PharmaSUG China 2024 - Paper PO-10088

Seeing is not always believing

-----The data you observe and the encoding behind its storage in SAS

Tiantian HE, Pfizer (Wuhan) Research and Development Co.,Ltd. China

ABSTRACT

It is a common situation in the SAS dataset that the seemingly identical data values, including numeric data and character data, actually have underlying differences, the differences can lead to unexpected outcomes and considerable confusions for SAS beginners. The situation may occur due to inconsistencies in data internal storage representation in SAS datasets such as different encodings, unprintable special characters, or numeric precision. This article will discuss how the storage and presentation of data in SAS dataset lead to such discrepancies, and how to use macros to identify and better handle these issues to prevent the potential errors, especially when using 'PROC COMPARE' or filtering statements.

INTRODUCTION

As we all know, computers store data in binary format. SAS datasets consist of two data formats: character and numeric. Numeric data typically utilizes eight bytes per value, whereas text data requires character encoding to map characters to numeric codes for storage. However, discrepancies can arise where seemingly identical data may be encoded differently internally, leading SAS to perceive differences. This issue manifests during operations like PROC COMPARE or filtering commands, resulting in unintended outcomes due to the variance in internal storage.

This paper provides the examples where numeric data and character data in datasets can present confusing results. And it explores how numeric values and character data are stored in SAS datasets and how this can introduce problems. Finally, it identifies and manifest these issues to enhance accuracy in data handling.

THE PROBLEM

EXAMPLE 1:

There are two SAS datasets, A and B which have two variables: A is a character variable, and INDEX is a numeric variable equal to _N_.

WORK.A

Filter: (none)

Total rows: 4 Total columns: 2

	A	index
1	Test end	1
2	Test end	2
3	Test end	3
4	Test end	4

WORK.B

Filter: (none)

Total rows: 4 Total columns: 2

	A	index
1	Test end	1
2	Test end	2
3	Test end	3
4	Test end	4

Display 1. SAS Dataset A And Dataset B

Now these two datasets appear to be identical. We will use the 'PORC COMPARE ' function to compare these two datasets and output the RESULT dataset.

```
PROC COMPARE base=A comp=B out=result outbase outcomp outdiff listall;
```

```
RUN;
```

WORK.RESULT

Filter: (none)

Total rows: 12 Total columns: 4

	TYPE	_OBS_	A	index
1	BASE	1	Test end	1
2	COMPARE	1	Test end	1
3	DIF	1	....X...	0
4	BASE	2	Test end	2
5	COMPARE	2	Test end	2
6	DIF	2	.....	0
7	BASE	3	Test end	3
8	COMPARE	3	Test end	3
9	DIF	3	....X...	0
10	BASE	4	Test end	4
11	COMPARE	4	Test end	4
12	DIF	4	....X...	0

Display 2. Output for Compare Result

We observed that the records show differences in variable A between datasets A and B according to output, except for the record where INDEX=2. It may be confused. Next, we use the IF filtering command to view the output results from dataset A.

```
data want;
     set a;
     if a='Test end';
run;
```

WORK.WANT

Total rows: 1 Total columns: 2

	A	index
1	Test end	2

Display 3. Output for Dataset WANT

Similarly, the output dataset contains only the record with index=2, and the result is still somewhat unexpected.

EXAMPLE 2:

There are two SAS datasets, C and D which have two variables: A and INDEX are numeric variables and INDEX is equal to _N_.

WORK.C

Filter:

Total rows: 3 Total columns: 2

	A	Index
1	0.1	1
2	0.3	2
3	0.8	3

WORK.D

Filter: (none)

Total rows: 3 Total columns: 2

	A	Index
1	0.1	1
2	0.3	2
3	0.8	3

Display 4. SAS Dataset C And Dataset D

We use the same code as in example 1 to compare these two datasets and output the RESULT dataset.

WORK.RESULT

Filter: (none)

Total rows: 9 Total columns: 4

	TYPE	_OBS_	A	Index
1	BASE	1	0.1	1
2	COMPARE	1	0.1	1
3	DIF	1	0	0
4	BASE	2	0.3	2
5	COMPARE	2	0.3	2
6	DIF	2	-5.55112E-17	0
7	BASE	3	0.8	3
8	COMPARE	3	0.8	3
9	DIF	3	1.110223E-16	0

Display 5. Output for Compare Result

The result still shows differences between dataset C and dataset D, but WHY?

