Missing value hell

Steve Simon

2022-07-04

There’s no mincing words here. Missing values are hell for every statistician. Here are a few examples.

### The null license plate

Joseph Tartaro thought it would be funny to get the following California vanity license plate.

![Figure 1. Null license plate](data:text/html;base64,)

Figure 1. Null license plate

The joke is that NULL is the missing value code used by many databases. But, as described in Wired Magazine, his life turned into a living hell. Police officers will always write the license plate number of any car that was caught speeding or parking illegally. Well, almost always. In the very rare cases that the officer forgets, this is converted to a NULL value in the database. This is then matched to Joseph Tartaro and the ticket was sent his way. He got tickets from all over California–Corona, Fresno, Rancho Cucamonga–mostly to places he’d never been. These tickets were written for all sorts of different automobiles that he had never driven–Mercedes, Toyotas. He got hundreds of tickets worth tens of thousands of dollars. It was not an easy problem to fix.

This is caused by a programming error. The four letters N-U-L-L are not the same thing as the internal code of Null in a database. But a programmer writes a statement like

if var="NULL" then ...

and everything goes to hell.

### The $9,700 bargain

A while back I was browsing through the geekier parts of Office Max and noticed the following tag.

![Figure 2. Printer pricing tage](data:text/html;base64,)

Figure 2. Printer pricing tage

The computer database used the common practice of using all 9’s to denote a missing value. That’s fine, but then whoever programmed the calculation of how much saving you would get by buying the printer at $299.99 dollars instead of the actual price, figured through a simple subtraction that you would save a whopping $9,700!

### How to avoid missing value problems

Whenever I see stories like this, I think of the quote from John Bradford, a sixteenth century preacher–“There but for the grace of God go I.”

How easy it is to forget to account for missing values. To avoid this, you must develop an obsession about missing values. You must track their number through every version of every dataset. This number can grow, if you are not careful, so monitor it with a religious fervor.

If you recode any variable, always account for missing values first and last. So start off any recodes with the case where the original variable is missing and end with a catch-all code. Consider a variable for gender with 1 for male, 2 for female, and 9 for unknown. Then a recode in SAS might look like

if gender=9 then gender\_label="Unknown";  
 else if gender=1 then gender\_label="Male";  
 else if gender=2 then gender\_label="Female";  
 else gender\_label="Coding error";

The syntax changes slightly with other statistical packages, but the general concept applies. Account for missing value codes first and then catch anything that fails to match at the end.

### Hunting for missing values

Every statistical package has a different internal code used for missing values, and this can cause problems if you are not aware of them.

SAS stores the missing value as the most extremely negative number that could be stored in floating point precision. You are not likely to have extremely negative values in a real application, so this protects you from a legitimate data value being mistaken for a missing value.

But there’s a trap waiting for you. Consider the following statement:

if age<=18 then child="Yes";  
 else child="No";

The child variable would be “Yes” for any 18 or younger or anyone with a missing age. That’s probably not what you wanted.

In Stata, the missing value is coded as the largest possible floating point number. Again, a reasonable choice, but now the child variable would be “No” for all adults and all missing values.

In R, comparisons involving missing values are handled quite differently. If you test whether a missing age is less than or equal to 18, it won’t tell you it is true, like SAS does. It won’t tell you false, like Stata does. It returms missing, which you can interpret as “I don’t know if this statement is true or false.” This seems fair enough, but it does cause problems with equality checks.

The statement

10==NA

returns a missing logic value. If the right hand side of the comparison is unknown, it might equal 10 if it were known, but it might not. So a missing logic value seems reasonable. Likewise

NA==10

returns a missing logic value because the left hand side of the comparison is unknown. What’s totally surprising is that

NA==NA

also returns a missing logic value. The left hand side could be anything 10, 82, 53. The right hand side could be anything, 23, 10, 92. So maybe the two unknown values are equal and maybe they are unequal. You can’t be sure, so you get a missing logic value.

The solution in all of these cases is to hunt for the missing values explicitly. SAS and Stata have a missing function. In R, you check for missing with the is.na function. Use these functions aggressively in any code that might have missing values.

### Conclusions

Missing values are one of the most troublesome aspects of data management. Every statistical package stores missing values differently. Be aware of how your package treats missing values, use the correct functions detecting missing value codes, and account for missing values at the start and at the end of any recodes.