Coupon Collector Calculator (CCC)

See the PDF version of this README file if you want to view the mathematical symbols the way they were intended.

Let $X_1, X_2, \ldots$ be i.i.d. random variables taking values in $\{1, \ldots, N\}$. Let $p_j = P(X_1 = j)$. This represents a sequence of trials, where on each trial, we collect one of N possible coupons. The probability of collecting the j-th coupon on any given trial is p_j.

Fix $n \in \mathbb{N}$, nonempty $S \subset \{1, \ldots, N\}$, and $k \in \{1, \ldots, |S|\}$. The main purpose of CCC is to calculate

$$P(|\{X_1, \ldots, X_n\} \cap S| \geq k),$$

which is the probability of collecting at least k coupons from S in n trials.

To use CCC, save the script, `CCC.R`, and `README.md` to the same folder, then run the script in R.

`CCC.R`

When you run this script, it will load four functions, `combnPure`, `combnGE`, `coupExact`, and `coup`. The main function is `coup`. The usage details are below, but here is a brief summary.

`coup(p,n)` returns the probability of collecting all coupons by the n-th trial. Normalizing p is unnecessary. For example, $p = (0.2, 0.8)$ and $p = (1, 4)$ both represent a scenario with two coupons, the first of which is four times rarer than the second. You can also use single integers for p. For example, $p = 4$ represents four equally likely coupons.

`coup(p,n,k)` returns the probability of collecting at least k coupons by the n-th trial.

`coup(p,n,k,g)` returns the probability of collecting at least k coupons from S by the n-th trial. Here, `g` is a vector of length |S| whose components are the elements of S.

For example, suppose there are 15 equally likely coupons numbered 1 through 15. The probability of collecting, in 40 trials, at least 7 distinct coupons with numbers between 1 and 10 is `coup(15,40,7,1:10)`.

`combnPure`

Let S be a set and let `g` be a vector of length $|S|$ whose components are the elements of S. Let `k` be an element of $\{0, 1, \ldots, |S|\}$. Then `combnPure(g,k)` returns a list with $\binom{|S|}{k}$ items. Each item is itself a list of the elements in a particular subset of S. In this way, `combnPure(g,k)` lists all subsets of S of size `k`, including the empty set when `k` equals zero. Unlike the built-in function, `combn`, the behavior of `combnPure` is the same even when `length(g)` equals one or zero.

`combnGE`

This function is the same as `combnPure`, except it lists all subsets of size greater than or equal to `k`. The default value for `k` is zero, so that `combnGE(g)` returns all subsets of S, including the empty set.

`coupExact`

Let `p` be a vector of nonnegative real numbers with length $N \geq 2$. If they do not add up to one, then `coupExact` will normalize the vector to be a probability vector. If the user enters an integer $N \geq 2$ for `p`, then `coupExact` will replace it by the vector of length N whose every component is $\frac{1}{N}$.

Let `n` be any positive integer.

Let $S \subset \{1, \ldots, N\}$ be nonempty. Let `g` be a vector of length $|S|$ whose components are the elements of S. As a shortcut, entering zero for `g` is equivalent to taking $S = \{1, \ldots, N\}$.

Then `coupExact(p,n,g)` returns $P(\{X_1, \ldots, X_n\} = S)$. This function utilizes the formula described in [this paper](). The default value for `g` is zero, so that `coupExact(p,n)` returns the probability of collecting all coupons in `n` trials.

`coup`

Let `p`, `g`, and `n` be as above. Let `k` be an element of $\{1, \ldots, |S|\}$. As a shortcut, entering zero for `k` is equivalent to taking `k` equal to $|S|$.

Then `coup(p,n,k,g)` returns

$$P(|\{X_1, \ldots, X_n\} \cap S| \geq k),$$

the probability of collecting at least k coupons from S in n trials. The default values for `k` and `g` are both zero. Thus, `coup(p,n)` is the same as `coupExact(p,n)`, and `coup(p,n,k)` is the probability of collecting at least `k` coupons in `n` trials.