LAB 1 - Counting Fish

ER Deyle

Fall 2024; Marine Semester Block 3

Introduction

Welcome to your "first" Quantitative Fisheries Analysis Lab.

Fisheries Context

"How many fish are in the ocean?" is a fundamental question for managing fishing activity. Although it is an easy question to state, it is can be a rather difficult question to answer! Today, we will experiment with a simple approach of "mark" and "recapture". This technique is used one way or another in many fishing contexts from ponds to oceans. It's also used widely in other branches of ecology (aquatic and terrestrial alike). It's also a great way to experiment with some very basic building blocks of coding.

Computational Approach

In this lab, the computational approach will be described in detail through-out the assignment below. However, at the end of all this you will need to write a lab-report in Rmarkdown notebook. This will be most convenient if you start in an Rmarkdown notebook from the get-go! Download the Lab_1_template.Rmd file from OneDrive. As in Lab 0, rename it to the effect of "BI_521_LAB_1_YourName.Rmd." Also make sure to change the Author and Title in the header.

Task list:

- Experiment with Mark and Re-capture *In Plastico*.
- Read about mark and re-capture techniques and statistics.
- Experiment with Mark and Re-capture *In Silico*.
- Use Rmarkdown to make a lab report.

In Plastico Experiment

Materials:

- Let's Go Fishin' Game.
- Small plastic bands.
- Excel or other data entry method.

Design Need:

- Use a timer to standardize effort.
- Replicate every experiment.
- Experiment with different true population sizes.

Collect Data

For each experiment, record the mark and recapture results, as well as the true population size. Record your data in Excel/Google Sheets. Then export the sheet as a "csv" file, and use `read.table()` to import the data into R.

Perform Statistical Analysis

Discuss with your peers how to translate the mark and recapture data into estimates of population size.

TASK: Estimate the size of your fish population. How does this compare to the true population size?

In Silico Experiment

In our *In Plastico* experiment, the population sizes were quite small, even in the XL Deep-Sea edition! Now we're going to generate mark-recapture data on the computer; this will allow us to experiment with much larger population sizes and much larger sets of replicates. First we will build a short script to simulate a single *in silico* mark-recapture experiment; we will then build it out using `for` loops to replay simulations and build a data set of replicates. Finally, we will introduce variation in parameters of the *in silico* mark-recapture to probe their quantitative workings.

Simulating a Single Mark-Recapture

There's rarely one single way to accomplish a computational task. Here's the basic ingredients we need:

- A way to represent a population of fish of some size, N.
- A way to simulate random selection of individuals in two independent samples.
- A way to track the overlap.

To get started, let's think of it this way. We will create a vector of length N. All entries will be "0" " to start. These represent untagged fish. Next we select a random sample of them and set their value to "1". This represents tagging in the initial *mark* phase. We then take a second random sample, and count what fraction of those values were tagged previously, i.e. the number of recaptures.

Now about generating that random sample. R is in many ways a stats package, and so has a lot of ways to generate random numbers. Type `?Distributions` to have a quick look. Lots of choices! Random number generators have existed much longer than computers, however. We know, for example, that Romans used "coin flips" to decide outcomes. Early on in computing, sequences of random numbers were actually pulled from tables that had been previously generated (from a physical process considered **RANDOM**). Later, people came up with various algorithms for generating **psuedo**-random numbers, which act like random numbers for any reasonable statistical application.

More on the history here: https://hal.inria.fr/hal-01561551/document

These produced uniform distributions either over the inters 0-9 or long strings of decimal numbers over the interval 0-1. In some sense, then, the most fundamental random number generator in R is `runif()`. Types `?runif`. Generate some random numbers!

```
runif(5)
```

```
## [1] 0.0238528 0.8135295 0.6630461 0.2850584 0.2194284
```

You shouldn't have gotten the same numbers as above. However, as mentioned above, these are **psuedo**-random numbers generated from an algorithm. Unlike a coin toss, the sequence can actually be repeated exactly if you know how. Generally, these algorithms will have a "seed", and if the seed is set to the same value at two different times, the same output will be generated. This is very useful if you want to have complete reproducibility of code even if you're simulating a stochastic process!

```
set.seed(617353)
runif(5)
```

```
## [1] 0.96869207 0.36163479 0.27939561 0.82995531 0.02229791
```

Okay it won't do to have everyone's RNG synced up though, so set your seed again and do it to the computer's internal clock.

```
set.seed(Sys.time())
runif(5)
```

```
## [1] 0.55614759 0.31053643 0.09511999 0.25510828 0.58393732
```

We can all be uniquely psuedo-random again! That was a long tangent. We don't actually need uniformly distributed random numbers; what we really want is a random selection of individuals from our population. If our population is size `N` this amounts to randomly sampling from the integers 1-`N` without replacement. We can use `sample.int()`. Type `?sample.int`. To sample 10 random integers from 100, we'd type:

```
sample.int(100,size=10,replace=FALSE)
```

```
##  [1] 67 89 30 60 13 96 91 75  7 99
```

Ok. Now we have all the ingredients we need to execute the plan. What was the plan again?

