EM and component-wise boosting for Hidden Markov Models: a machine-learning approach to capture-recapture

Robert W. Rankin[a,*]

[a] *Cetacean Research Unit, School of Veterinary and Life Sciences, Murdoch University, Australia*

Abstract

This study presents a new boosting method for capture-recapture models, rooted in predictive-performance and machine-learning. The regularization algorithm combines Expectation-Maximization and boosting to yield a type of multimodel inference, including automatic variable selection and control of model complexity. By analyzing simulations and a real dataset, this study shows the qualitatively similar estimates between AICc model-averaging and boosted capture-recapture for the CJS model. I discuss a number of benefits of boosting for capture-recapture, including: i) ability to fit non-linear patterns (regression-trees, splines); ii) sparser, simpler models that are less prone to over-fitting, singularities or boundary-value estimates than conventional methods; iii) an inference paradigm that is rooted in predictive-performance and free of p-values or 95% confidence intervals; and v) estimates that are slightly biased, but are more stable over multiple realizations of the data. Finally, I discuss some philosophical considerations to help practitioners motivate the use of either prediction-optimal methods (AIC, boosting) or model-consistent methods. The boosted capture-recapture framework is highly extensible and could provide a rich, unified framework for addressing many topics in capture-recapture, such as spatial capture-recapture, individual heterogeneity, and non-linear effects.

Keywords: capture-recapture, boosting, machine-learning, model-selection, marked animals, high-dimensional data

1. Introduction

In this study, I introduce boosting for Hidden-Markov Models (HMM) with a particular focus on capture-recapture models. It is targeted at capture-recapture practitioners who desire model parsimony under low-sample sizes and high-dimensional settings. Capture-recapture systems are perennially in an situation of high model-uncertainty (Johnson & Omland, 2004) and would benefit from an inference-paradigm that is flexible, extensible and rooted in good *predictive performance*. Some questions are the following. Can we find a simple model out of the hundreds or millions of plausible "fixed-effects" models? Can we correctly identify

*Corresponding author. E-mail: robertw.rankin@gmail.com

a sparse set of highly influential covariates in high-dimensional situations? Can the method accommodate non-linear relationships and interactions (e.g., regression trees, kernels and splines) without over-fitting? Can the method avoid the scourge of singularities and boundary-value estimates that trouble MLE-based models and their model-averaged derivatives? How does the method compare to other popular multimodel inference techniques, such as AICc model-averaging?

A motivating model will be the Cormack-Jolly-Seber (CJS) capture-recapture model, with a focus on which covariates influence the survival of an open population of marked animals under imperfect detection. While there are many regularization and variable selection techniques in univariate regression models, the problem becomes combinatorially difficult for HMMs such as capture-recapture models: we must consider multiple plausible specifications for both the transition process (survival), as well as the emission process (capture probability).

The issues of model selection and multimodel inference are front-and-centre in most capture-recapture studies. For example, the popular Program MARK (White & Burnham, 1999) is strongly allied to the model-averaging ideas of Burnham, Anderson, Buckland and others (Buckland et al., 1997; Anderson et al., 2000; Burnham & Anderson, 2004, 2014). By default, the program offers AICc-weighted averages (Akaike, 1974) of survival and capture probability. The widespread use of model-averaging in the capture-recapture field reflects an early appreciation by researchers for the *model uncertainty* inherent to capture-recapture: every analysis has dozens or thousands of plausible fixed-effect models, including, at a minimum, time-varying vs time-invariant processes. However, such *post-hoc* model-selection and/or averaging become computationally unfeasible with just a few extra covariates, due to the combinatorial explosion in the number of plausible models. Secondly, even if one could realistically compute every model, the AIC/AICc tends to favour more complex models (Shao, 1997; Hooten & Hobbs, 2015), which, in a capture-recapture context, can have singularities or boundary-value estimates (like 100% survival or 100% capture probability; Rankin et al., 2016; Hunt et al., 2016). This latter problem is rarely appreciated, but has motivated the development of Bayesian models to encourage parsimony under sparse data (Schofield et al., 2009; Schofield & Barker, 2011; Rankin et al., 2014, 2016)

Clearly, methods are needed to address the dual challenge of variable selection and low-sample sizes. Also, we should favour flexible techniques that can accommodate different functional forms (such as regression trees, splines, random effects) and find covariate-interactions, without over-fitting or producing boundary-value estimates.

Hand & Vinciotti (2003) and Burnham & Anderson (2004) hinted at a possible contender to the model-averaging approach when they suggested a parallel between multimodel inference and boosting: whereas model-averaging weights many fixed-effect models in a *post-hoc* manner, boosting sequentially combines hundreds or thousands of simple *weak learners* to yield a strong statistical model in aggregate. Most ecologists are familiar with boosting for univariate regression and classifications tasks (Elith et al., 2008; Kneib et al., 2009; Oppel et al., 2009; Hothorn et al., 2010; Tyne et al., 2015), but the recently developed *component-wise* *boosting* and *gamboostLSS* algorithms (Bühlmann & Yu, 2003; Schmid & Hothorn, 2008b; Schmid et al., 2010; Mayr et al., 2012) opened the way for complex hierarchical distributions with many components (Hothorn et al., 2010; Hutchinson et al., 2011; Schmid et al., 2013; Hofner et al., 2014). Under this boosting framework, each boosting iteration alternates between fitting the capture probability parameter (conditional on survival), and then fitting the survival component (conditional on the capture probabilities). Plus, boosting offers a wide variety of possible weak learners, from ordinary least squares to splines and CART-like trees (Hothorn et al., 2006; Bühlmann & Hothorn, 2007). This gives boosting much appeal over other sparsity-inducing variable selection paradigms, such as the Lasso (Tibshirani, 2011; Efron et al., 2004), Elastic-Net, Support Vector Machines, Hierarchical Bayesian shrinkage-estimators (Rankin et al., 2016). In this way, component-wise boosting offers a unified framework to address high-dimensional variable selection, interaction-detection, and non-linear relationships, while encouraging model parsimony through a prediction-optimized control on model complexity.

The contribution of this study is to develop a special boosting algorithm suitable for capture-recapture models. This study focuses on the simple two parameter Cormack-Jolly-Seber model (CJS) in order to introduce and validate the technique (hereafter, CJSboost). However, the ultimate goal is to expand the technique to a wider class of capture-recapture models. The central challenge of capture-recapture boosting is the serially dependent nature of observations. Hitherto, the gradient-descent procedure underlying boosting required independent data points (for estimating negative gradients). The CJSboost approach is to garner such conditional independence by imputing "two-slice marginal expectations" of pairs of *latent states* $\mathbf{z}$ (here, alive or dead). In CJSboost, we alternate between boosting the parameters (conditional on latent states) and imputing expectations of the latent states (conditional on the parameters). I provide two different techniques to impute such expectations: i) a special typ of Expectation-Maximization (called CJSboost-EM), and ii) Monte-Carlo approximation of the marginal distribution of latent states (CJSboost-MC). As I will show, both algorithms lead to approximately the same estimates. Furthermore, the estimates are qualitatively very similar to the model-averaged estimates by AICc weighting. The AIC is also motivated by optimal (asymptotic) predictive performance.

The idea of interweaving boosting and an Expectation-step was first suggested in the appendix of Ward et al. (2009) in their study of presence-only species distribution data. In an HMM context, boosting requires expectations of the *two-slice marginal* distributions of the latent states pairs, i.e. $p(z_{t-1}, z_t|\mathbf{y}_{1:T})$. It is this insight that paves the way to generalize boosting for a broad class of capture-recapture models.

This article proceeds with simulations and an analysis of an European Dipper dataset from Lebreton et al. (1992), with particular emphasis on comparing estimates from linear and non-linear models (e.g., CART-like trees), and comparisons to Maximum Likelihood estimation and AICc model-averaging (Burnham & Anderson, 2004) using Program MARK (White & Burnham, 1999). Simulations will also challenge CJSboost to perform a model-selection task that is nearly impossible for conventional methods: finding a sparse set of influential covariates among 21×21 different covariates.

There are two potential audiences for this paper. First, HMM practitioners will be interested in a general approach to boosting and HMMs, which opens new possibilities for incorporating regularization, semi-parametric learners and interaction detection to a vast catalogue of applications. For the second audience of mark-recapture practitioners, I offer a fresh view of mark-recapture from a *prediction* or *learning* perspective. For example, we can observe the degree to which regularization and bootstrap-validation suggest simpler models than those implied by AICc model-averaging. Boosting also offers capture-recapture an alternative means of inference that is principled and free of p-values and 95% Confidence Intervals (Anderson et al., 2000; Hoekstra et al., 2014; Morey et al., 2016). Furthermore, this new capture-recapture paradigm can easily accommodate a range of hot-topics in capture-recapture, such as individual-heterogeneity and spatial capture-recapture, by leveraging the wide variety of base-learners available in the `mboost` family of R packages (Bühlmann & Hothorn, 2007; Hothorn et al., 2006; Mayr et al., 2012; Hofner et al., 2012).

