TESTING ASSUMPTIONS AND BIAS OF A CARIBOU POPULATION ESTIMATOR THROUGH MONTE CARLO SIMULATION

By

Christopher Lloyd Roberts

A Project Submitted in Partial Fulfillment of the Requirements
for the Degree of

Master of Science
in
Statistics

University of Alaska Fairbanks
May 2022

APPROVED:

Ronald Barry, Committee Chair

Margaret Short, Committee Member

Scott Goddard, Committee Member

John Rhodes, Department Chair

Department of Mathematics and Statistics

Abstract

The Rivest method is the standard way to estimate caribou herd sizes in Alaska and the northern Canadian provinces. Biologists employ radio telemetry to detect discrete groups that make up the wider herd; the Rivest estimator provides an approximate herd size by enumerating the collared and uncollared animals within each group. A key assumption of this technique is that collared caribou mix randomly amongst the wider herd. In this report I scrutinize the accuracy of the Rivest estimator and evaluate three competing hypothesis tests for testing its random-mixing assumption under simulated conditions. The Fisher's Exact Test is the optimal test for detecting violations of random-mixing. I found the Rivest method underestimates caribou herd size in simulations where the random-mixing assumption was violated to a large degree.

1 Introduction

Postcalving surveys are a widespread method for estimating caribou population sizes for herds across Alaska and northern Canada. In short, these surveys entail the use of radio-telemetry to locate collared caribou approximately one month after calving season. During the high summer, caribou aggregate in large groups in windy areas to avoid insect harassment [5]. These groups can number in the hundreds of thousands [1]. Biologists use radio signals emitted by the collared individuals to locate groups of caribou from aircraft. Once groups are located, photographs are taken and the group is later enumerated.

Some previous studies have used classical Lincoln-Petersen techniques to estimate overall herd size from the postcalving survey counts, but this suffers from negative bias due to its assumption that all groups are equally likely to be found [1]. In 1998, *Rivest, Couturier, & Crepeau* [5] proposed an improved estimator that does not require all groups have equal location probability, but does assume that collared animals disperse randomly amongst said groups. In a 2018 review *Boulanger, Adamczewski, & Davison* [1] found that the Rivest estimator yields particularly precise estimates when a reasonably high proportion of animals are collared and there is an adequate amount of aggregation. An R package that implements the Rivest methodology became available in 2012 [3].

For my Master's project, I employ Monte Carlo simulation methods to investigate the key assumption of the Rivest estimator that the collared caribou mix randomly into the herd. In these simulations I examine the efficacy of three competing hypothesis tests for evaluating the equal mixing assumption. For each test, I run 6 simulations which iterate through different numbers of collared caribou. In one simulation the equal mixing assumption is met, and the five others have different degrees to which the assumption is broken.

The three tests under consideration are Fisher's Exact Test (FET), a Poisson Dispersion Test (PDT), and a Zero-Truncated Negative Binomial Dispersion Test (ZTNBDT). The PDT has a near-perfect Type I error rate under conditions where the random mixing assumption is met, but suffers from low power when even high levels of collar clustering exist. The opposite is true for the ZTNBDT - high power to detect even slight departures from the random mixing assumption, but a poor Type I error rate that is exacerbated when there are more collars in the population. I argue that the FET strikes the optimal balance with a respectable mean Type I error rate of 10.9% and power nearly as good as the ZTNBDT under high clustering conditions.

Another consideration of my Master's project is how the Rivest estimator performs under conditions when the random-mixing assumption is not valid. For each simulated caribou population I also obtained the Rivest estimate. Under all clustering conditions I found the Rivest estimate to slightly underestimate the known simulated herd size on average. All code and results of simulations are organized in an R package caribouSim and available on GitHub [6].

2 Notation

I begin with a brief description of notation from *Rivest, et al.* that will be adopted for use throughout this report.

- T is the total herd size
- M is the number of individual groups making up the herd
- $m \leq M$ is the number of groups that contain at least one collared animal
- $m' \leq m$ is the number of detected groups with at least one collared individual
- N_i is the size of the i^{th} group
- X_i is the number of radio collars found in the i^{th} group
- $n = \sum_{i=1}^{m} X_i$ is the total number of collars in the herd

where i indexes detected groups with at least one collar.

3 Rivest Estimator

During the summer season, it is typical for caribou to aggregate in large groups to provide collective relief from insect harassment in the arctic tundra [5]. These groups may number in the hundreds of thousands, and a single caribou herd might be made up of dozens of groups. Before this occurs, biologists opportunistically sample road-accessible caribou to outfit with radio collars. The idea is to use radio-telemetry to locate individuals which lead biologists to large groups. These groups are documented with high-resolution photos the number of indivduals is later counted [1].

The Rivest method computes an estimate for herd size (T) from the enumerated group sizes (N_i), the number of collars in each group (X_i), and the total number of collars in the population (n). If all collars deployed are located during postcalving surveys, then $\sum_{i=1}^{m'} X_i = n$. Table 1 shows a sample data set from which a Rivest estimate may be obtained.

The collared caribou do not represent a simple random sample, and the Rivest estimator doesn't require it [5]. It does require that, after the caribou are collared, they randomly disperse into the groups and do not cluster together (or actively avoid each other). So the sampling regimes here can be thought of as two-fold: the dispersion of collared caribou amongst the wider herd and the location of groups via radio-telemetry. This is referred to as phase I and phase II sampling, respectively.

3.1 Phase I Sampling

Phase I sampling refers to the process of n collared animals dispersing amongst groups in the wider herd. Let π_i be the probability that there is at least one collared animal in the i^{th} group. If the randomness assumption is met, then

$$\pi_i = 1 - \frac{\binom{T-N_i}{n}}{\binom{T}{n}}.$$

If $T - N_i$ is much larger than n, this can be approximated as

$$\pi_i \approx 1 - \left(\frac{T - N_i}{T}\right)^n$$

Thus, there are m groups (where $m \leq M$) in the herd with at least one collared caribou that are available to biologists to be detected during the postcalving survey. The probability that the i^{th} group is one of the m groups with collared animals is π_i.

3.2 Phase II Sampling

Of course, not all of the m groups with collared animals will be detected. Furthermore, not all groups will have equal probability of detection. The process of detecting groups with at least one collared animal is referred to as phase II sampling.

In order to estimate the overall herd size, it is necessary to estimate the the detection probabilty for each

Group	# Collars (X_i)	# Caribou (N_i)
1	1	1498
2	1	12730
3	14	83619
4	1	3587
5	2	3701
6	12	43607
7	57	135556
8	5	18858
9	2	7593
10	1	12

Table 1: 2011 survey counts of detected groups in the Western Arctic Herd, Alaska. Of the 97 collars deployed, 96 were located during postcalving surveys.

group. Let p_i be the probability of detection for the i^{th} group with collared animals. Unlike Lincoln-Petersen, the Rivest estimator allows for the detection probability to vary from group to group. The detection probability p_i is a function of an unknown parameter r; estimating r (and thus p_i) is critical for the estimating the total herd size.

There are many geographical and biological considerations that could affect the value p_i; *Rivest et al.* proposes three potential models. I give a brief description of each here, but note that my simulations only consider the Independence model. The caribouSim package which implements these simulations is capable of considering all models.