- We will create a vector of length N.
- All entries will be "0" " to start. These represent untagged fish.
- Next we select a random sample of them and set their value to "1". This represents tagging in the initial *mark* phase.
- We then take a second random sample, and count what fraction of those values were tagged previously, i.e. the number of recaptures.
- Start by picking the total population size `N` and the number selected in each "capture" to correspond to the *in plastico* experiment.

TASK: Using code described above, generate an equivalent data set *in silico* to what you generated *in plastico*. Are the results similar?

The `for` Loop

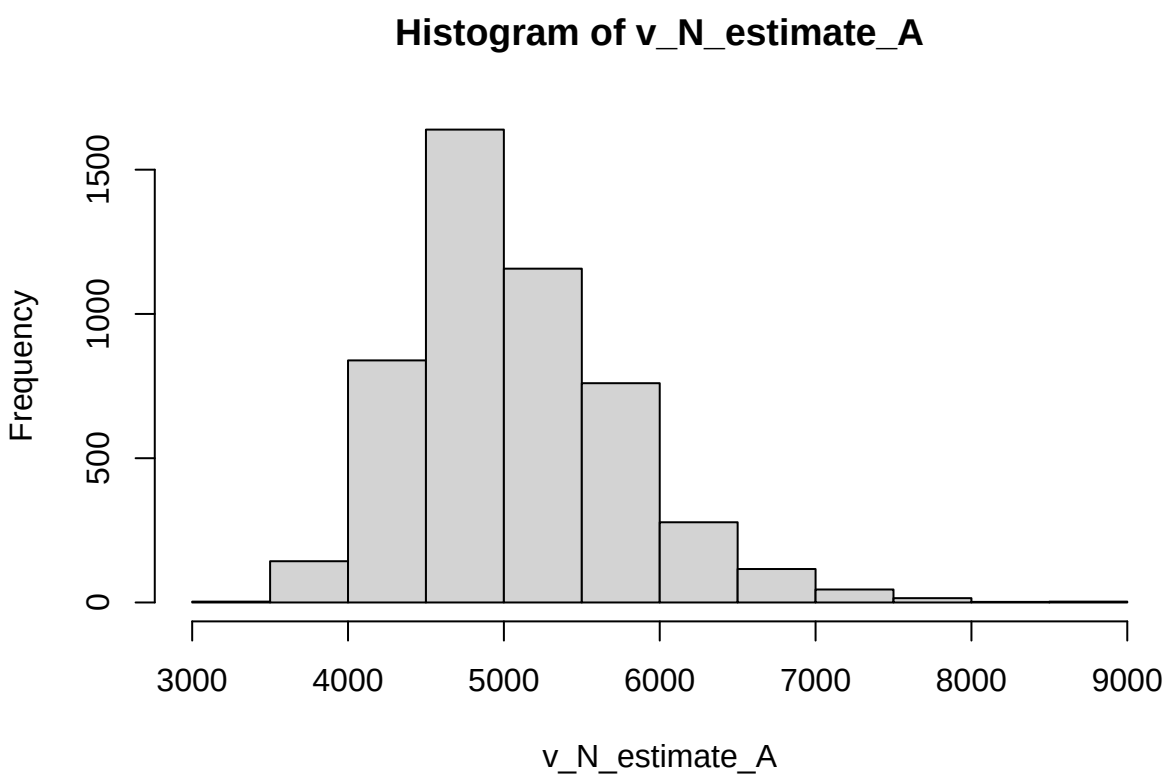
Above, we just stored the results in single variables. Now that we are trying to replicate, we need to store results in vectors or arrays, so that the values from each loop are preserved and organized.

TASK: Now repeat the *in silico* experiment many more times (let's say five hundred).

> What is your estimate of the population size? You should get something noticably larger than the actual population size. Try it again! Same result? What gives!?

We can repeat the experiment for different population or sampling parameters. For example, we can keep the total population size constant (N = 5000) and change the sample size in both the mark and recapture. For treatment "A", use m = K= 500; for treatment "B", use m = K = 1500.