2. Methods

2.1. Organization

The manuscript begins by introducing some basic ideas of statistical learning theory (Section 2.2) and the Cormack-Jolly-Seber model. Section 2.3 describes two boosting algorithms, CJSboost-EM and CJSboost-MC, for capture-recapture models. Section 2.4 discusses some important practical considerations about regularization and base-learners. Section 2.5 describes a simulation to compare the estimates from CJSboost-EM and CJSboost-MC, as well as AICc model-averaging and MLEs (results in 3.1). Section 2.6 describes a reanalysis of of dipper dataset using CJSboost-EM and AICc model-averaging (results in 3.2). Section 2.7 uses simulations to assess the performance of CJSboost-EM under a high-dimensional model-selection problem (results in 3.3). The manuscript finishes with a discussion about how to interpret the results from CJSboost and poses some new questions (Section 4). A summary is provided in Section 5. For R code and a tutorial, see the online content at `http://github.com/faraway1nspace/HMMboost/`.

2.2. Background

2.2.1. The Prediction perspective

From a prediction perspective, our goal is to estimate a prediction function G that maps covariate information $\mathbb{X}$ to our response variable (i.e., $G\colon \mathbb{X} \to \mathbb{Y}$). Our data $\{y_j, \mathbf{x}_j\}_{j=1}^{n}$ arises from some unknown probability distribution P. Our optimal prediction function is that which minimizes the *generalization error*:

$$\mathcal{L}(y, G(\mathbf{x})) = \int \ell\big(y, G(\mathbf{x})\big) dP(y, \mathbf{x}) \tag{1}$$

where ℓ is a *loss* function (it scores how badly we are predicting y from $\mathbf{x}$) and $\mathcal{L}$ is the *expected loss*, a.k.a, the *risk* (our loss integrated over the entire data distribution). Our goal is to minimize the loss on new, unseen data drawn from the unknowable data distribution P (Bühlmann & Yu, 2003; Meir & Rätsch, 2003; Murphy, 2012a). It should be noted that for many disciplines, making good predictions is the primary goal (e.g., financial forecasting). In mark-recapture, we usually wish to make inference about covariates $\mathbb{X}$ and their functional relationship (G) to the response variable, such as estimating survival from capture histories, rather than making predictions *per se*. In such cases, the generalization criteria (1) serves as a principled means of "model parsimony": our model is as complex as is justified to both explain the observed data and make good predictions on new data. This is very different from Maximum-Likelihood Estimation (as in Program MARK) whose estimate $\hat{G}$ is that which maximizes the likelihood of having seen the observed data $\mathbf{y}$. It is, however, similar to AIC selection, which is implicitly motivated by minimizing expected loss (Vrieze, 2012), i.e., optimal predictive performance.

One cannot measure the generalization error (1); instead, we must proceed by minimizing the *empirical risk* measured on our observed data:

$$L(\mathbf{y}, G(\mathbf{X})) = \sum_{j=1}^{n} \ell(y_j, G(\mathbf{X}_j)) \tag{2}$$

Minimizing $L(\mathbf{y}, G(\mathbf{X}))$ until convergence is easy but will obviously *over-fit* a sample and make bad predictions. However, it can be shown that if we constrain the complexity of our function space (Bühlmann & Yu, 2003; Meir & Rätsch, 2003; Mukherjee et al., 2003) we can pursue a strategy of "regularized risk minimization" and bound the generalization error. In learning algorithms, this entails at least one regularization parameter that smooths or constrains the complexity of G. In other words, we do not seek the estimator that best fits the data. In boosting, the principal means of regularization is via shrinkage (taking only small steps along the risk gradient) and early-stopping (*not* running the algorithm until the risk convergences). These correspond to hyperparameters ν and m, respectively, the shrinkage weight and the number of boosting iterations. For a small m, the model is strongly constrained and very conservative; as m gets big, the model becomes more complex. Likewise, $\nu \ll 1$ ensures that the influence of any one boosting step is tiny. Practically, one fixes ν and finds an optimal m via cross-validation. Figure 1 shows an example of bootstrap-validation to find an optimal stopping criteria m_{CV}, used in the dipper CJS analysis (Section 3.2).

2.2.2. Motivation for regularization

The unregularized boosted model with prediction function $G^{(m \to \infty)}$ results in a *fully-saturated model*, which (depending on the prediction function) is equivalent to the Maximum Likelihood solution (Mayr et al., 2012). At finite sample sizes and a large candidate-set of covariates, MLEs do not result in good predictions: they may be unbiased, but they will be wildly sensitive to noise in the data, especially for capture-recapture. For a regularized model $G^{(m \ll \infty)}$, learning algorithms should preferentially select influential covariates and shrink the coefficients of less-important covariates close to zero. This shrinkage induces a bias (Bühlmann & Yu, 2003; Bühlmann & Hothorn, 2007), but the predictions are more robust to noisy data (i.e. low-variance; Murphy, 2012a). In this light, we see the practical similarity between regularization and the more popular model parsimony strategies in capture-recapture, such as model-selection, model-averaging, and subjective Bayesian models. Hooten & Hobbs (2015) implore ecologists to unify these techniques under a Bayesian perspective; for example, the AIC, Lasso/L2Boosting, Ridge-regression can be reformulated in such a way that they differ according to the priors on the ℓ_0, ℓ_1 and ℓ_2-norm of regression-coefficients, respectively. Even

