Paper 193-2023

MISSING Mysteries for SAS Beginners

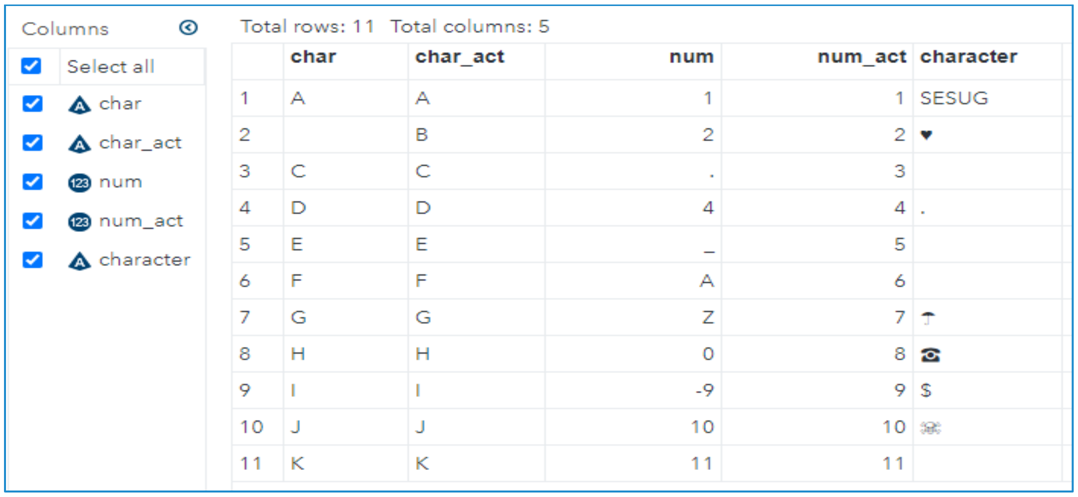
Jinson Erinjeri

ABSTRACT

SAS datasets comprise of variables which are mainly of two types, numeric and character. Observations with missing values for these types of variables are common in real applications. For a beginner in SAS programming, understanding how the missing values operate is critical because improper usage can lead to programming as well as reporting errors. This paper will present such error scenarios, the cause of these errors and ways to avoid them. The scenarios are presented with example dataset and code for better understanding and retention, thereby unraveling the mystery of missingness.

INTRODUCTION

In real word data, missing values are ubiquitous and how these values are processed based on an application becomes important to avoid downstream cataclysmic consequences. Incorrect handling of missing values can lead to programmatic errors as well as reporting errors. Programmatic errors may spew out in the logs, and this can be handled, if vigilant. On the other hand, reporting errors might be challenging and may seep through the cracks at the Quality Control phase.



Total rows: 11 Total columns: 5

	char	char_act	num	num_act	character
1	A	A	1	1	SESUG
2		B	2	2	♥
3	C	C	.	3	
4	D	D	4	4	.
5	E	E	_	5	
6	F	F	A	6	
7	G	G	Z	7	⛱
8	H	H	0	8	☎
9	I	I	-9	9	$
10	J	J	10	10	☠
11	K	K	11	11	

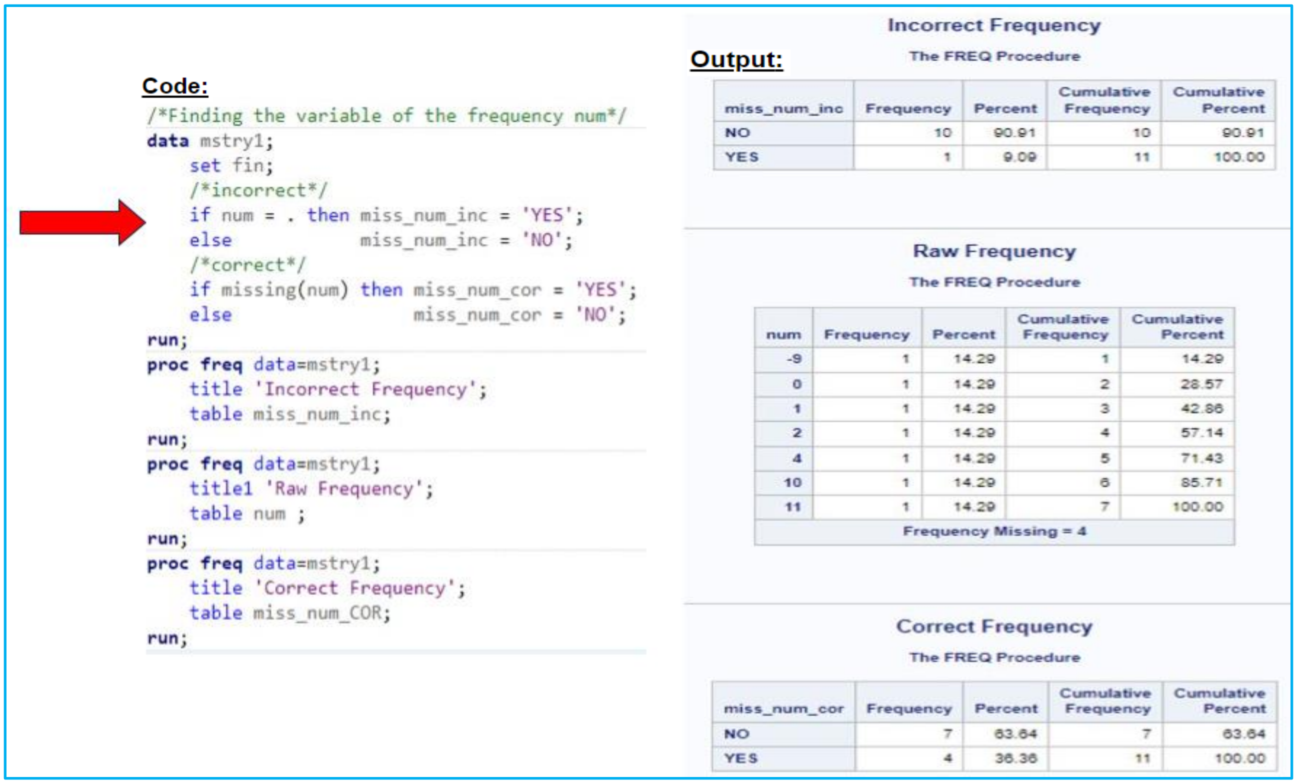
Figure 1. Screenshot of Dataset with Missing Observations for Character and Numeric Data Types

Some of the common scenarios faced by SAS beginners with regards to how missing values are accounted for or unaccounted for are presented in this paper. For each scenario, a task is presented and depending on the task, the issues with regards to detection and treatment will be discussed.

