Umsetzen des signifikanten Rundens in SAS

Matthias Lehrkamp
PAREXEL International
Spandauer Damm 130
14050 Berlin
matthias.lehrkamp@parexel.com

Zusammenfassung

Während der Darstellung von Ergebnissen stellt sich immer wieder die Frage, wie ein Wert gerundet werden soll, damit eine bestmögliche und vor allem „richtige" Darstellung erzielt wird. Die Rundungsfunktion in SAS bietet die Möglichkeit, anhand eines multiplen Vielfachen zu runden. Dabei werden in der Praxis meistens Zahlen ausgewählt, die auf bestimmte Anzahlen der Nachkommastellen runden. Ein einfacher Weg die Rundungsstelle abhängig von der jeweiligen Zahl zu machen, ist die Rundung auf signifikanten Stellen, umgangssprachlich auch „bedeutungstragende Stellen" genannt. In dieser Publikation werden neben Argumenten für das signifikante Runden zwei Methoden aufgezeigt, mit der die signifikante Rundung in SAS umgesetzt werden kann und was dabei zu beachten ist.

Schlüsselwörter: signifikante Rundung, Runden, SAS-Funktion, PROC FCMP, SAS-Makro

1 Definition und Beispiele

Die signifikante Rundung kann bei allen Statistiken und Auswertungen angewendet werden. Durch die dynamische Anpassung der Kommastellen an die Größe des Wertes, eignet sich die Rundungsmethode besonders bei berechneten Werten oder Messreihen, die mehrere Zehnerpotenzen beschreiten. Um zu verstehen, wie die signifikante Rundung funktioniert, muss zunächst der Begriff signifikante Stellen näher betrachtet werden. Eine Definition der signifikanten Stellen findet sich im Folgenden aus der DIN-Norm 1333, welche die Normen für Zahlenangaben definiert [1].

"*Alle Stellen eines Zahlensymbols des Zehner (b-)Systems von der ersten von Null verschiedenen Stelle bis zur Rundungsstelle.*"

In Tabelle 1 werden drei Zahlenbeispiele betrachtet, wobei die erste von Null verschiedene Zahl mit N und die Rundungsstelle mit R gekennzeichnet wird.

Tabelle 1: Beispiele für Umfang der Erde auf verschiedene Anzahl signifikanter Stellen gerundet

Zahl	Beschreibung

`N R`
`1236.67`
= 6 signifikante Stellen
	Die 1 ist die erste von Null verschiedene Zahl und die 7 ist die Rundungsstelle, damit ergeben sich 6 signifikante Stellen.

`N R`
`0.00498`
= 3 signifikante Stellen
	Die 4 ist die erste von Null verschiedene Zahl und die 8 ist die Rundungsstelle, damit ergeben sich 3 signifikante Stellen.

`N RRR`
`123400`
= 4-6 ??? signifikante Stellen
	Die 1 ist die erste von Null verschiedene Zahl, die Rundungsstelle ist nicht eindeutig identifizierbar. Es kann auf die Hunderter-, Zehner- oder Einerstelle gerundet worden sein. Somit ergeben sich 4 bis 6 signifikante Stellen.

Um die genaue Anzahl der signifikanten Stellen zu bestimmen, sollte immer die Messgenauigkeit beziehungsweise die Ursprungszahl(en) betrachtet werden. Ist der Originalwert in der Datenbank hinterlegt, so kann dieser Wert zur Bestimmung der signifikanten Stellen benutzt werden. Das Problem, die Anzahl der signifikanten Stellen zu bestimmen, kann bei dem Runden auf signifikanten Stellen umgangen werden, indem eine feste Anzahl signifikanter Stellen vorgegeben wird. Im folgenden Beispiel wird der Umfang der Erde, auf verschiedene Anzahl signifikanter Stellen gerundet, hier mit 40 075,017 km angegeben.

Tabelle 2: Umfang der Erde auf verschiedene Anzahl signifikanter Stellen gerundet

Signifikante Stellen	Erdumfang
1	40 000 km
2	40 000 km
3	40 100 km
4	40 080 km
5	40 075 km
6	40 075,0 km
7	40 075,02 km
8	40 075,017 km

In der Tabelle ist zu erkennen, dass einige Rundungen das gleiche Ergebnis liefern. Bei der Rundung auf 8 signifikante Stellen wird der Originalwert abgebildet, da dieser gerade 8 signifikante Stellen enthält.