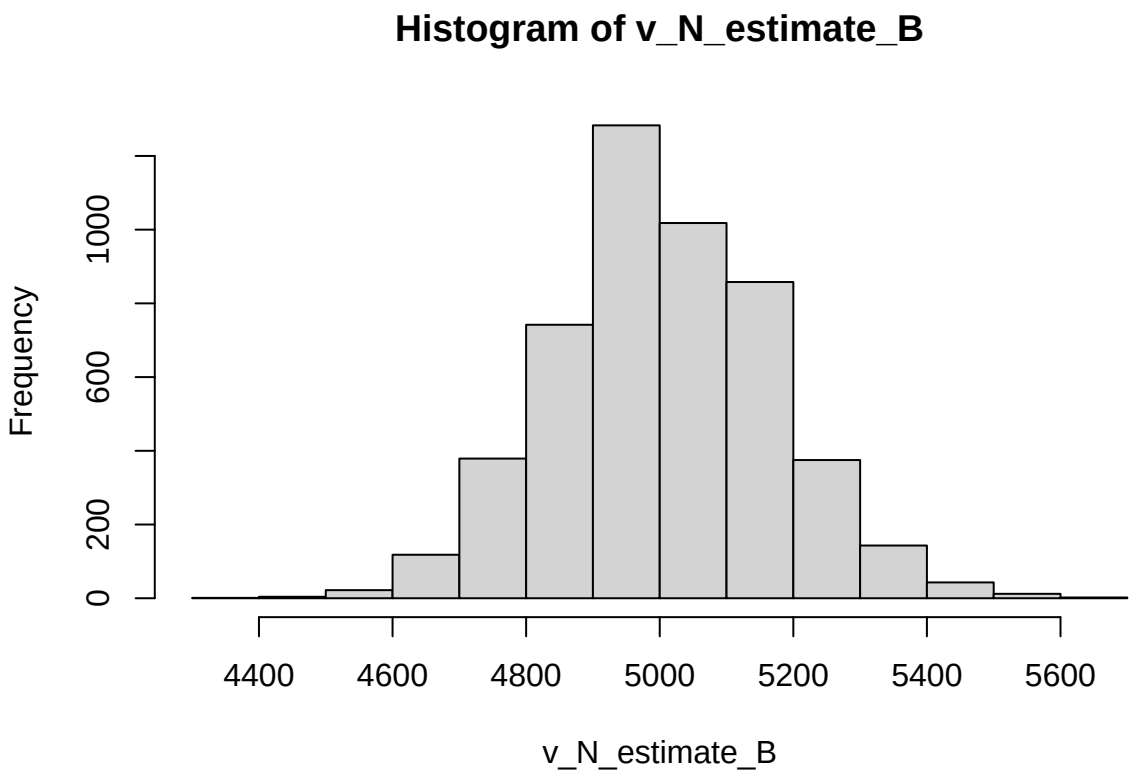
```
hist(v_N_estimate_A)
```




```
mean(v_N_estimate_A)
```

```
## [1] 5087.494
```

```
hist(v_N_estimate_B)
```




```
mean(v_N_estimate_B)
```

```
## [1] 5005.458
```

It turns out, this way of estimating the population size is biased! Let's look a little more.

CHALLENGE: Using a `for` loop to repeat the *in silico* mark-recapture process, investigate the following:

- How is the degree of bias affected by the size of the population, `N`? What happens if `N` is really large?
- How is the degree of bias affected by the fraction of the population (re)captured?
- Try this estimate of the population size instead:

$$\hat{N} = \frac{(m+1)(K+1)}{r+1} - 1$$

That proposed expression above is based on a Bayesian statistical derivation. If you're curious, see this wikimedia document:

https://en.wikipedia.org/wiki/Talk:Mark_and_recapture#Statistical_treatment

It should look unbiased!

Parametric variance

A natural question in any statistical application is "what's the confidence on this estimate?" Couldn't we just calculate the standard deviation? Yes and no. Yes we can calculate the standard deviation; but no it doesn't correctly answer the underlying question of confidence.

The statistics of recaptures in this simple mathematical treatment follow the "hypergeometric distribution". If you go back to `?Distributions`, you can find it in R! The commands are `dhyper()`, `phyper()`, `qhyper()`, and `rhyper()`. R knows a lot about statistics! In fact, the manual for the distribution functions gives the parametric form of the variance. R doesn't seem to know about fish, though. Something about white and black balls in an urn. If you look in other places (e.g. if you just looked up "Hypergeometric distribution" on Wikipedia), you might see the distribution written with slightly different variable choices.