The screenshot of the dataset in Figure 1 will be used in the entire paper to present the various scenarios. There are five variables with character, and numeric data types. The variables “char”, “num” and “character” are the ones with missing values and those with suffix “_act” are the actual values of the variables without any missing. The variable “char” has values from A to K including a blank, whereas variable “character” has some special characters along with blanks. The numeric variables have numbers ranging from 1 to 11 along with missing values.

Scenario 1:

Task: Find the frequency of the variable “num”.



Code:

```
/*Finding the variable of the frequency num*/
data mstry1;
    set fin;
    /*incorrect*/
    if num = . then miss_num_inc = 'YES';
    else            miss_num_inc = 'NO';
    /*correct*/
    if missing(num) then miss_num_cor = 'YES';
    else                 miss_num_cor = 'NO';
run;
proc freq data=mstry1;
    title 'Incorrect Frequency';
    table miss_num_inc;
run;
proc freq data=mstry1;
    title1 'Raw Frequency';
    table num ;
run;
proc freq data=mstry1;
    title 'Correct Frequency';
    table miss_num_COR;
run;
```

Output:

Incorrect Frequency

The FREQ Procedure

miss_num_inc	Frequency	Percent	Cumulative Frequency	Cumulative Percent
NO	10	90.91	10	90.91
YES	1	9.09	11	100.00

Raw Frequency

The FREQ Procedure

num	Frequency	Percent	Cumulative Frequency	Cumulative Percent
-9	1	14.29	1	14.29
0	1	14.29	2	28.57
1	1	14.29	3	42.86
2	1	14.29	4	57.14
4	1	14.29	5	71.43
10	1	14.29	6	85.71
11	1	14.29	7	100.00
Frequency Missing = 4				

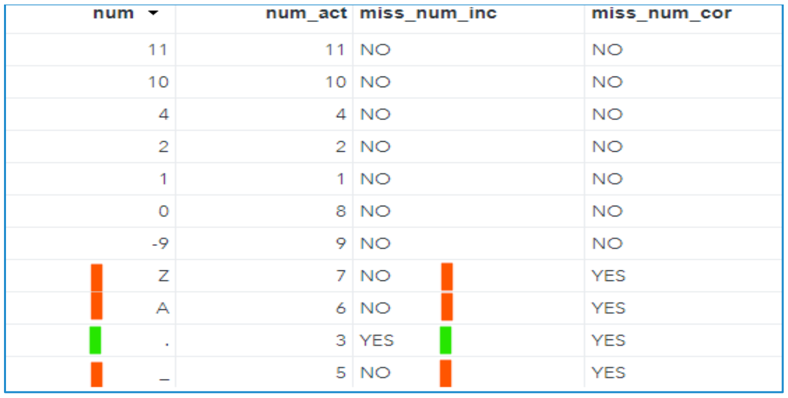
Correct Frequency

The FREQ Procedure

miss_num_cor	Frequency	Percent	Cumulative Frequency	Cumulative Percent
NO	7	63.64	7	63.64
YES	4	36.36	11	100.00

Figure 2. Screenshot of Code as well as Output for Finding Frequency of Numeric Variable

The task appears very simple, and the relevant code highlighted by the red colored arrow in Figure 2 shows the first attempt where an if-else condition is applied to create an explicit variable for finding missing values. The following PROC FREQ command for the explicit variable “miss_num_inc” gives the frequency of missing as one in the output titled “Incorrect Frequency”. Note that the frequency of the raw variable (num) shows the number of missing as four under the title “Raw Frequency”. Thus, it can be concluded that there exist differences in frequency between the explicit variable(miss_num_inc) and the raw variable(num) with regards to numerical missing values.



num	num_act	miss_num_inc	miss_num_cor
11	11	NO	NO
10	10	NO	NO
4	4	NO	NO
2	2	NO	NO
1	1	NO	NO
0	8	NO	NO
-9	9	NO	NO
🟥 Z	7	NO 🟥	YES
🟥 A	6	NO 🟥	YES
🟩 .	3	YES 🟩	YES
🟥 _	5	NO 🟥	YES

Figure 3. Screenshot of Output Dataset(mstry1) for Finding Frequency of Numeric Variable

