CRISP: Capture-Recapture Interactive Simulation Package

George Volichenko
Carnegie Mellon University
Pittsburgh, PA
gvoliche@andrew.cmu.edu

December 17, 2012

Contents

1 Executive Summary

Capture-recapture estimation is a technique commonly used to estimate population sizes without having to enumerate every single unit. This family of estimation methods heavily relies on a set of assumptions, most of which are not realistically satisfied, especially in the domain of estimating the size of population of a nation. I construct a simulation architecture that provides a very flexible way to study the adverse effects that violation of these assumptions can have on the final estimates. To demonstrate its functionality, two types of capture-recapture estimators - Petersen and Horvitz-Thompson, are compared. When capture probabilities vary as a function of some demographic variable (e.g. age), the Petersen estimator demonstrates consistently inaccurate estimates, while a Horvitz-Thompson styles estimator tackles this issue by adjusting for the covariate. More generally, the package can be used by the Census Bureau employees, and researches alike, to test the performance of various capture-recapture estimators given the specifics of their population of interest.

2 Introduction

Capture-recapture (CRC) estimation methods are widely used to estimate population sizes without resorting to the more expensive census enumeration. Alternatively, it can be used alongside the census to improve its accuracy. To be able to compute such an estimate, at least two capture events are required. A good illustration of this method is attempting to estimate the size of a fish population in a pond. A researcher would have to cast a net, count and mark all the fish caught. Then, after a short period of time, researcher goes back to the pond, casts the net again and counts the number of fish that were previously marked as well as the ones that appear for the first time. One of the CRC estimators can then be applied to estimate the fish population size. The U.S. census bureau utilizes CRC methodology by essentially following a similar process in select areas of the country. The second capture event is called the CCM (Census Coverage Measurement) and its purpose is to test the accuracy of the national census figures, which correspond to the first capture event.

Traditional CRC estimators make strong assumptions about the population of interest:

1. *Closed population.* The true population to be enumerated stays fixed over time. This assumption excludes the possibility of births, deaths, and migration during the generation of lists (capture events).
2. *Perfect matching.* Every individual observed at least once requires a complete capture history. In case of the CCM, people surveyed in the CCM (capture list 2) need to be matched with their records from the main census (list 1). There are numerous record linkage algorithms available, however, none of them guarantee that everyone's records will be perfectly matched.
3. *Homogeneity.* All the members of the target population are equally likely to be captured. The diversity of a national population makes this assumption seem like a crude oversimplification and, in this simulation, a lot of attention is paid to breaking this assumption by introducing covariates, such as age or race, for a more realistic heterogeneity structure.
4. *List independence.* Event of capture of an individual on one list has to be independent of his capture on any other list. This assumption can be violated with various list effects, which are implemented in the package under the umbrella term "correlation bias".

This and other assumptions are very often violated when it comes to real-life populations. As a result, such estimations and the corresponding confidence intervals can have questionable accuracy.

I construct a simulation framework that provides insight into the sampling distributions of CRC estimators under a variety of data generation scenarios. The scenarios of most interest are naturally the ones where the assumptions are intentionally violated.

3 Methods

3.1 Simulation Components

CRISP is implemented in R and C programming languages and consists of the following three main components:

1) *Capture Probabilities*
The two essential arguments that the user has to specify are the desired true population size (N) and number of capture events or lists (K). The $N \times K$ matrix of capture probabilities can then be generated. By default all of these probabilities are set to 0.5, however, there are multiple arguments that can be specified to include a heterogeneity or dependence structure.
First, the user specifies whether he is interested in presence of covariates. These can come from the list of three main covariates (age, race and gender) or can be user-defined. The package provides a great deal of flexibility for defining covariates and, more importantly, how they affect the capture probabilities.

The list independence assumption can be intentionally violated by including a correlation bias between lists. So far, the only type of list effects implemented is the behavioral effect, which makes the capture probability on each list contingent on the capture outcome of the previous list.

