The Stata Journal (2007)
7, Number 3, pp. 434–435

Stata tip 48: Discrete uses for uniform()

Maarten L. Buis
Department of Social Research Methodology
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
m.buis@fsw.vu.nl

The `uniform()` function produces random draws from a uniform distribution between 0 and 1 ([D] **functions**). `uniform()` is an unusual function. It takes no arguments, although the parentheses are essential to distinguish it from a variable name, and it returns different values each time it is invoked—as many as are needed. Thus, if `uniform()` is used to generate a variable, a different value is created in each observation. This Stata tip focuses on one of its many uses: creating random draws from a discrete distribution where each possible value has a known probability.

A uniform distribution between 0 and 1 means that each number between 0 and 1 is equally likely. So the probability that a random draw from a uniform distribution has a value less than 0.5 is 50%, the probability that such a random draw has a value less than 0.6 is 60%, and so on. The first example below shows how this fact can be used to create a random variable, where the probability of drawing 1 is 60% and that of drawing 0 is 40%. (Kantor and Cox [2005] give a tutorial on the `cond()` function.)

```
gen draw = cond(uniform() < .6, 1, 0)
```

The same result can be achieved even more concisely, given that in Stata a true condition is evaluated as 1 and a false condition as 0 (Cox 2005). `uniform() < .6` is true, and thus evaluated as 1, whenever the function produces a random number less than .6 and is false, and thus evaluated as 0, whenever that is not so.

```
gen draw = uniform() < .6
```

The probability need not be constant. Suppose that the probability of drawing 1 depends on a variable `x`. We can simulate data for a logistic regression with a constant of -1 and an effect of `x` of 1. In this example, the variable `x` contains draws from a standard normal distribution.

```
gen x = invnorm(uniform())
gen draw = uniform() < invlogit(-1 + x)
```

Nor is this method limited to random variables with only two values. Consider a distribution in which 1 has probability 30%, 2 probability 45%, and 3 probability 25%.

```
gen rand = uniform()
gen draw = cond(rand < .3,  1, cond(rand < .75, 2, 3))
```

A special case of this distribution is one for `k` integers, say, 1 to `k`, in which each value is equally likely. Suppose that we simulate throwing a six-sided die, so that values from 1 to 6 are assumed to have probability 1/6. So we need to map all values up to

pr0032

1/6 to 1, those up to 2/6 to 2, and so forth. That goal is easily achieved by multiplying by 6 and rounding up, using the ceiling function `ceil()`. Other applications of `ceil()` were discussed in Stata tip 2 (Cox 2003).

```
gen draw = ceil(6 * uniform())
```

We can use the same principle to simulate draws from a binomial distribution. Recall that a binomial distribution with parameters `n` and `p` is the distribution of the number of "successes" out of `n` trials when the probability of success in each trial is `p`. One way of sampling from this distribution is to do just that; i.e., draw `n` numbers from a uniform distribution, declare each number a success if it is less than `p`, and then count the number of successes (Devroye 1986, 524). Here Mata is convenient. Its equivalent to `uniform()`, `uniform(r, c)`, creates an `r` × `c` matrix filled with random draws from the uniform distribution. Thus we can create a new variable, `draw`, containing draws from a binomial(100, .3) distribution:

```
mata:
n = 100
p = .3
draw = J(st_nobs(),1,.)              // matrix to store results
for(i = 1; i <= rows(draw); i++) {  // loop over observations
    trials = uniform(1, n)           // create n trials
    successes = trials :< p          // success = 1 failure = 0
    draw[i,1] = rowsum(successes)    // count the successes
}
idx = st_addvar("int", "draw")
st_store(., idx, draw)               // store the variable
end
```

This code is deliberately spun to make its logit clear. Mata learners and experts alike might enjoy working out how to eliminate the loop and how to use fewer variables, while also pondering the possibility of a problem with memory demand for large datasets.

References

Cox, N. J. 2003. Stata tip 2: Building with floors and ceilings. *Stata Journal* 3: 446–447.

———. 2005. FAQ: What is true or false in Stata? http://www.stata.com/support/faqs/data/trueorfalse.html.

Devroye, L. 1986. *Non-Uniform Random Variate Generation*. New York: Springer.

Kantor, D., and N. J. Cox. 2005. Depending on conditions: A tutorial on the `cond()` function. *Stata Journal* 5: 413–420.