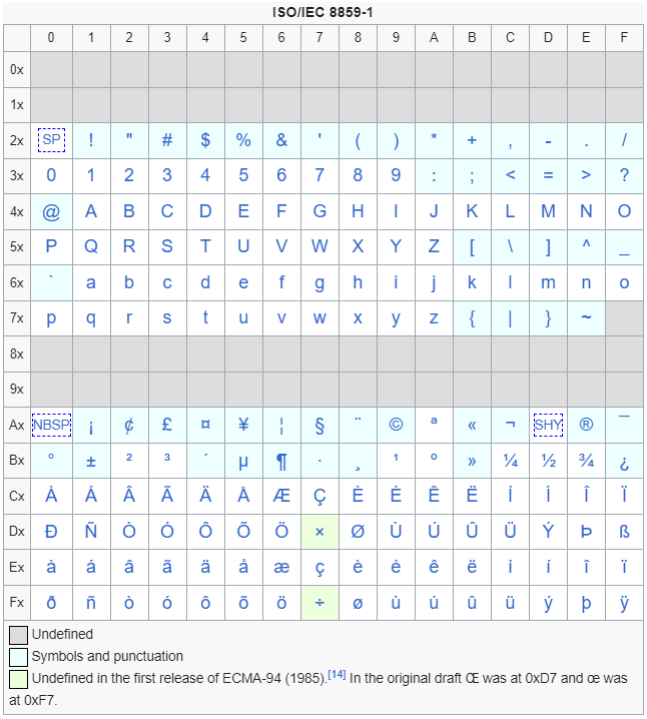
HOW NUMBERS AND CHARACTERS ARE STORED

In the previous discussion, it was mentioned that SAS uses different character encodings when storing characters. ASCII is one of the earliest and most widely used character encoding standards. It uses 7 bits to represent 128 characters, including control characters (e.g., newline, tab) and printable characters (e.g., letters, digits, symbols). Depending on various SAS system used, there may be different character encodings. We can use the 'SYSENCODING' function to check the used character encoding in SAS. In this article, SAS Studio use latin1 (ISO 8859-1) standard, which encompasses 256 characters.

```
73          %put &sysencoding=;
latin1=
```

Display 5. Character Encoding Standard Used in SAS Studio

The first 128 characters in latin1 correspond to ASCII characters. Figure 1 is the encoding for latin1, where the numbers on the left and above combine to give the hexadecimal representation of the character's value. For example, the third row of first column shows SP (Space), which stores the value 20 (hexadecimal), equivalent to 32 in decimal. The character NBSP (Non-breaking space) in the eleventh row of first column stores the value A0 (hexadecimal), which converts to 160 in decimal. The first 32 characters in the encoding table are control characters, which control peripherals such as printers. Details about these control characters can be found in figure 2.



ISO/IEC 8859-1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x																
1x																
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8x																
9x																
Ax	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Undefined
Symbols and punctuation
Undefined in the first release of ECMA-94 (1985).[14] In the original draft Œ was at 0xD7 and œ was at 0xF7.

Figure 1. Latin1 Encoding

ASCII Control Characters

The following table contains the ASCII control characters that can be represented by decimal values:

Value	Character	Value	Character	Value	Character
0	Null character	11	Vertical tab	22	Synchronous idle
1	Start of header	12	Form feed	23	End of transmission block
2	Start of text	13	Carriage return	24	Cancel
3	End of text	14	Shift out	25	End of medium
4	End of transmission	15	Shift in	26	Substitute
5	Enquiry	16	Data link escape	27	Escape
6	Acknowledgment	17	Device control 1	28	File separator
7	Bell	18	Device control 2	29	Group separator
8	Backspace	19	Device control 3	30	Record separator
9	Horizontal tab	20	Device control 4	31	Unit separator
10	Line feed	21	Negative acknowledgment	127	Delete

Figure 2. Control Characters

Alternatively, you can use code like the following to obtain the character corresponding to the encoding.

```
data encoding;
  do i=0 to 255;
    char=byte(i);
    hex=put(char,hex.);
    output;
  end;
run;
```

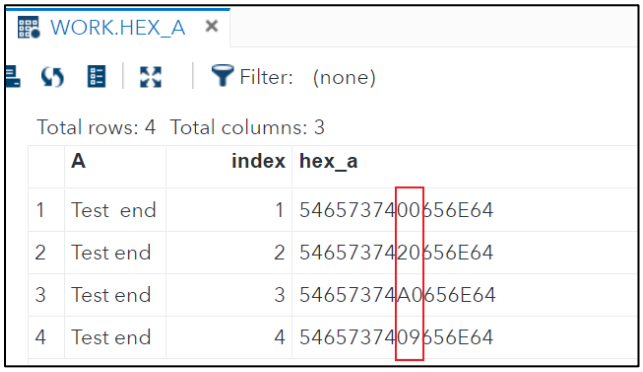
The CHAR variable represents 256 characters, and HEX is the hexadecimal value of the storage code.