Im Folgenden wird eine Zusammenfassung des Datensatzes SASHELP.CLASS dargestellt. Hierbei wird N als ganze Zahl dargestellt, die Ergebnisse von Min, Max und der Prozente auf 3 signifikante Stellen und alle übrigen Statistiken auf 4 signifikante Stellen gerundet. Zusammengefasst werden die Parameter Größe und Gewicht.

```
Summary of height and weight by sex
Parameter      Statistic  Female      Male
-----------------------------------------------

Height [inch]  n (%)        9 (47.4)   10 (52.6)
               Mean        60.59       63.91
               Std          5.018       4.938
               Min         51.3        57.3
               Median      62.50       64.15
               Max         66.5        72.0

Weight [lbs]   n (%)        9 (47.4)   10 (52.6)
               Mean        90.11      109.0
               Std         19.38       22.73
               Min         50.5        83.0
               Median      90.00      107.3
               Max        113         150
```

Üblicherweise werden in statistischen Analyseplänen immer die Kommastellen separat angegeben, z.B. mit Min und Max mit der gleichen Anzahl wie die Originalwerte, Mean und Median mit einer Dezimalstelle mehr, sowie die Standardabweichung (Std) mit 2 Dezimalstellen mehr. Diese Rundungsgenauigkeit wird durch die Angabe der Rundung auf signifikante Stellen automatisch erzeugt, wie am Beispiel des Parameters Größe (Height) in inch zu erkennen ist. Bei dem Parameter Gewicht (Weight) in lbs werden der Mittelwert und die Standardabweichung jeweils mit 2 Nachkommastellen angegeben. Das ist nachvollziehbar, da die Standardabweichung bereits so groß ist, dass eine dritte Nachkommastelle nicht nötig ist.

Zusammenfassend lässt sich sagen, dass die Anzahl der signifikanten Stellen manchmal nicht eindeutig feststellbar ist. Dieses Problem tritt bei der Rundung auf signifikante Stellen nicht auf, daher ist diese für alle Statistiken und Auswertungen geeignet. Insbesondere ist diese aber sehr nützlich bei berechneten Werten oder Messreihen, die mehrere Zehnerpotenzen beschreiten.

2 Runden auf signifikante Stellen

Funktionen gehören in analytischen Programmen zur Grundausstattung. In SAS werden Funktionen noch viel zu selten benutzt, da diese erst mit der Version 9.2 eingeführt wurden. Um eine Funktion für das signifikante Runden zu erstellen, wird zunächst der Algorithmus festgelegt. Im Folgenden wird die Zahl 1236.67 benutzt und diese soll auf 5 signifikante Stellen gerundet werden.

2.1 Die 10er Potenzen zwischen denen der Wert liegt

Die signifikanten Stellen beginnen mit der ersten von Null verschiedenen Zahl. An welcher Stelle diese Zahl liegt, wird durch die 10er Potenz angegeben. Der Wert **x=1236.67** liegt zwischen 1000 und 10000, also zwischen $10^3 <= x < 10^4$. Die 10^4 ist aus dem In-

tervall ausgeschlossen, da die 10000 bereits 5 Vorkommastellen besitzt. Um nun die 10er Potenz zu ermitteln, wird der Zehnerlogarithmus verwendet, der gerade die Umkehrfunktion der Zehnerpotenz darstellt. Da der Logarithmus für negative Zahlen nicht definiert ist, wird immer der Absolutbetrag verwendet.

LOG10(ABS(x))=3.092
FLOOR(LOG10(ABS(x)))=3

Somit liegt die Zahl x im 1000er Bereich (10^3).

2.2 Rundungszahl bestimmen

Die Rundungszahl wird bestimmt, indem die Anzahl der signifikanten Stellen **s=5** von der 10er Potenz abgezogen wird. Da die erste Stelle mitgerechnet wird, muss noch eine 1 hinzuaddiert werden.

FLOOR(LOG10(ABS(x))) + 1 – **s** = 3 + 1 - **5** = -1 = r

Daraus ergibt sich die Rundungszahl $10^{-1} = 0.1$

2.3 Zahl runden

Um die Zahl zu runden, greifen wir auf die bereits in SAS vorhandene Rundungsfunktion ROUND() zu. Diese Funktion wird mit dem zu rundenden Wert und einem Multiplikator aufgerufen.

