

Estimating Population Size with Randomized Algorithms: The Mark-and-Capture Method

Agniva Chowdhury

Lecture 5: COMP 480/580
09/11/2025

Motivations

- **Real problem:** counting turtles in a pond.
- **Broader goal:** understand how randomized estimation works in algorithms and data structures.

How can we estimate the number of turtles in a pond (or items in a dataset) without counting them all, using random sampling and probability?

Our strategy

How can we estimate the number of turtles in a pond (or items in a dataset) without counting them all, using random sampling and probability?

- **Key idea:** capture, mark, release, recapture → use randomness to estimate population.

Step 1: First Capture

- Let n be the total number of turtles in the pond (unknown).
- In the first sampling, you catch K_1 turtles.
- You **mark** all K_1 turtles (e.g., with a tag) and then release them back into the pond.
- So, after this step, exactly K_1 turtles in the population of n are marked.
- The fraction of marked turtles in the whole pond is: $\frac{K_1}{n}$

Step 2: Mixing period

- You wait (say 10 days) so the marked turtles can mix uniformly with the rest of the population.
- **Assumption:** every turtle in the pond is equally likely to be captured in the next sampling.

Step 3: Second capture

- Now you capture another K_2 turtles from the pond.
- Some of these will be marked (recaptured), and some will not.
- Let M = number of marked turtles in the second capture.

Setup with indicator r.v.s

- Define indicator random variables:

$$X_i = \begin{cases} 1 & \text{if turtle } i \text{ is marked} \\ 0 & \text{otherwise} \end{cases}$$

- **After first capture:** $\sum_{i=1}^n X_i = K_1$
- **After recapture:** $M = \sum_{j=1}^{K_2} X_j$

Expectation of recaptures

- $\mathbb{P}(X_i = 1) = \frac{K_1}{n}$
- Using the linearity of expectation:

$$\mathbb{E}(M) = \mathbb{E}\left(\sum_{j=1}^{K_2} X_j\right) = \sum_{j=1}^{K_2} \mathbb{E}(X_j) = \sum_{j=1}^{K_2} \frac{K_1}{n} = \frac{K_1 K_2}{n}$$

Distribution of M

- Without replacement: $M \sim \text{Hypergeometric}(n, K_1, K_2)$

$$\mathbb{P}(M = m) = \frac{\binom{K_1}{m} \binom{n-K_1}{K_2-m}}{\binom{n}{K_2}}$$

- $\mathbb{E}(M) = K_2 \frac{K_1}{n}$
- $\text{Var}(M) = K_2 \cdot \frac{K_1}{n} \cdot \left(1 - \frac{K_1}{n}\right) \cdot \frac{n-K_2}{n-1}$

Distribution of M when $n \rightarrow \infty$

- Without replacement \equiv independent with replacement
- With replacement: $M \sim \text{Binomial}\left(K_2, \frac{K_1}{n}\right)$

$$\mathbb{P}(M = m) = \binom{K_2}{m} \left(\frac{K_1}{n}\right)^m \left(1 - \frac{K_1}{n}\right)^{K_2-m}$$

- $\mathbb{E}(M) = K_2 \frac{K_1}{n}$
- $\text{Var}(M) = K_2 \cdot \frac{K_1}{n} \cdot \left(1 - \frac{K_1}{n}\right)$

When the population is huge relative to the sample, the chance of “colliding” (drawing the same turtle twice without replacement) is negligible, so sampling without replacement behaves like sampling with replacement.

How to estimate n ?

- We already have

$$\mathbb{E}(M) = \frac{K_1 K_2}{n}$$

- Therefore, $n = \frac{K_1 K_2}{\mathbb{E}(M)}$
- If $M \approx \mathbb{E}(M)$, we have

$$n \approx \frac{K_1 K_2}{M}$$

- Lower variance \Rightarrow Better estimate

This is the Lincoln–Petersen estimator

Chernoff bound

Let X_1, X_2, \dots, X_n be independent Bernoulli random variables, and define $M = \sum_{i=1}^n X_i$ and $\mu = \mathbb{E}(M)$. Then, we have

$$\mathbb{P}(M \geq (1 + \delta)\mu) \leq \exp\left(-\frac{\delta^2\mu}{2+\delta}\right), \text{ for any } \delta \geq 0$$

and

$$\mathbb{P}(M \leq (1 - \delta)\mu) \leq \exp\left(-\frac{\delta^2\mu}{2}\right), \text{ for any } 0 \leq \delta \leq 1$$

Example 2

- A website wants to estimate how many unique users visited during the past month.
- Storing and counting every user ID is too expensive.

Example 2

1. First capture (Mark):

- On Day 1, sample a subset of users randomly (say, log 10,000 visitors).
- **Mark them by storing their IDs in a hash table.**

2. Second capture (Recapture):

- On Day 15, take another random sample of 10,000 users.
- Count how many of these are already in the stored hash table (say, 2000).

Example 2

3. Estimation:

- By analogy,

$$n \approx \frac{K_1 K_2}{M}$$

where $K_1 = 10\text{k}$, $K_2 = 10\text{k}$, and $M = 2\text{k}$.

- So, $n \approx \frac{10,000 \times 10,000}{2,000} = 50,000$

- Exactly the same math as counting turtles.
- Instead of physical tags, we “mark” users with a data structure.

Another example

- You have a hash table distributed across a network, but you don't know how many keys are stored in total.
- Table is spread over many servers
- Full enumeration is too expensive.

Other estimators

- **Schnabel estimator:** uses repeated capture–recapture samples (not just 2).
 - A weighted average of several Lincoln–Petersen estimates.
 - Increases accuracy by reducing variance.
- **Jolly–Seber model:** allows for births, deaths, immigration, and emigration in the population. In randomized algorithms literature, this is close in spirit to sliding-window / dynamic sketch models in streaming algorithms.
- **MLE / Bayesian CR models:** Fit full probabilistic models to the observed recapture counts.

Conclusions

- **Mark-and-Capture is a simple but powerful idea:** sample, tag, resample, and use overlap to estimate the whole.
- It shows how randomness can make *hard* counting problems feasible.
- Randomized estimation turns impossible measurements into practical solutions.

**Thank you
Questions?**