Regarding numerical values, floating-point representation is a computer numerical type used to represent real numbers. If consists of four fundamental components: Sign, Base, Exponent, and Mantissa. While due to the finite precision of floating-point numbers, rounding errors and precision issues may arise during calculations, especially with decimals.

VIEW INTERNAL ENCODING

Based on the storage method described earlier, since data is stored using a sequence of numbers, it would be clearer to see the internal encoding of both character or numeric data. We can use the HEX format in SAS to display the internal storage of data, using dataset A as an example.

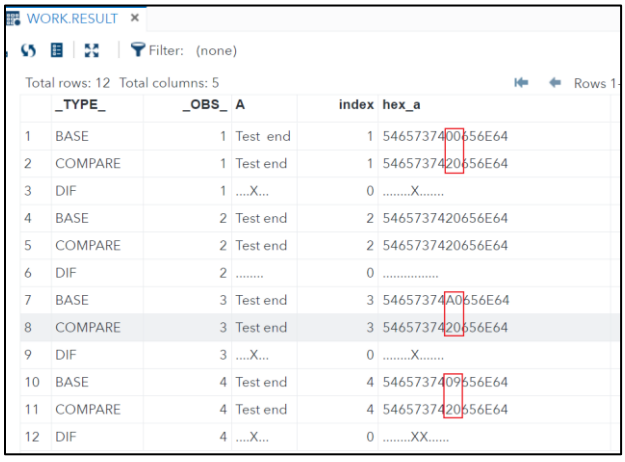
```
data hex_a;
     set a;
     hex_a=put(a, hex.);
run;
```



	A	index	hex_a
1	Test end	1	5465737400656E64
2	Test end	2	5465737420656E64
3	Test end	3	54657374A0656E64
4	Test end	4	5465737409656E64

Display 6. Output Dataset HEX_A

The variable HEX_A displays the hexadecimal values that variable A is stored in dataset A1. Each character is represented by two hexadecimal digits. We can observe that even though the fifth character of these four records appears to be a space, the internal storage codes are different. Referring to the Latin1 table mentioned earlier, we can determine that '00' represents the control character 'Null character', '20' represents 'SP', 'A0' represents 'NBSP', and '09' represents the control character 'Horizontal tab'. Similarly, by converting dataset B into hexadecimal format, we can clearly see the differences between the dataset A and dataset B.



	TYPE	_OBS_	A	index	hex_a
1	BASE	1	Test end	1	5465737400656E64
2	COMPARE	1	Test end	1	5465737420656E64
3	DIF	1	....X...	0	........X.......
4	BASE	2	Test end	2	5465737420656E64
5	COMPARE	2	Test end	2	5465737420656E64
6	DIF	2	....	0	....
7	BASE	3	Test end	3	54657374A0656E64
8	COMPARE	3	Test end	3	5465737420656E64
9	DIF	3	....X...	0	........X.......
10	BASE	4	Test end	4	5465737409656E64
11	COMPARE	4	Test end	4	5465737420656E64
12	DIF	4	....X...	0	........XX......

Display 7. Output for Compare Result

We can see that the spaces in dataset B are encoded as '20', which represents ‘Space’ without special characters. This is the reason for the difference between dataset A and B.

Using the same method, we can determine the storage code differences between dataset C and dataset D, which contain numeric variables.

SOLUTIONS THAT ENSURE EXPECTED RESULTS

Since these unexpected results occur due to differences in internal storage codes, we can resolve this issue by changing the internal storage codes.

METHOD 1

One method is to use translate to convert special characters or control characters into commonly used characters.

```
data a1;
     set a;
     a=translate(a,'' , 'A0'x); /* translate non-breaing space to space */
run;
```

WORK.RESULT

Filter: (none)

Total rows: 12 Total columns: 4

	TYPE	_OBS_	A	index
1	BASE	1	Test end	1
2	COMPARE	1	Test end	1
3	DIF	1	....X...	0
4	BASE	2	Test end	2
5	COMPARE	2	Test end	2
6	DIF	2	.....	0
7	BASE	3	Test end	3
8	COMPARE	3	Test end	3
9	DIF	3	.....	0
10	BASE	4	Test end	4
11	COMPARE	4	Test end	4
12	DIF	4	....X...	0

Display 8. Output for Compare Result

We see that the third record is currently displayed the same as dataset B due to the replacement of characters.

METHOD 2

We also can use COMPRESS to eliminate spaces. Adding the modifier 'c' and 's' and adding 'A0'x to characters essentially can eliminate most common space types (blank, horizontal tab, vertical tab, carriage return, line feed, form feed, and NBSP). But this method is more suitable when control characters are at the beginning or end.

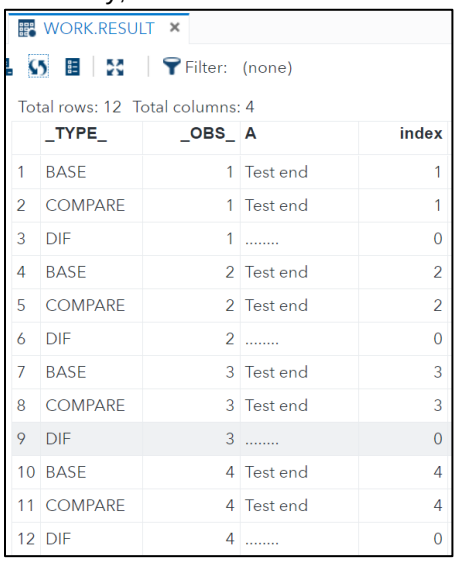
```
data a2;
     set a;
     a2=compress(a,'A0'x ,"cs"); /* delete control characters and blank*/
     hex_a2=put(a2, hex.);
run;
```

METHOD 3

The regular expressions also can be used to replace special characters and control characters with space. According to the Latin1 table, we can see that the first 32 characters and the characters from 7F to A0 are control characters and special characters. They can be replaced with simple space.

```
data a3;
    set a;
    a = prxchange('s/[\x00-\x1F\x7F-\xA0]/ /', -1, a); /*change 00-1F/7F-A0
to space*/
run;
```

Ultimately, it is evident that there are no discrepancies with dataset B .



WORK.RESULT

Filter: (none)

Total rows: 12 Total columns: 4

	TYPE	_OBS_	A	index
1	BASE	1	Test end	1
2	COMPARE	1	Test end	1
3	DIF	1	.....	0
4	BASE	2	Test end	2
5	COMPARE	2	Test end	2
6	DIF	2	.....	0
7	BASE	3	Test end	3
8	COMPARE	3	Test end	3
9	DIF	3	.....	0
10	BASE	4	Test end	4
11	COMPARE	4	Test end	4
12	DIF	4	.....	0

Display 9. Output for Compare Result

METHOD 4

For numeric variables, the best way is to use the ROUND function. To preserve data integrity as much as possible, we can round decimals to the nearest 10^{-15} places.

```
data c1;
    set c;
    a1=round(a,1E-15);
run;
```

CONCLUSION

Whether dealing with numeric or character data, if you do not understand how they are stored, the unintended results can be got during data processing. We can use the HEX format to inspect their encoding, then use TRANSLATE to replace them with desired characters, or use COMPRESS to remove unnecessary characters. Alternatively, we can directly replace control characters and special characters with space using regular expressions through PRXCHANGE. For numeric variables, especially decimals, using the ROUND function is probably the most convenient method.

REFERENCES

Paul Stutzman. 2014. “What do you mean 0.3 doesn't equal 0.3? Numeric Representation and Precision

in SAS and Why it Matters." *PharmaSUG*

ISO/IEC 8859-1

Available at https://en.wikipedia.org/wiki/ISO/IEC_8859-1

ASCII Control Characters

Available at https://documentation.sas.com/doc/en/engelref/2.8/p1orlsbqj20nirn1p17o8j1zsyti.htm

CONTACT INFORMATION

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

Tiantian HE
Pfizer (Wuhan) Research and Development Co.,Ltd. China
Tiantian.HE@pfizer.com