ROUND(x,10^r) = ROUND(1236.67,0.1) = 1236.7 = x_round

Damit wurde die Zahl x=1236.67 auf 5 signifikante Stellen = 1236.7 gerundet. Bei unterschiedlichen Dezimalstellen muss in SAS bei numerischen Werten das Best Format verwendet werden. Bei diesem Format gehen nachkommende Nullen leider verloren. Um diesen Nachteil auszugleichen, sollte das Ergebnis grundsätzlich als Textformat gespeichert werden. Der numerische Wert muss also in eine Zeichenkette umgewandelt werden.

2.4 Umwandlung als Zeichenkette

Für die Umwandlung in eine Zeichenkette wird ein Format benötigt. Das Zahlenformat für Dezimalzahlen mit einer bestimmten Anzahl Kommastellen wird über das *w.d*-Format abgebildet. Wobei das *w* für die Anzahl der Zeichen steht (Kommazeichen wird mitgezählt) und das d für die Anzahl der Kommastellen. Der maximale Wert für eine Zahl in SAS beträgt 32 Zeichen. Daher kann der Wert fest auf 32 gesetzt werden. Die Anzahl der Dezimalstellen wird dynamisch festgelegt und entspricht dem entgegengesetzten Wert (*-1) von r, der Rundungszahl. Zusammengesetzt erhält man das Format.

CATS("32.",MAX(-1*r,0)) = 32.1 = x_format

Das Maximum wird benötigt, um die Anzahl der Kommastellen auf Null zu beschränken, da negative Werte für die Anzahl der Kommastellen nicht definiert sind. Um das gerundete Ergebnis in eine Zeichenkette umzuwandeln, wird die Funktion PUTN() benutzt. Die Funktion PUT() kann nur mit einem fest angegebenen Format umgehen und daher nicht verwendet werden.

PUTN(x_round, x_format) = “1236.7“

Damit ist die Vorgehensweise vollständig definiert und die Umsetzung als SAS-Funktion kann beginnen. Wichtig ist auch zu beachten, dass der Logarithmus für die 0 nicht definiert ist, daher muss eine zusätzliche Regel festgelegt werden. Diese lautet konkret LOG10(ABS(0))=0, sprich 0 bleibt 0.

3 Umsetzung als SAS-Funktion

Am einfachsten beginnt man, indem die gewünschte Funktionalität in einem DATA-Step erprobt wird. Da der Algorithmus im Abschnitt 2 schon mit einem SAS Code beschrieben wurde, kann dieser direkt übernommen werden.

```
/* Testzahlen */
DATA sr01_numbers;
  DO x= 1236.67, 1000, 450, 100, 50.005, 10, 7.6,
        1, 0.12345, 0.1, 0.01437;
    OUTPUT;
  END;
RUN;

/* DATA-Step calculation */
DATA sr02_hand;
  SET sr01_numbers;
  * set significant digits;
  s= 3;
  * determine power of number rounding;
  r= FLOOR(LOG10(ABS(x))) + 1 - s;
  * round number;
  x_round= ROUND(x,10**r);
  * set character format;
  x_format= CATS("32.",MAX(-1*r,0));
  * convert to character format;
  x_char= PUTN(x_round, x_format);
RUN;
```

Wenn das gewünschte Ergebnis erreicht ist, kann der Programmcode aus dem DATA-Step direkt in der Funktion verwendet werden.

Eine Funktion wird über die Prozedur FCMP erstellt. Alle Funktionen werden in einem Datensatz gespeichert. Dieser Datensatz muss über die Option OUTLIB festgelegt werden. In jedem Fall muss zusätzlich ein Paketname angegeben wird, unter dem alle in der Prozedur erstellten Funktionen abgelegt werden.

```
PROC FCMP OUTLIB=<dataset-name>.<package-name>;
```

Funktionen können immer nur einen Rückgabewert vom Typ numerisch oder alphanumerisch besitzen. Daher benötigen wir zwei Funktionen. Die erste Funktion gibt den numerischen Wert zurück, falls mit diesem Wert noch weitere Berechnungen durchgeführt werden müssen. Um das Ergebnis als Zeichenkette auszugeben, wird eine zweite Funktion benötigt. Die Funktion selbst wird mit dem Schlüsselwort FUNCTION eingeleitet und mit ENDSUB beendet. Im FUNCTION-Statement werden der Name der Funktion, die benötigten Parameter und der Ausgabetyp festgelegt.

```
PROC FCMP OUTLIB=work.funcs.myFuncs;
  /* numeric output function */
  FUNCTION sigRoundN(x,s);
    IF FLOOR(s) ~= s OR s < 1 THEN DO;
      PUT "ERROR: Invalid argument s, s should be  integer >= 1!";
      x_round= .;
    END;
    ELSE IF x = 0 THEN DO;
      x_round= 0;
    END;

    ELSE DO;
      * determine power of number rounding;
      r= FLOOR(LOG10(ABS(x))) + 1 - s;
      * round number;
      x_round= ROUND(x,10**r);
    END;
    RETURN(x_round);
  ENDSUB;
  /* character output function */
  FUNCTION sigRoundC(x,s) $32;
    IF FLOOR(s) ~= s OR s < 1 THEN DO;
      PUT "ERROR: Invalid argument s, s should be integer >= 1!";
      x_char= " ";
    END;
    ELSE IF x = 0 THEN DO;
      x_char= PUT(x,32.0);
    END;
    ELSE DO;
      * determine power of number rounding;
      r= FLOOR(LOG10(ABS(x))) + 1 - s;
      * round number;
      x_round= ROUND(x,10**r);
      * set character format;
```

```
      x_format= CATS("32.",MAX(-1*r,0));
      * convert to character format;
      x_char= PUTN(x_round, x_format);
    END;
    RETURN(x_char);
  ENDSUB;
RUN;
```

Der Algorithmus zur Rundung der Zahl ist grau hinterlegt. Der Rückgabewert wird über die Funktion RETURN() bestimmt. Die ersten beiden Bedingungen werden zur Parametervalidierung benötigt. Mit `FLOOR(s) ~= s` wird überprüft, ob die Anzahl der signifikanten Stellen s eine ganze Zahl ist. Weiterhin wird überprüft, ob s größer gleich 1 ist. Wird der Wert 0 übergeben, so wird die 0 ohne Kommastellen zurückgegeben.

Bevor die Funktion aufgerufen wird, muss über die globale Option CMPLIB der Datensatzname, in der die Funktionen abgelegt sind, festgelegt werden.

OPTIONS CMPLIB=work.funcs;

Danach kann die Funktion genauso wie eine SAS interne Funktion aufgerufen werden. Dies gilt auch für SQL und Makrocode (%SYSFUNC).

```
DATA sr10_testFuction;
  SET sr01_numbers;
  resultn= sigRoundN(x,3);
  resultc= sigRoundC(x,3);
RUN;
```

	x	resultn	resultc
1	1236.67	1240	1240
2	1000	1000	1000
3	450	450	450

4 Umsetzung als SAS-Makro

Funktionen können erst ab der SAS Version 9.2 erstellt werden. Daher soll im Folgenden die Umsetzung als SAS-Makro gezeigt werden. Generell ist aber zu erwähnen, dass eine Funktion sicherer ist als ein SAS-Makro, da diese unabhängig vom DATA-Step ausgeführt wird. Würde bei dem SAS-Makro einfach der Programmcode wie bei der Funktion übernommen werden, so wäre es möglich, Variablen im DATA-Step zu überschreiben, falls gleichnamige Variablen vorhanden sind. Dieses Problem kann übergangen werden, indem alle Schritte in einem SAS-Ausdruck (SAS expression) zusammengefasst werden. Benötigte Bedingungen können über die Funktionen IFN() und IFC() realisiert werden. Ein Vorteil bei dem SAS-Makro ist, dass ein Makro sowohl eine Zei-

chenkette als auch den numerischen Wert zurückgeben kann. Daher soll das Makro drei Eingabeparameter besitzen, diese wären der Eingangswert x, die Anzahl der signifikanten Stellen des Ausgangswertes s und der Ausgangstyp type. Der SAS-Ausdruck für die Erzeugung des numerischen Wertes hat die folgende Form.

```
IFN( &var. ~= 0
   , ROUND(&var.,10**(FLOOR(LOG10(ABS(&var.))) + 1 - &s.)
   , 0
   );
```

Der Ausdruck für den Rückgabewert als Zeichenkette sieht etwas unübersichtlicher aus.

```
IFC( &var. ~= 0
   , PUTN(ROUND(&var.,10**(FLOOR(LOG10(ABS(&var.))) + 1 - &s.))
        ,CATS("32.",MAX(-1*(FLOOR(LOG10(ABS(&var.))) + 1 - &s.),0))
   , PUT(0,32.0)
   );
```

Die Umwandlung in einem Ausdruck ist erstens fehleranfälliger und zweitens bei einer eventuellen Fehlersuche störender.