2) *Captures*
After generating capture probabilities for all of the individuals on every list, the actual capture process can be simulated. Essentially, there are $N \times K$ Bernoulli variables with known probabilities and the result is an $N \times K$ indicator matrix of 0's and 1's where 1 means that the particular individual was captured on that list.

3) *Estimator function*
The top-level estimator function carries out the previous two steps and actually applies the selected estimator to the capture matrix to come up with an estimate value. The built-in estimators so far are Petersen and Horvitz-Thompson estimators, however, any other estimator function should be compatible. For our purposes, it is important to understand the variability of estimations and so this top-level function includes functionality for replication. It is also possible to apply the estimator function over a range of true population sizes. In that case, the population attributes (covariates) are generated incrementally, such that when we go from a population of 5,000 to 6,000 the simulation only needs to generate new covariate values only for that extra 1,000 while keeping all the previous covariates.

3.2 Computing Issues

After implementing all of the functionalities in R, the run times were quite dissatisfactory, especially when the desired population is large and multiple covariates need to be generated. To alleviate the issue the capture matrix generation was implemented in the C programming language. The run time improvement was very substantial (see Figure 1). C deals with iterative procedures much more effectively since it is a compiled language as opposed to R, which is interpreted.

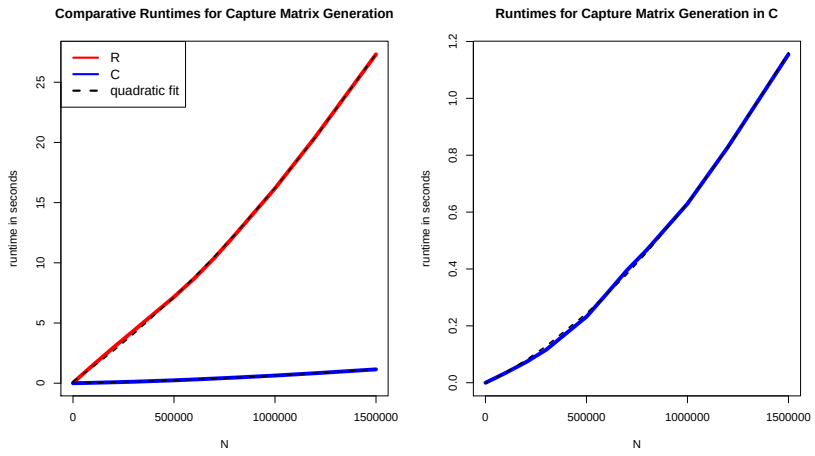


Figure 1: Comparison of run times produced by R and C scripts by N - pop size

Further Improvements

As a result of implementing these simulation components, especially given that two programming languages are interfaced, the code contained numerous unnecessary or excessive parts. Most of these were irrational violations of general guidelines of structured programming. For instance, for default covariates like age, race and gender, there were default probability-generating functions implemented as part of the top-level function, which produces the main capture matrix itself. I realize now that it makes a lot more sense to implement as separate functions, that can be called from within the top-level function when needed (when user does not specify any

other covariate function). This addresses the benefits of structured programming, where hierarchy of functions is maintained for simplicity and flexibility.
Another big improvement addressed the functionality of the simulation package. The original intention was to make the package as flexible as possible for the user. This flexibility could be very useful for the user to study various heterogeneity structures of interest. Say, the user specifies some covariate function (for generating capture probabilities), but he can also supply a separate set of arguments for every list. Alternatively, a different function can be specified for every list. The bottom line is that the simulation is intended to accommodate for any combinations of covariates and the ways they define capture probabilities.
For convenience purposes, the estimator functions are called from within the top-level function. The user simply specifies which estimates are of interest by specifying the corresponding function names. It can be any subset of built-in estimators or external estimator functions (e.g. log-linear models from `Rcapture` package).

