

Missing Data

In its default setting, RESAMPLING STATS recognizes a period "." as missing data. In the workings of the program, this is represented by a very large negative number (larger than any you can define for the program). Thus, you can use relational operators ">", "<" and "=" with missing values. Missing values are equal only to other missing values. They are less than any non-missing value.

RESAMPLING STATS, as an option in data import and export, can interpret any single number as a missing data code. For example, suppose your data has **-99999** as a code for missing entries.

The following option included in the **READ** command will so instruct RESAMPLING STATS:

READ file "mydata" missing -99999 a b c

Any non-valid values (such as blank fields, letters or other non-numerical characters) are automatically treated as missing. Similarly, you can specify how RESAMPLING STATS identifies missing data in the files it writes:

WRITE file "mydata" missing -99999 a b c

You can specify missing values in commands where it makes sense:

SET 10 a Create a vector of length 10, each element "missing."

What happens to missing values in program operations?

When adding, subtracting, multiplying or dividing vectors, the presence of a missing element causes the result to be missing. For example:

Vector a: 2 4 . 6

Vector b: 6 . 4 2

Multiplying a x b yields the vector 12 . . 12

SUMming a vector containing a missing element causes the result to be missing:

SUM a yields the single element vector (scalar): "."

Dividing by zero also yields "." (missing) as the result.

What should you do with missing data?

In most cases you will want to cleanse your data of records with missing values prior to using it in RESAMPLING STATS. Otherwise, tests may not make sense, or your trials may contain unequal numbers of observations. Consider the question of deciding whether two pig rations produce the same weight gains (a two-sample “permutation” or “randomization” test):

Table 16.

Ration A:	32	.	29	35	29	31	31	27	.	.	38
Ration B:	34	33	.	36	34	.	.	29	29	34	39

To test the proposition that the two rations produce weight gains that are similar, we toss all the weight gains into a hat and shuffle them. Then we select, with replacement, random groups of 11, recording how often the two groups’ total weight gains diverge as much as do the observed groups.

As you can see, the observed data vectors each have 3 missing elements. When we begin resampling, however, there is no guarantee that the two “resample” vectors will also have 3 missing elements each. They could have anywhere from 0 to 6 missing elements.

The statistic we are interested in — the total weight gains for each group — will depend not simply on the value of the elements selected, but on whether any missing values are selected. Selecting more missing values in one test vector than in the other will yield totals that cannot be used for comparison. You could get around this particular problem by using the mean, but there are other statistics we might be interested in (the range, for example) where we must have equal numbers of observations. Keep in mind that there are two places where the problem of unequal numbers of observations affects your procedures:

- 1 Possible variation in the number of missing elements between vectors within one trial (as above); and
- 2 Possible variation in the number of elements on which a trial is based.

You should consider the effect of missing data in your initial program on both.

Cleaning data when the observations are associated

In some cases, observations are associated with each other and it is necessary to remove related observations when one of the associated values is missing. This is done automatically in the **REGRESS** (regression), **CORR** (correlation), **SUMABSDEV** and **SUMSQRDEV** (sum of absolute and squared deviations) commands; the following procedure can be used in those few instances where you may need to do it elsewhere.

Table 17.

Vector A:	1	3	4	.	7	5	10	8	6	9	.
Vector B:	1	2	3	4	5	6	.	8	.	9	10
Vector C:	3	.	2	1	9	7	10	6	5	4	8

MULTIPLY a b c abc

The vector **abc** will contain 5 missing elements, since multiplying a missing element by anything yields a missing element. We then **RECODE** all non-missing elements to 1:

RECODE abc <>. 1 cleaner

(Recall that "<>" means "not equal to")

Then, we multiply each of the original vectors by this new vector **cleaner**. That leaves each with their original data, except with missing values wherever **any** of the original vectors had a missing value. You can then **WEED** the original vectors to get rid of the missing values.