The Stata Journal (2003)
3, Number 4, pp. 446–447

Stata tip 2: Building with floors and ceilings

Nicholas J. Cox, University of Durham, UK
n.j.cox@durham.ac.uk

Did you know about the `floor()` and `ceil()` functions added in Stata 8?

Suppose that you want to round down in multiples of some fixed number. For concreteness, say, you want to round `mpg` in the auto data in multiples of 5 so that any values 10–14 get rounded to 10, any values 15–19 to 15, etc. `mpg` is simple, in that only integer values occur; in many other cases, we clearly have fractional parts to think about as well.

Here is an easy solution: `5 * floor(mpg/5)`. `floor()` always rounds down to the integer less than or equal to its argument. The name floor is due to Iverson (1962), the principal architect of APL, who also suggested the expressive $\lfloor x \rfloor$ notation. For further discussion, see Knuth (1997, 39) or Graham, Knuth, and Patashnik (1994, chapter 3).

As it happens, `5 * int(mpg/5)` gives exactly the same result for `mpg` in the auto data, but in general, whenever variables may be negative as well as positive, *interval* `*` `floor(`*expression*`/`*interval*`)` gives a more consistent classification.

Let us compare this briefly with other possible solutions. `round(mpg, 5)` is different, as this rounds to the nearest multiple of 5, which could be either rounding up or rounding down. `round(mpg - 2.5, 5)` should be fine but is also a little too much like a dodge.

With `recode()`, you need two dodges, say, `-recode(-mpg,-40,-35,-30,-25,-20,-15,-10)`. Note all the negative signs; negating and then negating to reverse it are necessary because `recode()` uses its numeric arguments as upper limits; i.e., it rounds up.

`egen, cut()` offers another solution with option call `at(10(5)45)`. Being able to specify a *numlist* is nice, as compared with spelling out a comma-separated list, but you *must* also add a limit, here 45, which will not be used; otherwise, with `at(10(5)40)`, your highest class will be missing.

Yutaka Aoki also suggested to me `mpg - mod(mpg,5)`, which follows immediately once you see that rounding down amounts to subtracting the appropriate remainder. `mod(,)`, however, does not offer a correspondingly neat way of rounding up.

The `floor` solution grows on one, and it has the merit that you do not need to spell out all the possible end values, with the risk of forgetting or mistyping some. Conversely, `recode()` and `egen, cut()` are not restricted to rounding in equal intervals and remain useful for more complicated problems.

Without recapitulating the whole argument insofar as it applies to rounding up, `floor()`'s sibling `ceil()` (short for ceiling) gives a nice way of rounding up in equal intervals and is easier to work with than expressions based on `int()`.

References

Graham, R. L., D. E. Knuth, and O. Patashnik. 1994. *Concrete Mathematics: A Foundation for Computer Science*. Reading, MA: Addison–Wesley.

Iverson, K. E. 1962. *A Programming Language*. New York: John Wiley & Sons.

Knuth, D. E. 1997. *The Art of Computer Programming. Volume I: Fundamental Algorithms*. Reading, MA: Addison–Wesley.