STA 220 - Data and Web Technologies for Data Analysis - Homework 1

The Coupon Collector's Problem

Beginning in the late 19th century, tobacco companies, such as the British manufacturer John Player and Son's, often included collector's cards in packets of cigarettes. Each packet contained one card, and a typical series would consist of 25 to 50 related cards. This gives rise to a version of the famous coupon collector's problem: how many packets does a consumer need to to buy in order to obtain a complete set of the cards/coupons?

Of course the number of purchases required is random, so a better question is:

> What is the probability distribution of the number of purchases required to get a complete set of coupons?

To answer this question, we will make a few (reasonable) simplifying assumptions. Let n represent the number of cards in a complete set.

a. Assume that each purchase is a random draw from the set of available cards.

b. Assume that the manufacturer produces the same number of each card in the collection.

c. Assume that the total number of cards manufactured is large enough that our purchases do not appreciably change the population of cards remaining be purchased.

The idea is that we are sampling randomly from an extremely large population in which the cards in the series appear in equal proportions. Because the sample size is very small relative to the size of the population, we can pretend that we are sampling *with replacement* from the series of cards.

There is an obvious direct way to simulate this experiment: draw one card at a time until we have at least one card of each type. Here is a function that uses this approach to simulate the number of purchases required to obtain one complete set:

```
coupon_slow <- function(n) {
    coupons <- rep(0L, n)
    while (any(coupons == 0L)) {
        new_coupon <- sample.int(n, 1)
        coupons[new_coupon] <- coupons[new_coupon] + 1
    }
    sum(coupons)
}
```

As discussed in class, one can speed things up by thinking more carefully about the problem. Let:

- X_1 be the number of draws needed to get our first card from the collection;
- X_2 be the number of additional draws needed to get a card different from the one we already have;
- X_3 be the number of additional draws needed to get a card different from the first two;
- etc.

Of course $X_1 \equiv 1$. Then, on our next purchase the probability that we get a card different from the first one is just $\frac{n-1}{n}$, and we will continue to buy packets with this same probability of "success" until we do get a

card different from the first one, with each purchase being independent of the last. This means that X_2 is a *geometrically distributed*[1] random variable, with $p_2 = \frac{n-1}{n}$.

Then, having obtained two distinct cards from the collection, we begin drawing to get a card different from the first two. At each attempt, the probability that we get a card different from the first two is $\frac{n-2}{n}$, and X_3, the number of draws required to achieve this goal is geometrically distributed with $p_3 = \frac{n-2}{n}$. Note also that X_3 is statistically independent of X_1 and X_2: the number of packets that we had to purchase to get 2 different cards has no influence on the distribution of the number of additonal packets we must buy to get our 3rd distinct card.

Continuing in this way, we see that the total number of purchase required to get a complete set of n distinct cards is

$$S = X_1 + X_2 + \cdots + X_n,$$

where $X_1, X_2, \ldots, X_n$ are independent and each X_k is geometrically distributed with

$$p_k = \frac{n - (k-1)}{n} = 1 - \frac{k-1}{n}.$$

This is useful in more than one way. For example, it is well known that a geometric random variable has expected value (or mean) $\frac{1}{p}$, and since the expected value of a sum of random variables is the sum of their expectations, we can easily calculate the expected value of S. Similarly, the variance of a geometric random variable is $\frac{1-p}{p^2}$, and because the variance of a sum of *independent* random variables is equal to the sum of their variances, we can also easily calculate the variance and hence the standard deviation of S.[2]

This formulation also allows us to simulate the number of purchases required to obtain a complete set of cards in a much more efficient way.

```
coupon_fast <- function(n) {
    p <- 1 - (0:(n-1))/n
    x <- rgeom(n, p)  ## R counts only failures
    n + sum(x)  ## so add n * 1 = n to get the total
}
```

Exercises

1. Write a function named `coupon_mean_sd` which computes the (theoretical) mean and standard deviation of S and use it compute the mean and standard deviation when $n = 25$, $n = 50$, and $n = 100$.
2. Use `system.time()` to compare the elaspsed times required for `coupon_slow()` and `coupon_fast()` to perform 10,000 replications of the coupon collectors problem for $n = 25$, $n = 50$, and $n = 100$. How many times slower is the slow function than the fast version? Does the speed ratio depend on n? (*Hint*: here and elsewhere in this assignment you should start with a smaller number of replications, say 100 or 1000, until you have everything worked out and debugged. Only do 10,000 replications after you have all the code working correctly.)
3. For $n = 50$, compute 10,000 simulated values of S. (Given the speed advantage, I would suggest that you use `coupon_fast()` for this, but you can use `coupon_slow()` if you prefer.)
    a. Compare the sample mean and standard deviation of your 10,000 values to the theoretical values. Do they agree reasonably well? (Note: a large discrepancy would suggest that you may have made an error either in computing the theoretical mean and standard deviation or in your simulation code.)

[1] We take the geometric distribution to represent distribution of the number of *trials* required to obtain the first success in a sequence of independent trials where the probability of a success in each trial is p. Some authors count the number of *failures* before the first sucess, so their geometric random variable is exactly one less than ours.

[2] In principal, we could also compute the exact distribution of S, but this would be beyond the scope of this class.

b. Use `hist()` to plot the estimated distribution of S. (Setting `nclass` to anything from 25 to 50 or so seems to work pretty well here. Use your judgement.)

c. Estimate the 10th, 50th, and 90th percentiles of S.

4. What if we were interested in the number of number of trials required to get m complete sets?

   Here is an adaptation of our earlier `coupon_fast()` function for this problem. Because of the additional bookkeeping required, its speed advantage over the slow version is much less pronounced.

```
mcoupon_fast <- function(n, m) {
    coupons <- rep(0L, n)  # Will count each one until we have m of that coupon, then stop
    incomplete <- (coupons < m)
    S <- 0
    while (any(incomplete)) {
        k <- sum(incomplete)    # Number of coupons with fewer than m accumulated so far
        x <- rgeom(1, k/n) + 1  # Success is getting any one of the coupons we still need
        S <- S + x              # Update number of purchases
        y <- sample.int(k, 1)   # Which of the not-yet-completed coupons did we draw
        coupons[incomplete][y] <- coupons[incomplete][y] + 1   # Increment incomplete coupon counts
        incomplete <- (coupons < m)
    }
    S
}
```

a. Add a second argument to `coupon_slow()` and modify the function to generate simulated values of S for this problem. Name your new function `mcoupon_slow()`.

b. Using either your `mcoupon_slow()` or my `mcoupon_fast()`, use 10,000 simulated replications of the experiment to estimate the mean, the standard deviation, and the 10th, 50th, and 90th percentiles of the number of purchases required to get $m = 2$ complete collections of a series of $n = 50$ coupons.

c. Repeat part (b) with $m = 3$.