Large Populations

As mentioned previously, capture-recapture estimation is often used for estimation of human populations, like those of a whole nation (e.g. the national Census). Therefore, it needs to cope well with very large population sizes. Of course, there is a general limitation of the computer's memory capacity, however, I found even with very limited memory code efficiency can be improved to establish consistently low runtimes.
In R, matrices are stored in memory by columns - it is essentially a vector of length N by K (N - rows, K - columns). The way I set up the R-C interface originally involved feeding the probability matrix into the C script, then accessing it by rows to generate K capture outcomes for each person in the population. This makes logical sense the way a human would think about this problem, however, it is inefficient from a computational perspective. When cells of a matrix created in R are accessed by row, they are not adjacent in memory and could even be in different memory blocks (used for faster access of related memory elements). I was not aware of this issue until I tried running the simulation with larger population sizes:

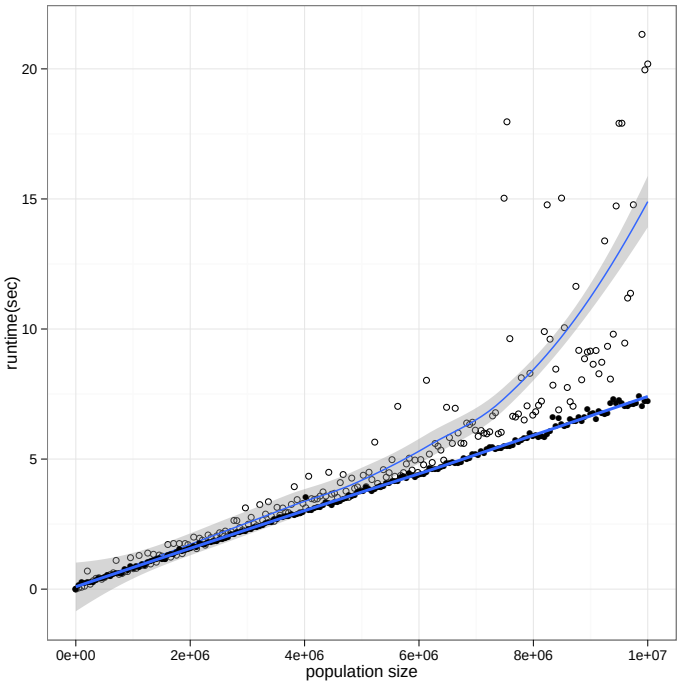


Figure 2: Comparison of run times of C scripts with probability matrix unwrapped by column (hollow points) vs by row (solid points) by N - pop size

The hollow points on the plot show that after approximately 5 million, there is apparent stochasticity in the runtimes, which is an even more serious issue for populations of over 10 million. My solution was to simplify the C script by supplying it the probability matrix as a vector, already unwrapped by columns in R. The C script does not put it back in matrix form, but simply makes a vector of identical length with 0's and 1's for capture events. The performance of this script is shown by solid black points and has a clear linear trend without any anomalies.

4 Results

The analysis below is just an example of the kind of insight that CRISP provides. It includes the comparison of the two implemented estimators, however, other estimators can be studied.
Using the graphing tool that was also implemented as part of the package, I was able to plot the multiple replicated values of the Petersen estimator for a range of population sizes from 1,000 to 50,000 (Figure 2). The confidence intervals and the mean of the replicated estimates are also included. Here the vertical axis is the ratio of the estimated population size and the true population size. Ideally, this should be 1 and this is visualized by the horizontal dashed line. In this most basic scenario where no assumptions are violated, the Petersen estimator performs fairly well.

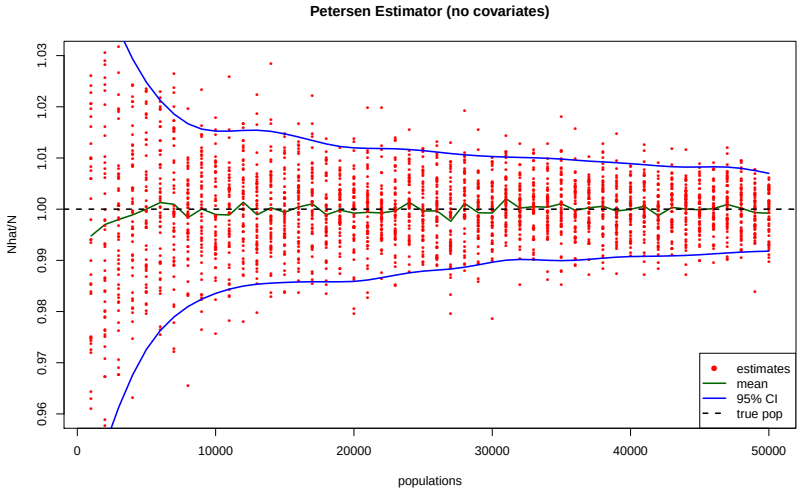


Figure 3: Petersen estimator applied to a range of homogenous populations.

