SESUG Paper FCS-49-2025

Reproducing the "BEST" Format in R

David J. Bosak, r-sassy.org

ABSTRACT

The "BEST" format is commonly used in SAS® programs. This format constrains the display of a numeric value to a specified width. Depending on the characteristics of the numeric values, it may be rounded or not. For very small or large values, the "BEST" format will display the value in scientific notation. If the value has decimals, the number of decimals shown by "BEST" can change depending on number of integer digits. The rules concerning the "BEST" format are in fact quite complicated. Is there any way to reproduce the "BEST" format in R? This paper will show you how to do it using a recently added feature of the **fmtr** R package.

INTRODUCTION

SAS and R offer many of the same capabilities when it comes to managing data. There are some things, however, that are quite specific to SAS. One of them is the “best” SAS format.

The “best” format was introduced early in SAS history, and is designed to produce the *best* representation of a number within a fixed width. Natively, R has no such format. Data formatted with the SAS “best” format can therefore be problematic when converting or translating between SAS and R.

Recently, some of the capabilities of the “best” format have been published in the R package **fmtr**. The **fmtr** package contains tools for formatting that are similar in concept to SAS. The purpose of this paper is to first explain the basic logic of the “best” format, and then illustrate how the **fmtr** package can reproduce this logic.

“BEST” OVERVIEW

The “best” format is actually the default format that SAS uses when printing a number. The syntax for the “best” format is “BEST*w*.”, where “w” is a width between 1 and 32. If no width is specified, the default is 12 characters. The format will attempt to fit any numeric value from your data within the given width.

Here is a simple example, showing basic usage of the format. The example shows a large decimal value and a small decimal value, each displayed with “best” formats of different widths:

```
data test;
length Test $6 Value $12;

Test = "best4";
Value = put(123456789.12345, best4.);
output;

Test = "best4";
Value = put(0.0000123456789, best4.);
output;

Test = "best8";
Value = put(123456789.12345, best8.);
output;

Test = "best8";
Value = put(0.0000123456789, best8.);
output;

Test = "best12";
Value = put(123456789.12345, best.);
```

```
output;

Test = "best12";
Value = put(0.0000123456789, best.);
output;

run;
```

The above code produces the following output data:

	Test	Value
1	best4	12E7
2	best4	0
3	best8	1.2346E8
4	best8	0.000012
5	best12	123456789.12
6	best12	0.0000123457

Notice that the same numeric values can be displayed very differently, depending on the width associated with the “best” format. That is one reason why the “best” format is so difficult to reproduce in R.

UNDERSTANDING THE “BEST” FORMAT

Let’s examine the “best” format more closely, and try to understand some of the logic. Here is an illustrative example. This example outputs the same value with a range of best formats from 1 to 12.

```
%macro TestBestFullRange;

%let v1 = 123456789.12345;

data TestBestFullRange;

length Test $6 Value $12;

%do nm = 1 %to 12;
  Test = "best&nm.";
  Value = put(&v1., best&nm..);
  output;
%end;

run;

%mend;

%TestBestFullRange;
```

Here is the output data from the above macro:

	Test	Value
1	best1	*
2	best2	**
3	best3	1E8
4	best4	12E7
5	best5	123E6
6	best6	1.23E8
7	best7	1.235E8
8	best8	1.2346E8
9	best9	123456789
10	best10	123456789
11	best11	123456789.1
12	best12	123456789.12

Observe that at small widths 1 and 2, the format cannot represent the number at all. Instead, it produces stars (*) of the desired width. With widths 3 to 5, the format emits a modified scientific notation with no decimal place. From widths 6 to 8, it outputs proper scientific notation. Finally, from 9 to 12, the “best” format simply rounds the numeric value to the desired width.

Note that these variations are not fixed for the specified width. A specified width can produce stars, scientific notation, or a rounded value depending on what kind of number is being formatted. Behold:

```
data test2;
length Test $6 Value $12;

Test = "best4";
Value = put(-123789123450000, best4.);
output;

Test = "best4";
Value = put(123789.12345, best4.);
output;


Test = "best4";
Value = put(12.12345, best4.);
output;

run;
```

The above code produces the following:

	Test	Value
1	best4	****
2	best4	12E4
3	best4	12.1

As can be seen above, the same format “best4” will vary the output depending on the input value. The logic is driven by what can be reasonably displayed in the given width.

For example, in row 1, the value is both large and negative. That means there is simply no room to display the negative sign, a number, and an exponent. Therefore, the format has to give up and display stars.