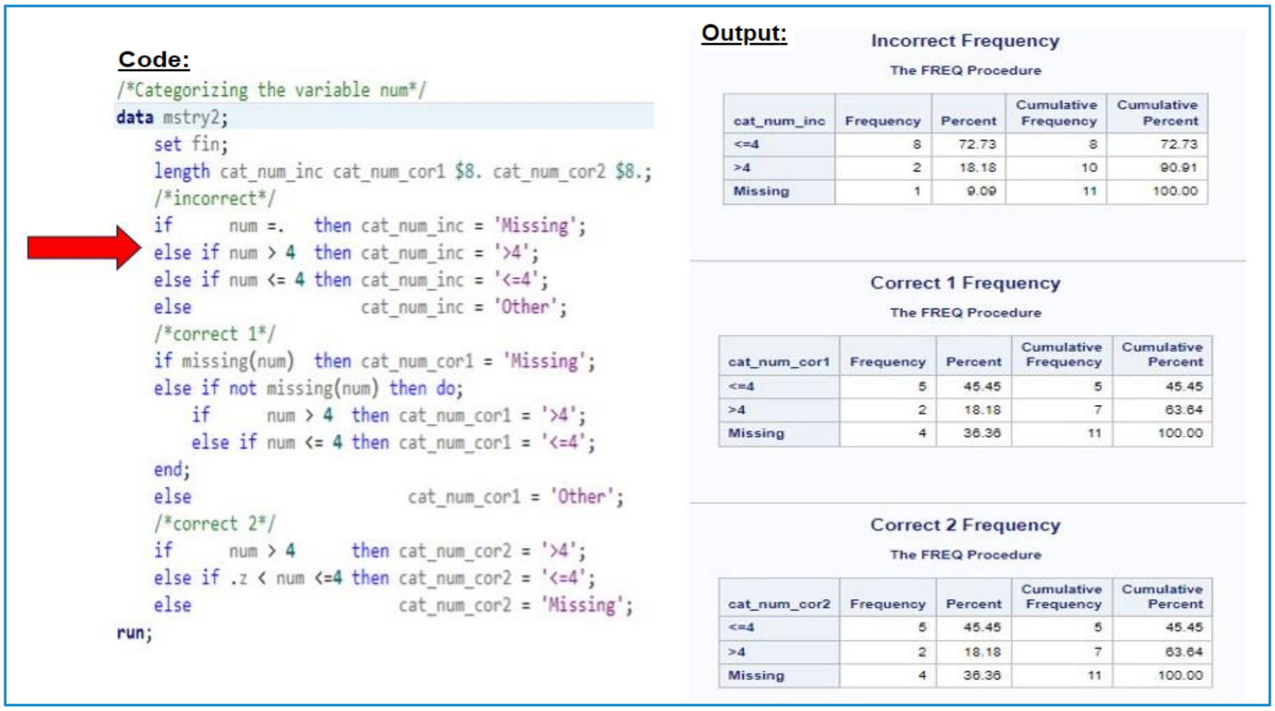
To make things clear, it is best to look at the output dataset(mstry1) shown in Figure 3. The values “_”, “A” and “Z” with red color bars are not captured as missing values by the logic applied for creating the explicit variable “miss_num_inc”. In fact, “_”, “A” and “Z” were entered as “._”, “.A” and “.Z” while creating the source SAS dataset. It is important to know that SAS can store 28 types of missing values in a numeric variable namely,

1. ._ (dot underscore)
2. . (dot)
3. .A (dot A) through .Z (dot Z)

Note that ".A" through ".Z" is not case sensitive. To create an explicit variable, it is best to account for all these types of missing values and the MISSING function in SAS can check if a variable contains a missing value. The function will return 1 if it contains a missing value and 0 if it does not. The application of MISSING function shown in the code section of Figure 2 is the correct way to detect missing numeric variables. The variable "miss_num_cor" is the explicit variable created using the MISSING function. With this approach, we can observe that the raw frequency output for missing counts matches the output using the MISSING function.

Scenario 2:

Task: Categorize variable "num" into above 4, less than or equal to 4 and missing. Also, find the counts of the newly created categories.



Code:

```
/*Categorizing the variable num*/
data mstry2;
    set fin;
    length cat_num_inc cat_num_cor1 $8. cat_num_cor2 $8.;
    /*incorrect*/
    if      num =.   then cat_num_inc = 'Missing';
    else if num > 4  then cat_num_inc = '>4';
    else if num <= 4 then cat_num_inc = '<=4';
    else                  cat_num_inc = 'Other';
    /*correct 1*/
    if missing(num)  then cat_num_cor1 = 'Missing';
    else if not missing(num) then do;
        if      num > 4  then cat_num_cor1 = '>4';
        else if num <= 4 then cat_num_cor1 = '<=4';
    end;
    else                          cat_num_cor1 = 'Other';
    /*correct 2*/
    if      num > 4      then cat_num_cor2 = '>4';
    else if .z < num <=4 then cat_num_cor2 = '<=4';
    else                      cat_num_cor2 = 'Missing';
run;
```

Output:

Incorrect Frequency

The FREQ Procedure

cat_num_inc	Frequency	Percent	Cumulative Frequency	Cumulative Percent
<=4	8	72.73	8	72.73
>4	2	18.18	10	90.91
Missing	1	9.09	11	100.00

Correct 1 Frequency

The FREQ Procedure

cat_num_cor1	Frequency	Percent	Cumulative Frequency	Cumulative Percent
<=4	5	45.45	5	45.45
>4	2	18.18	7	63.64
Missing	4	36.36	11	100.00

Correct 2 Frequency

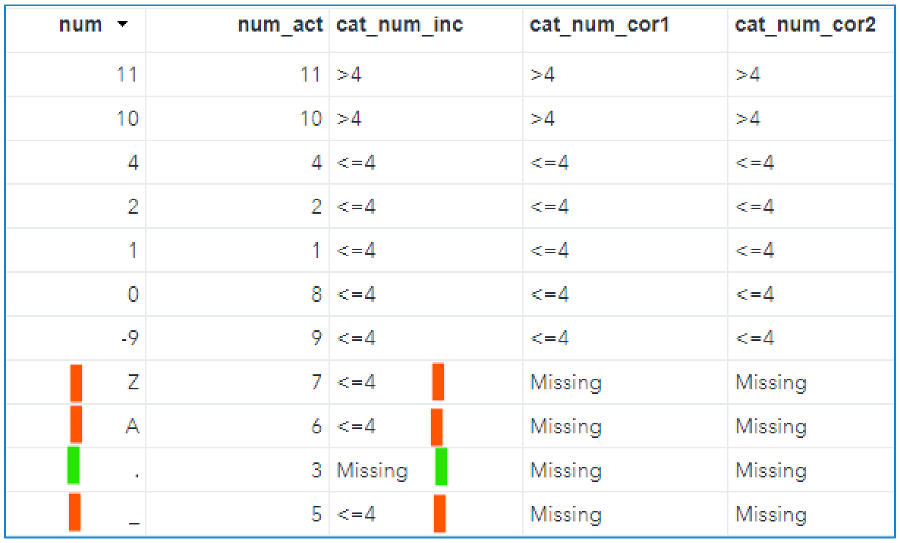
The FREQ Procedure

cat_num_cor2	Frequency	Percent	Cumulative Frequency	Cumulative Percent
<=4	5	45.45	5	45.45
>4	2	18.18	7	63.64
Missing	4	36.36	11	100.00

Figure 4. Screenshot of Code as well as Output for Categorizing Numeric Variable

The second task is common when you are involved with data wrangling and the task appears simple but the red colored arrow in Figure 4 shows the most common mistake a beginner makes. The output titled "Incorrect Frequency" displays the counts of the categories created with this approach. Using the MISSING function in conjunction with the NOT operator, one can categorize the numbers correctly as shown in the code in Figure 4 (correct 1) with the corresponding output under the title "Correct 1 Frequency".
In the previous scenario, we mentioned about 28 types of missing numeric values, but it is also important to note that in SAS, "._" has the lowest missing value, followed by ".", then ".A", and so forth with ".Z" having the highest missing value. With this information on the sort order of missing values, this task can also be coded by specifying ".Z" explicitly in the conditional statement thereby diverting the missing values to the else statement as shown in Figure 4 (correct 2).

The PROC FREQ command gives the frequency of the categories of the incorrect as well as correct approach. It can be seen that “<=4” category has a frequency of eight in the incorrect approach whereas it should have been five. To delve further, the output dataset is presented in Figure 5. Note that the observations with red color bars have been categorized incorrectly since it follows the SAS sort order for missing numeric values. The values “.Z”, “.A” and “.-” are less than four and hence assigned to “<=4” category for the variable “cat_num_inc”. In contrast, the variables “cat_num_cor1” and “cat_num_cor2” have the missing values categorized correctly with the frequency of “<=4” category as five.

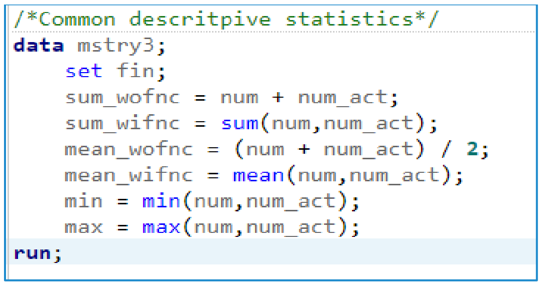


num	num_act	cat_num_inc	cat_num_cor1	cat_num_cor2
11	11	>4	>4	>4
10	10	>4	>4	>4
4	4	<=4	<=4	<=4
2	2	<=4	<=4	<=4
1	1	<=4	<=4	<=4
0	8	<=4	<=4	<=4
-9	9	<=4	<=4	<=4
🟥 Z	7	<=4 🟥	Missing	Missing
🟥 A	6	<=4 🟥	Missing	Missing
🟩 .	3	Missing 🟩	Missing	Missing
🟥 _	5	<=4 🟥	Missing	Missing

Figure 5. Screenshot of Output Dataset(mstry2) for Categorizing Numeric Variable

Scenario 3:

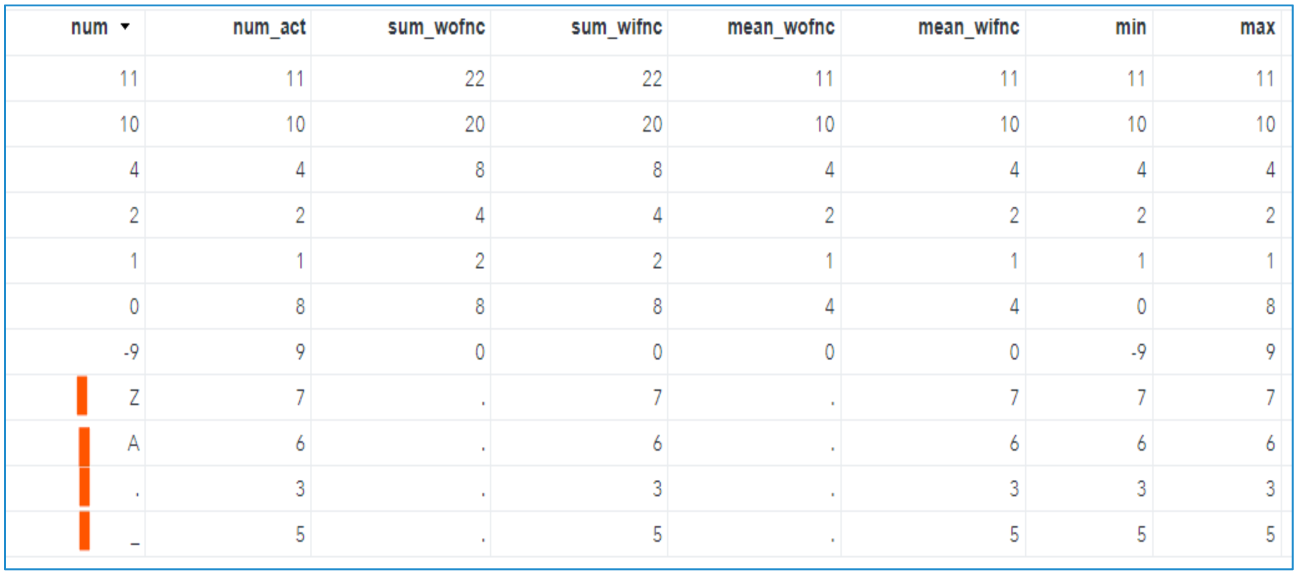
Task: Calculate common descriptive statistics such as sum, mean, min and max for the variable “num”.



```
/*Common descritpive statistics*/
data mstry3;
    set fin;
    sum_wofnc = num + num_act;
    sum_wifnc = sum(num,num_act);
    mean_wofnc = (num + num_act) / 2;
    mean_wifnc = mean(num,num_act);
    min = min(num,num_act);
    max = max(num,num_act);
run;
```

Figure 6. Screenshot of Code for Calculating Common Descriptive Statistics

The task of calculating common descriptive statistics seems to appear straightforward but there are certain things to be aware of with regards to the treatment of missing values. The code above in Figure 6 shows the way of calculating sum and mean of two variables without (sum_wofnc, mean_wofnc) and with (sum_wifnc, mean_wifnc) using the built-in function in SAS. Figure 7 shows the screenshot of the output dataset created using the code shown in Figure 6. When compared, the two approaches of calculating the required statistics give the same results except when missing values are present in the data. The red colored bars indicate observations with missing values and note that sum and mean without the built-in function gives a missing value since an operation with a missing value is calculated as missing. Note that the built-in function ignores the missing value while calculating the required statistics. This is more obvious in the calculation of mean where the mean is equal to the non-missing value (num_act) since the denominator is one. The min and max functions also ignore missing values as shown in the output dataset in Figure 7.

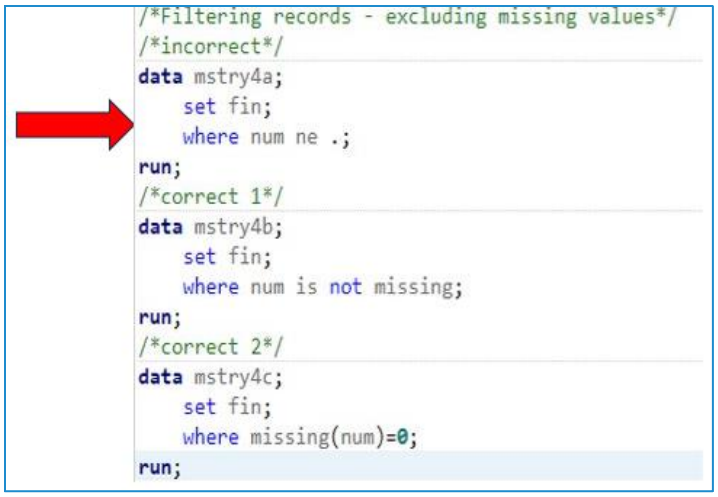


num	num_act	sum_wofnc	sum_wifnc	mean_wofnc	mean_wifnc	min	max
11	11	22	22	11	11	11	11
10	10	20	20	10	10	10	10
4	4	8	8	4	4	4	4
2	2	4	4	2	2	2	2
1	1	2	2	1	1	1	1
0	8	8	8	4	4	0	8
-9	9	0	0	0	0	-9	9
Z	7	.	7	.	7	7	7
A	6	.	6	.	6	6	6
.	3	.	3	.	3	3	3
_	5	.	5	.	5	5	5

Figure 7. Screenshot of Output Dataset(mstry3) for Calculating Common Descriptive Statistics

Scenario 4:

Task: Create a new dataset excluding missing values for the variable “num”.

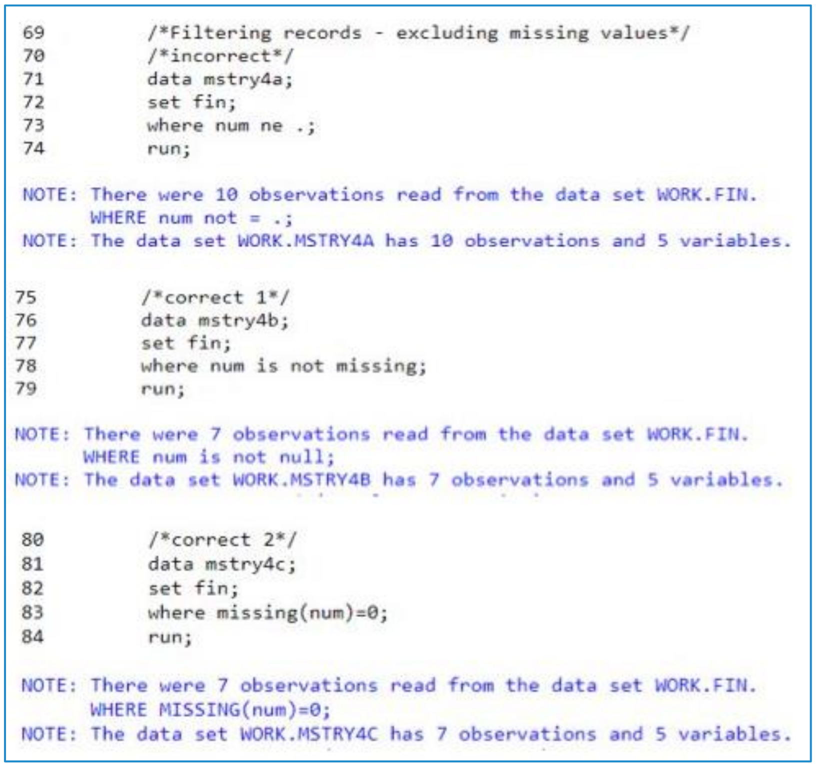


```
/*Filtering records - excluding missing values*/
/*incorrect*/
data mstry4a;
    set fin;
    where num ne .;
run;
/*correct 1*/
data mstry4b;
    set fin;
    where num is not missing;
run;
/*correct 2*/
data mstry4c;
    set fin;
    where missing(num)=0;
run;
```

Figure 8. Screenshot of Code for Creating Dataset without Numeric Missing Values

This task involves creating a dataset excluding observations with missing values and newcomers always proceed coding as shown by the red arrow in Figure 8. This approach results in the exclusion of only one type of missing value, “.”. The remaining 27 numeric missing values are retained in the dataset. This error can be rectified by employing the MISSING function in couple of ways as shown in Figure 8 (correct 1 and correct 2).

It is always good practice to look at logs when executing snippets of code to get an idea of what has been processed during the development phase. From the log in Figure 9, it can be observed that the first dataset (mstry4a) retained 10 observations whereas the remaining datasets (mstry4b and mstry4c) retained only 7 observations. The reasoning behind this difference is due to the fact that all types of numeric missing values have not been excluded.

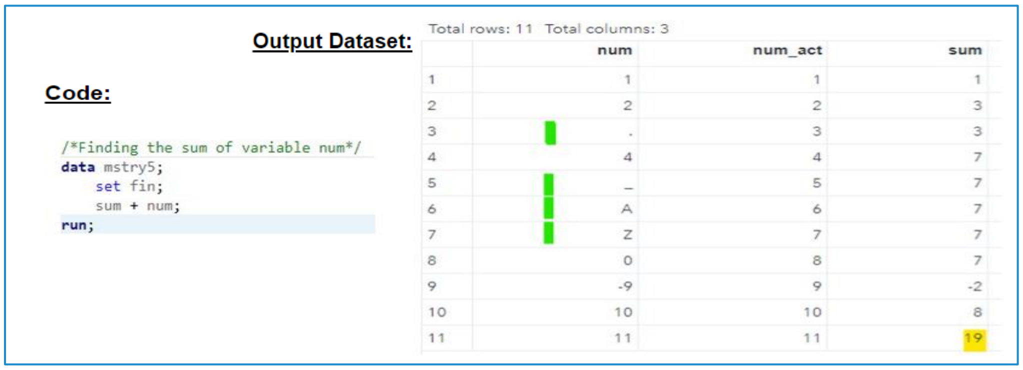


```
69          /*Filtering records - excluding missing values*/
70          /*incorrect*/
71          data mstry4a;
72          set fin;
73          where num ne .;
74          run;

NOTE: There were 10 observations read from the data set WORK.FIN.
      WHERE num not = .;
NOTE: The data set WORK.MSTRY4A has 10 observations and 5 variables.

75          /*correct 1*/
76          data mstry4b;
77          set fin;
78          where num is not missing;
79          run;

NOTE: There were 7 observations read from the data set WORK.FIN.
      WHERE num is not null;
NOTE: The data set WORK.MSTRY4B has 7 observations and 5 variables.

80          /*correct 2*/
81          data mstry4c;
82          set fin;
83          where missing(num)=0;
84          run;

NOTE: There were 7 observations read from the data set WORK.FIN.
      WHERE MISSING(num)=0;
NOTE: The data set WORK.MSTRY4C has 7 observations and 5 variables.
```

Figure 9. Screenshot of Log for Creating Datasets without Numeric Missing Values

Scenario 5:

Task: Find the sum of variable "num".



Code:

```
/*Finding the sum of variable num*/
data mstry5;
    set fin;
    sum + num;
run;
```

Output Dataset:

Total rows: 11 Total columns: 3

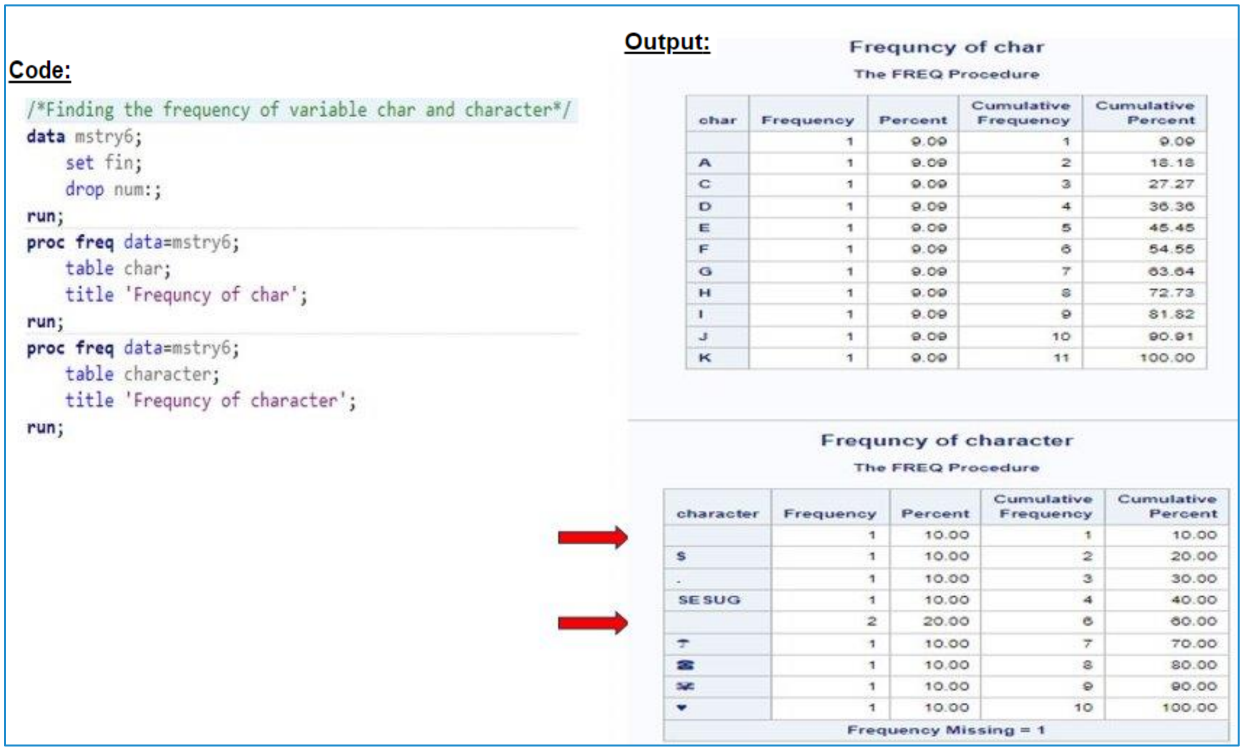
		num	num_act	sum
1		1	1	1
2		2	2	3
3	🟩	.	3	3
4		4	4	7
5	🟩	_	5	7
6	🟩	A	6	7
7	🟩	Z	7	7
8		0	8	7
9		-9	9	-2
10		10	10	8
11		11	11	19

Figure 10. Screenshot of Code and Output Dataset for Sum of a Variable

The sum of the variable "num" can be determined using the sum statement in SAS and it is always good to know how this is handled in the presence of various numerical missing values. Figure 10 shows the code as well as the output dataset created using the sum statement. Note that the cumulative value is retained when any of the 28 numeric missing values are present as highlighted by green bars. The value of 19, highlighted in yellow is the sum of the values of variable "num" ignoring any type of numeric missing values during the retaining step. This information is valuable as this can be applied per one's need.

Scenario 6:

Task: Find the frequency of variables “char” and “character”.



Code:

```
/*Finding the frequency of variable char and character*/
data mstry6;
    set fin;
    drop num:;
run;
proc freq data=mstry6;
    table char;
    title 'Frequncy of char';
run;
proc freq data=mstry6;
    table character;
    title 'Frequncy of character';
run;
```

Output:

Frequncy of char

The FREQ Procedure

char	Frequency	Percent	Cumulative Frequency	Cumulative Percent
	1	9.09	1	9.09
A	1	9.09	2	18.18
C	1	9.09	3	27.27
D	1	9.09	4	36.36
E	1	9.09	5	45.45
F	1	9.09	6	54.55
G	1	9.09	7	63.64
H	1	9.09	8	72.73
I	1	9.09	9	81.82
J	1	9.09	10	90.91
K	1	9.09	11	100.00

Frequncy of character

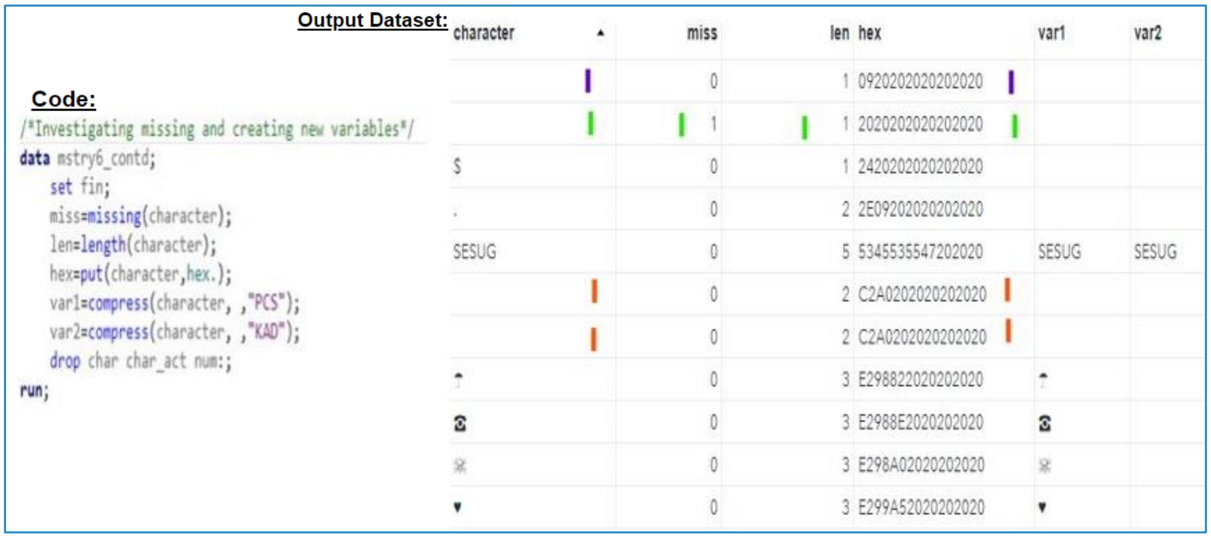
The FREQ Procedure

character	Frequency	Percent	Cumulative Frequency	Cumulative Percent
	1	10.00	1	10.00
$	1	10.00	2	20.00
.	1	10.00	3	30.00
SE SUG	1	10.00	4	40.00
	2	20.00	6	60.00
〒	1	10.00	7	70.00
☎	1	10.00	8	80.00
✂	1	10.00	9	90.00
▼	1	10.00	10	100.00
Frequency Missing = 1				

Figure 11. Screenshot of Code and Output for Frequency of Character Variables

Unlike numeric missing values, character missing values are officially represented by single value which is nothing but a blank. This makes coding easier for a novice programmer and the code for finding the frequency of character variables is shown in Figure 11. The frequency of variable “char” seems correct but the frequency output for the variable “character” is misleading as shown under the title “Frequency of character”. Note that there are two levels of blanks (highlighted in red arrow marks) in addition to the explicit statement of frequency missing. Comparing it with the source dataset in Figure 1, it can be inferred that the blanks are not purely blanks, basically non-blanks. In this data-driven world, data is being sourced from various sources and these occurrences can happen and hence it is aways good to check for these types of missing values before marching ahead.

The MISSING function can be used to detect missing values in character variables, but it can only capture pure blanks as shown by the second observation in the output dataset in Figure 12. For investigative purposes, we can use the LENGTH statement to find out how many characters are associated with the values. The “len” variable in the output dataset (Figure 12) shows that there are blanks with lengths 1 and 2 which suggests that the blanks are not real blanks. The best way to investigate is to convert the variable into a hexadecimal format using the “HEX.” format as shown in the code in Figure 12. This format converts strings/numbers to a hexadecimal representation as shown by the variable “hex” in the output dataset. It can be seen that the blanks have different values in the hexadecimal representation as shown by various colored bars (purple, green and red) in Figure 12. One will have to decide how character missing values are defined in such cases. For illustrative purposes, we have created two new variables (var1 and var2) using the COMPRESS function to show how the character missing values can be defined. The COMPRESS function is basically used to remove special characters from a string. The function comes with various optional parameters (modifiers) which can cater to various needs. In the code presented in Figure 12, the modifier" PCS" in the variable "var1" removes all punctuation (P), control characters(C) and space(S) characters whereas the modifier "KAD" in variable "var2" keeps(K) all alphabetic characters (A) and digits (D).



Output Dataset:

character	miss	len	hex	var1	var2
	0	1	0920202020202020		
	1	1	2020202020202020		
$	0	1	2420202020202020		
.	0	2	2E09202020202020		
SESUG	0	5	5345535547202020	SESUG	SESUG
	0	2	C2A0202020202020		
	0	2	C2A0202020202020		
☂	0	3	E298822020202020	☂	
☎	0	3	E2988E2020202020	☎	
☠	0	3	E298A02020202020	☠	
♥	0	3	E299A52020202020	♥	

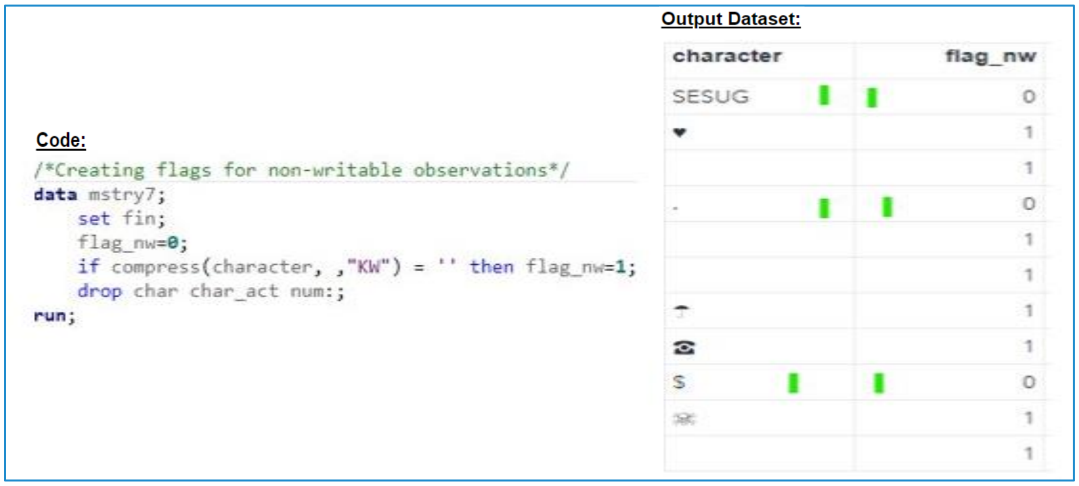
Code:

```
/*Investigating missing and creating new variables*/
data mstry6_contd;
    set fin;
    miss=missing(character);
    len=length(character);
    hex=put(character,hex.);
    var1=compress(character, ,"PCS");
    var2=compress(character, ,"KAD");
    drop char char_act num:;
run;
```

Figure 12. Screenshot of Code and Output Dataset for Handling Character Missing Values

Scenario 7:

Task: Create a flag for observations with non-writable characters associated with variable "character".



Output Dataset:

character	flag_nw
SESUG	0
♥	1
	1
.	0
	1
	1
☂	1
☎	1
$	0
☠	1
	1

Code:

```
/*Creating flags for non-writable observations*/
data mstry7;
    set fin;
    flag_nw=0;
    if compress(character, ,"KW") = '' then flag_nw=1;
    drop char char_act num:;
run;
```

Figure 13. Screenshot of Code and Output Dataset for Creating a Flag for Non-writable Characters

The non-writable characters can be detected using the COMPRESS function described in scenario 6. The relevant code and the output dataset created is shown in Figure 13. Note that the modifier "KW" used in the COMPRESS function means keeping(K) all writable characters(W). The observations with green bar shown in Figure 13 (flag_nw=0) indicate observations which are writable or printable whereas the rest of the observations are non-writable and flagged as 1 (flag_nw=1). The flag created in this task can be applied to subset the data or used for other categorization purposes.

CONCLUSION

Handling missing values is an important part of data wrangling and this article has dealt with the operational aspect of both numeric and character missing values in SAS which will aid in developing sound handling methods. In this paper, we have shown a few scenarios where a beginner can overlook some of the aspects of missingness while coding. The scenarios included both the incorrect and the correct approach detailed with code, output and datasets. These presented scenarios should hopefully begin to remove the curtain of MISSING mysteries in SAS.

REFERENCES

SAS Institute, Inc. Missing Values. Working with Missing Values. Most recently accessed on August 22, 2023.
https://documentation.sas.com/doc/en/lrcon/9.4/p175x77t7k6kggn1io94yedqagl3.htm

Reading, Pamela. 2015. "The COMPRESS Function: Hidden Superpowers". Proceedings of 2015 Southeast SAS Users Group (SESUG) Conference. Savannah, GA. Available at https://www.lexjansen.com/sesug/2015/138_Final_PDF.pdf

CONTACT INFORMATION

Your comments/questions/criticisms are valued and encouraged. Please contact the author at:

Jinson Erinjeri
E-mail: jinson_je@yahoo.com