSE	Placebo	0	2	82.5	3.54	82.5	(80, 85)
TEMP	A	1	2	99.55	2.475	99.55	(97.8, 101.3)
TEMP	B	1	2	98.95	0.636	98.95	(98.5, 99.4)
TEMP	Placebo	1	1	97.20	.	97.20	(97.2, 97.2)

Table 7: Final output on dummy vital signs data with precision variables

A simple transpose creates a summary table:

paramcd	_NAME_	A	B	Placebo
PULSE	n_	1	2	2
PULSE	mean_	88.0	86.0	82.5
PULSE	stdv_	.	4.24	3.54
PULSE	med_	88.0	86.0	82.5
PULSE	mnmx_	(88, 88)	(83, 89)	(80, 85)
TEMP	n_	2	2	1
TEMP	mean_	99.55	98.95	97.20
TEMP	stdv_	2.475	0.636	.
TEMP	med_	99.55	98.95	97.20
TEMP	mnmx_	(97.8, 101.3)	(98.5, 99.4)	(97.2, 97.2)

Table 8: Final transposed output on dummy vital signs data with precision variables

CONCLUSION

The PUTN function offers a simple solution to assigning precision. Without using this function, programmers are left with other techniques (often IF/THEN statements), which arrive at the same results. As the precision values vary (as we see in laboratory data), more IF/THEN statements are required. However adding more code clutters the program and copying/pasting more IF/THEN statements is prone to error. Using PUTN reduces the amount of code and relies on the data to dictate the number of decimal places.

ACKNOWLEDGMENTS

Eva J. Welch
John Ingersoll
Steve Noga

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Brandon Welch
Rho, Inc
6330 Quadrangle Dr., Ste. 500
Chapel Hill, NC 27517
Phone: 919-595-6592
Fax: 919-408-0999
Email: Brandon_Welch@rhoworld.com
Web: www.rhoworld.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies