HatchPBT

A statistical tool to conduct an *a priori* analysis of the precision and accuracy of the maximum likelihood estimator of the proportion of hatchery-origin spawners using parentage-based tagging

Richard A. Hinrichsen

June 4, 2022

INTRODUCTION

Assessments of the status of endangered Columbia River salmon populations require reliable estimates of the proportion of hatchery-origin spawners on the spawning grounds. Without such an estimate, it would be impossible to estimate a trend in the wild-origin population abundance or estimate population extinction risks (Hinrichsen 2003; McClure et al. 2003). Furthermore, quantifying the potential for interbreeding between hatchery-origin spawners and wild-origin spawners in the wild, which may reduce the genetic fitness of subsequent generations of wild-origin fish, also depends on this estimate (Waples 1991). The potential genetic risks of all 178 hatchery programs in the Columbia River basin were assessed using the proportion of hatchery-origin spawners (Mobrand et al. 2005; HSRG 2009). To allow distinction between natural-origin and hatchery-origin salmon in the Columbia Basin, the U.S. Congress presently requires the US Fish and Wildlife Service to visibly mark all hatchery

production intended for harvest.[1] Visible marking of hatchery releases is a widespread practice among hatchery operators in the Columbia River basin, though non-visible marking procedures are sometimes substituted for or added to visible marks.

Despite the importance of estimates of proportion of hatchery-origin fish (p) on the spawning grounds, until recently, reliable estimation techniques have been lacking. The statistical difficulty of estimating the proportion of hatchery-origin escapement when some hatchery-fish are not visibly marked has been recognized for over thirty years (Hankin 1982). The difficulty is especially pronounced when different source hatcheries do not use the same marking fraction. Hinrichsen et al. (2012) addressed the problem of estimating p in a hatchery program that used visible marks and coded-wire tag recoveries. The authors developed a generalized least squares estimator of the proportion of hatchery origin spawners and compared it to a simplified method of moments estimator.

An alternative to this CWT approach to estimating the proportion of hatchery-origin spawners is to use genetic tagging called parentage-based tagging (PBT) of hatchery releases, which can be used to mark a high percentage of juveniles released. In this alternative approach, some juveniles are visibly marked (VM), parentage-based tagged (PBT), or both. Many (but not all) hatchery juveniles in the Columbia River basin are released with a VM with an adipose fin clip, a ventral fin clip, or a visible implant elastomer tag. PBT involves genotyping hatchery broodstock (parents) and adding these genotypes to a database (Steele et al. 2011; Anderson and Garza 2005; Anderson and Garza 2006). Genotyped progeny of these parents collected as juveniles or adults can be assigned back to their parents, thus creating a tag identifying the hatchery of origin. Thus offspring of the genotyped brood stock are genetically tagged. Software

[1] On June 27, 2007, the House passed (amended) H.R. 2643, including a provision requiring the U.S. Fish and Wildlife Service to implement a system of mass marking of salmonid stocks that are released from federal hatcheries

used to assign genotyped progeny to their parents, SNPPIT 1.0 developed by Anderson (2010), is available online at http://www.mybiosoftware.com/population-genetics/6013.

In a carcass survey, VM fish are recovered as adults at a spawning area together with fish that are the progeny of salmon spawning in the wild. A subsample of the sampled carcasses is then drawn and each fish in the subsample is genotyped to determine if it is PBT. The genotypes of subsampled carcasses are compared to the genotypes of parents in a database. If there is a match, the sampled carcass represents a PBT spawner, and the hatchery of origin is determined. Note that the subsample potentially consists of spawners in three different categories: VM hatchery-origin spawners, not VM hatchery-origin spawners, and wild-origin spawners. Therefore, it is not guaranteed that each genotyped spawner will carry a PBT.

The goal of this documentation is to present a method for evaluating different study designs aimed at estimating the proportion of hatchery-origin spawners in a program that uses VM and PBT to identify hatchery-origin spawners. Equations for a maximum likelihood estimator and its precision and bias are developed as in Hinrichsen et al. (2016). These may be used to formulate an experimental design that has a high chance of delivering the best accuracy and precision given some design constraints. The designs explored allow the possibility of two or more source hatcheries. In general, these source hatcheries may use different VM fractions and (possibly) different PBT fractions. The precision of the MLE developed depends on several quantities, some of which may be selected by a researcher or manager: the number of carcasses sampled, the VM fractions, the PBT fractions, and the actual proportions of hatchery fish from each source hatchery. These represent management controls that may be manipulated to achieve a target level of precision in the proportion of hatchery-origin spawners.

METHODS

In order to estimate the proportion of hatchery-origin spawners, I specify a probability distribution for observed spawners using a two-stage sampling protocol (Figure 1). The protocol uses an initial sample of size N to determine how many fish are VM and not VM (denoted ~VM). Then a random (sub)sample of the VM group is taken and genotyped. And a random (sub) sample of the spawners ~VM is also taken and genotyped. One possibility, if the initial

sample is not prohibitively large, is to genotype all of the spawners from the initial sample. At another extreme, the random subsamples of the VM and ~VM groups could represent a small fraction of the initial sample. One of the main goals of this work is to determine how large of a subsample is sufficient to obtain a precise estimate of the proportion of hatchery-origin spawners. The steps for collecting spawner data necessary to estimate the proportion of hatchery-origin spawners are the following:

1) Take a random sample of size *N* of the spawners in the wild. Note that this sample will consist of two groups: fish that are VM and fish without a VM (i.e., ~VM)
2) Take a random (sub)sample of size n_1 from the VM group and a random (sub)sample of size n_2 from the ~VM group. Note that the total subsample size is $n = n_1 + n_2$.
3) Test these subsamples for PBT. Note the hatchery of origin for each fish that is PBT.

The method I develop in this documentation is a maximum likelihood technique (Mood et al. 1974). Other estimation techniques are also possible, for example, a generalized least squares method based on the method of moments (Hinrichsen et al. 2012; Kariya and Kurata 2004). I use maximum likelihood because it has a well-developed theory. Maximum likelihood estimators have desirable statistical properties such as asymptotic normality, functional invariance, and consistency (Mood et al. 1974). Furthermore, the variance of the MLE may be derived as the inverse of the Fisher Information Matrix, which is the negative of the expected value of the Hessian of the likelihood function (Mood et al. 1974).

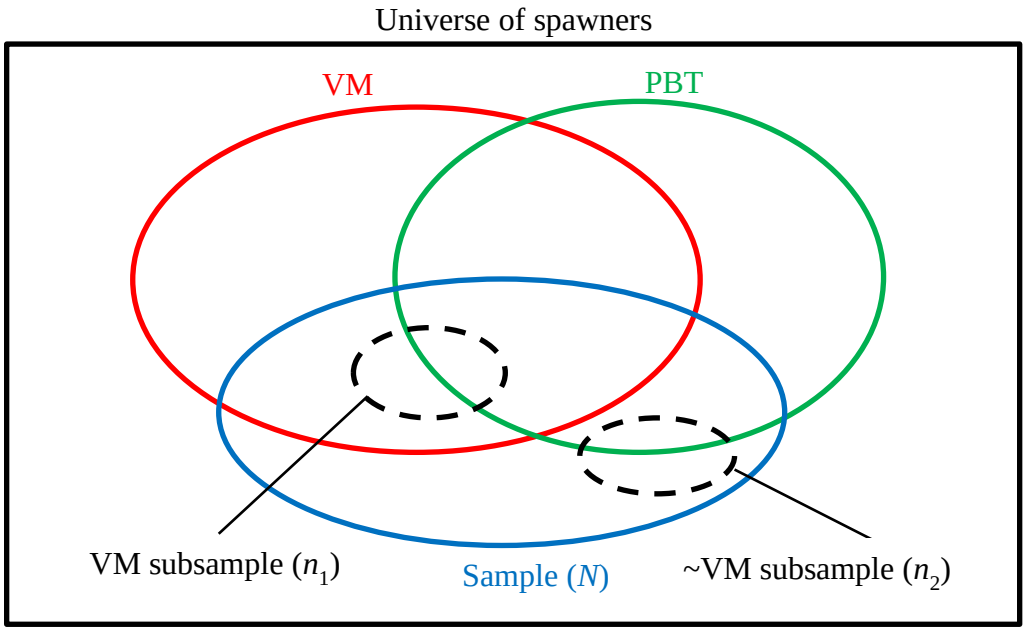


Figure 1.—Carcass sampling represented by a Venn diagram. The universe is represented by all returning spawners to a particular spawning ground in the wild. The sample of size N is a random sample of the universe of spawners. The subsamples of sizes n_1 and n_2 are random samples of the VM spawners and not VM (~VM) spawners, respectively. These subsamples, totaling $n = n_1 + n_2$, are checked for a PBT, while the remaining carcasses, totaling $N - n$, are not.

Experiment and Probability Model

This section is devoted to developing the maximum likelihood estimator (MLE) of the proportion of hatchery-origin spawners and its variance. I begin by defining the assumptions (Table 1) and the variables used in the study, which are used to develop the probability model. For convenience, variable names and their definitions are given in Appendix A. Statistical code for the analysis, written in the R programming language, may be found in Appendix B. Let p_i represent the fraction of spawners on the spawning grounds that originated at hatchery i. Let λ_i represent the VM fraction that is applied to hatchery fish releases from hatchery i, and let ϕ_i represent the PBT fraction that is applied to hatchery fish releases from hatchery i. Further

assume that the total number of spawners sampled is *N* (fixed) and that x_1 represents the number of sampled fish that are VM and $x_2 = N - x_1$ represents the number of sampled fish that are not VM. Here, x_1 is a binomial deviate with the number of 'trials' equal to *N* and the probability of 'success' equal to the probability that a spawner is VM, which is $\sum_{i=1}^{m} \lambda_i p_i$.

Table 1.—Assumptions[1]

(A1)	Hatchery-specific VM fractions and number of spawners sampled are known.
(A2)	Hatchery-specific PBT fractions are known.
(A3)	Every individual spawner has the same probability of being sampled.
(A4)	Every individual hatchery-origin spawner from the same hatchery has the same probability of having a VM.
(A5)	Every individual hatchery-origin spawner from the same hatchery has the same probability of having a PBT.
(A6)	Whether an individual is sampled has no effect on the probability that another individual is sampled.
(A7)	Whether an individual hatchery-origin spawner is VM has no effect on the probability that another individual will be VM.
(A8)	Whether an individual hatchery-origin spawner is PBT has no effect on the probability that another individual will be PBT.

[1] For convenience, I derived the estimators for releases grouped at the hatchery level. To split the data by release group instead, simply replace "hatchery" by "release" in the estimation method and interpret VM fractions and PBT fractions as release-specific.

Further assume that a subsample of sample of *N* spawners is tested for a PBT that allows investigators to determine the hatchery of origin. A total of n_1 VM spawners are genotyped and n_2 unmarked spawners are genotyped to determine whether they are PBT. The total subsample size is then $n = n_1 + n_2$. To make the subsample sizes meaningful, I used the following restrictions: $E(x_1) - (N - n) \le n_1 \le E(x_1)$, $E(x_2) - (N - n) \le n_2 \le E(x_2)$, and $0 < n \le N$. I use these restrictions because n_1 must be less than or equal to the number of VM fish in the sample (

x_1) and n_2 must be less than or equal to the number of unmarked fish in the total sample (x_2), and the subsample size must be less than the total sample size. The lower bounds follow from applying the identities $n = n_1 + n_2$ and $N = E(x_1) + E(x_2)$ to these upper bound inequalities. The number of genotyped marked fish originating at hatchery i and determined to have a PBT is denoted by the random variable y_i and the number of genotyped unmarked fish originating at hatchery i and determined to have a PBT is z_i. The values y_i are considered cell counts from a multinomial distribution with n_1 trials with cell probabilities $\lambda_i \phi_i p_i / \sum_{i=1}^{m} \lambda_i p_i$. The values z_i are cell counts from a multinomial distribution with n_2 trials and cell probabilities $(1-\lambda_i)\phi_i p_i / \left(1 - \sum_{i=1}^{m} \lambda_i p_i\right)$.