If X $Hypergeometric(m, n, K)$ where m is the number of success states (marked fish/white balls), n is the number of fail states (unmarked fish/black balls), and K is the number of selections (captures in second phase) then the variance in X (recaptures in second phase) is:

$$Var(X) = K\frac{m}{m+n}\frac{n}{m+n}\frac{m+n-K}{m+n-1}$$

Except... this is the variance on the number of recaptures! We want to know the variance on the estimate of population size! Going back to the Bayesian estimate, the variance on the mean of the estimated N is

$$\sigma^2 = \frac{(m-1)(K-1)}{r-2}\frac{(m-r+1)(K-r+1)}{(r-2)(r-3)}$$

Okay, great, there's algebraic expressions for an unbiased estimator of N and associated variance. Deriving this, or even following the derivation, is a big ask, however. Now, non-parametric statistics is sometimes a term spoken in hushed terms through the halls. However, here's a secret. *If you have a computer, non-parametric statistics are often much easier to calculate!*

Boot-strap variance

A simple non-parametric estimate of variance on the mean is to use a "boot-strap". The variance is supposed to reflect the distribution of estimates if you repeat the calculation for many different realizations of the same process. Wait did you say "repeat the calculation"? Hey this is what computers do best! Boot-straps, specifically use re-sampling of values to estimate variances. Say we have 100 measurements that we think are normally distributed with unknown mean and unknown variance.

```
v_observations <- rnorm(100,mean=14.5,sd=2.5)
mean(v_observations)
```

```
## [1] 14.42529
```

We recalculate the mean by drawing a "boot-strap sample" from our observed sample. Check `?sample`. It's important to do this **with replacement**.

```
v_bootstrap <- sample(v_observations,100,replace=TRUE)
mean(v_bootstrap)
```

```
## [1] 14.50975
```

If you check close, there are some repeats! Now repeat this many times, and calculate the variance on the estimated means.

```r
v_bootstrap_means <- vector(length=500)

for(i_ in 1:length(v_bootstrap_means)){
  v_bootstrap_i <- sample(v_observations,100,replace=TRUE)
  v_bootstrap_means[i_] <- mean(v_bootstrap_i)
}

var(v_bootstrap_means)
```

```
## [1] 0.07822067
```

This should be pretty close to the parametric answer 0.0625! Now, the variance is directly related to confidence intervals for the normal distribution, but does not hold for other distributions (including the hypergeometric). The same bootstrap resampling principle can also be used to calculate confidence intervals based on quantiles, but instead of calculating the variance of our bootstrap estimates of the mean, we calculate the 0.025 and 0.975 quantiles.

TASK: Compute estimates of the 0.025 and 0.975 quantiles for an *in silico* mark-recapture experiment.

> Use large enough numbers to avoid getting 0-recaptures (since this gives an undefined estimate of population size). Suggestion is 5000 total individuals and sample catches of 1000.
>
> Does the true population size fall within the confidence intervals?

Note that a true bootstrap approach only needs a single experiment to work. So in that case, each bootstrap generates a new estimate of population size from the same mark-recapture. This is useful in lab and field studies where producing true replicates isn't feasible. However, with our in-silico approach, we can just use iteration to create true replicates instead of boot-strap resamples to approximate replicates.

Optional Extensions

"Catchability" is a parameter (and concept) in fisheries analysis that mediates the relationship between catch rate and the true population size. It is "fish caught per fish available per effort unit and per time unit". It is generally treated as both species and size dependent, influenced by both biological and technological factors.
Example: Larger fish can swim out of trawls, small fish slip through nets, and medium fish are doomed!
Example: Schooling changes catchability, but schooling behavior depends on population thresholds in many species.

More reading:
- https://spo.nmfs.noaa.gov/sites/default/files/pdf-content/2006/1043/clark.pdf
- https://www.fao.org/3/y4593e/y4593e03.htm#bm03.1
- https://spo.nmfs.noaa.gov/sites/default/files/weinberg.pdf

Modify any combination of the *in plastico* experiment, *in plastico* statistical analysis, *in silico* experiment, and *in silico* analysis to examine how variation in catchability might bias estimates of population size in mark-recapture study. Questions you might try to address:
- If catchability increases with age, how might mark-recapture estimates differing between populations with lots of large individuals or lots of small individuals?
- Is catchability *in plastico* constant across the experiment? Does that matter?

Further Interest

There are extensive records of Mark-Recapture programs– a trove for an ambitious marine data scientist! Here's an example for sharks:

https://data.noaa.gov/dataset/dataset/cooperative-shark-mark-recapture-database-mrdbs1

Make a Lab Report

Your lab report should include all code chunks that address the TASK headings, provide answers to the questions within those tasks (including any figures you might need to support), and you should also make sure to provide response to any **Reflection Questions**.

Reflection Questions

- Both *in plastico* and *in silico* models of mark and recapture are highly simplified. What is sorely missing? What is missing but won't necessarily matter?