On the other hand, the input values on rows 2 and 3 can be reasonably represented in only 4 characters.

INTRODUCTION TO THE “FMTR” PACKAGE

Given the complex logic of the “best” format, is there any way to reproduce this style of formatting in R? Yes!

The **fmtr** R package was released in 2020, and contains several tools that replicate the style of formatting used in SAS.

For instance, the **fmtr** package has a function called `fapply()`, which stands for “format apply”. This function is used very much like a datastep `put()` function in SAS. You can pass a data value and a format, and the function will apply that format to the data value, and return the formatted value as a character string. Like this:

```
# Apply 8.2 format
ret <- fapply(12345.12345, "%8.2f")

# View Results
ret
# [1] "12345.12"
```

The above example exhibits how to apply a simple format to a decimal value. Here, the R format code “%8.2f” corresponds to the SAS format “8.2”. In R, the percent sign introduces the format code, and the “f” indicates that you want to format a floating point number. The “8.2” corresponds exactly to the SAS format. That is, “8.2” indicates that you want the value rounded to 2 decimal places and a width of 8 characters.

For further information on the **fmtr** package, see the following link: https://fmtr.r-sassy.org/index.html

For further explanation of R formatting codes, see the following:

https://fmtr.r-sassy.org/reference/FormattingStrings.html

USING THE “BEST” R FORMAT

To achieve a “best” style format with the **fmtr** package, we will simply use the `fapply()` function shown above, and pass the name of the “best” format as the second parameter. So easy!

Here is how it is done:

```
# Apply best6 format
  ret <- fapply(12345678.12345, "best6")


# View Results
  ret
  # [1] "1.23E7"
```

Voila! Using the **fmtr** package, you can reproduce the SAS "best" format with one line of code!

MORE EXAMPLES

Before moving on, let's look at some more examples. In the following code, we will reproduce the `%TestBestFullRange` SAS macro from above. Here is how it is done in R:

```
# Assign value to format
v1 <- 123456789.12345

# Initialize vectors
tst <- c()
vls <- c()

# Apply formats dynamically
for (nm in 1:12) {

  tst <- append(tst, paste0("best", nm))
  vls <- append(vls, fapply(v1, paste0("best", nm)))

}

# Create data frame from vectors
res <- data.frame(Test = tst, Value = vls)

# View result
res
#      Test        Value
# 1   best1            *
# 2   best2           **
# 3   best3          1E8
# 4   best4         12E7
# 5   best5        123E6
# 6   best6       1.23E8
# 7   best7      1.235E8
# 8   best8     1.2346E8
# 9   best9    123456789
# 10 best10    123456789
# 11 best11  123456789.1
# 12 best12 123456789.12
```

By comparing the output of the above R code to the corresponding SAS macro, you will see that the output values of the "best" R format are identical to the SAS "best" format for each of the specified widths. Hurray!

LIMITATIONS OF R "BEST"

Note that there are some limitations of the R "best" format.

Firstly, at this time, the format only reproduces the "BEST*w*." syntax. That is to say, the package does not currently support "BEST*w.d*" syntax, which allows you to specify a number of decimal places for the output value.

The package also does not support the "BESTD*w.p*" variant of the "BEST" format. This variant is often used for decimal alignment.

Finally, note that the logic of the BEST format is quite complicated, and there may be cases that do not match perfectly with SAS. If you encounter such a case, please submit an issue to the **fmtr** package issue list here: https://github.com/dbosak01/fmtr/issues.

CONCLUSION

R has no native capability to replicate the SAS "best" format. Therefore, discrepancies can arise when trying to reproduce SAS outputs in R. Fortunately, the **fmtr** package has reverse-engineered the basic functionality of the "best" format, and provided it to the open-source community for free. The format can be replicated by passing the "best" format specification to the `fapply()` function. The function will input a numeric value, and output a character representation of the number in the indicated width. The R "best" format is easy to use, and can greatly reduce the difficulty of trying to match the corresponding SAS output.

REFERENCES

Bosak, David J. 2025. "Why SASSY?" *Proceedings of PharmaSUG*. Available at https://pharmasug.org/proceedings/2025/AP/PharmaSUG-2025-AP-263.pdf

Bosak, David J. "The SASSY System." 2025. Available at https://r-sassy.org/.

CONTACT INFORMATION

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

David J. Bosak
Archytas Clinical Solutions, LLC
dbosak01@gmail.com
www.r-sassy.org