In developing the marking and PBT program, it is critical that all hatchery fish are released randomly, and each hatchery fish is equally likely to have a VM or PBT, and that the probability a hatchery fish is VM does not influence the probability that it is PBT and visa versa. (i.e., the event that a hatchery fish is VM is independent of the event that it is PBT). Note, hatchery managers need to be aware that these hatchery fish are all treated equally, as if any group is treated in a different manner, the Table 1 assumptions will be violated. Likewise, in the second event sampling (i.e., drawing of the subsamples), the subsamples taken should be truly random and representative of the populations (i.e. if complete mixing is not occurring/homogeneity of the sample is violated, this will not work).

The assumptions in Table 1 allow one to express the joint distribution of spawner counts as a product of multinomial distributions:

$$f(x_1, x_2, y_1, \ldots y_m, z_1, \ldots z_m) = \binom{N}{x_1, x_2} \left(\sum_{i=1}^{m} \lambda_i p_i\right)^{x_1} \left(1 - \sum_{i=1}^{m} \lambda_i p_i\right)^{x_2} \quad (1)$$

$$\times \binom{n_1}{y_1,\ldots y_m, n_1-\sum_{i=1}^{m} y_i} \prod_{i=1}^{m} \left(\frac{\lambda_i \phi_i p_i}{\sum_{k=1}^{m} \lambda_k p_k} \right)^{y_i} \left(1 - \frac{\sum_{k=1}^{m} \lambda_k \phi_k p_k}{\sum_{k=1}^{m} \lambda_k p_k} \right)^{n_1 - \sum_{k=1}^{m} y_k}$$

$$\times \binom{n_2}{z_1,\ldots z_m, n_2-\sum_{i=1}^{m} z_i} \prod_{i=1}^{m} \left[\frac{(1-\lambda_i)\phi_i p_i}{1-\sum_{k=1}^{m} \lambda_k p_k} \right]^{z_i} \left[1 - \frac{\sum_{i=1}^{m} (1-\lambda_i)\phi_i p_i}{1-\sum_{k=1}^{m} \lambda_k p_k} \right]^{n_2 - \sum_{k=1}^{m} z_k}.$$

Given the joint distribution of the observations in equation (1), it is now possible to form the log-likelihood function of the unknown parameters $p_1, p_2, \ldots p_m$ by taking the natural log of the joint distribution and treating the result as a function of the parameters:

$$l(p_1, p_2 \ldots, p_m) = x_1 \log\left(\sum_{i=1}^{m} \lambda_i p_i \right) + x_2 \log\left(1 - \sum_{i=1}^{m} \lambda_i p_i \right) \tag{2}$$

$$+ \sum_{i=1}^{m} y_i \log\left(\frac{\lambda_i \phi_i p_i}{\sum_{i=1}^{m} \lambda_i p_i} \right) + \left(n_1 - \sum_{k=1}^{m} y_k \right) \log\left(1 - \frac{\sum_{k=1}^{m} \lambda_k \phi_k p_k}{\sum_{i=1}^{m} \lambda_k p_k} \right)$$

$$+ \sum_{i=1}^{m} z_i \log\left[\frac{(1-\lambda_i)\phi_i p_i}{1-\sum_{k=1}^{m} \lambda_k p_k} \right] + \left(n_2 - \sum_{k=1}^{m} z_k \right) \log\left[1 - \frac{\sum_{k=1}^{m} (1-\lambda_k)\phi_k p_k}{1-\sum_{k=1}^{m} \lambda_k p_k} \right].$$

Score Function

To determine the MLEs of p_i, I set the partial derivatives of the log likelihood function equal to zero and solve for the unknown values of p_i. The partial derivatives of the log likelihood function are

$$\frac{\partial l}{\partial p_i} = \frac{\lambda_i (x_1 - n_1)}{\sum_{k=1}^{m} \lambda_k p_k} - \frac{\lambda_i (N - x_1 - n_2)}{1 - \sum_{k=1}^{m} \lambda_k p_k} + \frac{y_i + z_i}{p_i} + \frac{(n_1 - \sum_{k=1}^{m} y_k)(1-\phi_i)\lambda_i}{\sum_{k=1}^{m} (1-\phi_k)\lambda_k p_k} \tag{3}$$

$$- \frac{(n_2 - \sum_{k=1}^{m} z_k)\left[(1-\phi_i)\lambda_i + \phi_i\right]}{1 - \sum_{k=1}^{m} \left[(1-\phi_k)\lambda_k + \phi_k\right] p_k}.$$

The MLEs of p_i (which are the zeroes of the score function) are found numerically using Fisher's Scoring Method, which will be described in detail later in this documentation. Notice that there are special cases that must be considered where the log-likelihood function in equation (3) is undefined: namely, when the PBT fractions (ϕ_i) are all one, or when the VM fractions (λ_i) are all zero.

Special case (all visually marked releases PBT).—In the special case where there are some VM releases and all the visually marked releases are PBT, the log-likelihood equation becomes

$$\frac{\partial l}{\partial p_i} = \frac{\lambda_i (x_1 - n_1)}{\sum_{k=1}^{m} \lambda_k p_k} - \frac{\lambda_i (N - x_1 - n_2)}{1 - \sum_{k=1}^{m} \lambda_k p_k} + \frac{y_i + z_i}{p_i} - \frac{(n_2 - \sum_{k=1}^{m} z_k)\phi_i}{1 - \sum_{k=1}^{m} \phi_k p_k}. \quad [(1-\phi_i)\lambda_i = 0 \text{ for } i = 1, \ldots, n] \tag{4}$$

Special case (no releases VM).—In the special case where no releases are VM, then the log-likelihood equation becomes

$$\frac{\partial l}{\partial p_i} = \frac{z_i}{p_i} - \frac{(n_2 - \sum_{k=1}^{m} z_k)\phi_i}{1 - \sum_{k=1}^{m} \phi_k p_k} \qquad (\lambda_i = 0 \text{ for } i = 1, \ldots, n) \tag{5}$$

Fisher Information Matrix

The next step in deriving the theoretical formulas for precision of the MLEs is to derive the Fisher Information Matrix. The inverse of the Fisher Information Matrix will supply the variances and covariances of the MLEs of p_i. The Fisher Information Matrix is the negative of the expected value of the Hessian of the likelihood function. The Hessian of likelihood function (**H**) is found by further differentiating the log likelihood. Its diagonal elements are given by

$$H_{ii} = -\frac{\lambda_i^2 (x_1 - n_1)}{\left(\sum_{k=1}^{m} \lambda_k p_k\right)^2} - \frac{\lambda_i^2 (N - x_1 - n_2)}{\left(1 - \sum_{k=1}^{m} \lambda_k p_k\right)^2} - \frac{(y_i + z_i)}{p_i^2} - \frac{\left(n_1 - \sum_{k=1}^{m} y_k\right)(1 - \phi_i)^2 \lambda_i^2}{\left[\sum_{k=1}^{m} (1 - \phi_k)\lambda_k p_k\right]^2} \tag{6}$$

$$- \frac{\left(n_2 - \sum_{k=1}^{m} z_k\right)\left((1 - \phi_i)\lambda_i + \phi_i\right)^2}{\left\{1 - \sum_{k=1}^{m} \left[(1 - \phi_k)\lambda_k + \phi_k\right] p_k\right\}^2},$$

and off-diagonal elements

$$H_{ij} = H_{ji} = -\frac{\lambda_i \lambda_j (x_1 - n_1)}{\left(\sum_{k=1}^{m} \lambda_k p_k\right)^2} - \frac{\lambda_i \lambda_j (N - x_1 - n_2)}{\left[1 - \sum_{k=1}^{m} \lambda_k p_k\right]^2} - \frac{\left(n_1 - \sum_{k=1}^{m} y_k\right)(1-\phi_i)(1-\phi_j)\lambda_i \lambda_j}{\left[\sum_{k=1}^{m} (1-\phi_k)\lambda_k p_k\right]^2}$$

$$- \frac{\left(n_2 - \sum_{k=1}^{m} z_k\right)\left[(1-\phi_i)\lambda_i + \phi_i\right]\left[(1-\phi_j)\lambda_j + \phi_j\right]}{\left\{1 - \sum_{k=1}^{m} \left[(1-\phi_k)\lambda_k + \phi_k\right] p_k\right\}^2}, \tag{7}$$

where $i \neq j$; $i = 1,2,\ldots m$; and $j = 1,2,\ldots m$. I assume that not all of the p_i are zero, not all of the VM fractions are zero, and not all of the PBT fractions are equal to one; otherwise equations (6) and (7) will be undefined. The special case where all hatchery-specific PBTs equal one will be considered separately. I will also consider the special case where all VM fractions are zero.

To determine the Fisher Information Matrix, the expected values of x_1, x_2, y_i, and z_i are needed. These are given by

$$E(x_1) = N \sum_{i=1}^{m} \lambda_i p_i \,, \tag{8}$$

$$E(x_2) = N\left(1 - \sum_{i=1}^{m} \lambda_i p_i\right), \tag{9}$$

$$E(y_i) = \frac{n_1 \lambda_i \phi_i p_i}{\sum_{k=1}^{m} \lambda_k p_k}, \tag{10}$$

and

$$E(z_i) = \frac{n_1 (1-\lambda_i) \phi_i p_i}{1 - \sum_{k=1}^{m} \lambda_k p_k}. \tag{11}$$

Taking the expected value of the Hessian by using the expected values of the observations of x_1, x_2, y_i, and z_i yields the following Fisher Information Matrix ($\mathbf{I}$) with diagonal entries:

$$I_{ii} = -E(H_{ii}) = \frac{N\lambda_i^2 (1-\theta_1)}{\sum_{k=1}^{m} \lambda_k p_k} + \frac{N\lambda_i^2 (1-\theta_2)}{1 - \sum_{k=1}^{m} \lambda_k p_k} + \frac{N\phi_i \left[\theta_1 \lambda_i + \theta_2 (1-\lambda_i)\right]}{p_i} + \frac{N\theta_1 (1-\phi_i)^2 \lambda_i^2}{\sum_{k=1}^{m} (1-\phi_k) \lambda_k p_k} \tag{12}$$

$$+ \frac{N\theta_2 \left[(1-\phi_i)\lambda_i + \phi_i\right]^2}{1 - \sum_{k=1}^{m} \left[(1-\phi_k)\lambda_k + \phi_k\right] p_k},$$

and off-diagonal entries

$$I_{ij} = I_{ji} = -E(H_{ij}) = \frac{N\lambda_i\lambda_j(1-\theta_1)}{\sum_{k=1}^{m}\lambda_k p_k} + \frac{N\lambda_i\lambda_j(1-\theta_2)}{1-\sum_{k=1}^{m}\lambda_k p_k} + \frac{N\theta_1(1-\phi_i)\lambda_i(1-\phi_j)\lambda_j}{\sum_{k=1}^{m}(1-\phi_k)\lambda_k p_k} \tag{13}$$

$$+ \frac{N\theta_2[(1-\phi_i)\lambda_i + \phi_i][(1-\phi_j)\lambda_j + \phi_j]}{1-\sum_{k=1}^{m}[(1-\phi_k)\lambda_k + \phi_k]p_k},$$

where $\theta_1 = n_1(N\sum_{i=1}^{m}\lambda_i p_i)^{-1}$ and $\theta_2 = n_2\left[N\left(1-\sum_{i=1}^{m}\lambda_i p_i\right)\right]^{-1}$. I assume that not all of the p_i are zero, not all of the VM fractions are zero, and not all of the PBT fractions are equal to one; otherwise equations (11) and (12) would be undefined.

Special case (all visually marked releases PBT).—In the special case where some releases are visually marked and all the visually marked releases are PBT, the information matrix is given by diagonal entries

$$I_{ii} = \frac{N\lambda_i^2(1-\theta_1)}{\sum_{k=1}^{m}\lambda_k p_k} + \frac{N\lambda_i^2(1-\theta_2)}{1-\sum_{k=1}^{m}\lambda_k p_k} + \frac{N[\theta_1\lambda_i + \theta_2(1-\lambda_i)]}{p_i} \qquad [(1-\phi_i)\lambda_i = 0 \text{ for } i = 1,\ldots,n] \tag{14}$$

$$+ \frac{N\theta_2\phi_i^2}{1-\sum_{k=1}^{m}\phi_k p_k}$$

and off-diagonal entries

$$I_{ij} = I_{ji} = \frac{N\lambda_i \lambda_j (1-\theta_1)}{\sum_{k=1}^{m} \lambda_k p_k} + \frac{N\lambda_i \lambda_j (1-\theta_2)}{1-\sum_{k=1}^{m} \lambda_k p_k} + \frac{N\theta_2 \phi_i \phi_j}{1-\sum_{k=1}^{m} \phi_k p_k}. \qquad [(1-\phi_i)\lambda_i = 0 \text{ for } i = 1,\ldots,n] \tag{15}$$

Special case (no releases VM).—In the special case where no releases are VM, the information matrix is given by diagonal entries

$$I_{ii} = \frac{N\phi_i \theta_2}{p_i} + \frac{N\theta_2 \phi_i^2}{1-\sum_{k=1}^{m} \phi_k p_k} \qquad (\lambda_i = 0 \text{ for } i = 1,\ldots,n) \tag{16}$$

and off-diagonal entries

$$I_{ij} = I_{ji} = \frac{N\theta_2 \phi_i \phi_j}{1-\sum_{k=1}^{m} \phi_k p_k} \qquad (\lambda_i = 0 \text{ for } i = 1,\ldots,n) \tag{17}$$

Precision

Given that the Fisher Information Matrix is calculated and it is invertible, it is possible to derive the variance of the MLE of the proportion of hatchery-origin spawners. From the variance, the standard error and coefficient of variation, both useful measures of precision may be calculated. The vector of hatchery-specific maximum likelihood estimators

$$\hat{\mathbf{p}} = \begin{bmatrix} \hat{p}_1 & \hat{p}_2 & \ldots & \hat{p}_m \end{bmatrix}' \tag{18}$$

is given by

$$\text{var}(\hat{\mathbf{p}}) = \mathbf{I}^{-1}, \tag{19}$$

which is the inverse of the Fisher Information Matrix. This formula holds provided that **I** is invertible, which is not guaranteed. To determine the variance of the estimator of the overall proportion of hatchery-origin spawners, $\hat{p} = \sum_{i=1}^{m} \hat{p}_i$, I use the formula

$$\text{var}(\hat{p}) = \text{var}(\mathbf{e}'\hat{\mathbf{p}}) = \mathbf{e}'\,\text{var}(\hat{\mathbf{p}})\mathbf{e}, \tag{20}$$

where $\mathbf{e}$ is an *m*-vector of 1s. The standard error of $\hat{p}$ is then

$$SE(\hat{p}) = \sqrt{\text{var}(\hat{p})}, \tag{21}$$

and the coefficient of variation of $\hat{p}$ is

$$CV(\hat{p}) = \frac{SE(\hat{p})}{\hat{p}}. \tag{22}$$

The minimum SE is obtained when both n_1 and n_2 are as large as possible. This occurs when $n = N$, $n_1 = E(x_1)$, and $n_2 = E(x_2)$. To find the minimal standard error, $SE_{\min}(\hat{p})$, substitute these values in the Fisher Information matrix equations, then use equations (19)-(21). The $CV_{\min}(\hat{p})$ may be found by employing equations (19)-(22). This minimum SE is useful when evaluating the precision of alternative choices for subsample sizes n_1 and n_2.

Fisher's Scoring Method

The MLEs of the hatchery-specific proportions of hatchery-origin spawners are calculated using Fisher's Scoring Method, which is similar to Newton's method for solving nonlinear equations (Press et al. 1992), but modified to use the expected Hessian (i.e., **-I**) in place of the usual Hessian (Jennrich and Sampson 1976). In this case, Fisher's Scoring Method becomes

$$\hat{\mathbf{p}}_{l+1} = \hat{\mathbf{p}}_l + \mathbf{I}^{-1}(\hat{\mathbf{p}}_l)\nabla l(\hat{\mathbf{p}}_l), \quad (23)$$

where $\hat{\mathbf{p}}_{l+1}$ is the maximum likelihood estimator of $\mathbf{p}$ on iteration l+1, $\hat{\mathbf{p}}_l$ is the maximum likelihood estimator of $\mathbf{p}$ on iteration l, $\mathbf{I}^{-1}(\hat{\mathbf{p}}_l)$ is the inverse of the Fisher Information Matrix evaluated at $\hat{\mathbf{p}}_l$, and $\nabla l(\hat{\mathbf{p}}_l)$ is the gradient or score function, which represents the vector of first partial derivatives of the likelihood function evaluated at $\hat{\mathbf{p}}_l$. The algorithm proceeds by making a first initial guess at the MLE, then iterating equation (23) until the estimate changes by no more than a given error tolerance (e.g., 10^{-5}).

Estimated proportions of hatchery-origin spawners equal to zero.—A special case emerges when conducting Monte Carlo simulations with the Fisher's Scoring Method. If a hatchery has a positive expected number of tag recoveries the marking fraction is zero and zero PBT recoveries then the estimate of the proportion of hatchery-origin spawners is zero for that hatchery (i.e., $\hat{p}_k^* = 0$). In this case, the Fisher Information Matrix (evaluated at $\hat{p}_k^* = 0$) is not defined due to division by zero. To remedy this difficulty, the hatchery contributing an estimated zero spawners is removed from the estimation procedure, effectively reducing the dimension of the estimation problem and eliminating the division by zero.

Special singular cases in the estimation problem

There are important special cases to consider in the estimation of *p* when the Fisher Information is singular and therefore Fisher's Scoring Method cannot be applied as described in the previous section. In the first case I present, the dimension of the estimation problem may be reduced allowing estimation of *p* to proceed. In the second case, the estimation problem cannot be saved.

Singular case for the Fisher Information Matrix where p is estimable.—In the special case where two or more hatcheries have expected tag recoveries of zero, the Fisher Information matrix is singular and it is impossible to estimate the individual contributions of these individual hatcheries. However, if each of these hatcheries uses the same marking rate, *p* is still estimable, an alternative, lower dimension Fisher Information matrix may be formulated. To estimate *p*,

combine the hatcheries with expected tag recoveries of zero, i.e., $E(y_i + z_i) = 0$, into a single hatchery for the purposes of estimation. If there are m_0 hatcheries that have expected tag recoveries of zero (and constant marking fractions), then the revised "number of hatcheries" used for the purposes of analysis is $m_{new} = m - m_0 + 1$. Assume that the hatchery indices are arranged so that the first m_0 hatcheries those with expected number of tag recoveries equal to zero. Then the VM fractions for the purpose of analysis are given by $\left[\lambda_1, \lambda_{m_0+1}, \ldots, \lambda_m\right]'$, where λ_1 is the marking fraction of the first m_0 hatcheries. The PBT fractions for the purpose of analysis are given by $\left[0, \phi_{m_0+1}, \ldots, \phi_m\right]'$. I chose zero for the PBT fraction of the hatcheries with expected tag recoveries of zero, but any PBT fractions would suffice because they do not enter into the calculation of $\hat{p}$. Note that when a single hatchery alone has an expected number of tag recoveries of zero and a positive visual marking fraction, then the original Fisher Information Matrix is not singular, and estimation of all the hatchery-specific proportions of hatchery-origin spawners may proceed without reducing the dimension of the estimation problem.

Note that when marking fractions at all source hatcheries are identical and there are no expected PBT recoveries from any of the hatcheries, this fits the preceding special case. This is the case of the constant marking fraction, and the statistical properties of the MLE of *p* (equivalent to the GLSE of *p* in this case) are well-known Hinrichsen et al. (2012).

Singular cases for the Fisher Information Matrix where p is not estimable.—There are important singular cases where it is impossible to estimate the fraction of hatchery-origin spawners. This occurs when the expected number of PBT recoveries is zero at two or more hatcheries and VM fractions differ at these hatcheries or are zero. Estimation is also ruined when a single hatchery has an expected number of PBT recoveries of zero and a visual marking fraction of zero. In these cases, the estimation problem cannot be saved by reducing its dimension: a redesign of the study is required. One possible redesign that would ensure that *p* is estimable would be to use the same visual marking fractions at hatcheries with expected PBT recoveries of zero. Another possible redesign would be to insist that the expected number of PBT recoveries for hatchery is not zero.

Monte Carlo Estimates of Precision and Bias

As an alternative approach to estimating the precision of $\hat{p}$, Monte Carlo simulation is used. Additionally, this approach yields an estimate of accuracy (bias). Monte Carlo estimates of precision and accuracy do not rely on asymptotic theory as in the previous sections. That is, Monte Carlo estimates of precision and bias will work with small sample sizes (*N*) and small subsample sizes, n_1 and n_2. The Monte Carlo method proceeds by drawing NSIM random samples from the joint distribution of $x_1, x_2, y_1, \ldots, y_m, z_1, \ldots z_m$ given by equation (1), then calculating the MLE of the proportion of hatchery-origin spawners for each Monte Carlo sample using Fisher's Scoring Method. This yields NSIM replications of the MLE, denoted by $\hat{p}_1^*, \hat{p}_2^*, \ldots, \hat{p}_{NSIM}^*$. The Monte Carlo estimate of the standard error of the MLE of the proportion of hatchery-origin spawners is then equal to the square root of the sample variance of the Monte Carlo replications:

$$SE^*(\hat{p}) = \sqrt{\sum_{k=1}^{NSIM} \frac{\left(\hat{p}_k^* - \bar{\hat{p}}^*\right)^2}{NSIM - 1}} \ , \tag{24}$$

where $\bar{\hat{p}}^*$ is the sample mean of the Monte Carlo replications. The coefficient of variation of $\hat{p}$ is

$$CV^*(\hat{p}) = \frac{SE^*(\hat{p})}{\hat{p}} \ . \tag{25}$$

The relative bias is calculated as

$$bias^*(\hat{p}) = \frac{\hat{p} - \bar{\hat{p}}^*}{\hat{p}} \ . \tag{26}$$

Note Monte Carlo estimates of minimum standard error, denoted $SE^{*}_{\min}(\hat{p})$, and minimum coefficient of variation, denoted $CV^{*}_{\min}(\hat{p})$, are obtained by using $n = N$ and $n_1 = E(x_1)$ in the simulations. These quantify the precision obtained by testing the entire sample of size *N* for PBT.

Developing an Experimental Design

The equations for precision and bias may be used to formulate an experimental design that has a high chance of delivering the best accuracy and precision given some design constraints. The precision, as measured by SE of $\hat{p}$, is a function of VM fractions, PBT fractions, sample size (N), and subsample sizes n_1 and n_2, which, to some extent, may be manipulated to give lowest possible SE. For example, as sample size and subsample sizes and PBT fractions are increased, precision will always increase. One possible constraint on the experimental design may be the number of carcasses tested for PBT. Suppose this quantity is fixed at, say, $n = 20$. The level of precision one can expect from such a study will vary according to the fraction of these 20 fish drawn from the VM group. By varying n_1 and n_2 such that they sum to 20, and calculating the SE for each possible combination, it will be possible to determine the optimal subsample sizes, which will yield the greatest precision (the combination giving smallest SE). This will yield valuable insight into a design: one can calculate the very best precision possible from a given design. This can in turn be used to select a design. The R-code in Appendix B or may be used to solve this problem. By checking the box "optimize marked subsample size," HatchPBT will find the value of n_1 that minimizes the SE of $\hat{p}$ given a fixed subsample size *n*. The following application demonstrates such a use of the program HatchPBT.

Note that alternatively, there may be no restrictions on the number of fish tested for PBT. In that case, to maximize precision, all of the N spawners sampled should be tested for a PBT. This simple cases is handled by the above equations by setting $n_1 = E(x_1)$ and $n_2 = E(x_2)$. This case represents that maximum amount of precision that may be attained when using a sample of size *N*.

APPLICATIONS

Design problem #1.—Assume that a carcass sample of size N=100 is drawn from the spawning grounds and 10% of the spawners are of hatchery origin. Further assume that there are two source hatcheries that contribute spawners to the spawning grounds, hatcheries A and B, where each contributes 5% of the total spawners. Also assume that the PBT fraction at both hatcheries is 0.95. There are two different scenarios we wish to test: (a) equal VM fractions of 0.5 ($\lambda_1 = \lambda_2 = 0.5$); (b) and unequal VM fractions of $\lambda_1 = 0.5$ and $\lambda_2 = 0.9$, at hatcheries A and B, respectively. In each of these cases, we assume that a subsample of $n = 50$ is drawn from the sample of 100. The goal is to determine how many of this subsample of 50 should be drawn from the group of VM spawners and how many should be drawn from the group of ~VM spawners in order to maximize precision; in other words, what are the optimal values of n_1 and n_2? Which of these two scenarios (a or b) gives the smallest CV($\hat{p}$) ?

To solve this design problem, the R-code for the theoretical formulas was used. The CV($\hat{p}$) associated with each feasible combination of n_1 and n_2 was plotted, and the combination of giving lowest CV($\hat{p}$) represented the optimal design. The CV($\hat{p}$) of the optimal design for scenario (a) was then compared to that of scenario (b). In solving this problem, the subsamples were constrained so that n_1 did not exceed $E(x_1)$ and that n_2 did not exceed $E(x_2)$.

The optimal designs for scenarios a and b are evident in Figure 2. Notice that when VM fractions are equal ($\lambda_1 = \lambda_2 = 0.5$), then the optimal design uses $n_1 = 0$ and $n_2 = 50$. When common VM fractions are used, precision is always maximized by making n_2 as large as possible. In the scenario where $\lambda_1 = 0.5$ and $\lambda_2 = 0.9$, the optimal design uses $n_1 = 3$ and $n_2 = 47$. Scenario b gives a smaller CV($\hat{p}$) than scenario a. Scenario (b) gives a minimum CV($\hat{p}$) of 0.3338 while scenario (a) gives a minimum CV($\hat{p}$) of 0.3535.

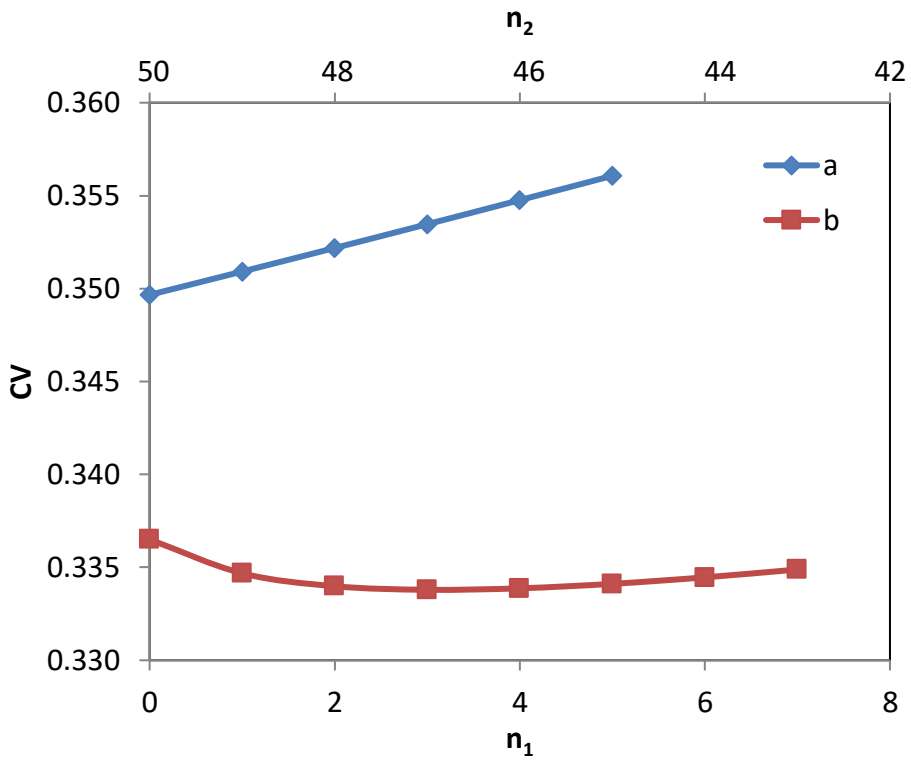
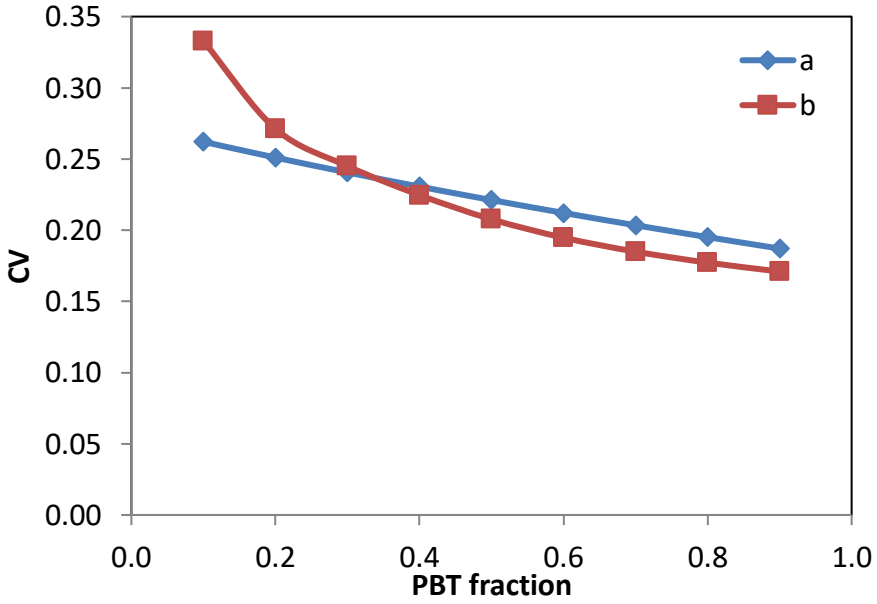


Figure 2. Plots of CV($\hat{p}$) for two different design scenarios (a) equal VM fractions and (b) unequal VM fractions.

Design problem #2.— In this problem, the benefit of having a high PBT fraction is demonstrated. Assume that a carcass sample of size N=40 is drawn from the spawning grounds and 50% of the spawners are of hatchery origin. Further assume that there are two source hatcheries that contribute spawners to the spawning grounds, hatcheries A and B, where each contributes 25% of the total spawners. Also assume that the PBT fraction at both hatcheries are identical and vary over the values 0, 0.1, 0.2,…, 1.0. As in the last design problem, there are two different scenarios we wish to test: (a) equal VM fractions of 0.5 ($\lambda_1 = \lambda_2 = 0.5$); and (b) unequal VM fractions of $\lambda_1 = 0.5$ and $\lambda_2 = 1.0$, at hatcheries A and B, respectively. In each of these cases, we assume that a subsample of $n = 20$ is drawn from the sample of 40. Assume that n_1 and n_2 are chosen so that CV($\hat{p}$) is as small as possible. Show how CV($\hat{p}$) under scenarios (a) and (b) varies with PBT fraction.

To determine the minimum CV($\hat{p}$) for a particular design with $n = 20$, CV($\hat{p}$) was calculated for each feasible pair of n_1 and n_2, and the smallest CV($\hat{p}$) over this set of pairs was chosen as the minimum value.[2] The function phos.pbt.estimates from Appendix B was used to calculate the CV($\hat{p}$) associated with the alternative designs. The smallest CV($\hat{p}$) over feasible values of n_1 and n_2 were then plotted against PBT fraction.

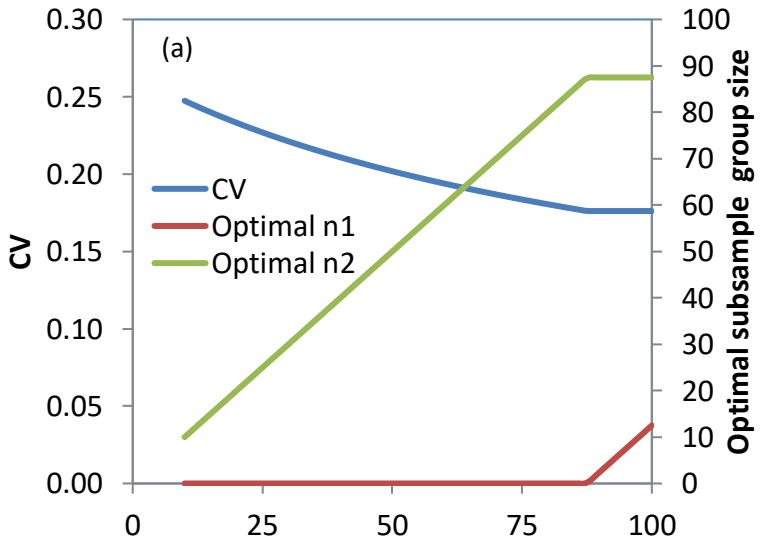
The results of this analysis are shown in Figure 3. The CV($\hat{p}$) declines with the PBT fraction for both the (a) equal VM fraction and (b) unequal VM fraction scenarios. Also, when PBT fraction is small, the CV of scenario (a) has lower CV than scenario (b) even though (a) has smaller VM fractions. This occurs because when VM fractions are the same, estimating *p* does not hinge on numbers of PBT detections in adults: a viable estimate is possible without knowing anything about the PBT detections (see equation (24)). However, when VM fractions differ, then estimating *p* does hinge on the PBT detections. Thus when VM fraction differ and expected PBT detections are low (due to small values of PBT fractions), the precision of $\hat{p}$ is expected to be poor. This was also observed in Hinrichsen et al. (2012).



[2] Subsamples were constrained so that n_1 did not exceed $E(x_1)$ and that n_2 did not exceed $E(x_2)$.

Figure 3. The CV($\hat{p}$) of scenarios (a) equal VM fractions ($\lambda_1 = \lambda_2 = 0.5$) and (b) unequal VM fractions ($\lambda_1 = 0.5$ and $\lambda_2 = 1.0$) plotted against the value of PBT fraction.

Design problem #3.— In this problem, the benefit of having a high PBT fraction is demonstrated. Assume that a carcass sample of size N=100 is drawn from the spawning grounds and 25% of the spawners are of hatchery origin. Further assume that there are two source hatcheries that contribute spawners to the spawning grounds, hatcheries A and B, where each contributes 12.5% of the total spawners. Also assume that the PBT fraction at both hatcheries is 0.95. As in the last design problem, there are two different scenarios we wish to test: (a) equal VM fractions of 0.5 ($\lambda_1 = \lambda_2 = 0.5$); (b) and unequal VM fractions of $\lambda_1 = 0.5$ and $\lambda_2 = 1.0$, at hatcheries A and B, respectively. In each of these cases, we assume that a subsample of $n = 1,2,\ldots 100$ is drawn from the total sample if size 100. Assume that n_1 and n_2 are chosen so that CV($\hat{p}$) is as small as possible. Show how CV($\hat{p}$) under scenarios (a) and (b) varies with the subsample size *n*. Show also how the optimal value of n_1 changes with *n*.



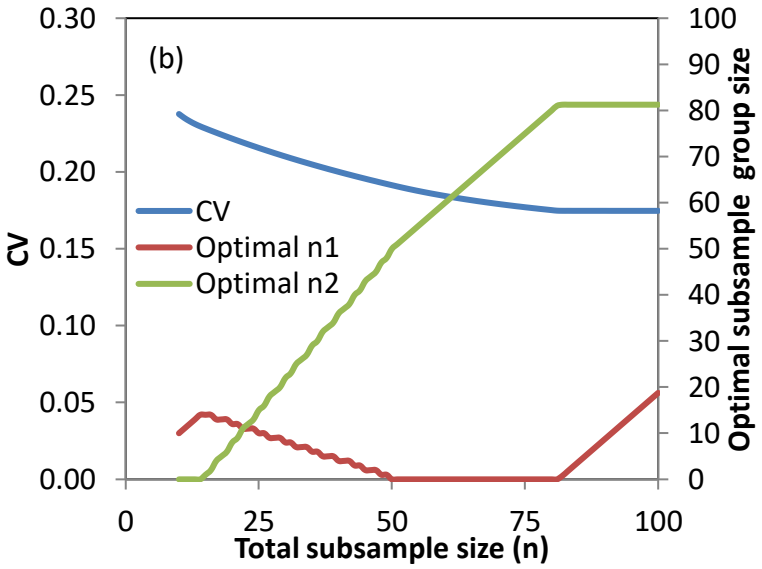


Figure 4. CV($\hat{p}$) plotted against the total sample size. The optimal n_1 and n_2 represent the values of n_1 and n_2 that minimize CV($\hat{p}$) subject to the constraints $n_1 + n_2 = n$, $n_1 \le E(x_1)$, and $n_2 \le E(x_2)$ using two scenarios: scenarios (a) equal VM fractions ($\lambda_1 = \lambda_2 = 0.5$) and (b) unequal VM fractions ($\lambda_1 = 0.5$ and $\lambda_2 = 1.0$).

The results of this analysis are shown in Figure 4. When VM fractions were equal (i.e., $\lambda_1 = \lambda_2 = 0.5$) it was always optimal to test more unmarked spawners for PBT than marked. In contrast, when VM fractions are unequal (i.e., $\lambda_1 = 0.5$ and $\lambda_2 = 1.0$), notice that for small total sample sizes (*n*), the optimal policy was to test only VM fish for PBT (Figure 4b). As the total sample size grows toward its maximum value of 100, the optimal policy was to send $n_1 = E(x_1) = 18.75$ and $n_2 = E(x_2) = 81.25$ for PBT testing. (In practice, these fractional numbers of fish should be rounded to the nearest whole number). Over the majority of the range of subsample sizes, it was optimal to test more unmarked spawners for PBT than marked spawners.

Design problem #4.—In this case, the benefit of visibly marking hatchery releases is demonstrated. Assume that 25% of the spawners are of hatchery origin and that there are two source hatcheries hatcheries A and B, and each contributes 12.5% of the total spawners. Also assume that equal VM fractions are used ($\lambda_1 = \lambda_2 = \lambda$) where λ ranges over the values 0.0, 0.1,…,1.0, all sampled carcasses are tested for PBT, and three alternative sample sizes N=25,

50, and 100. Also assume that the PBT fraction at both hatcheries takes on the alternative values of 0.50 and 0.95. Demonstrate how CV($\hat{p}$) changes with increasing VM fraction, λ . How does the relationship between CV($\hat{p}$) and λ change with sample size and PBT fraction?

This design problem is solved by plotting a graph of CV($\hat{p}$) versus λ using the alternative sample sizes N=25, 50, and 100 and alternative PBT fractions of 0.50 and 0.95 (Figure 5). Notice that when PBT fraction is small (0.50), the marking fraction has a large effect on CV($\hat{p}$), but when PBT is large (0.95), the marking fraction has little effect on CV($\hat{p}$). This shows that when PBT is high for all source hatcheries, and all sampled carcasses are tested for a PBT, then visibly marking releases does little to increase the precision of the estimate of *p*.

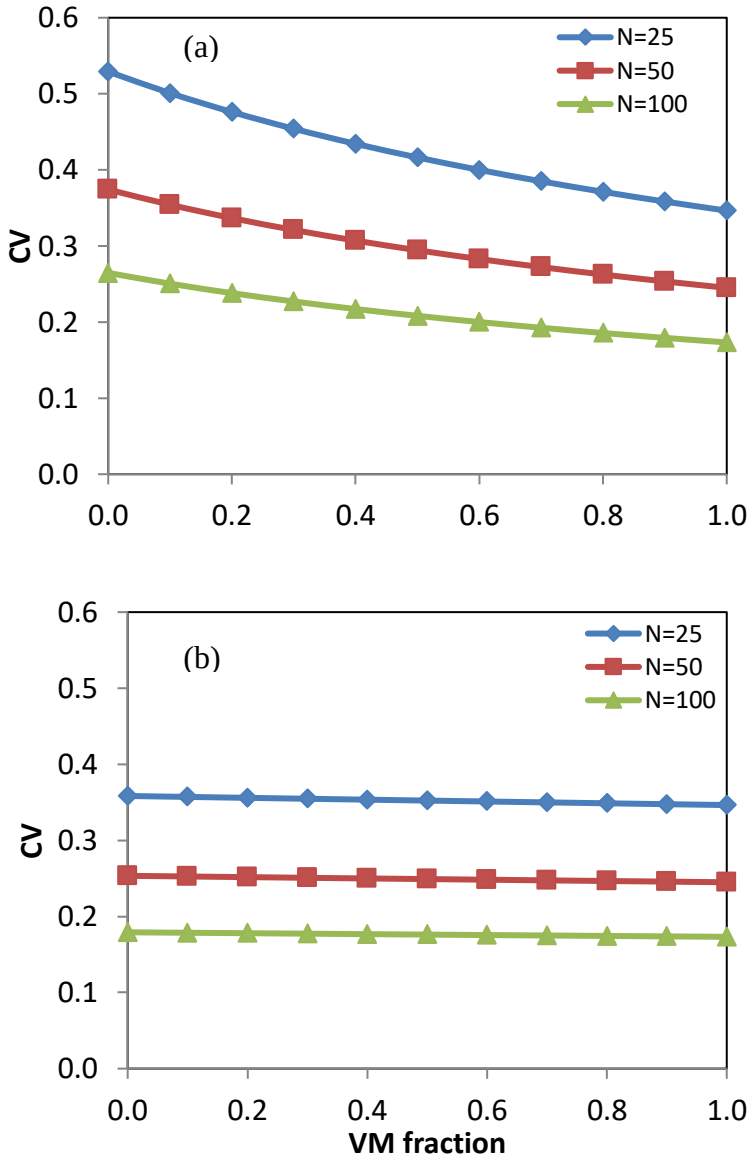


Figure 5. CV($\hat{p}$) plotted against VM fraction for three alternative sample sizes N=25, 50, and 100, when PBT fractions are (a) 0.50 and (b) 0.95. All sampled carcasses were tested for PBT.

The above results for design problem #4 were based on the assumption that 100% of the sampled carcasses were tested for PBT. How do the results change if 50% of the sampled carcasses are randomly selected and tested for PBT? This sensitivity analysis is illustrated in Figure 6. Notice that when 50% of the sampled carcasses are tested for PBT instead of 100% that the CV($\hat{p}$) is much more sensitive to the VM fraction. This demonstrates that when not all of the

sampled carcasses are tested for PBT, it is possible that increasing the VM marking fraction will greatly increase precision of $\hat{p}$, even when PBT fraction is high (0.95).

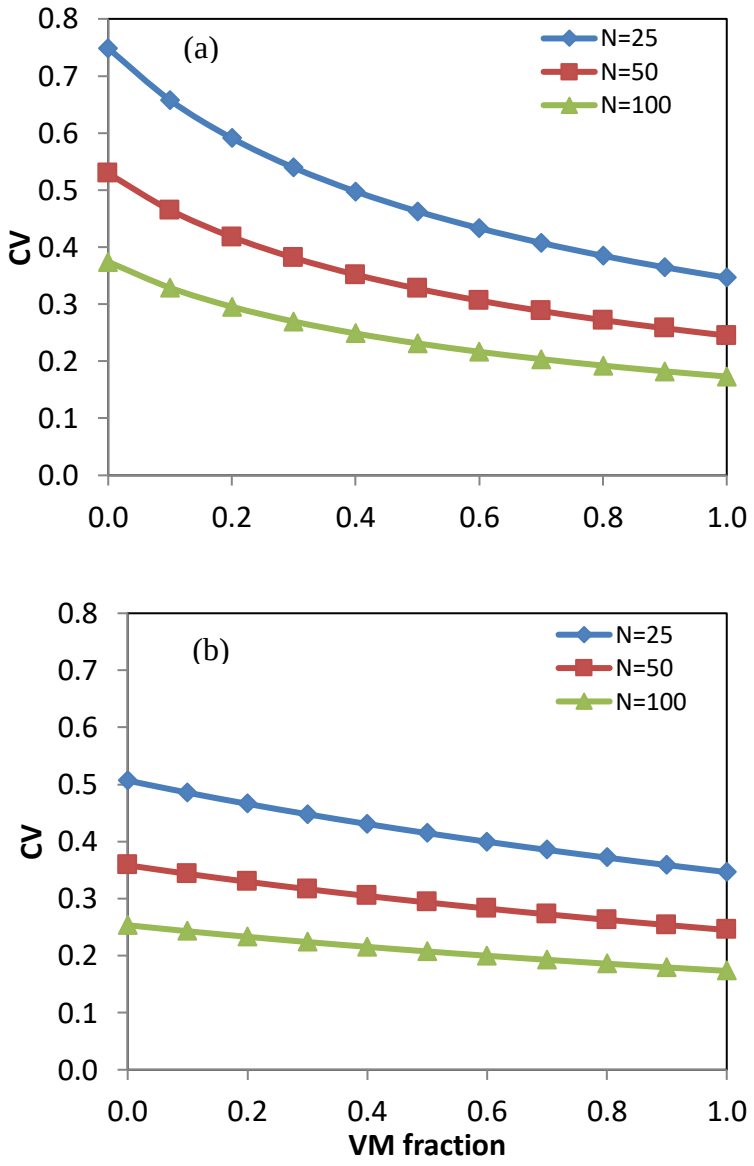


Figure 6. CV($\hat{p}$) plotted against VM fraction for three alternative sample sizes N=25, 50, and 100, when PBT fractions are (a) 0.50 and (b) 0.95. Half of the sampled carcasses were tested for PBT.

ACKNOWLEDGEMENTS

The author gratefully acknowledges the reviews of Mike Ackerman, Matt Campbell, Maureen Hess, Brian Maschhoff, Shawn Narum, Rishi Sharma, Barbara Shields, Craig Steele, and Bill Young. This work was supported by Bonneville Power Administration. Amber Parsons reviewed this documentation and tested the R-code.

REFERENCES

Anderson, E.C. 2010. Computational algorithms and user-friendly software Computational algorithms and user-friendly software for parentage-based tagging of Pacific salmonids. Final report submitted to the Pacific Salmon Commission's Chinook Technical Committee (US Section). 46 p. Available: http://swfsc.noaa.gov/uploadedFiles/Divisions/FED/Staff_Pages/Eric_Anderson/PBT_PSC_final_report.pdf.

Anderson, E. C., and J. C. Garza. 2005. A description of full genotyping. Report submitted to the Pacific Salmon Commission, Vancouver, British Columbia. 11p. Available: http://swfsc.noaa.gov/publications/FED/00675.pdf.

Anderson E. C., and J. C. Garza. 2006. The power of single-nucleotide polymorphisms for large-scale parentage inference. Genetics 172: 2567–2582.

Hankin, D.G. 1982. Estimating escapement of pacific salmon: marking practices to discriminate wild and hatchery fish. Transpactions of the American Fisheries Society 111:286-298.

HSRG (Hatchery Scientific Review Group). 2009. Columbia River hatchery reform systemwide report. Available: www.hatcheryreform.us.

Hinrichsen, R. A. 2003. The power of experiments for estimating the relative reproductive success of hatchery-origin spawners. Canadian Journal of Fisheries and Aquatic Sciences 60:864–872.

Hinrichsen, R.A., R. Sharma, T.R. Fisher. 2012. Precision and accuracy of estimators of the proportion of hatchery-origin spawners. Transactions of the American Fisheries Society 142:437-454.

Hinrichsen, R.A., Steele, C.A., Ackerman, M.W., Campbell, M.R., Narum, S.R., Hess, M.A., Young, W.P., Shields, B.A. and Maschhoff, B.L. 2016. Maximum likelihood estimation of the proportion of hatchery-origin fish on spawning grounds using coded wire tagging

and parentage-based tagging. Transactions of the American Fisheries Society, 145(3): 671–686.

Jennrich, R. I., and P. F. Sampson. 1976. Newton-Raphson and related algorithms for maximum likelihood variance component estimation. Technometrics 18:11-17.

Kariya, T. and H. Kurata. 2004. Generalized Least Squares. Wiley Series in Probability and Statistics. Wiley. New York, New York.

McClure, M.M., E.E. Holmes, B.L. Sanderson, and C.E. Jordan. 2003. A large-scale multispecial status assessment: anadromous salmonids in the Columbia River basin. Ecological Applications 13:964-989.

Mobrand, L. E., J. Barr, L. Blankenship, D. D. Campton, T. T. P. Eveleyn, T.A. Flagg, C. V. W. Mahnken, L. W. Seeb, P. R. Seidel, and W. W. Smoker. 2005. Hatchery reform in Washington State: principles and emerging issues. Fisheries 30(6):11–23.

Mood, A. M., F. A. Graybill, and D. C. Boes. 1974. Introduction to the theory of statistics, 3rd edition. McGraw-Hill, New York.

Press, W. H., S.A., Teukolsky, W.T. Vetterling, and B.P. Flannery, Brian P. 1992. Numerical Recipesin Fortran: The Art of Scientific Computing (2rd ed.). New York: Cambridge University Press.

Steele, C., M. Ackerman, J. McCane, M. Campbell, M. Hess, N. Campbell, and S. Narum. 2011. Parentage-based tagging of Snake River hatchery steelhead and Chinook salmon. 2010 Annual Report. Idaho Department of Fish and Game and Columbia River Inter-Tribal Fish Commision. IDFG Report Number 11-111. June 2011. Available: https://research.idfg.idaho.gov/Fisheries%20Research%20Reports/Res11-111Steele2010%20Parentage%20Based%20Tagging%20Snake%20River%20Steelhead%20Salmon.pdf

Waples, R.S. 1991. Genetic interactions between hatchery and wild salmonids: lessons from the Pacific Northwest. Canadian Journal of Fisheries and Aquatic Sciences. 48(Suppl. 1): 124-133.

APPENDIX A. NAMES OF VARIABLES

Table A.1.—Variables.

R code	Mathematical derivation	Definition
nhatch	m	Number of source hatcheries.
phos	p	Proportion of hatchery-origin spawners. The MLE of p is denoted by $\hat{p}$.
phosi[i]	p_i	Hatchery-specific proportion of hatchery-origin spawners for hatchery i. The MLE of p_i is denoted by $\hat{p}_i$.
ppbt[i]	ϕ_i	PBT fraction for hatchery i.
lambda[i]	λ_i	VM fraction for hatchery i.
Nsamp	N	Total sample size.
x1	x_1	Number of VM spawners in total sample
x2	x_2	Number of unmarked spawners in total sample
y[i]	y_i	Number of subsampled VM spawners testing positive for PBT from hatchery i.
z[i]	z_i	Number of the subsampled unmarked spawners testing positive for PBT from hatchery i.
n	n	Subsample size.
n1	n_1	Number of VM spawners tested for PBT.
n2	n_2	Number of unmarked spawners tested for PBT.
Ex1	$E(x_1)$	Expected value of x_1.
Ex2	$E(x_2)$	Expected value of x_2.
I	$\mathbf{I}$	Fisher Information Matrix.

SE.phos	$SE(\hat{p})$	Standard error of the estimate of the proportion of hatchery-origin spawners. Note $SE^*(\hat{p})$ denotes a Monte Carlo estimate.
CV.phos	$CV(\hat{p})$	Coefficient of variation of the estimate of the proportion of hatchery-origin spawners. Note $CV^*(\hat{p})$ denotes a Monte Carlo estimate.
SE_MIN.phos	$SE_{\min}(\hat{p})$	Minimum standard error of the estimate of the proportion of hatchery-origin spawners, achieved with $n_1 = E(x_1)$ and $n_2 = E(x_2)$, where $n = N$. Note $SE^*_{\min}(\hat{p})$ denotes a Monte Carlo estimate.
CV_MIN.phos	$CV_{\min}(\hat{p})$	Minimum coefficient of variation of the estimate of the proportion of hatchery-origin spawners achieved with $n_1 = E(x_1)$ and $n_2 = E(x_2)$, where $n = N$. Note $CV^*_{\min}(\hat{p})$ denotes a Monte Carlo estimate.
BIAS.phoshat	$bias^*(\hat{p})$	Monte Carlo relative bias which is bias divided by the true value of p.

APPENDIX B. R-CODE

Table B.1. R-code

```
#Program to calculate statistical properties of phos estimates using maximum likelihood theory
#and Monte Carlo Simulation. This code allows inputs from several hatcheries with
#potentially different visual marking (VM) fractions and different parentage-based tagging (PBT) fractions.
#FILE: pbt-web-7-10-2013.s
#AUTHOR: Richard A. Hinrichsen, 10 July 2013

#Variables and parameters used in the analysis
#inputs
#phosi = true proportions of hatchery origin spawners (hatchery-specific)
#Nsamp = total number of spawners sampled on spawning grounds
#n = total number of spawners tested for PBT
#n1 = number of visually marked spawners tested for PBT (when OPT=FALSE)
#lambda = marking fraction (hatchery-specific)
#ppbt = fraction of fish that are PBT (hatchery-specific)
#OPT = FALSE when n1 is user input, TRUE when program to select an optimal value of n1
#MONTE = FALSE for theoretical results, TRUE for Monte Carlo results
#NSIM = number of Monte Carlo simulations (needed if MONTE=TRUE)
#
#
#Select intermediate variables
#nhatch = number of hatcheries supplying spawners to spawning grounds
#I = Fisher Information Matrix
#x1 = number of visually marked spawners in sample of size Nsamp
#x2 = number of unmarked spawners in sample of size Nsamp
#n2 = number of unmarked spawners tested for PBT
#y = number of visually marked spawners tested that were PBT (hatchery-specific)
#z = number of unmarked spawners tested that were PBT (hatchery-specific)
#Ey = expected value of y
#Ez = expected value of z

#Results
#phos = true proportion of hatchery-origin spawners
#Ex1 = expected number of visually marked spawners (summing over hatcheries)
#Ex2 = expected number of not visually marked spawners (summing over hatcheries)
```

```
#SE_MIN.phos = standard error (SE) when all sampled fish are tested for PBT
#CV_MIN.phos = Coefficient of variation when ALL sampled fish are tested for PBT
#SE.phos = standard error (SE)
#CV.phos = Coefficient of variation
#BIAS.phos = relative bias estimate (NA if MONTE=FALSE)
#n1 = optimal number of visually marked spawners tested for PBT (when OPT=TRUE)

#top level function
phos.pbt.main<-function(phosi=.1*c(1/20,1/20,9/20,9/20),
                    Nsamp=1000,n=200,n1=50,
                    lambda=c(1,.95,.5,.5),
                    ppbt=c(.95,.95,.95,.95),
                    OPT=FALSE,
                    MONTE=FALSE,NSIM=1000){
      check.inputs(phosi,Nsamp,n,n1,lambda,ppbt,OPT,MONTE,NSIM)
      if(!OPT){
      if(!MONTE){res<-
phos.pbt.estimates(phosi=phosi,Nsamp=Nsamp,n=n,n1=n1,lambda=lambda,ppbt=ppbt,suppress=FALSE)}
      if(MONTE){res<-
phos.pbt.estimates2(NSIM=NSIM,phosi=phosi,Nsamp=Nsamp,n=n,n1=n1,lambda=lambda,ppbt=ppbt)}}
      if(OPT){res<-optimize(phosi=phosi,Nsamp=Nsamp,n=n,lambda=lambda,ppbt=ppbt)}
      final.res<-list(OPT=OPT,
            MONTE=res$MONTE,
            NSIM=res$NSIM,
            phosi=res$phosi,
            Nsamp=res$Nsamp,
            n=res$n,
            n1=res$n1,
            lambda=res$lambda,
            ppbt=res$ppbt,
            phos=res$phos,
            Ex1=res$Ex1,
            Ex2=res$Ex2,
            SE_MIN.phos=res$SE_MIN.phos,
            CV_MIN.phos=res$CV_MIN.phos,
            SE.phos=res$SE.phos,
            CV.phos=res$CV.phos,
```

```
            BIAS.phos=res$BIAS.phos)
    return(final.res)
}

#check of feasibility of optimization
#avoid unusual case where hatcheries with zero expected tags recoveries
#and these hatcheries do not all use the same visible marking fraction
is.feas<-function(phosi,Nsamp,n,n1,lambda,ppbt){
 n2<-n-n1
 onelambda2<-TRUE
 iii<-n1*lambda*ppbt+n2*(1-lambda)*ppbt==0
 onelambda2<-sum(mean(lambda[iii])==lambda[iii])==sum(iii)
 onelambda2<-onelambda2&(mean(lambda[iii])>0)
 if((!onelambda2)&sum(iii))return(FALSE)
 return(TRUE)
}

#Find the value of n1 that minimizes CV
optimize<-function(phosi,Nsamp,n,lambda,ppbt){
 #loop over all feasible values of n1
 #note that n1 might be a non-integer because the constraints are not necessarily integers
 Ex1<-Nsamp*sum(lambda*phosi)
 min.n1<-max(Ex1-(Nsamp-n),0)
 max.n1<-min(Ex1,n)
 min.int<-ceiling(min.n1)
 max.int<-floor(max.n1)
 if(min.int>=max.int){n1s<-min.n1}
 if(min.int<max.int){n1s<-c(min.n1,min.int:max.int,max.n1)}
 n1s<-unique(n1s)
 nfeas<-length(n1s)
 cv<-1.e10
 icount<-0
 for(ii in 1:nfeas){
  if(is.feas(phosi=phosi,Nsamp=Nsamp,n=n,n1=n1s[ii],lambda=lambda,ppbt=ppbt)){
  icount<-icount+1
  res.new<-
phos.pbt.estimates(phosi=phosi,Nsamp=Nsamp,n=n,n1=n1s[ii],lambda=lambda,ppbt=ppbt,suppress=TRUE)
```

```
 if(res.new$CV.phos<cv){res<-res.new;n1<-res.new$n1;cv<-res.new$CV.phos}}
 }
 if(icount==0)stop("Error in optimize: no feasible values are available for n1")
 return(res)
}

#make sure the inputs make sense
check.inputs<-function(phosi,Nsamp,n,n1,lambda,ppbt,OPT,MONTE,NSIM){
 if(!is.logical(OPT))stop("OPT must be TRUE or FALSE")
 if(!is.logical(MONTE))stop("MONTE must be TRUE or FALSE")
 if(OPT){
 if(MONTE){stop("For the optimization option, the program cannot be run in Monte Carlo mode")}}
 if(MONTE){
 if(floor(NSIM)!=NSIM){stop("NSIM must be a positive integer")}
 if(NSIM<=0){stop("NSIM must be a positive integer")}}
 if(floor(Nsamp)!=Nsamp){stop("Nsamp must be a positive integer")}
 if(Nsamp<=0){stop("Nsamp must be a positive integer")}
#check dimension of inputs
 k1<-length(phosi);k2<-length(lambda);k3<-length(ppbt)
 mytest<-abs(k1-k2)+abs(k2-k3)
 if(mytest>0) stop("dimensions of phosi, lambda, and ppbt must match")
#check constraints on ppbt, phosi, and lambda
 if(sum(ppbt<0))stop("ppbts must all be greater than or equal to zero")
#check that each ppbt is less than or equal to one
 if(sum(ppbt>1))stop("ppbts must all be less than or equal to 1.0")
#check that all lambdas are between zero and 1.0
 if(sum(lambda<0))stop("lambdas must all be greater than or equal to zero")
 if(sum(lambda>1))stop("lambdas must all be less than or equal to one")
#check that all phosi are between zero and 1.0
 if(sum(phosi<=0))stop("phosis must all be greater than zero")
 if(sum(phosi>1))stop("phosis must all be less than or equal to one")
#check that the subsample size is less than sample size
 if(n>Nsamp)stop(paste("n must be less than or equal to Nsamp=",Nsamp))
#get expected values of observations
 Ex1<-Nsamp*sum(lambda*phosi)
 Ex2<-Nsamp-Ex1
#check subsample. Note in the case of OPT==TRUE, n1 might be changed
```

```
 if(!OPT){
 if(n1<0)stop("n1 must be nonnegative")
 if(n<n1)stop("n must be greater than or equal to n1")
 if(n1>Ex1)stop(paste("n1 must not exceed the expected number of VM spawners in sample=",Ex1))
 if(n1<(n-Ex2))stop(paste("n1 must be >= n minus the expected number of ~VM spawners in sample=",n-Ex2))}
 n2<-n-n1
 if(!OPT){
 iii<-n1*lambda*ppbt+n2*(1-lambda)*ppbt==0}
#note in the case of OPT that n1 might be changed to allow estimation
if(OPT){
 iii<-(lambda*ppbt+(1-lambda)*ppbt==0)|(n==0)}
 onelambda2<-sum(mean(lambda[iii])==lambda[iii])==sum(iii)
 onelambda2<-onelambda2&(mean(lambda[iii])>0)
 if((!onelambda2)&sum(iii)){
  stop("Expected tag recoveries must not be zero when marking fractions differ or are zero")}

 return(NULL)
}

#Theoretical results using maximum likelihood theory
phos.pbt.estimates<-function(phosi=.1*c(1/20,1/20,9/20,9/20),
Nsamp=1000,n=200,n1=50,lambda=c(1,.95,.5,.5),ppbt=c(.95,.95,.95,.95),
suppress){
 n2<-n-n1
 nhatch<-length(lambda)
#get expected values of observations
 Ex1<-Nsamp*sum(lambda*phosi)
 Ex2<-Nsamp-Ex1
 Ey<-n1*lambda*ppbt*phosi/sum(lambda*phosi)
 Ez<-n2*(1-lambda)*ppbt*phosi/(1-sum(lambda*phosi))
 Ey2<-Ex1*lambda*ppbt*phosi/sum(lambda*phosi)
 Ez2<-Ex2*(1-lambda)*ppbt*phosi/(1-sum(lambda*phosi))
 if(sum(lambda)==0){
  Ey<-0
  Ez<-n2*ppbt*phosi
  Ey2<-0
  Ez2<-Ex2*ppbt*phosi
```

```
 }
 phos<-sum(phosi)
#get estimates using pbte routines
 res<-phos.pbte.estimates(x1=Ex1,x2=Ex2,n1=n1,n2=n2,y=Ey,z=Ez,lambda,ppbt,suppress=suppress)
 res2<-phos.pbte.estimates(x1=Ex1,x2=Ex2,n1=Ex1,n2=Ex2,y=Ey2,z=Ez2,lambda,ppbt,suppress=suppress)
 myres<-list(MONTE=FALSE,
              NSIM=NA,
              phosi=phosi,
              Nsamp=Nsamp,
              n=n,
              n1=n1,
              lambda=lambda,
              ppbt=ppbt,
              phos=phos,
              Ex1=Ex1,
              Ex2=Ex2,
              SE_MIN.phos=res2$SE.phos,
              CV_MIN.phos=res2$CV.phos,
              SE.phos=res$SE.phos,
              CV.phos=res$CV.phos,
              BIAS.phos=NA)
 return(myres)
}

#Monte Carlo estimates of standard error and relative bias
phos.pbt.estimates2<-function(NSIM=1000,phosi=.1*c(1/20,1/20,9/20,9/20),
 Nsamp=1000,n=n,n1=50,lambda=c(1,.95,.5,.5),ppbt=c(.95,.95,.95,.95)){
 n2<-n-n1
 nhatch<-length(phosi)
#get expected values of observations
 Ex1<-Nsamp*sum(lambda*phosi)
 Ex2<-Nsamp-Ex1
 Ey<-n1*lambda*ppbt*phosi/sum(lambda*phosi)
 Ez<-n2*(1-lambda)*ppbt*phosi/(1-sum(lambda*phosi))
 Ey2<-Ex1*lambda*ppbt*phosi/sum(lambda*phosi)
 Ez2<-Ex2*(1-lambda)*ppbt*phosi/(1-sum(lambda*phosi))
 if(sum(lambda)==0){
```

```
 Ey<-0
 Ez<-n2*ppbt*phosi
 Ey2<-0
 Ez2<-Ex2*ppbt*phosi
}
phos<-sum(phosi)
res<-phos.pbte.estimates2(NBOOT=NSIM,x1=Ex1,x2=Ex2,n1=n1,n2=n2,y=Ey,z=Ez,lambda,ppbt)
res2<-phos.pbte.estimates2(NBOOT=NSIM,x1=Ex1,x2=Ex2,n1=Ex1,n2=Ex2,y=Ey2,z=Ez2,lambda,ppbt)

myres<-list(MONTE=TRUE,
            NSIM=NSIM,
            phosi=phosi,
            Nsamp=Nsamp,
            n=n,
            n1=n1,
            lambda=lambda,
            ppbt=ppbt,
            phos=phos,
            Ex1=Ex1,
            Ex2=Ex2,
            SE_MIN.phos=res2$SE.phos,
            CV_MIN.phos=res2$CV.phos,
            SE.phos=res$SE.phos,
            CV.phos=res$CV.phos,
            BIAS.phos=res$BIAS.phos)
 return(myres)
}

#Maximum Likelihood Theory results
phos.pbte.estimates<-function(x1,x2,n1,n2,y,z,lambda,ppbt,suppress){
 nhatch<-length(lambda)
 Nsamp<-x1+x2

#An important case for combining cells occurs when the expected w=x+y=0
#in this case, constant lambdas over these cells saves the estimation.
#note that this also takes care of the case with a single lambda for all hatcheries
iii<-n1*lambda*ppbt+n2*(1-lambda)*ppbt==0
```

```
onelambda2<-sum(lambda[iii]==mean(lambda[iii]))==sum(iii)
if((sum(iii)>1)&onelambda2){
 if(!suppress)warning("collapsing cells with expected tag recoveries of zero into single cell since lambda is
constant")
 lambda1.new<-mean(lambda[iii])
 ppbt1.new<-0.0
 lambda.new<-c(lambda1.new,lambda[!iii])
 ppbt.new<-c(ppbt1.new,ppbt[!iii])
 y.new<-c(0,y[!iii])
 z.new<-c(0,z[!iii])
 nhatch.new<-length(lambda.new)
 res<-phos.pbte.estimates(x1=x1,x2=x2,n1=n1,n2=n2,y=y.new,z=z.new,lambda=lambda.new,ppbt=ppbt.new,
suppress=suppress)
 phosi<-rep(NA,nhatch)
 SE.phosi<-rep(NA,nhatch)
 phosi[!iii]<-res$phosi[2:nhatch.new]
 SE.phosi[!iii]<-res$SE.phosi[2:nhatch.new]

 myres<-list(BOOT=FALSE,
             NBOOT=NA,
             Nsamp=Nsamp,
             x1=x1,
             x2=x2,
             n=n1+n2,
             n1=n1,
             n2=n2,
             y=y,
             z=z,
             lambda=lambda,
             ppbt=ppbt,
             phosi=phosi,
             SE.phosi=SE.phosi,
             phos=res$phos,
             SE.phos=res$SE.phos,
             CV.phos=res$CV.phos,
             BIAS.phos=NA)
 return(myres)
```

```
}


#get initial estimate of phosi
phosi.init<-init(x1,x2,n1,n2,y,z,lambda,ppbt,suppress=suppress)

myres<-get.estimates2(phosi.init,x1,x2,n1,n2,y,z,lambda,ppbt,suppress=suppress)
phos<-myres$phos
phos.var<-myres$phos.var
phosi<-myres$phosi
SE.phosi<-sqrt(myres$phosi.var)


SE.phos<-sqrt(phos.var)
CV.phos<-SE.phos/phos
SE.phos<-as.numeric(SE.phos)
CV.phos<-as.numeric(CV.phos)

myres<-list(BOOT=FALSE,
            NBOOT=NA,
            Nsamp=Nsamp,
            x1=x1,
            x2=x2,
            n=n1+n2,
            n1=n1,
            n2=n2,
            y=y,
            z=z,
            lambda=lambda,
            ppbt=ppbt,
            phosi=phosi,
            SE.phosi=SE.phosi,
            phos=phos,
            SE.phos=SE.phos,
            CV.phos=CV.phos,
            BIAS.phos=NA)
 return(myres)
```

```
}

#Fisher Information Matrix (general case)
getI<-function(Nsamp,n1,n2,lambda,ppbt,phosi){
 Ex1<-Nsamp*sum(lambda*phosi)
 Ex2<-Nsamp-Ex1
 theta1<-n1/Ex1
 theta2<-n2/Ex2

 v<-lambda
 I<- v%*%t(v)*Nsamp*(1-theta1)/sum(v*phosi)
 I<-I +v%*%t(v)*Nsamp*(1-theta2)/(1-sum(v*phosi))

 v<-(1-ppbt)*lambda
 I<-I+v%*%t(v)*Nsamp*theta1/sum(v*phosi)

 v<-lambda*(1-ppbt)+ppbt
 I<-I+v%*%t(v)*Nsamp*theta2/(1-sum(v*phosi))

#fix diagonal
 mydiag<-diag(I)+Nsamp*ppbt*(theta1*lambda+theta2*(1-lambda))/phosi
 diag(I)<-mydiag
 return(I)
}

#Fisher Information matrix used when all VM releases PBT
getI2<-function(Nsamp,n1,n2,lambda,ppbt,phosi){
 Ex1<-Nsamp*sum(lambda*phosi)
 Ex2<-Nsamp-Ex1
 theta1<-n1/Ex1
 theta2<-n2/Ex2

 v<-lambda
 I<- v%*%t(v)*Nsamp*(1-theta1)/sum(v*phosi)
 I<-I +v%*%t(v)*Nsamp*(1-theta2)/(1-sum(v*phosi))

 v<-ppbt
```

```
 I<-I+v%*%t(v)*Nsamp*theta2/(1-sum(v*phosi))

#fix diagonal
 mydiag<-diag(I)+Nsamp*ppbt*(theta1*lambda+theta2*(1-lambda))/phosi
 diag(I)<-mydiag
 return(I)
}

#Fisher Information matrix (used when all lambdas are zero)
getI3<-function(Nsamp,n1,n2,lambda,ppbt,phosi){
 Ex2<-Nsamp
 theta2<-n2/Ex2
 v<-ppbt
 I<-v%*%t(v)*Nsamp*theta2/(1-sum(v*phosi))
#fix diagonal
 mydiag<-diag(I)+Nsamp*ppbt*theta2/phosi
 diag(I)<-mydiag
 return(I)
}

#Bootstrap estimates of standard error and bias
#consider cases where some of the phosi are missing
phos.pbte.estimates2<-function(NBOOT,x1,x2,n1,n2,y,z,lambda,ppbt){
 nhatch<-length(lambda)
 Nsamp<-x1+x2
#get MLE using theoretical results
 res<-phos.pbte.estimates(x1,x2,n1,n2,y,z,lambda,ppbt,suppress=FALSE)
 phosi<-res$phosi
 phosi.orig<-res$phosi
 phos<-res$phos
 phos.sim<-rep(NA,NBOOT)
 phosi.sim<-matrix(NA,nrow=NBOOT,ncol=nhatch)
 if(!is.na(phos)){
 for(ii in 1:NBOOT){
  iii<-is.na(phosi)
  phosi[iii]<-rep(phos-sum(phosi,na.rm=T),sum(iii))/sum(iii)
  mysim<-pbtsim1(phosi,Nsamp,n1,n2,res$lambda,res$ppbt)
```

```
 my.n1<-n1
 my.n2<-n2
 if(mysim$x1<n1)my.n1<-mysim$x1
 if(mysim$x2<n2)my.n2<-mysim$x2
 res<-phos.pbte.estimates(x1=mysim$x1,x2=mysim$x2,n1=my.n1,n2=my.n2,
          y=mysim$y,z=mysim$z,lambda=lambda,ppbt=ppbt,suppress=TRUE)
 phos.sim[ii]<-res$phos
 phosi.sim[ii,]<-res$phosi

}}

SE.phosi<-apply(phosi.sim,c(2),var,na.rm=T)
SE.phosi<-sqrt(SE.phosi)
SE.phos<-sqrt(var(phos.sim,na.rm=T))
CV.phos<-SE.phos/phos
mymean<-mean(phos.sim,na.rm=T)
BIAS.phos<-(mymean-phos)/phos

myres<-list(BOOT=TRUE,
          NBOOT=NBOOT,
          Nsamp=Nsamp,
          x1=x1,
          x2=x2,
          n=n1+n2,
          n1=n1,
          n2=n2,
          y=y,
          z=z,
          lambda=lambda,
          ppbt=ppbt,
          phosi=phosi.orig,
          SE.phosi=SE.phosi,
          phos=phos,
          SE.phos=SE.phos,
          CV.phos=CV.phos,
          BIAS.phos=BIAS.phos)
return(myres)
```

```
}

#simulate data when phosi available
pbtsim1<-function(phosi,Nsamp,n1,n2,lambda,ppbt){
 m<-length(phosi)
#first get binomial sample of fish marked and unmarked
 P<-sum(phosi*lambda)
 x1<-rbinom(n=1,size=Nsamp,prob=P)
 x2<-Nsamp-x1
#next use multinomial distribution to simulate
#how many fish are pbt and how many are not pbt
 py<-ppbt*phosi*lambda/P
 pz<-ppbt*phosi*(1-lambda)/(1-P)
 if(P>0){y<-rmultinom(n=1,size=min(n1,x1),prob=c(py,1-sum(py)))}
 if(P==0){y<-matrix(0,ncol=1,nrow=length(phosi))}
 if(P<1){z<-rmultinom(n=1,size=min(n2,x2),prob=c(pz,1-sum(pz)))}
 if(P==1){z<-matrix(0,ncol=1,nrow=length(phosi))}

 return(list(Nsamp=Nsamp,x1=x1,x2=x2,n1=n1,n2=n2,y=y[1:m,1],z=z[1:m,1]))
}

#Use R.A. Fisher's scoring algorithm to estimate phosi
#phosi represents the intial guess on input
get.estimates2<-function(phosi.init,x1,x2,n1,n2,y,z,lambda,ppbt,suppress){
 Nsamp<-x1+x2
 NTRIAL<-100
 tolx<-1.e-5
 nhatch<-length(phosi.init)
 nhatch.orig<-nhatch
 w<-y+z
 nas<-rep(NA,nhatch)
 phosi<-nas
 phosi.var<-nas
 phos<-NA
 phos.var<-NA

#in a rare case init can return zeroes even though the true estimate is not zero
```

```
#which would defeat Fisher's scoring method
jjj<-phosi.init==0
phosi.init[jjj]<-phosi.init[jjj]+.00001
#find zeroes that occur when lambda=0 and n1*lambda*ppbt+n2*(1-lambda)*ppbt>0
jjj<-(n1*lambda*ppbt+n2*(1-lambda)*ppbt>0)&(lambda==0)&(w==0)
if(sum(jjj)>=1){
 y<-y[!jjj]
 z<-z[!jjj]
 ppbt<-ppbt[!jjj]
 lambda<-lambda[!jjj]
 phosi.init<-phosi.init[!jjj]
 nhatch<-length(y)}

#check for special cases
#check to see if all VM releases PBT
pbttest<-sum((ppbt-1)*lambda)==0
#check to see if all lambdas are zero (special case)
lambdatest<-sum(lambda==0)==nhatch

#get the right Fisher Information function
 if(lambdatest){
  my.getI<-getI3
  dlike<-dlike3
 }
 else{

 if(pbttest){
  my.getI<-getI2
  dlike<-dlike2
 }
else{
  my.getI<-getI
  dlike<-dlike1
 }
 }

#use R.A. Fisher's scoring algorithm to find where the partial derivatives of the
```

```
#log-likelihood are zero. Use Fisher Information matrix in Newton's method to approx -Hessian
 phosi<-phosi.init
 errf<-0.0
 alpha<-0.9
 for(ii in 1:NTRIAL){
  I<-my.getI(Nsamp,n1,n2,lambda,ppbt,phosi)
  df<-dlike(phosi,x1,x2,n1,n2,y,z,lambda,ppbt)
  size<-prod(dim(I))
  if(size==0){
    if(!suppress)warning("dimension of I is 0 x 0")
    return(list(phosi=nas,phosi.var=nas,phos=NA,phos.var=NA))}
  if(is.na(rcond(I))){
    if(!suppress)warning("condition number of I is NA")
    return(list(phosi=nas,phosi.var=nas,phos=NA,phos.var=NA))}
  if(rcond(I)<1.e-15){
    if(!suppress)warning("computationally singular information matrix")
    return(list(phosi=nas,phosi.var=nas,phos=NA,phos.var=NA))}
  delx<-solve(I)%*%df
  phosi<-phosi+delx*(1-alpha)
  phosi<-abs(phosi)
  errx<-sum(abs(delx))/sum(abs(phosi))
  alpha<-alpha*alpha
  if(errx<=tolx)break
 }
 if(ii==NTRIAL){
  if(!suppress)warning("maximum number of iterations was reached")
  return(list(phosi=nas,phosi.var=nas,phos=NA,phos.var=NA))}
 phos<-sum(phosi)
 e<-rep(1,nhatch)
 myvar<-solve(I)
 phos.var<-t(e)%*%myvar%*%e
 phosi.var<-diag(myvar)
 full.phosi.var<-rep(0,nhatch.orig)
 full.phosi<-rep(0,nhatch.orig)
#recall that jjj represents hatcheries with estimates of phosi=0
#reduction occurs only when sum(jjj>=1)
 if(sum(jjj)>=1){
```

```
 full.phosi.var[!jjj]<-phosi.var
 full.phosi[!jjj]<-phosi
 }
 if(sum(jjj)<1){
 full.phosi.var<-phosi.var
 full.phosi<-phosi

 }
 return(list(phosi=full.phosi,phosi.var=full.phosi.var,phos=phos,phos.var=phos.var))
}

#get gradient of the log likelihood function
#phosi is the current best estimate of phosi
#used in most general case
dlike1<-function(phosi,x1,x2,n1,n2,y,z,lambda,ppbt){
#estimate phosi
 Nsamp<-x1+x2
 sum1<-sum(lambda*phosi)
 sum2<-sum((1-ppbt)*lambda*phosi)
 sum3<-sum(phosi*ppbt)
 res<-lambda*(x1-n1)/sum1-lambda*(Nsamp-x1-n2)/(1-sum1)+y/phosi
 res<-res+(n1-sum(y))*(1-ppbt)*lambda/sum2+z/phosi
 res<-res-(n2-sum(z))*(lambda*(1-ppbt)+ppbt)/(1-sum2-sum3)
 return(res)
}

#get gradient of the log likelihood function
#phosi is the current best estimate of phosi
#used in the case where sum(lambda*(1-ppbt))===0
dlike2<-function(phosi,x1,x2,n1,n2,y,z,lambda,ppbt){
#estimate phosi
 Nsamp<-x1+x2
 sum1<-sum(lambda*phosi)
 sum3<-sum(phosi*ppbt)
 res<-lambda*(x1-n1)/sum1-lambda*(Nsamp-x1-n2)/(1-sum1)+y/phosi
 res<-res+z/phosi
 res<-res-(n2-sum(z))*ppbt/(1-sum3)
```

```
 return(res)
}

#get gradient of the log likelihood function
#phosi is the current best estimate of phosi
#used when lambda=0 at all hatcheries
dlike3<-function(phosi,x1,x2,n1,n2,y,z,lambda,ppbt){
#estimate phosi
 Nsamp<-x1+x2
 sum3<-sum(phosi*ppbt)
 res<-z/phosi
 res<-res-(n2-sum(z))*ppbt/(1-sum3)
 return(res)
}

#get initial estimates of phosi
#by equating x1,y,and z to their expectations
#this yields 2*nhatch +1 equations with nhatch unknowns
#which is, in general, overdetermined.
init<-function(x1,x2,n1,n2,y,z,lambda,ppbt,suppress){
Nsamp<-x1+x2
nhatch<-length(lambda)
A1<-lambda
B1<-x1/Nsamp
A2<--n1*diag(lambda*ppbt,ncol=nhatch,nrow=nhatch)
LAMBDAMAT<-matrix(lambda,ncol=nhatch,nrow=nhatch)
LAMBDAMAT<-t(LAMBDAMAT)
YDIAG<-diag(y,nrow=nhatch,ncol=nhatch)
ZDIAG<-diag(z,nrow=nhatch,ncol=nhatch)
A2<-A2+YDIAG%*%LAMBDAMAT
B2<-rep(0,nhatch)
A3<-n2*diag((1-lambda)*ppbt,nrow=nhatch,ncol=nhatch)
A3<-A3+ZDIAG%*%LAMBDAMAT
B3<-z
A<-rbind(A1,A2,A3)
B<-c(B1,B2,B3)
if(rcond(t(A)%*%A)<1.e-15){
```

```
    if(!suppress)warning("matrix t(A)%A in init() is computationally singular")}
phosi<-solve(t(A)%*%A)%*%t(A)%*%B
return(abs(phosi))
}
```