This plot becomes a lot more interesting when heterogeneity structure is introduced by including the age covariate (ages are sampled based on the census demographic information):

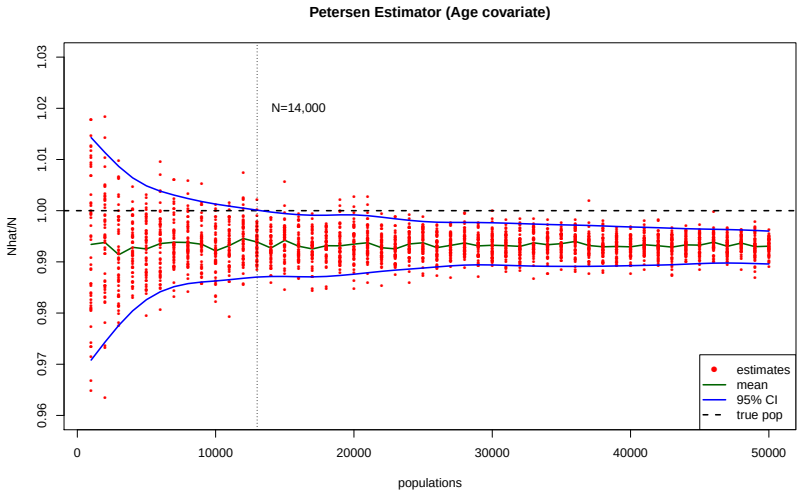


Figure 4: Petersen estimator applied to heterogeneous populations (age covariate)

Heterogeneity resulted in major underestimation by the Petersen estimator. It is apparent that when the true population size exceeds about 14,000 even the upper confidence band curves below the true Nhat/N=1 line. This was an expected result because homogeneity is one of the main assumptions for the Petersen estimator and since it does not model the covariate in any way, the resulting estimations are of questionable quality. The reason for the scenario with presence of a covariate having smaller variance in estimates is the nature of the age heterogeneity structure. The average capture probability in this case was around 0.7, substantially higher than 0.5 in the default case.

After exploring the properties of the Petersen estimator in detail (see Theory appendix), I implemented a more advanced CRC estimator - Horvitz-Thompson style estimator. The idea is that by modeling the covariates (see Theory) it provides more accurate estimates. To tackle the issue of heterogeneity in capture probabilities, the U.S. Census Bureau uses a combination of the Petersen estimator with post-stratification and logistic regression, however, I chose to explore the appropriateness of a Horvitz-Thompson style estimator. Here is a visualization of how this estimator solves the underestimation problem in heterogenous populations:

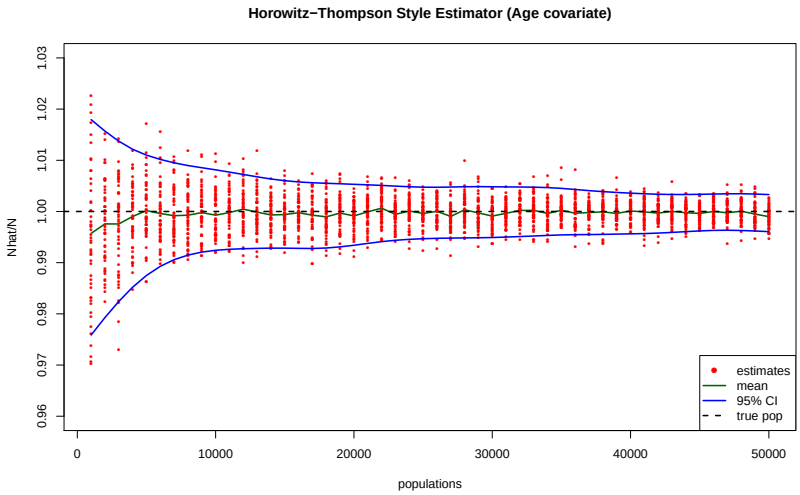


Figure 5: Horvitz-Thompson style estimator applied to heterogenous populations

5 Discussion/Future Work

The simulation framework described here can be used by researchers, as well as Census Bureau employees, for exploring the properties of Capture-Recapture estimators. I, myself, used it to look into the performance of the Petersen estimator in the most basic scenario and in those cases where its assumptions are violated. It shows to come up with severe underestimates in presence of heterogeneity structure in capture probabilities. A Horvitz-Thompson style estimator, however, provides much better estimates in presence of a covariate.

Another advantage of this simulation architecture is that it can be used by anyone, who needs to simulate a large population with some set of demographical data (age, race, gender etc.) For example, an epidemiologist could be interested in simulating a large population with disease contraction probabilities instead of capture probabilities. This could be easily and quickly done using CRISP. With some additional work on formalizing the code and writing documentation, this could be turned into a complete R package available to download from CRAN.

My next immediate steps are going to include:

1. Cleaning up and organizing the code.
2. Implementing more iterative procedures in C.
3. Including more functionality like imperfect matching and more list effects.
4. Using elegant `ggplot2` graphics.
5. Preparing software documentation.

6 Theory

Bold (non-cursive): Can be measured		Enumerated in List 2?		
		YES	NO	Total
Enumerated in List 1?	YES	$\mathbf{N_{11}}$	N_{12}	$\mathbf{N_{1+}}$
	NO	N_{21}	N_{22}	N_{2+}
	Total	$\mathbf{N_{+1}}$	N_{+2}	N

6.1 Petersen Estimator

Petersen estimator uses the following approximation:

$$\frac{N_{11}}{N_{+1}} \approx \frac{N_{1+}}{N} \tag{1}$$

Then, the estimate for the population size is:

$$\hat{N} = (N_{1+}) \left(\frac{N_{+1}}{N_{11}} \right) \tag{2}$$

6.2 Horvitz-Thompson Style Estimator

The Horvitz-Thompson (HT) style estimator models the probabilities of appearing on each list (π_{01},π_{10}, π_{11}) as a function of the covariate (x) to find the detection probabilities for each individual. A generalized additive

model (GAM) is used to approximate the covariate function. The estimate can then be computed by summing the reciprocals of all detection probabilities for everyone captured at least once:

$$\begin{aligned}\pi_{00}(x) &= \frac{\pi_{10}(x)\pi_{01}(x)}{\pi_{11}(x)} \\ \hat{\phi}_i &= \frac{\pi_{10}(x) + \pi_{01}(x) + \pi_{11}(x)}{\pi_{10}(x) + \pi_{01}(x) + \pi_{11}(x) + \frac{\pi_{10}(x)\pi_{01}(x)}{\pi_{11}(x)}} \\ \hat{N} &= \sum_{i:M_i=1} \frac{1}{\hat{\phi}_i} \end{aligned} \tag{3}$$

6.3 Petersen Estimator Bias

An important characteristic of any estimator is its bias. I was interested in finding out if the Petersen estimator was biased or not (when assumptions are satisfied). The issue with doing it in a purely analytical way is that the expected value of the Petersen is not defined: $\hat{N} = (N_{1+}) \left(\frac{N_{+1}}{N_{11}}\right)$

The denominator, which is the number of people appearing on both lists can potentially be 0, and then the expected value becomes infinity. Even though such an outcome is highly unlikely, it still has to be considered and, thus, I cannot use an analytical approach to find the expected value of the Petersen estimator.