3.2.1 Homogeneity Model

The homogeneity model supposes that the detection probability is not a function of the structure of the caribou herds themselves, but rather due to location at the time of sampling. Perhaps the group is too far to away from the sampling path to be detected, or in a small canyon that impedes the range of radio signals. Thus, the the probability of detection is constant across groups

$$p_i^H = r$$

where the r parameter gives the constant probability of detection for all groups.

3.2.2 Independence Model

It may be more likely that detection probability increases with the number of collared animals in a group. If we assume that within a group collars are detected independently of one another, then this suggests the following model

$$p_i^I = 1 - r^{X_i}.$$

Here, r describes the probability of not detecting an individual collared caribou. Again, this is the only model considered in the analysis provided in this report.

3.2.3 Threshold Model

The last model postulates that there may be some threshold of collared animals, B, such that when exceeded the probability of detection is 1 and some constant otherwise

$$p_i^B = \begin{cases} 1 & X_i \geq B \\ r & 1 \leq X_i < B \end{cases}$$

If the threshold B is not exceeded for the i^{th} group, then the probability of detection is homogeneous and equals the constant r.

3.3 Rivest Estimator of T

The first step to estimate T is to define an estimating equation for the phase II detection probability, p_i, and its parameter r. As noted above, p_i depends on the detection model used and here I specifically consider the independence model. There is no explicit form for $\hat{r}$ for the independence model, and it is instead calculated numerically.

Once the estimate $\hat{p}_i$ is obtained, the abundance estimate for the caribou herd is the solution to the equation,

$$\hat{T} = \sum_{i=1}^{m'} \frac{N_i}{\hat{p}_i[1 - (1 - N_i/\hat{T})^n]},$$

which also must be solved numerically.

4 Hypothesis Tests

It is unclear how the Rivest estimator performs when the random-mixing assumption is broken. Violation of this assumption could lead to bias in the caribou herd size estimates, which has potentially negative implications for caribou management. Underestimating the herd sizes might deny key recreational and subsistence hunt opportunities for rural northern communities, whereas overestimating may lead to overharvest.

It is critically important that wildlife biologists have reliable methods for detecting when the Rivest assumptions are met or not. This would help wildlife managers decide when their estimates are reliable or not. Unfortunately, an optimal test for detecting collar clustering isn't widely known in the literature. In this section I describe three potential tests for detecting violations of the random-mixing assumption. Later, I consider the consequences of clustering amongst collared caribou.

4.1 Fisher's Exact Test (FET)

Table 1 above shows a typical caribou postcalving count dataset. This example shows 2011 counts of the number of collared and uncollared caribou for the Western Arctic Herd in Alaska. Note that the numbers given in the Group column are arbitrary identifiers for distinct groups found during the survey. This sample dataset is given in the caribou R package [3].

If the assumption is valid and collared individuals do indeed mix randomly with the rest of the herd, then the probability that any one caribou is collared is constant across the population. It follows that the number of collared caribou in any group follows a hypergeometric distribution [5]. Thus, the probability mass function of k collared animals in group i is

$$P(K = k) = \frac{\binom{N_i}{k}\binom{T-N_i}{n-k}}{\binom{T}{n}}.$$

Therefore a test of independence between the columns of the dataset should detect if the assumption is met, and a Fisher's Exact Test will do exactly that.

4.2 Overdispersion Tests

4.2.1 Poisson Dispersion Test (PDT)

If the random assumption is met and collared animals are randomly distributed amongst the groups of a herd, then the number of collared animals per group (given group size) should follow an approximate Poisson distribution [5]. Then the offset loglinear model under consideration is

$$X_i|N_i \sim Poisson(\lambda) \qquad \implies \qquad \log X_i = \lambda + \log N_i + \epsilon_i$$

where ϵ is the error term. However if the assumption is not met, then the data collected will have larger or smaller variance than expected and should alternatively follow an dispersed Poisson distribution. This reasoning forms the basis for the next hypothesis test evaluated in my simulations.

The AER package implements a Wald test for dispersion in Poisson GLM's [4]. If the rate of collars per group of caribou, X_i/N_i, is a dispersed Poisson random variable with constant parameter λ, then the variance of X_i takes the form

$$\log X_i = \lambda + \alpha\lambda + \log N_i + \epsilon_i$$

where α is an overdispersion parameter. This parameter can be treated as a coefficient in an intercept-only OLS model,

$$\frac{(x_i - \hat{x}_i)^2 - x_i}{\hat{x}_i} = \alpha + \epsilon_i$$

where the i^{th} response is a function of the i^{th} observed number of collars and fitted value [2]. Fitting this auxillary model, an estimate and standard error for α is obtained and thus a Wald statistic $t = \hat{\alpha}/se(\hat{\alpha})$. Under the null hypothesis of no dispersion, the distribution of t is asymptotically standard normal [2]. This test is implemented in the function dispersiontest().

4.2.2 Zero-Truncated Negative Binomial Dispersion Test (ZTNBDT)

Since biologists locating caribou only include groups with collared animals in their population estimation, it is probably more realistic to consider a zero-truncated model since it is impossible to observe zero values in these data. Zero-truncation in a probability mass function accounts for the reality that a random variable cannot take the value zero [8]. Thus, an alternative model for radio collars per group is a Zero-Truncated Negative Binomial (ZTNB). Aside from being more realistic, this distribution has the advantage of having a dispersion parameter that can be directly tested. Let $Y_i = X_i/N_i$ be a random variable that follows such a distribution. Then an expression for the probability mass function for Y_i, a Zero-Truncated Gamma-Poisson mixture model [8], is

$$P(Y_i = y_i) = \frac{\Gamma(y_i + \alpha^{-1})}{\Gamma(y_i + 1)\Gamma(\alpha^{-1})[1 - (1 + \alpha\lambda)^{-\alpha^{-1}}]}(\alpha\lambda)^{y_i}(1 + \alpha\lambda)^{-(y_i + \alpha^{-1})} \quad y_i = 1, 2, 3, \ldots$$

where λ is the shape parameter and α the dispersion parameter. Like the Poisson test, if the random mixing assumption is violated then we should see overdispersion in the data [8]. Thus, the dispersion parameter should be significant. Also, as with the PDT, I model the number of collars X_i offset by group size N_i.

Yehia (2021) proposed implementations of likelihood ratio, score, and Wald tests of the dispersion parameter of this model. By performing Monte Carlo simulations to compare power amongst the three tests, they found that the Wald test had the highest power for detecting overdispersion. Based on these results, I will choose this Wald test as my last hypothesis test to evaluate. Similar to the previous test, if $\hat{\alpha}$ is the maximum likelihood estimate of α then the Wald statistic is $t = \hat{\alpha}/se(\hat{\alpha})$ and is asymptotically standard normal [8]. This test is implemented by fitting a vglm model using the package VGAM [7].

5 Simulations

The R package caribouSim provides functions for simulating caribou herds, phase I, and phase II sampling. Discussion of how these functions work is provided in this section.

5.1 Algorithm for Simulating Caribou Populations

The overall approach for generating caribou groups is to first define target herd size, Θ, and randomly select the m number of groups that approximately Θ caribou will be distributed amongst. Using the algorithm below I vary the size of each group to create a highly right-skewed distribution of animals per group (N_i).

The function sim_Ni implements this algorithm and takes a single input, Θ, and returns a vector of caribou groups sizes that make up the total synthetic herd. Note that Θ is merely a fixed target herd size, and the actual herd size T will generally be slightly larger. A summary for how sim_Ni is as follows:

```
1. Define sampling frame of integers $1, \ldots, \frac{\Theta}{2}$
2. Let T_sim $= \Theta$
3. Let $i = 1$, while T_sim $> 0$
    (a) $N\_i$ is a sampled integer from the sample frame. The probability of an
        integer begin selected is inversely proportional to the value of the integer
    (b) T_sim $=$ T_sim $- N_i$
    (c) $i = i + 1$
4. Stop if T_sim $\leq 0$
```

The purpose for altering sample probability in step 3(a) is to give the size of groups a right-skew distribution. See the function sim_Ni in the appendix to see source code for the implementation of this algorithm.

5.2 Algorithm for Simulating Collared Animals

The next step after T animals (where $T = \sum_{i=1}^{M} N_i$) are distributed amongst M groups is to assign a number of collared animals to each group. Given the set of groups generated by sim_Ni, the number of collars will follow a mixture of two binomial distributions. If there is random-mixing then these two binomial distributions will be the same. If there is clustering then one distribution will have a higher success probabilty (ρ) than the other, where "success" is the event that any individual caribou has a collar.

This is accomplished by the function sim_Xi. This function takes three arguments as inputs and returns a vector of the collars in each group. The input arguments are the desired total number of collars, ν; the simulated herd, N_i; and a parameter ϕ which controls the degree to which the random mixing assumption is broken.

1. Randomly select half of the M groups generated by sim_Ni
2. Give these groups a weight factor of $w_i = N_i(1 + \phi)$
3. Give the other half a weight factor of $w_i = N_i(1 - \phi)$
4. Define $\rho_i = \frac{w_i}{\sum w_i}$ to be a vector of success probabilities
5. For each group $X_i \sim$ Binomial(ν, ρ_i)

Observe that the parameter $0 \leq \phi \leq 1$ works by increasing the binomial probability for the number of collars in half of the groups - and decreases the probability in the other half - thus resulting in clustering of collared animals (and increasing dispersion). If $\phi = 0$, then each group has a success probability strictly proportional to the group size and the equal mixing assumption is met. The maximum amount of clustering occurs when $\phi = 1$, resulting in half of the groups having 0 probability of acquiring collars.

Much like Θ, the desired number of collars (ν) is approximated by the simulated actual number of collars (n). Since the number of collars in each group X_i is a random variable, they don't necesarily sum to n. This is a realistic outcome as biologists aren't necessarily guarenteed to collar n number of animals each year; they attempt to collar as many animals as they can and this number changes from year to year. Again see sim_Xi in the appendix for source code that implements this algorithm.

5.3 Phase I and Phase II Sampling

Functions are provided to simulate phase I and phase I sampling, as discussed in section 2.1 and 2.2. Phase I sampling is implicit in the previous simulation functions; sample_phaseI is a utility function that formats the dataset. sample_phaseII takes into account that not all groups will be detected, and enables users to select the detection model. Only the independence model is considered in this report.

5.4 Monte Carlo Simulations

The function iterate extends the capability of sim_Ni and sim_Xi to iterate simulations many times. In each iteration, a single population with collars is generated and phase II detection (using independence model) simulated. The estimated population is reported, as is the difference between the true and estimated T. The parameter r is estimated as well as its standard error. Then the three hypothesis tests are performed and their p-values saved, and an overall rejection rate for each is reported.

Six total Monte Carlo simulations are performed, each with varying degrees of the random mixing assumption broken. These six simulations differ only in the parameter ϕ which is passed to the sim_Xi function. In one simulation the assumption is met ($\phi = 0$); the other five simulations correspond to $\phi = \{0.1, 0.2, 0.3, 0.4, 0.5\}$.

Within each simulation 100 iterations are performed for an increasing number of collared animals, generating a power curve. Thus, a rejection rate is computed for each hypothesis test along intervals of 10 in a sequence from $n = 10$ to $n = 1000$. More explicitly, the iterations begin with $1/100000$ of the animals collared and end with $1/1000$ of the of the herd collared.

6 Results

6.1 Power Analysis

6.1.1 Random Mixing Assumption Valid

Figure 1 below shows the resulting power curves for the Monte Carlo simulation when the random mixing assumption is valid. The y-axis of this plot gives the overall rejection rate for 100 simulations within each targeted collar abundance, ν. The x-axis gives the number of collared animals simulated, beginning with 10 and ending with 1000 (out of 1000000 total caribou). The bold lines give moving averages for each hypothesis test, and the transparent lines are the raw results.

Theoretically my hypothesis tests should fail to reject since the null hypothesis (random-mixing) is true. The Poisson test has a near-zero Type I error rate of 0.1%, and the Fisher's test has a respectable Type I error rate of 10.9%. The Negative-Binomial test has high mean rejection rate of 34.05%, but is not at all constant and worsens with more collars in the herd.

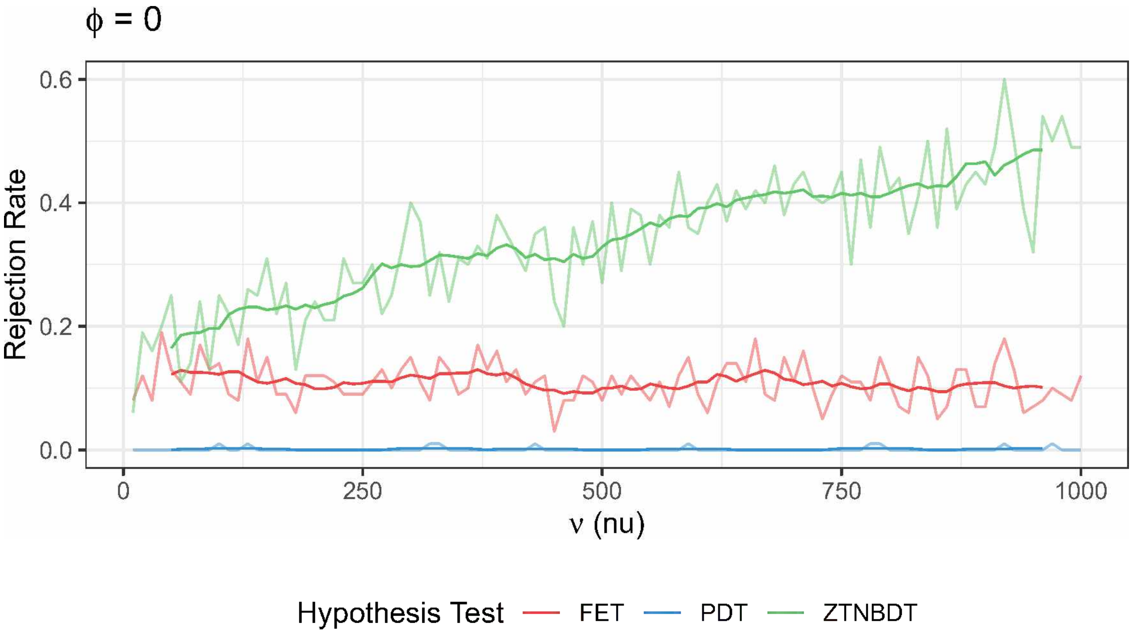
6.1.2 Random Mixing Assumption Violated

There are five power curves in figure 2 that correspond to Monte Carlo simulations of varying degrees of violation of the random mixing assumption. The smallest degree of collar clustering is $\phi = 0.1$ and the maximum amount is $\phi = 0.5$.

Table 2 gives the approximate number of collars required to achieve a power of at least 50% and 80% for each value of ϕ for each test. The FET and ZTNBDT did an excellent job at detecting departures from random mixing at medium to high levels of clustering. When $\phi = 0.4$ or greater, both tests achieved greater than 80% power when less than $1/10000$ of the population was collared. As few as 10 (out of 1000000) collars were required to correctly reject at least half of the time. At low levels of ϕ, the FET and ZTNBDT required large numbers of collars in the population to achieve reasonable power. For example, 80% power was not reached, even with 1000 collars, when $\phi = 0.1$.

It should be noted that the ZTNBDT performed slightly better than the FET at all levels of collar clustering, but comes with the caveat of a terrible Type I error rate. In contrast the PDT had a poor Type II error rate, and never rejected much more frequently than 75% of the time when the random-mixing assumption was violated. All three tests rejected more often the worse the random-mixing assumption was violated.

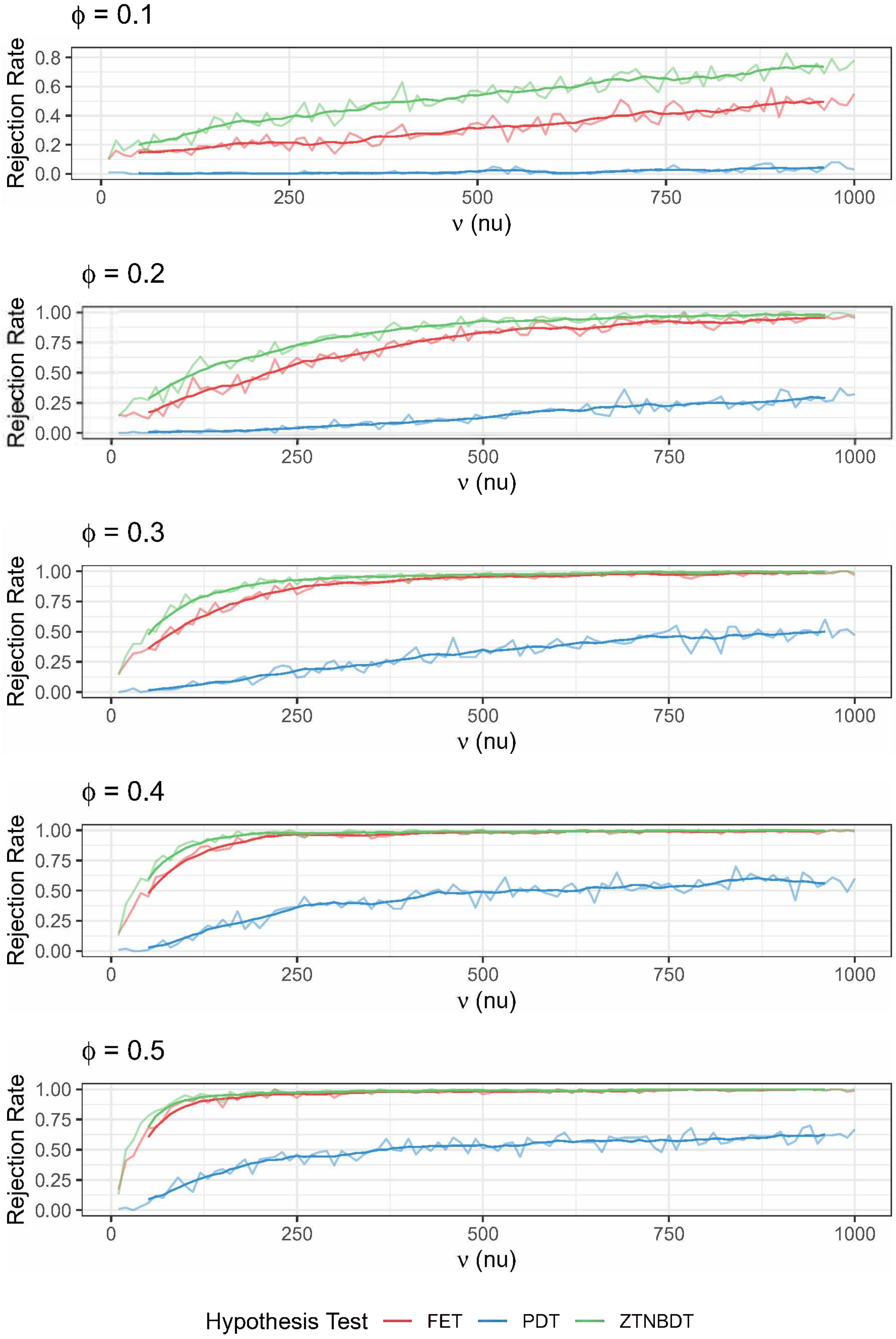
Figure 1: Power Curve When Random Mixing Assumption is Met. Bold lines give $k = 9$ moving averages for FET (red), PDT (blue) and ZTNBDT (green); transparent lines are raw simulation results.



	FET		PDT		ZTNBDT	
Power	50%	80%	50%	80%	50%	80%
$\phi = 0.1$	≈ 1000	> 1000	> 1000	> 1000	360	> 1000
$\phi = 0.2$	180	440	> 1000	> 1000	80	280
$\phi = 0.3$	60	170	920	> 1000	20	100
$\phi = 0.4$	20	90	540	> 1000	10	50
$\phi = 0.5$	10	50	330	> 1000	10	30

Table 2: Approximate n required to reach 50% and 80% power

Figure 2: Power Curves When Random Mixing Assumption is Violated. Bold lines are moving averages for FET (red), PDT (blue) and ZTNBDT (green); transparent lines are unsmoothed results.



ϕ = 0.1

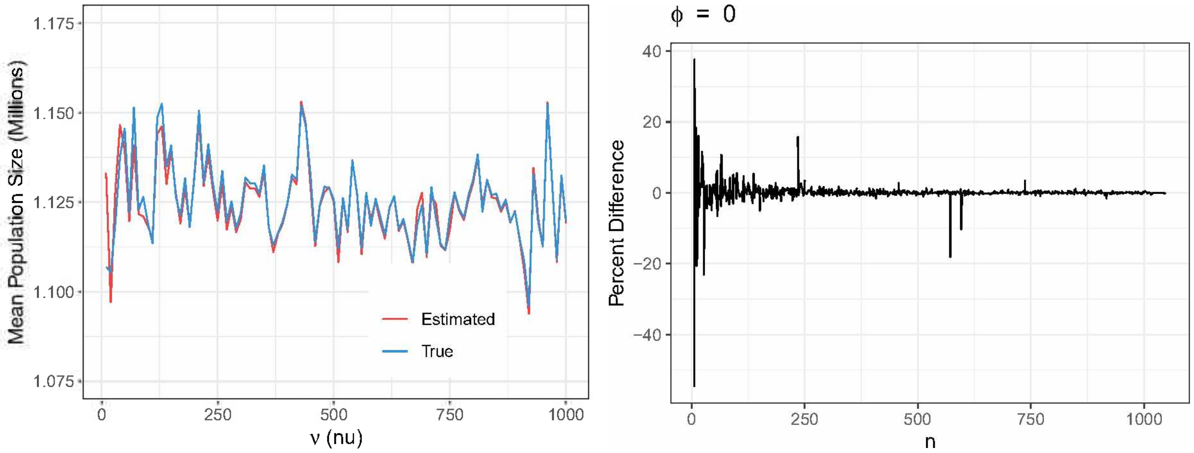
ϕ = 0.2

ϕ = 0.3

ϕ = 0.4

ϕ = 0.5

Figure 3: Comparison of Simulated and Estimated T when $\phi = 0$



6.2 Simulated Accuracy of Rivest Estimator

At low values of ϕ all tests did a relatively poor job of accurately rejecting the equal mixing assumptions. But how do low levels of clustering affect the overall estimate for herd size, T? Is the Rivest estimator still appropriate when there is some clustering? Simulated results are presented here to address these questions.

6.2.1 Random Mixing True: $\phi = 0$

When the random mixing assumption is true, the Rivest estimator did an excellent job of estimating the caribou herd size, T. The left plot in figure 3 shows the mean estimated herd size $\hat{T}$ and mean simulated herd size T for each target number of collars, ν. This shows that these two values are very similar to each other, especially when there are more than 300 or so collars in the herd. This matches up with the right plot in figure 3, which shows the percent difference between $\hat{T}$ and T for a sample of individual simulations. Notable outliers aside, the percent difference between estimated and true herd size converges pretty close to 0. Overall, the mean percent difference between T and $\hat{T}$ is a slight overestimation of 0.0566%.

6.2.2 Random Mixing Violated: $\phi = 0.2$

Overall, the difference between estimated and simluated herd sizes is very similar when there is minor clustering. The plots in figure 4 are almost indistinguishable to that of figure 3 - if anything there seems be more bias at low values of ν. There also appears to be more variance in T (and thus $\hat{T}$), but this is noise in the simulation and not a result of clustering. The mean percent difference of T and $\hat{T}$ is an underestimation at -0.1695%. See table 3 for a comparison of sum of squared errors for all ϕ. Interestingly, the SSE increases two orders of magnitude from when $\phi = 0.1$ to $\phi = 0.2$.

ϕ	SSE
0.0	1.2090×10^{13}
0.1	1.5670×10^{13}
0.2	1.5809×10^{15}
0.3	9.8245×10^{15}
0.4	1.3634×10^{16}
0.5	1.5670×10^{16}

Table 3: Sum of Squared Errors Between Simulated and Estimated T

6.2.3 Random Mixing Violated: $\phi = 0.5$

At high amounts of clustering, the estimated herd sizes are more severely underestimated (-0.5695%, on average). The sum of squared errors is much larger as well. Most worrying is that this underestimation is exacerbated when there are less than 200 or so collars in the population. When consider simulations where $n < 200$, the mean percent difference between T and $\hat{T}$ increases to -3.7737%.

6.3 Pairwise Comparisons of Estimator Bias

It appears that there is not much of a difference in the performance of the Rivest estimator between low and no collar clustering, but that is not the case when there is high clustering. However I want more concrete evidence that the discrepency between simulated and estimated T is greater when there is a higher degree of clustering. A two-sided paired Wilcoxon two-sample test is appropriate for detecting such a difference.

Let $\omega_k = E[\hat{T}] - T$ denote the bias for the Rivest estimator for the k^{th} value of ϕ. Thus an estimate for the bias is $\hat{\omega}_k = \hat{T} - T$, and I want to test if there is a difference in bias between simulations with different levels of

Figure 4: Comparison of Simulated and Estimated T when $\phi = 0.2$

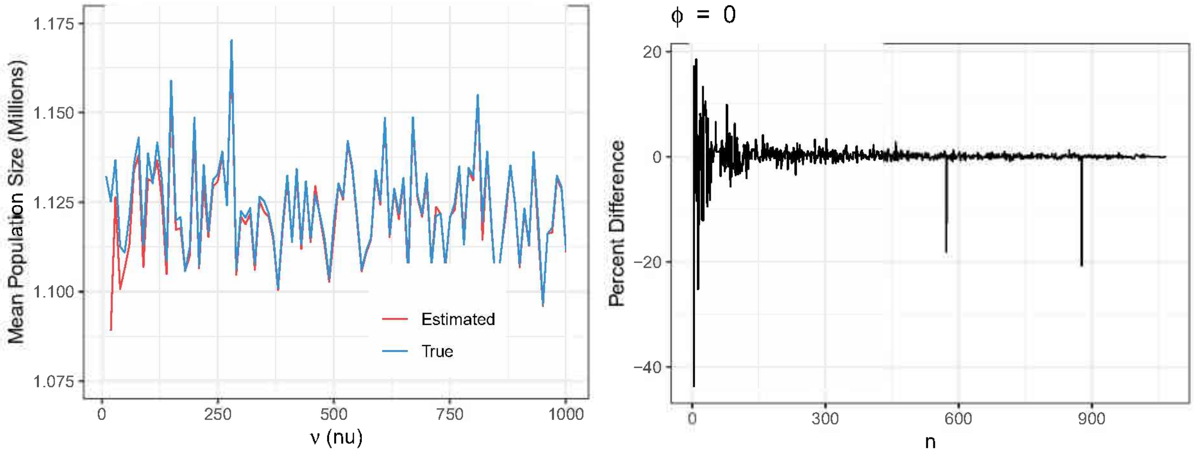
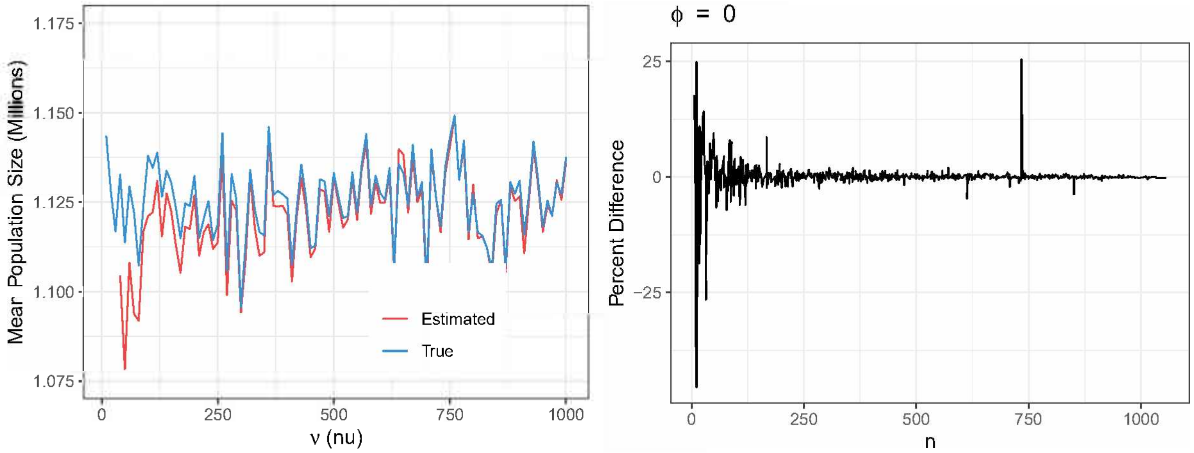


Figure 5: Comparison of Simulated and Estimated T when $\phi = 0.5$



				$\phi = k$			
		0.0	0.1	0.2	0.3	0.4	0.5
$\phi = j$	0.0	0.0566%	0.1571	0.0151^*	0.0011^*	$< 0.0001^*$	$< 0.0001^*$
	0.1	-	-0.1261%	0.2780	0.0687	$< 0.0001^*$	$< 0.0001^*$
	0.2	-	-	-0.1695%	0.1991	$< 0.0001^*$	$< 0.0001^*$
	0.3	-	-	-	-0.2303%	$< 0.0001^*$	$< 0.0001^*$
	0.4	-	-	-	-	-0.3727%	0.0003^*
	0.5	-	-	-	-	-	-0.5695%

Table 4: P-values from Pairwise Wilcoxon Tests; (*) indicates significance at the $\alpha = 0.05$ level. Diagonal entries give the mean percent difference between the simulated and estimated herd sizes.

clustering. The null and alternative hypotheses are

$$H_0 : \omega_k = \omega_j \text{ for } k \neq j$$

$$H_A : \omega_k \neq \omega_j \text{ for } k \neq j$$

Table 4 summarizes the results from these pairwise Wilcoxon tests. Since these tests are two-sided the table is symmetric, hence the lower left corner is omitted. The diagonal entries give the mean percent difference between simulated and estimated herd size for each level of ϕ.

There appears to be fairly clear delineation between low clustering ($\phi = 0.1$), medium clustering ($\phi = \{0.2, 0.3\}$), and high clustering ($\phi = \{0.4, 0.5\}$). There isn't sufficient evidence to say that the low clustering simulations had a median bias different from that of the simulations with random-mixing ($P = 0.1571$). On the other hand, there is moderate/strong evidence that medium clustering simulations had a different median bias ($P = 0.0151, 0.0011$) - and extremely strong evidence that the high clustering simulations had a different median bias ($P \approx 0$) - from the cluster-free simulations.

7 Discussion

Of the three hypothesis tests under consideration, the Fisher's Exact Test struck the best balance between high power and a low Type I error rate. Although the Poisson Dispersion Test had a near-perfect Type I error rate it had very low power for practical purposes, never consistently achieving better than a 75% rejection rate even when collared caribou were simulated with high clustering and made up $.1\%$ of the total population. Conversely, the Zero Truncated Negative Binomial Dispersion Test had the best power under all conditions but the cost being a Type I error rate approaching 50% in some cases. The FET had only slightly less power than the ZTNBDT, but had a relatively low Type I error rate.

Indeed, the Fisher's Exact Test appears close to optimal for detecting clustering amongst collared animals. When the assumption was met, the FET maintained a fairly constant rejection rate of approximately 10.9% regardless of how many collared animals were present in the population. At low amounts of clustering many collared caribou are required to detect violations to random mixing. At high amounts of clustering, however, clustering is detected fairly quickly. I recommend collaring at least approximately $1/10000^{th}$ of the population to achieve reasonable power for the FET.

What is less clear is the reason for the relatively poor performance of the dispersion tests. The issue lies within one or more of three possible sources: the theory, the implementation, or the simulation. The main idea behind both the PDT and NBDT was that if collared animals are not randomly mixing, then the distribution of collars should be overdispersed. I believe this logic to be sound but there are potential issues. First, the Wald test used in both cases relies on the result that the distribution of the Wald statistic is *asymptotically* normal. Considering the number of groups M in my simulations is random and generally lies between 5 and 20, I believe the asymptotic requirement is not met. Perhaps this could be mitigated by implementing a score test for both distributions. Another issue might be the choice of distribution. As noted in section 3.2.2, groups of 0 collars are not observed. So a Zero-Truncated Poisson model is more appropriate than a conventional Poisson. On the other hand the ZTNB model performed poorly, but perhaps the assumption that a Negative Binomial could replace the Poisson is not appropriate in this case.

The asymptotic requirements and choice of distribution are rendered moot if test implementations are flawed. As I did not write the code for either the PDT or ZTNBDT, I cannot vouch for their veracity. I would argue that the functions and libraries performing these calculations are nonstandard in the R community. The AER library serves a specific field (applied econometrics)[4], and the ZTNB distribution might be obscure enough that bugs have gone undetected. I computed this using the posnegbinomial() function in a vglm model in the VGAM library [7].

A last explanation for the poor performance of the dispersion tests might be that the issue lies in some property of how the simulated caribou populations are generated. Perhaps the algorithm that simulates collared groups is flawed, for example. However, given the success of the FET, this is unlikely.

In any case, it's probably not critical for the FET to reject when there are low levels of clustering in the collared population. When $\phi = 0.1$, the difference in the estimated and the simulated caribou herd size was not significantly different than when the random mixing assumption was met. In other words, under low clustering conditions the Rivest estimator performs about as well as it does when there is no clustering.

What is most worrying is that the Rivest estimator tended to underestimate the simulated carribou population when high amounts of collar clustering was present. Although on average the underestimation was not severe, it was still quite significant (-3.7737%) when there were less than 200 collared animals in the population. So when violations to the random-mixing assumption occur, it requres substantially more effort to obtain precise estimates.

This clear bias has negative implications for real-life caribou population estimation and management. In practical terms, biologists rarely collar more than 100 or so animals in large caribou herds [1]. For example, if there was a high degree collar clustering in a caribou herd of 1,000,000 and biologists collar 100 of those animals, then based on these data they might underestimate the herd by almost 400,000 animals. Nevertheless, the Fisher's test would most likely detect clustering in this hypothetical scenario.

It is important to note that there are many more variables to explore than considered here. For example, the work presented here simulated phase II sampling using a constant value of $r = 0.5$ and under the independence model only. The Rivest estimation was similarly only computed using the independence model. How do results differ when the homogeneity model is used for phase II estimation? Threshold model? What happens when phase II estimation uses one model and the population estimate is computed using a different model? How does altering the value of the r parameter affect the estimates and rejection rates? How close does $\hat{r}$ come to estimating the true value of r? Does this change under different simulated conditions? All of these questions have practical management implications and could be easily investigated using the caribouSim package [6].

References

[1] John Boulanger, Jan Adamczewski, and Tracy Davison. Estimates of caribou herd size using post-calving surveys in the northwest territories and nunavut, canada: A meta-analysis. *Rangifer*, 38(1):39–78, Dec. 2018.

[2] A Colin Cameron and Pravin K Trivedi. *Regression analysis of count data*, volume 53. Cambridge University Press, 2013.

[3] Helene Crepeau, Louis-Paul Rivest, Serge Couturier, and Sophie Baillargeon. *caribou: Estimation of caribou abundance based on large scale aggregations monitored by radio telemetry*, 2012. R package version 1.1.

[4] Christian Kleiber and Achim Zeileis. *Applied Econometrics with R*. Springer-Verlag, New York, 2008. ISBN 978-0-387-77316-2.

[5] Louis-Paul Rivest, Serge Couturier, and Hélène Crépeau. Statistical methods for estimating caribou abundance using postcalving aggregations detected by radio telemetry. *Biometrics*, 54(3):865–876, 1998.

[6] CL Roberts. *caribouSim: Functions for evaluating Rivest Estimator*, 2022. R package version 0.1.0.

[7] Thomas W. Yee. The VGAM package for negative binomial regression. *Australian and New Zealand Journal of Statistics*, 61, 2020.

[8] Enas Gawdat Yehia. Power of overdispersion tests in zero-truncated negative binomial regression model. *American Journal of Theoretical and Applied Statistics*, 10(3):152–157, 2021.

Appendix

Relavent Functions in caribouSim

```r
sim_Ni <- function(Theta) {
    T_sim <- Theta
    Ni <- c()
    sample_frame <- 1:(round(0.5 * T_sim))
    # note that a group cannot be greater than
    # half the total pop size
    while (T_sim > 0) {
        Ni <- append(Ni, sample(sample_frame, size = 1,
            prob = 1/(sample_frame)))
        T_sim <- Theta - sum(Ni)
    }
    return(sort(Ni, decreasing = TRUE))
}

sim_Xi <- function(n, Ni, phi = 0) {
    T <- sum(Ni)
    M <- length(Ni)
    ind <- sample(1:M, size = round(M/2), replace = FALSE)
    places <- c(c(M:1)[ind], c(M:1)[-ind])
    weights <- c(Ni[ind] * (1 + phi), Ni[-ind] * (1 -
        phi))[rev(order(places))]
    probs <- weights/sum(weights)
    Xi <- rep(0, M)
    Xi <- rbinom(M, size = n, prob = probs)
    return(Xi)
}

sample_phaseI <- function(Xi, Ni) {
    pop <- data.frame(Xi = as.numeric(Xi), Ni = as.numeric(Ni))[which(Xi !=
        0), ]
    return(pop)
}

sample_phaseII <- function(pop, r, model = c("H", "I",
    "T"), B) {
    if (model == "H") {
        pi <- rep(r, nrow(pop))
    } else if (model == "I") {
        pi <- 1 - r^pop[, 1]
    } else if (model == "T") {
        pi <- ifelse(pop[, 1] >= B, 1, r)
    }
    detected <- ind <- c()
    for (i in 1:nrow(pop)) {
        detected[i] <- rbinom(n = 1, size = pop[i,
            1], prob = pi[i])
        ind[i] <- ifelse(detected[i] > 0, i, NA)
    }
    out <- pop[na.omit(ind), ]
    return(out)
}

iterate <- function(iters, T, n, phi, r, modelSim = c("H",
    "I", "T"), modelEst = c(modelSim, "H", "I", "T"),
    BSim, BEst = BSim) {
    ntrue <- X <- Ttrue <- That <- se_That <- p_fisher <- c()
    p_dispersion <- p_overdisp <- rhat <- se_rhat <- iteration <- c()
    library(VGAM)
    for (i in 1:iters) {
        Ni <- sim_Ni(T = T)
        Xi <- sim_Xi(n = n, Ni = Ni, phi = phi)
        pop <- sample_phaseI(Xi = Xi, Ni = Ni)
        dat <- sample_phaseII(pop, model = modelSim,
            r = r, B = BSim)

        if (ncol(dat) < 2 | nrow(dat) < 2)
            next

        ntrue[i] <- sum(Xi)
        X[i] <- sum(dat[, 1])
        Ttrue[i] <- sum(Ni)
        tryCatch({
            tmp <- caribou::abundance(mat = dat, n = n,
                model = modelEst, maxT.hat = 10^8,
                B = BEst)
            tmp2 <- vglm(Xi ~ offset(log(Ni)), family = posnegbinomial(),
                data = dat)
            p_fisher[i] <- fisher.test(dat, simulate.p.value = TRUE)$p.value
            p_dispersion[i] <- AER::dispersiontest(glm(Xi ~
                offset(log(Ni)), family = poisson,
                weights = Ni, data = dat))$p.value
            p_overdisp[i] <- coef(summaryvglm(tmp2))[2,
                4]
        }, error = function(e) {
            cat("ERROR :", conditionMessage(e), "\n")
        })
        That[i] <- tmp$T.hat
        se_That[i] <- tmp$se_T.hat
        rhat[i] <- tmp$rr
        se_rhat[i] <- tmp$se_rr
        iteration[i] <- i
    }

    diff <- Ttrue - That
    pctDiff <- ((Ttrue - That)/Ttrue) * 100

    p_fisher <- na.omit(p_fisher)
    p_dispersion <- na.omit(p_dispersion)
    p_overdisp <- na.omit(p_overdisp)

    alpha1 <- sum(ifelse(p_fisher < 0.05, TRUE, FALSE))/length(p_fisher)
    alpha2 <- sum(ifelse(p_dispersion < 0.05, TRUE,
        FALSE))/length(p_dispersion)
    alpha3 <- sum(ifelse(p_overdisp < 0.05, TRUE, FALSE))/length(p_overdisp)

    out <- structure(list(collars = cbind(iteration,
        ntrue, X), abundance = cbind(iteration, Ttrue,
        That, se_That, diff, pctDiff), p_values = cbind(iteration,
        p_fisher, p_dispersion, p_overdisp), rhat = cbind(iteration,
        rhat, se_rhat), reject_rate = cbind(alpha1,
        alpha2, alpha3)), class = "caribouSim")
    return(out)
}
```

Code for Simulations

```
############ phi = 0 #############

T <- 1e+06
n <- seq(1e-05 * T, 0.001 * T, by = 10)
iters <- 100

results <- vector(mode = "list", length = length(n))
rejections <- matrix(nrow = length(n), ncol = 3)
```

```r
for (i in seq(n)) {
    tmp <- iterate(iters = iters, T = T, n = n[i],
        phi = 0, r = 0.5, modelSim = "I")
    print(tmp)
    results[[i]] <- cbind(i, tmp$collars, tmp$abundance,
        tmp$p_values, tmp$rhat)
    rejections[i, ] <- tmp$reject_rate
}

write.csv(do.call(rbind, results), "simulations/results_phi0c.csv")
write.csv(rejections, "simulations/rejections_phi0c.csv")

############ phi = .1 #############

results <- vector(mode = "list", length = length(n))
rejections <- matrix(nrow = length(n), ncol = 3)
for (i in seq(n)) {
    tmp <- iterate(iters = iters, T = T, n = n[i],
        phi = 0.1, r = 0.5, modelSim = "I")
    print(tmp)
    results[[i]] <- cbind(i, tmp$collars, tmp$abundance,
        tmp$p_values, tmp$rhat)
    rejections[i, ] <- tmp$reject_rate
}

write.csv(do.call(rbind, results), "simulations/results_phi1c.csv")
write.csv(rejections, "simulations/rejections_phi1c.csv")

############ phi = .2 #############

results <- vector(mode = "list", length = length(n))
rejections <- matrix(nrow = length(n), ncol = 3)
for (i in seq(n)) {
    tmp <- iterate(iters = iters, T = T, n = n[i],
        phi = 0.2, r = 0.5, modelSim = "I")
    print(tmp)
    results[[i]] <- cbind(i, tmp$collars, tmp$abundance,
        tmp$p_values, tmp$rhat)
    rejections[i, ] <- tmp$reject_rate
}

write.csv(do.call(rbind, results), "simulations/results_phi2c.csv")
write.csv(rejections, "simulations/rejections_phi2c.csv")
```

```r
############ phi = .3 #############

results <- vector(mode = "list", length = length(n))
rejections <- matrix(nrow = length(n), ncol = 3)
for (i in seq(n)) {
    tmp <- iterate(iters = iters, T = T, n = n[i],
        phi = 0.3, r = 0.5, modelSim = "I")
    print(tmp)
    results[[i]] <- cbind(i, tmp$collars, tmp$abundance,
        tmp$p_values, tmp$rhat)
    rejections[i, ] <- tmp$reject_rate
}

write.csv(do.call(rbind, results), "simulations/results_phi3c.csv")
write.csv(rejections, "simulations/rejections_phi3c.csv")


############ phi = .4 #############

results <- vector(mode = "list", length = length(n))
rejections <- matrix(nrow = length(n), ncol = 3)
for (i in seq(n)) {
    tmp <- iterate(iters = iters, T = T, n = n[i],
        phi = 0.4, r = 0.5, modelSim = "I")
    print(tmp)
    results[[i]] <- cbind(i, tmp$collars, tmp$abundance,
        tmp$p_values, tmp$rhat)
    rejections[i, ] <- tmp$reject_rate
}

write.csv(do.call(rbind, results), "simulations/results_phi4c.csv")
write.csv(rejections, "simulations/rejections_phi4c.csv")


############ phi = .5 #############

results <- vector(mode = "list", length = length(n))
rejections <- matrix(nrow = length(n), ncol = 3)
for (i in seq(n)) {
    tmp <- iterate(iters = iters, T = T, n = n[i],
        phi = 0.5, r = 0.5, modelSim = "I")
    print(tmp)
    results[[i]] <- cbind(i, tmp$collars, tmp$abundance,
        tmp$p_values, tmp$rhat)
    rejections[i, ] <- tmp$reject_rate
}

write.csv(do.call(rbind, results), "simulations/results_phi5c.csv")
write.csv(rejections, "simulations/rejections_phi5c.csv")
```

Code Used in This Report

```
rejections_phi0 <- read.csv("../simulations/rejections_phi0.csv")
rejections_phi1 <- read.csv("../simulations/rejections_phi1.csv")
rejections_phi2 <- read.csv("../simulations/rejections_phi2.csv")
rejections_phi3 <- read.csv("../simulations/rejections_phi3.csv")
rejections_phi4 <- read.csv("../simulations/rejections_phi4.csv")
rejections_phi5 <- read.csv("../simulations/rejections_phi5.csv")
phi0 <- na.omit(read.csv("../simulations/results_phi0.csv"))
phi1 <- na.omit(read.csv("../simulations/results_phi1.csv"))
phi2 <- na.omit(read.csv("../simulations/results_phi2.csv"))
phi3 <- na.omit(read.csv("../simulations/results_phi3.csv"))
phi4 <- na.omit(read.csv("../simulations/results_phi4.csv"))
phi5 <- na.omit(read.csv("../simulations/results_phi5.csv"))
# fig1
library(caribouSim)
nu <- seq(10, 1000, by = 10)
plot_sim(dat = rejections_phi0, legend = "bottom", nu = nu) +
  labs(title = expression(paste(phi, " = 0")))
# fig2

plot1 <- plot_sim(dat = rejections_phi1, nu = nu) +
                  labs(title = expression(paste(phi, " = 0.1")))
plot2 <- plot_sim(dat = rejections_phi2, nu = nu) +
                  labs(title = expression(paste(phi, " = 0.2")))
plot3 <- plot_sim(dat = rejections_phi3, nu = nu) +
                  labs(title = expression(paste(phi, " = 0.3")))
plot4 <- plot_sim(dat = rejections_phi4, nu = nu) +
                  labs(title = expression(paste(phi, " = 0.4")))
plot5 <- plot_sim(dat = rejections_phi5, nu = nu) +
                  labs(title = expression(paste(phi, " = 0.5")))

library(cowplot)
```

```r
plots <- plot_grid(plot1, plot2, plot3, plot4, plot5, ncol = 1)
legend <- get_legend(plot_sim(dat = rejections_phi1, legend = "bottom", nu = nu))
plot_grid(plots, legend, rel_heights = c(1, .05), ncol = 1)
# table 2
Ni <- seq(10, 1000, by = 10)
tmp <- as.data.frame(sapply(rejections_phi1, FUN = function(x) zoo::rollmean(x, k = 9))))
power1 <- Ni[c(nearest(tmp$V1, .5), nearest(tmp$V1, .8),
               nearest(tmp$V2, .5), nearest(tmp$V2, .8),
               nearest(tmp$V3, .5), nearest(tmp$V3, .8))] # repeat for other values of phi
# mean Ttrue and That for each sample size n

# table 3
tab <- data.frame(phi = 0:5, SSE = c(
                  sum((phi0$Ttrue - phi0$That)^2),
                  sum((phi1$Ttrue - phi1$That)^2),
                  sum((phi2$Ttrue - phi2$That)^2),
                  sum((phi3$Ttrue - phi3$That)^2),
                  sum((phi4$Ttrue - phi4$That)^2),
                  sum((phi5$Ttrue - phi5$That)^2))
)
set.seed(123)
plot_grid(plot_meanTs(phi0, legend = c(.7, .2), nu = nu),
          plot_pctDiff(phi0, phi = 0.0, prop = .1), ncol = 2)
plot_grid(plot_meanTs(phi2, legend = c(.7, .2), nu = nu),
          plot_pctDiff(phi2, phi = 0.0, prop = .1), ncol = 2)
plot_grid(plot_meanTs(phi5, legend = c(.7, .2), nu = nu),
          plot_pctDiff(phi5, phi = 0.0, prop = .1), ncol = 2)
phi0_means <- data.frame(nu = nu,
                         Ttrue = tapply(phi0[,"Ttrue"], INDEX = factor(phi0$i), FUN = mean),
                         That = tapply(phi0[,"That"], INDEX = factor(phi0$i), FUN = mean))

phi1_means <- data.frame(nu = nu,
                         Ttrue = tapply(phi1[,"Ttrue"], INDEX = factor(phi1$i), FUN = mean),
                         That = tapply(phi1[,"That"], INDEX = factor(phi1$i), FUN = mean))

phi2_means <- data.frame(nu = nu,
                         Ttrue = tapply(phi2[,"Ttrue"], INDEX = factor(phi2$i), FUN = mean),
                         That = tapply(phi2[,"That"], INDEX = factor(phi2$i), FUN = mean))

phi3_means <- data.frame(nu = nu,
                         Ttrue = tapply(phi3[,"Ttrue"], INDEX = factor(phi3$i), FUN = mean),
                         That = tapply(phi3[,"That"], INDEX = factor(phi3$i), FUN = mean))
```

```r
phi4_means <- data.frame(nu = nu,
                         Ttrue = tapply(phi4[,"Ttrue"], INDEX = factor(phi4$i), FUN = mean),
                         That = tapply(phi4[,"That"], INDEX = factor(phi4$i), FUN = mean))

phi5_means <- data.frame(nu = nu,
                         Ttrue = tapply(phi5[,"Ttrue"], INDEX = factor(phi5$i), FUN = mean),
                         That = tapply(phi5[,"That"], INDEX = factor(phi5$i), FUN = mean))

# table 4
mat <- matrix(NA, nrow = 6, ncol = 6)
t(mat)
for(i in 1:6){
  for(j in 1:6){
    if(i == j) next
    mat[i,j] <- wilcox.test(c(eval(parse(text = paste0("phi", i-1, "_means$That")))
                            - eval(parse(text = paste0("phi", i-1, "_means$Ttrue")))),
                            c(eval(parse(text = paste0("phi", j-1, "_means$That")))
                            - eval(parse(text = paste0("phi", j-1, "_means$Ttrue")))),
                              paired = TRUE, alternative = "less")$p.value
  }
}
mat
```