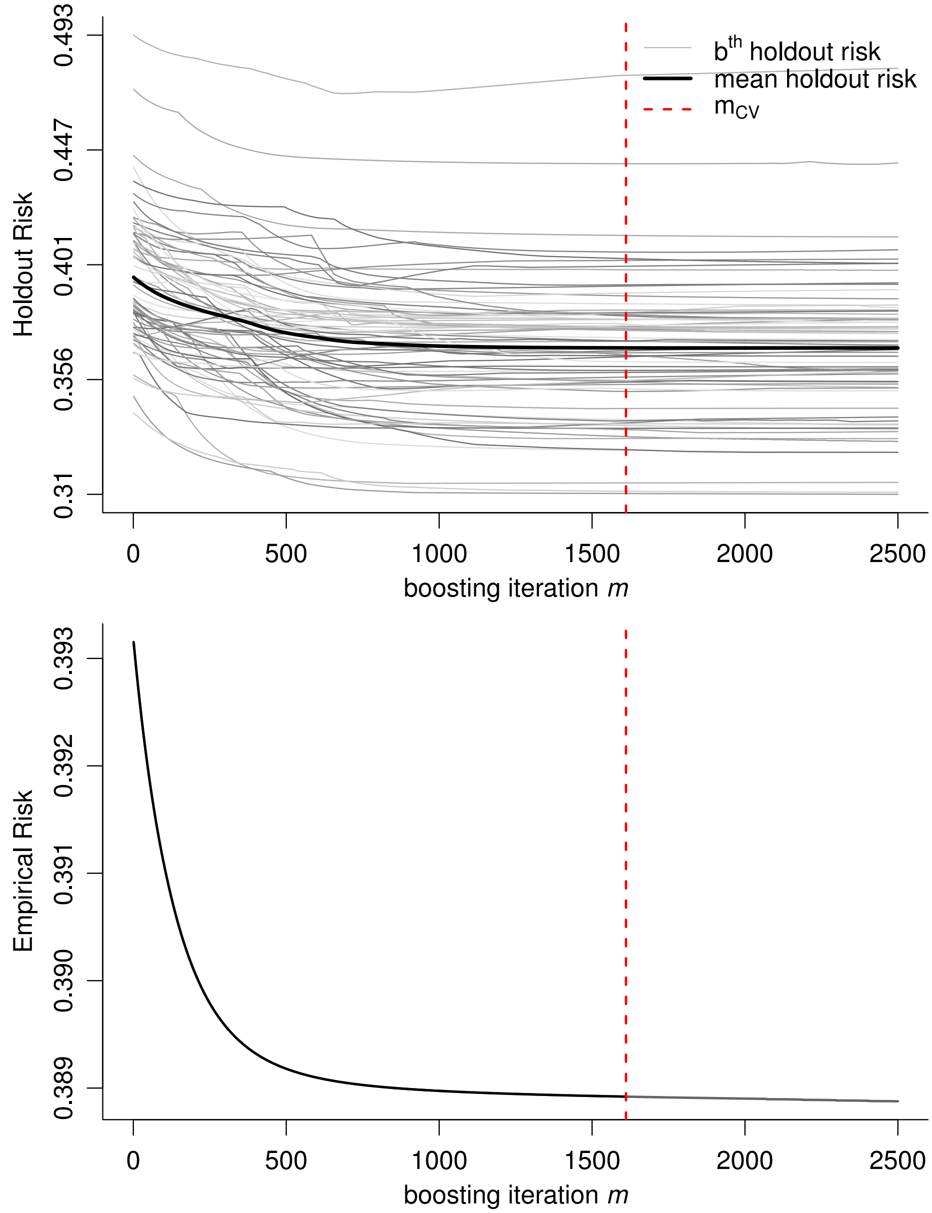

Figure 1: Monitoring the risk minimization (negative CJS log-Likelihood) for the Dipper analysis, using CJSboost-EM with CART-like trees as base-learners. Each boosting iteration m takes a step towards minimizing the empirical risk and selects a new shrunken base-learner to add to the ensembles. *Top*: Estimating the optimal stopping criteria at m_{cv} (red dashed line) via bootstrap-validation. Each grey line represents the holdout-risk predicted on a subset of the capture-histories, from a model trained on a bootstrap-sample of capture-histories. m_{cv} minimizes the mean holdout-risk over all bootstrap runs, an estimate of the *expected loss*. *Bottom*: The empirical risk of the final statistical model using the full dataset; stopping early at m_{cv}, well before convergence.

a simple Bayesian prior can be understood as a type of regularization by shrinking estimates away from their MLE values and towards the conservative expectations of a prior (a.k.a "natural shrinkage"; Hooten & Hobbs, 2015).

Today, most mark-recapture practitioners are implicitly using a prediction criteria for inference. For example, the AIC is popular in mark-recapture studies (Johnson & Omland, 2004), thanks in large part to the Frequentist and Information-Theoretic leanings of Program MARK (White & Burnham, 1999). The AIC is asymptotically prediction-optimal, whose maximum risk is minimal among all potential models, and has connections with leave-one-out-cross-validation (LOOCV; Stone, 1977; Shao, 1993, 1997; Vrieze, 2012). However, statisticians consider the AIC to be a bit too permissive, especially if the "true model" is sparse (Shao, 1993; Burnham & Anderson, 2004; Hooten & Hobbs, 2015). For practical mark-recapture analysis, the AIC/AICc can favour overly-complex models which can suffer singularities or boundary-value estimates (like 100% survival or 100% capture probability; Rankin et al., 2016; Hunt et al., 2016). Boosting is also prediction-optimal (Bühlmann & Yu, 2003), but skirts the issues of singularities and boundary-value estimates by fitting very simple models, called *base-learners* in a step-wise manner. At finite sample sizes, boosting should lead to slightly sparser models than the AIC/AICc.

In an extreme case of sparsity, when being prediction-optimal is not the chief concern, and one wishes to instead uncover a "true model" with just a few important covariates, boosting has another desirable property. Regularized risk-minimizers (in a univariate setting) can be made model-selection consistent by hard-thresholding unimportant covariates to zero weight (Bach, 2008; Meinshausen & Bühlmann, 2010; Murphy, 2012c). These sparse solutions may be more interesting for capture-recapture practitioners when inference about covariates or estimating survival is the chief concern.

2.2.3. Introduction to boosting

Boosting is an iterative method for obtaining a statistical model via gradient descent (Breiman, 1998; Friedman et al., 2000; Friedman, 2001; Breiman, 1999; Schmid et al., 2010; Robinzonov, 2013). The key insight is that one can build a strong predictor $F = G(X)$ by the step-wise addition of many weak *base-learners*, $b(y, x) \Rightarrow g$, $g(x)$: $x \to y$ (Schapire, 1990; Kearns & Valiant, 1994). Remarkably, a base-learner need only have a predictive performance of slightly better than random chance for the entire ensemble to be strong. The ensemble results in a smooth additive model of adaptive complexity:

$$F^{(m)} = G(\mathbf{X}) = \sum_{m=1}^{m_{\text{stop}}} \nu \cdot g_k^{(m)}(\mathbf{X}_k) \tag{3}$$

where each g_k is a base-learner's prediction function, shrunken by ν. The ensemble is constructed as follows: i) initialize the prediction vector $F^{(m=0)}$ at some uniform estimate (like the MLE of an intercept model); ii) fit base-learners b to $\hat{\mathbf{u}}$, the estimated negative-gradient of the loss function (the residual variation unexplained by $F^{(m-1)}$), $b(\hat{\mathbf{u}}, \mathbf{x}) \Rightarrow g$; iii) shrink each base-learners' prediction $g(\mathbf{x}) = \hat{f}^{(m)}$ by a small fraction ν; iv) update the overall prediction $F^{(m)} = F^{(m-1)} + \nu \hat{f}^{(m)}$; v) repeat for m_{stop} iterations. m_{stop} is the key parameter that governs model complexity (Bühlmann & Yu, 2003; Schmid & Hothorn, 2008a) and must be tuned by cross-validation or bootstrap-validation. Variable selection can be directly embedded within each boosting iteration by choosing only one best-fitting base-learner per m iteration, discriminating among a large candidate set of base-learners $\{b(\mathbf{u}, \mathbf{x}_1), b(\mathbf{u}, \mathbf{x}_2), ..., b(\mathbf{u}, \mathbf{x}_k)\}$, and where each candidate only includes a small subset of the covariates $\mathbf{X}$. For linear base-learners, this boosting algorithm is generally considered equivalent to ℓ_1 regularization (Efron et al., 2004; Bühlmann & Hothorn, 2007), a.k.a the Lasso.

Base-Learners may be simple Least-Squares estimators, b_{OLS}, in which case an unregularized boosted model will estimate regression coefficients that are practically identical to a frequentist GLM. However, Bühlmann & Yu (2003) showed that for L2Boosting, good overall predictive performance depends on the fact that base-learners are very weak. Therefore, practitioners commonly use highly-constrained base-learners such as Penalized Least Squares b_{PLS}, or recursive-partitioning trees b_{trees} (a.k.a CART), or low-rank splines b_{spline}. Despite their weakness, Bühlmann & Yu note that for a fixed constraint (such as low degrees-of-freedom in b_{spline} or low tree-depth in b_{trees}), the overall boosted ensemble will typically have a much greater complexity than its constituent base-learners and that this complexity is adaptive.

There are many flavours of boosting. CJSboost hails primarily from the component-wise boosting and gamboostLSS frameworks (Bühlmann & Yu, 2003; Schmid & Hothorn, 2008b; Schmid et al., 2010; Mayr et al., 2012). Here, the prediction vector is now a set $\mathcal{F} = (F_1, F_2, ..., F_k)$ of k components, each representing one of the parameters in the likelihood function (e.g., ϕ and p). Each parameter has its own ensemble of base-learners. The loss function is the negative log-likelihood of the data-generating model $\ell_i = -\log p(\mathbf{y}_i|\phi_i, p_i) = -\log p\left(\mathbf{y}_i\middle|\frac{1}{1+e^{-F_\phi}}, \frac{1}{1+e^{-F_p}}\right)$ (see the CJS likelihood 4). Each components' gradient can be estimated from the negative partial-derivatives of the loss function with respect to F_k, i.e., $\hat{u}_{k,i} = -\frac{\partial \ell_i}{\partial F_k}$, conditional on the values of the other prediction vectors $F_{\neg k}$. Each k parameter is updated once per boosting iteration.

2.2.4. The Cormack-Jolly-Seber model and Hidden Markov Models

The above component-wise boosting framework is not suitable for serially dependent observations in an HMM time-series: consider that the negative gradient in traditional boosting must be estimated point-wise for each independent data pair (y_i, X_i). Instead, the CJS likelihood is evaluated on individuals' entire *capture histories* $\mathbf{y}_i = (y_1, y_2, ..., y_T)^\intercal$ over T capture periods:

$$p(\mathbf{y}_i|\phi, p, t_i^0) = \Big(\prod_{t>t_i^0}^{t_i^*} \phi_{i,t}(p_{i,t})^{y_{i,t}}(1-p_{i,t})^{1-y_{i,t}}\Big)\chi_i^{(t_i^*+1)} \quad (4)$$

Where i indexes the n uniquely identified individuals constituting our dataset; $t = 1{:}T$ indexes the T equally spaced capture periods (time); $y_{i,t} \in [0, 1]$ scores whether individual i was observed in capture period t; $\phi_{i,t}$ is the probability of surviving from capture period t–1 to t (note the one-time-step difference from the definition of ϕ_t used in Program MARK); $p_{i,t}$ is the capture probability of individual i in capture period t (a.k.a, our observation error, or the "emission process" in HMM parlance); t_i^0 is the first capture period in which individual i was first observed; t_i^* is the last period when individual i was observed. Finally, $\chi_i^{(t_i^*+1)}$ is the probability of never being seen again after t_i^* until the end of the study, $\chi_i^{(t)} = (1 - \phi_{i,t}) + (1 - p_{i,t})\phi_{i,t}\chi_i^{(t+1)}$, and whose recursive calculation exemplifies the serially dependent nature of the model. $\mathbf{Y}^{n\times T}$ is the full matrix of our capture-histories.

Mark-recapture practitioners will be interested to note: i) the model conditions on first-capture $\{t_i^0\}_{i=1}^n$; ii) the model can potentially allow for individual heterogeneity in capture probabilities $p_{t,i}$ (which otherwise results in a negative-bias in population abundance estimates; Carothers, 1973; Burnham & Overton, 1978;

Rankin et al., 2016); and iii) certain parameters cannot be separated in Maximum-Likelihood Estimation, such as p_T and ϕ_T, but this is less of an issue under constrained base-learners and regularization.

In order to boost the CJS model, we need independence of data pairs $(y_{i,t}, X_{i,t})$. If we reformulate the capture-recapture system as a HMM, we can garner conditional independence via the concept of latent states $z_{i,t} \in \{0, 1\}$ to represent $\{\text{dead}, \text{alive}\}$. When $z_{i,t} = 1$, then individual i is alive and available for capture at time t, and the probability of a capture is simply $p(y_{i,t}=1|z_{i,t}=1) = p_{i,t}$. However, if $z_{i,t} = 0$ then individual i is dead and unavailable for capture at time t; therefore the probability of a capture is zero.

Obviously, one never knows with certainty the latent states of a trailing sequence of zeros $\mathbf{y}_{t:T} = (0, ..., 0)^{\intercal}$, but we can utilize well-developed HMM tools to estimate the state-sequence $\mathbf{z}$ in various ways. In particular, "CJSboost-EM" 2.3.1 utilizes the marginal *expectations* of (z_t, z_{t-1}) in an Expectation-Maximization step. "CJSboost-MC" 2.3.2 utilizes Monte-Carlo integration by drawing random values of $\mathbf{z}$ from the posterior $\pi(\mathbf{z}|\mathbf{y}, \phi, p)$. We can interweave these two methods within a boosting algorithm: both will allow us to estimate point-wise negative gradients for all *complete-data* points $\big(\{y_{i,t}, z_{i,t}, z_{i,t-1}\}, X_{i,t}\big)$ and proceed with the gradient descent algorithm.

2.3. CJSboost

I will now formally describe the CJSboost variants "CJSboost-EM" and "CJSboost-MC". In practise, I will show that they lead to approximately the same estimates, but have different computation disadvantages under different scenarios. When the number of discrete states in the HMM process is low (2-3), then the deterministic EM algorithm is significantly faster and less prone to approximation error. For example, in our CJS example, we just have two latent states $\{0, 1\} := \{\text{dead}, \text{alive}\}$ with three legal transitions $\{1\to1, 1\to0, 0\to0, \cancel{0\to1}\}$. However, as the number of discrete states increases, the memory management of all the possible transitions becomes combinatorially expensive. In such scenarios, it is computationally easier to sample z from its posterior.

2.3.1. CJSboost by Expectation-Maximization

For a CJS model using CJSboost-EM, our target risk is the CJS negative log-likelihood. However, we use the principle of Expectation-Minimization to derive a slightly different loss function and subsequent negative gradients. Our loss is derived from the negative Complete-Data Log-Likelihood (CDL) which assumes we have estimates of the latent state $z_{i,t}, z_{i,t-1}$.

$$\begin{aligned} -\text{CDL}(y_{i,t}, z_{i,t}, z_{i,t-1}|F_{i,t,\phi}, F_{i,t,p}) = \; & -\mathbb{1}[z_{i,t-1}=1, z_{i,t}=1]\Bigg(\log\left(\frac{1}{1+e^{-F_{i,t,\phi}}}\right) + y_{i,t}\log\left(\frac{1}{1+e^{-F_{i,t,p}}}\right) \\ & + (1-y_{i,t})\log\left(\frac{1}{1+e^{F_{i,t,p}}}\right)\Bigg) \\ & -\mathbb{1}[z_{i,t-1}=1, z_{i,t}=0]\log\left(\frac{1}{1+e^{F_{i,t,\phi}}}\right) \\ & -\mathbb{1}[z_{i,t-1}=0, z_{i,t}=0] \end{aligned} \tag{5}$$

where y and z are defined as above in (4) and $F_{i,t,p}$ and $F_{i,t,\phi}$ are the prediction vectors for the capture probability component and the survival component, respectively, on the logit scale. In accordance with the principle of EM, we derive a "Q-function" to serve as our new loss, replacing the values of $(z_{i,t-1}, z_{i,t})$ with their *two-slice marginal* expectations: $w_t(q,r) := p\big(z_{t-1} = q, z_t = r|\mathbf{y}, \mathcal{F}\big)$. Conditional on the prediction vectors $\mathcal{F}$ and the capture history $\mathbf{y}$, the values of the two-slice marginals $\{w(1,1), w(1,0), w(0,0)\}$ can be easily computed using a standard "forwards-backwards" HMM algorithm (Rabiner, 1989; Murphy, 2012b), as detailed in Appendix A. We can also treat each $i \times t$ observation as being conditionally independent, resulting in the new index $j := (i,t)$. The Q-function is:

$$\begin{aligned}
\ell(y_j, \{F_{j,\phi}, F_{j,p}\}) &= -w_j(1,1)\left(\log\left(\frac{1}{1+e^{-F_{j,\phi}}}\right) + y_j \log\left(\frac{1}{1+e^{-F_{j,p}}}\right) + (1-y_j)\log\left(\frac{1}{1+e^{F_{j,p}}}\right)\right) \\
&\quad - w_j(1,0)\log\left(\frac{1}{1+e^{F_{j,\phi}}}\right) \\
&\quad - w_j(0,0)
\end{aligned} \tag{6}$$

According to the theory of EM, by minimizing the Q-function, we also minimize the target empirical risk: the negative CJS log-likelihood (4). The advantage of working with the Q-function is that it is easy to calculate the negative gradients (7) and proceed with the gradient descent.

I now describe the CJSboost-EM algorithm.

1. Set the regularization parameters: $m_{\text{stop}} \approx 10^2 - 10^3$; ν_ϕ, $\nu_p \approx 10^{-3} - 10^{-1}$;
2. Initialize: $m = 1$; $\mathcal{F}^{(0)} = \left\{F_\phi^{(0)} = \hat{\phi}^{\text{MLE}}(\cdot), F_p^{(0)} = \hat{p}^{\text{MLE}}(\cdot)\right\}$ (i.e., initialize the prediction vectors at the MLEs of a simple intercept model).
3. Estimate the two-slice marginal probabilities $\{w_j(1,1), w_j(1,0), w_j(0,0)\}_{j=1}^{J}$ for all individuals and capture-periods, using the forwards-backwards algorithm (see Appendix A.3).
4. Estimate the negative gradients:

$$\begin{aligned}
\hat{u}_{j,\phi}^{(m)} &= -\frac{\partial \ell_j}{\partial F_\phi^{(m-1)}} = \frac{w_j(1,1) - w_j(1,0)e^{F_{j,\phi}^{(m-1)}}}{\left(1 + e^{F_{j,\phi}^{(m-1)}}\right)} \\
\hat{u}_{j,p}^{(m)} &= -\frac{\partial \ell_j}{\partial F_p^{(m-1)}} = \frac{w_j(1,1)\left(1 + e^{F_{j,p}^{(m-1)}}\right) y_j - w_j(1,1)e^{F_{j,p}^{(m-1)}}}{1 + e^{F_{j,p}^{(m-1)}}}
\end{aligned} \tag{7}$$

5. For each component $\theta = \{\phi, p\}$:
   (a) fit k base-learners independently to the gradients: $b_k\left(\hat{\mathbf{u}}_\theta^{(m)}, X_k\right) \Rightarrow g_k(X_k)$;
   (b) each fitted learner makes an estimate of the gradient, $\hat{f}_k = g_k(X_k)$;
   (c) select the best-fitting base-learner $k^* = \underset{k}{\operatorname{argmin}}(\hat{\mathbf{u}}_\theta^{(m)} - \hat{f}_k)^2$ and append the fitted-learner to the ensemble $\mathcal{G}_\theta \leftarrow g_k^*$;
   (d) update the prediction vector: $F_\theta^{(m)} = \nu_\theta \hat{f}_k^* + F_\theta^{(m-1)}$;
6. Estimate the empirical risk on the full data $L(\mathbf{Y}, \mathcal{F}^{(m)})$, or estimate the holdout-risk on a test set $L(\mathbf{Y}_{\text{test}}, \mathcal{F}_{\text{test}}^{(m)})$ s.t. $\mathcal{F}_{\text{test}}^{(m)} = \{G_\phi^{(m)}(\mathbf{X}_{\text{test}}), G_p^{(m)}(\mathbf{X}_{\text{test}})\}$.

7. Update $m = m + 1$.
8. Repeat steps 3 to 7 until $m = m_{stop}$.

The outputs of the algorithm are the fit vectors $\mathcal{F}$ and the ensemble of fitted base-learners $\mathcal{G}_\phi$ and $\mathcal{G}_p$. The estimate of survival for individual i at time t can be retrieved $j := (i,t); \phi_j = \text{logit}^{-1}(F_j)$, and likewise for capture probability. For predicting ϕ and p on new covariate data $\mathbf{X}$, we merely process the data through the ensemble of fitted base-learners and shrink by ν, i.e., $F_\theta^{\text{pred}} = G_\theta(\mathbf{X}) = \nu_\theta \sum_{g_k \in \mathcal{G}_\theta} g_k(\mathbf{X})$.

The three regularization parameters m_{stop}, ν_ϕ, ν_p must be tuned by minimizing the holdout-risk averaged over many out-of-sample test sets, i.e., our estimate of the expected loss (see 2.4).

2.3.2. CJSboost by Monte-Carlo approximation

A second strategy to garner conditional independence of data-points $(y_j, \mathbf{x}_j)$ and estimate the negative gradients is to integrate over the latent state distributions $\pi(\mathbf{z}_i|\mathbf{y}_i, \mathcal{F}_i)$ with a large sample drawn from the posterior. A fast and simple "forward-filtering and backward-sampling" algorithm is used (Rabiner, 1989; Murphy, 2012b), detailed in Appendix A.4. Within each boosting iteration m, we sample S sequences of $\mathbf{z}_i$. Per s sequence, we estimate a separate negative-gradient, and fit base-learners to it. After fitting all S samples, we update the prediction vectors with the best-fitting base-learners from each sequence, $F_\theta^{(m+1)} = F_\theta^{(m)} + \nu_\theta \sum_s^S \hat{f}^{(s)}$. Over $S \times m$ draws, this is approximately equivalent to the EM algorithm. For comparable results to CJSboost-EM, the shrinkage parameters ν_{MC} should be set equal to $\frac{1}{S}\nu_{\text{EM}}$, i.e., the contribution of any one sequence $\mathbf{z}^{(s)}$ is small.

I now describe the CJSboost-MC algorithm:

1. Set parameters S, m_{stop}, ν_ϕ, and ν_p.
2. Initialize $m = 1$ and $\mathcal{F}^{(0)}$.
3. For $s = 1 : S$, do:
   (a) sample latent state sequence $\mathbf{z}_i^{(s)} \sim \pi(z|\mathbf{y}_i, \mathcal{F}_i)$ (see Appendix A.4);
   (b) estimate the negative gradients, conditional on $\mathbf{z}_i^{(s)}$:

$$\hat{u}_{i,t,\phi}^{(m,s)} = -\frac{\partial \ell_{i,t}}{\partial F_\phi^{(m-1)}} = \frac{\mathbb{1}[z_{i,t-1}^{(s)}=1, z_{i,t}^{(s)}=1] - \mathbb{1}[z_{i,t-1}^{(s)}=1, z_{i,t}^{(s)}=0] \cdot e^{F_{i,t,\phi}^{(m-1)}}}{1 + e^{F_{i,t,\phi}^{(m-1)}}}$$

$$\hat{u}_{i,t,p}^{(m,s)} = -\frac{\partial \ell_{i,t}}{\partial F_p^{(m-1)}} = \frac{\mathbb{1}[z_{i,t-1}^{(s)}=1, z_{i,t}^{(s)}=1] \left(\left(1 + e^{F_{i,t,p}^{(m-1)}}\right) y_{i,t} - e^{F_{i,t,p}^{(m-1)}}\right)}{1 + e^{F_{i,t,p}^{(m-1)}}}$$

   (c) for each component $\theta = \{\phi, p\}$:
      i. fit k base-learners independently to the gradients: $b_k\left(\hat{\mathbf{u}}_\theta^{(m,s)}, X_k\right) \Rightarrow g_k^{(s)}(X_k)$.
      ii. each fitted learner makes an estimate of the gradient, $\hat{f}_k^{(s)} = g_k^{(s)}(X_k)$
      iii. select the best-fitting base-learner $k^{(s)*} = \underset{k}{\text{argmin}}(\hat{\mathbf{u}}_\theta^{(m,s)} - \hat{f}_k^{(s)})^2$ and append the fitted-learner to the ensemble $\mathcal{G}_\theta \leftarrow g_k^{(s)*}$.
4. For each $\theta = \{\phi, p\}$: update the fit vectors, overall s: $F_\theta^{(m)} = F_\theta^{(m-1)} + \nu_\theta \sum_s^S \hat{f}^{(s)}$.

5. Estimate the empirical risk on the training data $L(\mathbf{Y}, \mathcal{F}^{(m)})$, or on a holdout test set $L(\mathbf{Y}_{\text{test}}, \mathcal{F}_{\text{test}}^{(m)})$.
6. $m = m + 1$
7. Repeat steps 3 to 6 until $m = m_{stop}$.

Just as in the CJSboost-EM algorithm, we must tune ν and m_{stop} through cross-validation or bootstrap-validation (Section 2.4).

Notice that although the two algorithms have different specific negative-gradients and loss functions, the empirical risk is always the negative log-likelihood of the CJS model.

2.4. Hyperparameters

In component-wise boosting, the three most important regularization parameters are m_{stop}, ν_ϕ, ν_p. These must be tuned by some form of holdout-validation. As per Schmid et al. (2013), I suggest sampling with replacement (bootstrapping) individuals' capture histories between 50 to 100 times, training a new model on each bootstrap sample. On average, each bootstrap leaves 36.5% of the capture-histories unused in the model fitting, which can then be used to estimate a holdout-risk. Averaged over all bootstraps, this is an estimate of the generalization error. Bootstrap-validation is preferable to k-fold or leave-one-out cross-validation because it is most similar to the multiple resampling/subsampling schemes of Shao (Monte-Carlo CV and Delete-d CV; 1993, 1997) which are model-selection consistent under a wider variety of conditions (e.g., sparsity, tapering). Finally, the K-bootstrap can also give us an estimate of posterior inclusion probabilities via stability-selection (Meinshausen & Bühlmann, 2010; Murphy, 2012c), which I use in section 2.7.

Because we can monitor the trajectory of the holdout-risk during each boosting iteration, we only need to perform one round of K-bootstrap-validation to find the optimal m. See Figure 1 for an example of monitoring the holdout-risk and estimating m_{cv}. Estimating optimal values of ν_ϕ and ν_p is more complicated because they are continuous; in practise we must discretize the set of plausible combinations, e.g., $(10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}) \times (10^{-4}, 10^{-3}, 10^{-2}, 10^{-1})$. Each combination requires a separate bootstrap-validation exercise. This is the most expensive step of CJSboost. See Appendix B for a suggestion on how to perform this task with only 7-10 K-bootstrap-validation runs.

The reader should note that other multivariate boosting techniques (such as gamboostLSS; Schmid et al., 2013; Mayr et al., 2012) instead have a single fixed ν for all parameters, and seek to optimize m_θ per parameter θ. This is inversely related to what I propose: optimizing a global m_{stop} for both parameters, while optimizing the *ratio* of ν_{θ_1} to ν_{θ_2}. The two methods are equivalent in their outcome. In other words, making ν_θ smaller for component θ is the same as decreasing m_θ for fixed ν, and *vice versa*. More importantly, other authors have claimed that there is little benefit in optimizing m and ν for each component (Schmid et al., 2013). This is untrue for CJSboost, where the optimal estimate of ν_ϕ may be several orders of magnitude different than the optimal ν_p.

There are theoretically many other hyperparameters, such as the base-learner parameters which control flexibility, e.g. the effective degrees-of-freedom of a spline, or the maximum tree-depth of a conditional inference tree (Hothorn et al., 2006). However, Bühlmann & Yu (2003) and Schmid & Hothorn (2008a) show that these can be safely fixed and one should instead focus on m_{stop}. A more important consideration is the *relative* flexibility of competing base-learners: multi-covariate learners and unpenalized learners have a greater freedom to (over)fit the residual variation and will be preferentially selected in the algorithm. Therefore, one should use penalties to enforce a similar effective degrees-of-freedom among all base-learners, as well as decompose higher-order interactions and non-linear curves into their constituent components. For example, if one wishes to learn about the role of covariates x_1 and x_2 and the possibility of an interaction between $x_1 \times x_2$, then one must add three PLS base-learners of equal effective-df: two for the main-effects and a separate base-learner for their interaction. Readers should refer to the practise of "centring" in Kneib et al. (2009) and Hofner et al. (2012).

2.5. Simulation 1: MC vs EM vs AICc vs MLE

The goals of this simulation were to compare estimates of survival and capture probabilities among the two boosting algorithms (CJSboost-EM and CJSboost-MC) and benchmark them against MLEs and AICc model-averaging. The simulated dataset was inspired by the European Dipper dataset from (Lebreton et al., 1992). The simulated dataset included $T = 10$ primary periods, and $n = 300$ individuals in two groups (male and female). Individuals' first-capture periods (t_i^0) were random. The true processes were smoothly time-varying effects plus an individual covariate (sex-effect). The true data-generating processes were: $p(t, \text{sex}) = \text{logit}^{-1}\left(0.5 + t^{\frac{\sin(t)}{17}}\right) - 10 \cdot \mathbb{1}[\text{sex} = 1]$ and $\phi(t, \text{sex}) = 0.91 - 0.01 \cdot t - 0.05 \cdot \mathbb{1}[t = 5, 6] + 0.05 \cdot \mathbb{1}[t = 9, 10] - 0.05 \cdot \mathbb{1}[\text{sex} = 1]$. Figure 3 graphs the true processes.

Figure 2A shows all combinations of p and ϕ parametrizations, which has 64 possible fixed-effect models for estimation by Maximum Likelihood and AICc model-averaging. The true model is best represented as $\phi(t, \text{sex})p(t, \text{sex})$. *Flood* is a dummy categorical variable that groups the captures periods $\{4, 5, 6\}$ (corresponding to a trough in either process): it simulates an analyst's hypothesis that high flood years (in periods 4,5,6) may influence dipper survival and capture probability. The MLE and AICc model-averaging analyses were conducted with Program MARK (White & Burnham, 1999) and RMark (Laake, 2013).

For the boosting analyses, four techniques were compared: i) linear-model CJSboost-EM (using OLS and PLS base-learners); ii) non-linear CJSboost-EM (using a CART-like base-learner called "conditional-inference trees"; Hothorn et al., 2006); iii) linear-model CJSboost-MC; and iv) non-linear CJSboost-MC. For the linear-models, the OLS and PLS base-learners included all base-learners listed in Figure 2B. See the `mboost` R package (Bühlmann & Hothorn, 2007; Hofner et al., 2012). Variable selection occurs as a consequence of the internal competition among base-learners to fit the gradient, per boosting iteration. The effective degrees-of-freedom of each base-learner were constrained with ridge penalties, as per Section 2.4. The non-linear CJSboost models had just one CART-like base-learner per ϕ and p. Variable-selection and interactions are implemented internally to the `ctree` algorithm, much like a black-box.

All 4 models used 70-times bootstrap-validation to estimate optimal values of m_{stop}, ν_ϕ and ν_p, as per section 2.4.

A) Linear Models (MLE)

$$\left\{\begin{array}{c}\phi(\cdot)\\ \phi(t)\\ \phi(\text{sex})\\ \phi(\text{flood})\\ \phi(t,\text{sex})\\ \phi(t\times\text{sex})\\ \phi(\text{flood},\text{sex})\\ \phi(\text{flood}\times\text{sex})\end{array}\right\} \times \left\{\begin{array}{c}p(\cdot)\\ p(t)\\ p(\text{sex})\\ p(\text{flood})\\ p(t,\text{sex})\\ p(t\times\text{sex})\\ p(\text{flood},\text{sex})\\ p(\text{flood}\times\text{sex})\end{array}\right\} : \text{64 fixed-effects models for model-averaging}$$

B) Equivalent Linear Model Base-learners

$$\left\{\begin{array}{c}b_{\text{OLS}}(u_\phi,\mathbf{1}^{N_J})\\ b_{\text{PLS}}(u_\phi,X_t;df=1)\\ b_{\text{OLS}}(u_\phi,X_{\text{sex}})\\ b_{\text{OLS}}(u_\phi,X_{\text{flood}})\\ b_{\text{PLS}}(u_\phi,X_{t,\text{sex}};df=1)\\ b_{\text{PLS}}(u_\phi,X_{t\times\text{sex}};df=1)\\ b_{\text{PLS}}(u_\phi,X_{\text{flood,sex}};df=1)\\ b_{\text{PLS}}(u_\phi,X_{\text{flood}\times\text{sex}};df=1)\end{array}\right\} \left\{\begin{array}{c}b_{\text{spline}}(u_\phi,X_t;df=1)\\ b_{\text{spline}}(u_\phi,X_{t\times\text{sex}};df=1)\end{array}\right\} + \left\{\begin{array}{c}b_{\text{OLS}}(u_p,\mathbf{1}^{N_J})\\ b_{\text{PLS}}(u_p,X_t;df=1)\\ b_{\text{OLS}}(u_p,X_{\text{sex}})\\ b_{\text{OLS}}(u_p,X_{\text{flood}})\\ b_{\text{PLS}}(u_p,X_{t,\text{sex}};df=1)\\ b_{\text{PLS}}(u_p,X_{t\times\text{sex}};df=1)\\ b_{\text{PLS}}(u_p,X_{\text{flood,sex}};df=1)\\ b_{\text{PLS}}(u_p,X_{\text{flood}\times\text{sex}};df=1)\end{array}\right\} \left\{\begin{array}{c}b_{\text{spline}}(u_p,X_t;df=1)\\ b_{\text{spline}}(u_p,X_{t\times\text{sex}};df=1)\end{array}\right\} : \text{1 boosted model with automatic base-learner selection}$$

C) Equivalent non-Linear Model Base-learners (CART)

$$b_{\text{trees}}(u_\phi,X_{\text{t,sex,flood}};\text{depth}=2) + b_{\text{trees}}(u_p,X_{\text{t,sex,flood}};\text{depth}=2) : \text{1 boosted model with automatic covariate selection}$$

Figure 2: Different notation for multimodel inference of a Cormack-Jolly-Seber model, comparing fixed-effects model-averaging and boosting. **A)** Each fixed-effect model includes one term for ϕ (*left*) and one for p (*right*). $\theta(\cdot)$ is an intercept model; $\theta(t)$ has different coefficients per T capture periods (with appropriate constraints on $t=T$); $\theta(a,b)$ is a linear combination of covariate a and b on the logit scale; $\theta(a\times b)$ is an interaction effect between a and b on the logit scale. **B)** Equivalent linear base-learners (Ordinary and Penalized Least Squares from `mboost`; Bühlmann & Hothorn, 2007) with penalties to constrain their effective-*df* (ridge penalty). All terms are available in one model; selection of base-learners is by component-wise boosting. **C)** Non-linear CJS model with CART-like trees, allowing complex interactions. Selection of covariates is by the `ctree` algorithm (Hothorn et al., 2006).

2.6. Analysis: dipper example

Using CJSboost-EM, I reanalyzed the European Dipper dataset from (Lebreton et al., 1992). I compared the results to the MLEs of the fully-saturated model ($\phi(t\times\text{sex})p(t\times\text{sex})$) as well as to AICc model-averaged estimates. The dataset has 294 individuals in $T=7$ capture periods. Covariates included time, sex, and flood, similar to Section 2.5. The model-building framework was the same as in Figure 2. A 70-times bootstrap-validation was used for optimizing m_{stop}, ν_ϕ and ν_p.

Interested readers can repeat this analysis using the online tutorial at `http://github.com/faraway1nspace/HMMboost/`.

2.7. Simulation 2: high-dimensional example

The final simulation addressed the issue of high-dimensionality and the ability of CJSboost (EM) to find a sparse set of important covariates out of many spurious covariates. This is a variant of the "small n big p" problem often studied in machine learning. However, this challenge is extraordinarily difficult for capture-recapture analysis, because one must consider all combinations of covariates for different parameters (ϕ, p). In this section, I simulated 21 multi-colinear, individual-level covariates (18 continuous, three discretized) drawn from a multivariate Gaussian with marginal variances of 1. The general model can be represented as:

$$\text{logit}(\theta_{i,t}) = \beta_{\theta,0} + \underbrace{\sum_{k=1}^{21} X_{i,k}\beta_{\theta,k}}_{\text{individual effects}} + \underbrace{\sum_{\tau=2}^{T} \beta_{\theta,\tau}\mathbb{1}[\tau = t]}_{\text{capture period effect}}$$

The intercepts were drawn randomly from $\beta_{p,0} \sim \text{U}(0.4, 0.6)$ and $\beta_{\phi,0} \sim \text{U}(0.55, 0.8)$. The true models were deliberately *sparse*, such that only three covariates' coefficients (β^*_θ) were non-zero. For continuous variables, β^*_θ had a norm of 1 (on the logit scale), while the categorical-variables had norms of 3, resulting in individual marginal effects spanning $0.8-0.9$ probability-units. Time-as-a-categorical-variable was also included as a possible influential covariate. The number of individuals varied randomly from $n = 200:300$, in $T = 10$ capture periods. The simulation was repeated 30 times, each time with new covariates and coefficients.

Such a model-averaging exercise cannot be performed in MARK, because there are more than 4 trillion different fixed-effects models (excluding two-way interactions or higher). Furthermore, the AIC is known to do poorly when the simulated true model is sparse by design (Burnham & Anderson, 2004).

For each simulated dataset, the boosting analyses used 23 different PLS base-learners ($df = 2$) for all continuous and categorical covariates, and included capture-period t as a categorical variable (a.k.a, the $\theta(t)$ model). A 70-times bootstrap-validation was performed to optimize m_{stop}, ν_p, and ν_ϕ. After optimization, the performance of the fitted models were assessed by calculating 2 point-wise statistics between the true (simulated) processes and the estimates of logit(ϕ) and logit(p): i) Pearson correlation $\rho(F^{(\text{true})}_\theta, \hat{F}_\theta)$; and ii) the slope between $s(F^{(\text{true})}_\theta, \hat{F}_\theta)$ from a simple linear regression, whereby $s = 1$ suggests that the estimates are unbiased. $\hat{\rho}_\theta$ is a measure of the precision of the linear relationship between the true and fitted values, while $\hat{s}_\theta$ can be likened to angular bias.

An extra topic explored in the online tutorial, but not in this paper, was the performance of CART-like trees (see `http://github.com/faraway1nspace/HMMboost/`).

In addition to studying the precision and bias of estimates, I also demonstrate the usefulness of inclusion probabilities (the probability that a covariate is selected in the model) to infer the importance of covariates. I used the technique of stability selection from Meinshausen & Bühlmann (2010), integrated within the 70-times bootstrap-validation. Stability selection probabilities $\hat{S}$ are estimated by scoring whether a k^{th} covariate X_k is selected in a b bootstrap before m iterations, $\hat{S}^{(m)}_{\theta,k} = \frac{1}{70}\sum_{b=1}^{70} \mathbb{1}[X_k \in \mathcal{G}^{(m,b)}_\theta]$; $\hat{S}^{(m)}_{\theta,k}$ is evaluated per m, per covariate X_k and per parameter $\theta \in \{\phi, p\}$. The average over all (reasonable) regularization hyperparameters yields a Frequentist approximation to posterior inclusions probabilities, $\pi(i_{\theta,k}|\mathcal{D}) \approx \frac{1}{m_{\max}}\sum_{m=1}^{m_{\max}} S^{(m)}_{\theta,k}$ (David Draper, 2010; Murphy, 2012c). Ideally, influential covariates should have very high inclusion probabilities ($\gg 0.5$, and perhaps close to 1). Such posterior probabilities are an important means of inference about the covariates, and are more intuitive than other familiar tools for inference, like 95%CI (Hoekstra et al., 2014;

Morey et al., 2016). Also, the *stability paths* (Figure 8) can be a valuable graphical tool for interpreting the importance of covariates (Meinshausen & Bühlmann, 2010).

Stability selection can also serve in a second-stage of "hard-thresholding" to find a sparse set of truly influential covariates (Bach, 2008; Meinshausen & Bühlmann, 2010). One picks an inclusion probability threshold between $0.5-0.99$, and discards non-influential covariates below this threshold. One can proceed to "debias" the coefficients by running a final boosting model using only the selected covariates (Murphy, 2012c) and setting $m \to \infty$. Choosing an appropriate threshold is a classic trade-off between Type I errors and Power: a high threshold ≈ 1 should correctly reject the non-influential covariates (low False Discovery Rate) but may wrongly reject some of the truly influential covariates (high False Rejection Rate); a low threshold < 0.5 will result in a higher False Discovery Rate but low False Rejection Rate. Ideally, there should be a wide range of thresholds between 0.5-1 where both the FDR and FRR are close to zero.

When the FDR and FRR are zero, a procedure is called "model-selection consistent": it can correctly shrink the coefficients of non-influential covariates to zero. It is also an "oracle" if it can accurately estimate coefficients as if the true model was known in advance (Zou, 2006). The Lasso, Ridge-regression, and Boosting do *not* have these properties (Zou, 2006; Bach, 2008; Bühlmann & Hothorn, 2010): there is a pernicious trade-off between predictive-performance and model selection consistency (Zou, 2006; Meinshausen & Bühlmann, 2006; Murphy, 2012c) which has to do with one's values (Vrieze, 2012). The AIC is also not model-selection consistent (Shao, 1997; Vrieze, 2012). Instead, the AIC and Boosting are motivated by good prediction performance and minimizing the expected loss, rather than the belief in a sparse true model. Many authors laud this latter perspective, and declare sparsity to be a purely human construct that is irrelevant to natural phenomena (Burnham & Anderson, 2004; Vrieze, 2012). Philosophical notions aside, there may be a strong practical imperative in capture-recapture to favour sparser solutions than what AIC or boosting can provide, as we demonstrate with the stability paths. Towards this goal, stability selection and inclusion probabilities can transform an ℓ_1 regularizer into a model-selection consistent procedure (Bach, 2008; Meinshausen & Bühlmann, 2010). Further debiasing can give it oracle properties. Such a multi-stage procedure is no longer strictly about prediction; rather, it considers regularization as an intermediary step towards an ultimate goal to recover a true sparse model.

One caveat to using stability selection for CJSboost is that base-learners must have equal flexibility/degrees-of-freedom; otherwise, the more complex base-learners (and their constituent covariates) will have a greater probability of being selected (Kneib et al., 2009). See Section 2.4.

A final note on debiasing and convexity of the loss function: after thresholding, the final model may not have a unique MLE, such as as the $\phi(t)p(t)$ model. In such cases, one must impose constraints (such as $\phi_T = \phi_T$) before attempting to debias the results and run the gradient descent until convergence $m \to \infty$. Regularized CJSboosting does not have this problem because of early-stopping and model-selection.