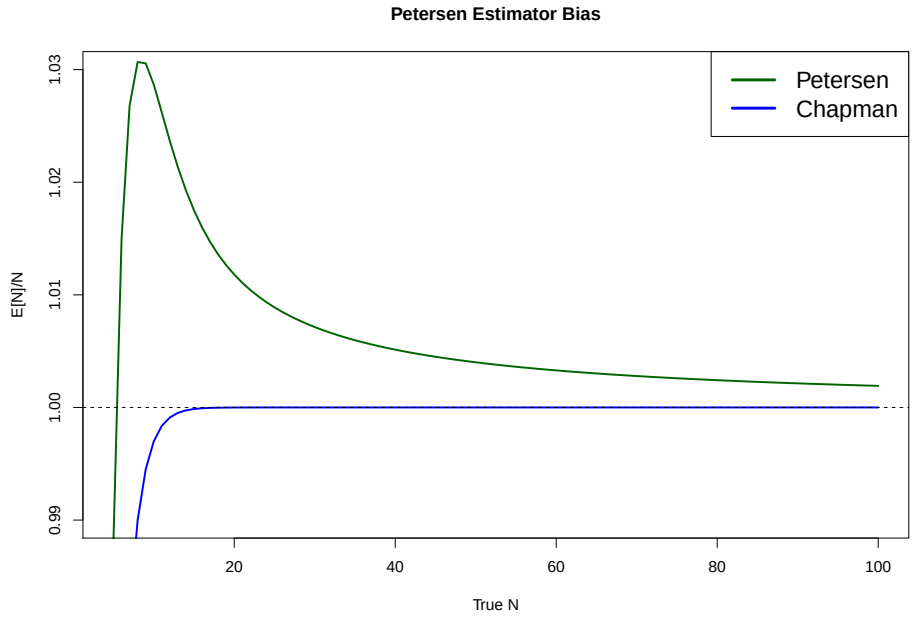
Instead, a quasi-analytical approach was used. The definition of the expectation of a random variable is the sum of all possible values the random variable can take weighted with respective probabilities of those outcomes:

$$\begin{aligned} E\left[f(n)\right] &= \sum_n f(n)P(n) \\ &= \sum_{n|n_{11}\neq 0} \frac{(n_{11}+n_{10})(n_{11}+n_{01})}{n_{11}} P(n = \{n_{11}, n_{10}, n_{01}, n_{00}\} | n_{11} \neq 0) \end{aligned} \tag{4}$$

The first part of the product within the sum is that same Petersen estimator from before, except represented with disjoint events (previously both N_{1+} and N_{+1} contained the entirety of N_{11}). To find the expected value we need to sum the product of the value of Petersen estimator with the respective probability across all possible outcomes for a given population size. The probability term can be calculated using the multinomial formula:

$$\begin{aligned} &= \binom{N}{n_{11}, n_{10}, n_{01}, n_{00}} P_{11}^{n_{11}} P_{10}^{n_{10}} P_{01}^{n_{01}} P_{00}^{n_{00}} \\ &= \frac{N}{n_{11}n_{10}n_{01}n_{00}} P_{11}^{n_{11}} P_{10}^{n_{10}} P_{01}^{n_{01}} P_{00}^{n_{00}} \end{aligned} \tag{5}$$

The bias curve for population sizes below a 100:



The vertical axis is the ratio of the computed expectation of Petersen estimator and the true population size. We can see that the regular Petersen estimator (green line) is biased upwards for small populations, but it is asymptotically unbiased, as seen from the bias curve approaching the horizontal dotted line.

The blue line represents bias of a modification of the Petersen estimator proposed by D. G. Chapman in 1951. It approaches the unbiased line a lot more rapidly, simply by adding a 1 to every part of the product, and subtracting a 1 from the result:

$$\hat{N} = (N_{1+} + 1)\left(\frac{N_{+1} + 1}{N_{11} + 1}\right) - 1 \tag{6}$$

Acknowledgements

I would like to acknowledge Bill Eddy, the U.S. Census Bureau and the Statistics Department at Carnegie Mellon University for the opportunity to perform this research. I would also like to thank Zach Kurtz for being very helpful and encouraging.