Bei dem Makro wird der gesamte Ausdruck zurückgegeben, wobei alle Makrovariablen bereits aufgelöst sind. Der restliche Programmcode im Makro kümmert sich um die Validierung der Eingangsparameter.

```
%MACRO sigRound(var=,s=,type=C);
  %IF %SYSFUNC(FLOOR(&s.)) ~= &s. OR &s. < 1 %THEN %DO;
    %PUT ERROR: Invalid argument s, s should be integer >= 1!;
    %ABORT;
  %END;
  /* numeric output */
  %IF %UPCASE(&type.) = N %THEN %DO;
    IFN( &var. ~= 0
       , ROUND(&var.,10**(FLOOR(LOG10(ABS(&var.))) + 1 - &s.)
       , 0
       );
  %END;
  /* character output */
  %ELSE %IF %UPCASE(&type.) = C %THEN %DO;
    IFC( &var. ~= 0
       , PUTN(ROUND(&var.,10**(FLOOR(LOG10(ABS(&var.))) + 1 - &s.))
            ,CATS("32."
                 ,MAX(-1*(FLOOR(LOG10(ABS(&var.))) + 1 - &s.),0)
                 )
       , PUT(0,32.0)
       );
  %END;
  %ELSE %DO;
    %PUT ERROR: Invalid argument type, type should be N or C!;
    %ABORT;
  %END;
%MEND sigRound;
```

Der Aufruf ähnelt dem der Funktion, wobei die erforderlichen Prozentzeichen vorangestellt werden müssen und der zusätzliche Eingangsparameter, zur Bestimmung des Typs vom Ausgangswert, angegeben werden muss.

```
DATA sr20_testMacro;
  SET sr01_numbers;
  resultc= %sigRound(var=x,s=3,type=C);
  resultn= %sigRound(var=x,s=3,type=N);
RUN;
```

Das Ergebnis ist identisch zu dem abgebildeten Ergebnis für die Funktionen.

5 Fazit

Mit Hilfe der signifikanten Rundung, lassen sich die geforderten Zahlenformate viel einfacher definieren. Das signifikante Runden ist außerdem gegen unerwartete Größenänderungen resistent. Das sich diese Rundungsart noch nicht so weit durchgesetzt hat, liegt zum einen an dem dynamischen Layout, welches etwas gewöhnungsbedürftig ist und zum anderen an der schwierigen Umsetzung im Programm. Letzteres sollte sich mit diesem Artikel deutlich vereinfacht haben. Es empfiehlt sich immer, eine SAS-Funktion zu erstellen, wenn folgende Voraussetzungen gegeben sind:

- es steht die SAS Version 9.2 oder höher zu Verfügung
- der Algorithmus lässt sich in einem DATA-Step umsetzen
- es wird nur ein Rückgabewert benötigt

Vor der Datenausgabe sollten die gerundeten Werte immer im Textformat vorliegen, da nachfolgende Nullen sonst verloren gehen können.

Durch die Offenlegung des Programmcodes können schnell auswertungsspezifische Anpassungen gemacht werden. Ein häufiger Kundenwunsch ist z.B. das Vorkommastellen nicht gerundet werden. Der Grund für diese Regel ist, dass diese Zahlen ohnehin ausgegeben werden müssen und daher auch genauer angegeben werden können. Ich persönlich finde die gerundeten Zahlen richtiger und übersichtlicher. Zudem besteht die Gefahr einer fälschlichen Übergenauigkeit (die Zahlen könnten genauer dargestellt werden, als es die Ausgangswerte oder Messgeräte hergeben). Weitere mögliche Features wären: die Anpassung der Zeichenkettenlänge, Ausrichtung mehrerer Werte am Dezimaltrennzeichen und die Änderung der Darstellungen der Null.

Ich hoffe, der Artikel hat ihnen gefallen und vereinfacht ihre tägliche Arbeit signifikant;-). Bei weiteren Fragen können sie mich gerne kontaktieren.

Literatur

[1] Zahlenangaben: Band 1333 von DIN-Normen: Deutsches Institut für Normung. Beuth Verlag, 1992