sample

syntax: `sample(n, datavector)` or `sample(n, urn)`

purpose: Takes `n` random samples, with replacement, from the specified dataset or urn.

To sample from a dataset, the data should be in the form of a simple vector, for example `[2.8 5.8 3.2 2 3.7 3.4]` or `1:10`. If your data set is a matrix, use RESAMP to sample it in a row-by-row fashion. In fact, you can use RESAMP for all resampling from datasets, either vectors or matrices.

For sampling from an urn, which is appropriate for simulation programs that need to sample from probability distributions, either of three forms can be used:

1. a matrix specifying multiplicities and values, e.g.

```
[  5 1;
  21 2;
  31 3]
```

   which consists of 5 ones, 21 twos and 31 threes. The multiplicities must be integers, but the values don't have to be.

2. a matrix specifying relative probabilities and values, e.g.

```
[ .05    1;
  .08532 2;
  .55    3]
```

   The relative probabilities are given in the first column and the corresponding values in the second. If the relative probabilities don't add up to 1, they will be scaled appropriately.

3. an urn created with the program URN. This is useful for creating complex, compound probability distributions. See the documentation of URN.

warning!: In forms (1) and (2) above, make sure to use the SEMI-COLON (and not the comma) if typing all the entries on one line, for example `[5 1; 21 2; 31 3]`.

for MATLAB experts:: Forms (1) and (2) can represent exactly the same information, but in different forms. Given a choice between forms

(1) and (2) — for instance if the probabilities are in percents — it's faster to use (1) rather than (2). For instance,

≫ `sample(100, [10 0; 10 1; 80 2])`

is generally faster than

≫ `sample(100, [.10 0; .10 1; .80 2])`

although the end result is the same. It's generally even faster to use `expand` to translate form (1) to a list of data points. (However, in those rare cases when you need a matrix that is very unbalanced, for instance, `[1000000 1; 1 2]`, it's better to use `[1000000.0 1; 1.0 2]` which will avoid the use of `expand` to create the entire extremely long set of points.)

see also: URN, RESAMP, EXPAND, NORMAL, UNIFORM, EXPONENTIAL