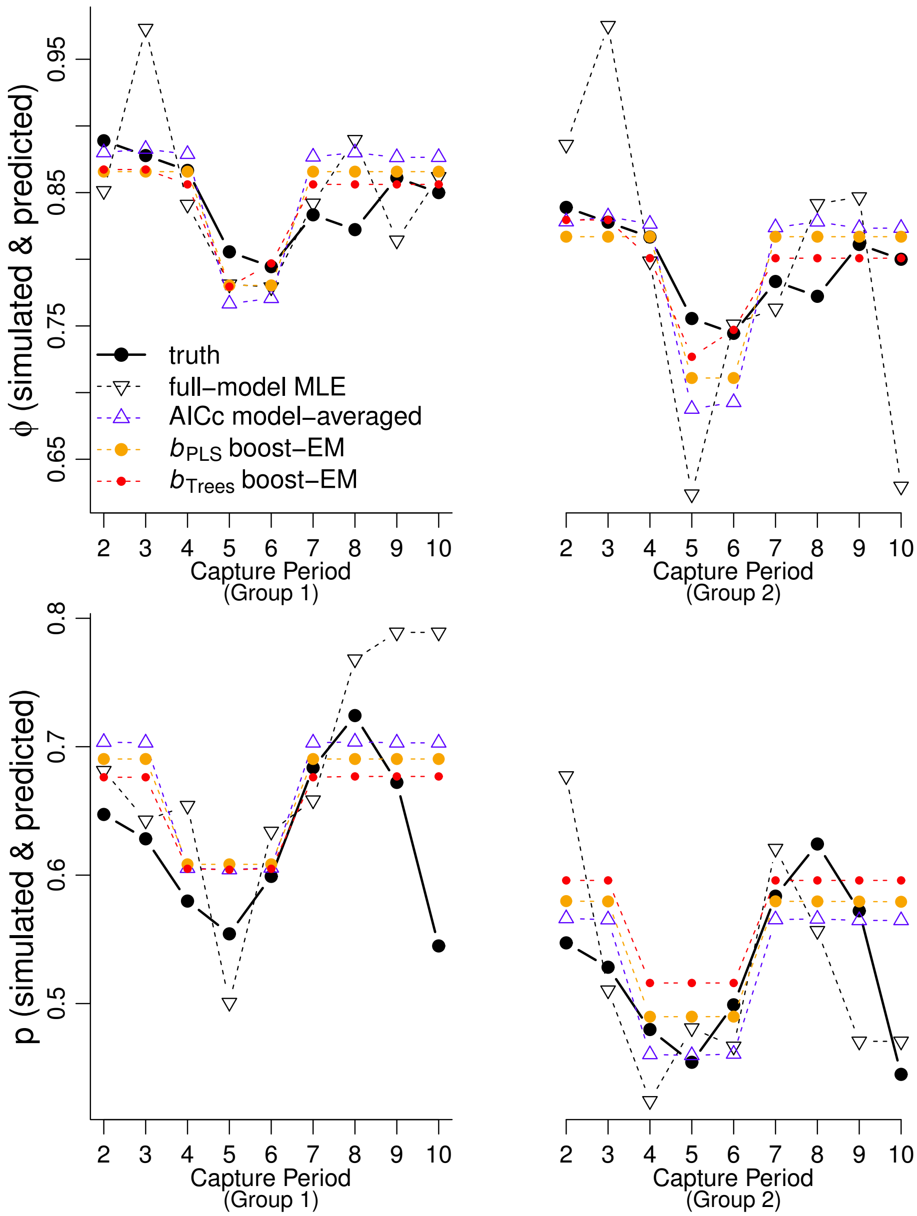

Figure 3: Simulation 1, demonstrating the CJSboost estimates from the Expectation-Maximization technique. A comparison of capture probability estimates $\hat{p}(t\times\text{sex})$ and survival estimates $\hat{\phi}(t\times\text{sex})$ from models composed of linear base-learners (OLS and PLS; in orange) and non-linear base-learners (CART-like trees; in red), as well AICc model-averaging (blue) and MLE (dashed black).

3. Results

3.1. Simulation 1: EM vs MC vs AICc vs MLE

Figure 3 compares the estimates from CJSboost-EM versus AICc model-averaging and MLEs from the full-model $\phi(t\times\text{sex})p(t\times\text{sex})$, as well as the true processes. Figure 4 does the same for the CJSboost-MC. The results can be summarized as follows:

i) The Expectation-Maximization algorithm and the Monte-Carlo algorithm yielded approximately the

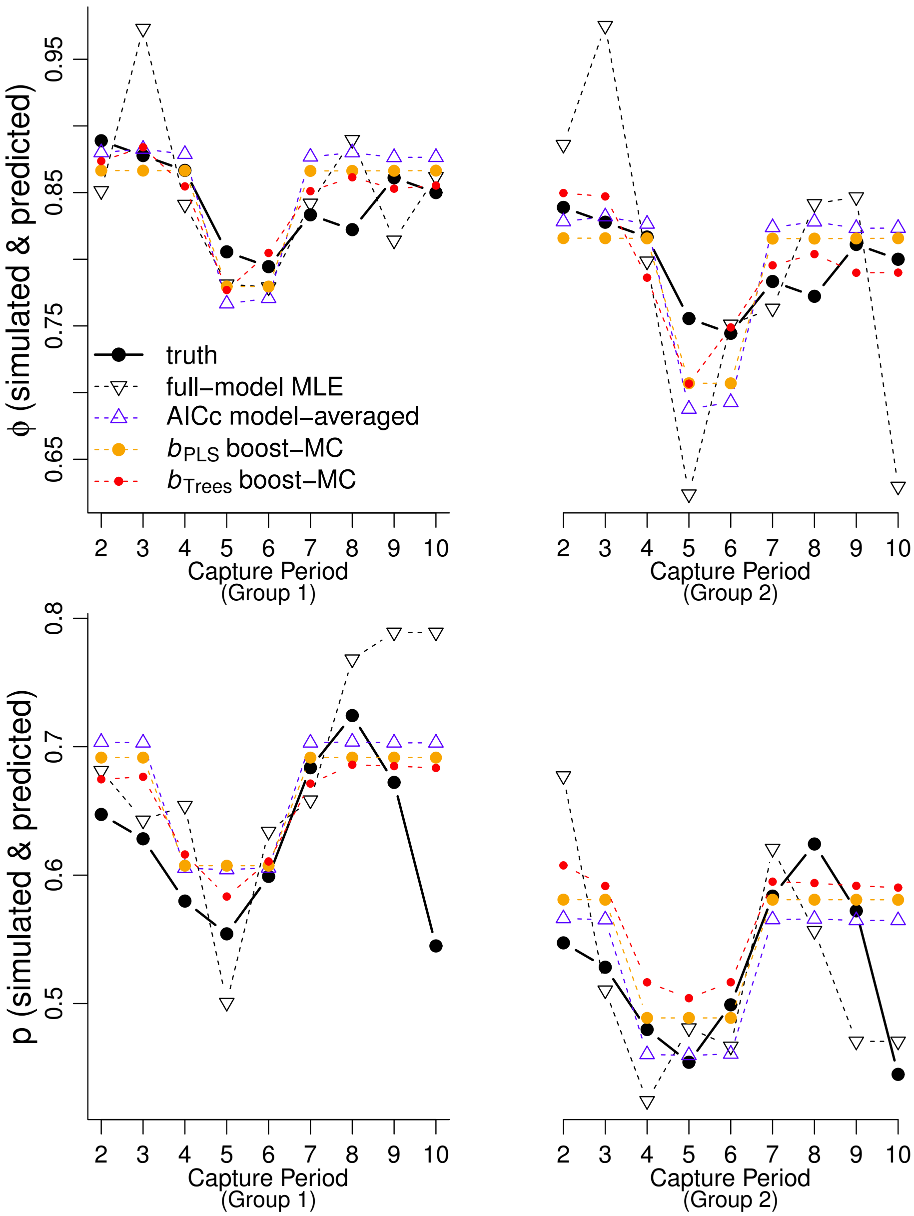

Figure 4: Simulation 1, demonstrating CJSboost estimates from the Monte-Carlo approximation technique. A comparison of capture probability estimates $\hat{p}(t{\times}\text{sex})$ and survival estimates $\hat{\phi}(t{\times}\text{sex})$ from models composed of linear base-learners (OLS and PLS; in orange) and non-linear base-learners (CART-like trees; in red), as well AICc model-averaging (blue) and MLE (dashed black).

same estimates for the linear models (b_{PLS}), but slightly different results for the non-linear CART-like base-learners (b_{Trees}).

ii) None of the four methods (MLE, AICc, b_{PLS}-boost or b_{trees}-boost) did a convincing job of approximating the true underlying processes (for both ϕ and p), although each model did uncover some aspect of the true processes.

iii) The similarities between the three predictive methods (AIC, b_{PLS}-boost, b_{trees}-boost) were thus:

(a) all three methods showed the same pattern for both for ϕ and p: low values during the flood periods

($t\!=\!4,5,6$), and a moderate sex effect (group 1 had higher values than group 2);

   (b) the b_{PLS}-boost model was most similar to AICc model-averaging;
   (c) the estimates from both boosted models were *shrunk to the mean* relative to model-averaged estimates; i.e., high model-averaged estimates were generally greater than the boosted estimates, and low model-averaged estimates were generally lower than the boosted estimates;
   (d) the non-linear b_{trees} estimates showed more shrinkage to the mean than the linear b_{PLS} estimates;

iv) The MLEs of the full-model $\hat{\phi}(t\!\times\!\text{sex})\hat{p}(t\!\times\!\text{sex})$ showed the most extremes values.

3.2. Results: Dipper example

This section shows the reanalysis of the European Dipper dataset from Lebreton et al. (1992). Figure 1 shows an example of the gradient descent of the empirical risk and the holdout-risk from the 70-times bootstrap-validation used to estimate the optimal m_{stop}. Comparisons were between the linear b_{PLS}-boost-EM model (Figure 5) and the non-linear b_{Trees}-boost-EM model (Figure 6), as well as AICc model-averaging and MLEs from the full-model $\phi(t\!\times\text{ sex})p(t\!\times\text{ sex})$. Both Figures also show the "regularization pathway" of their respective boosted model: the movement of the estimates from their initial uniform intercept model (at $m\!=\!0$) to their final values at $m\!=\!m_{\text{CV}}$, stratified by the percentage of the total reduction in the empirical risk. The results can be summarized thus:

i) For both survival ϕ and capture probability p, all three predictive methods (AICc, b_{PLS}-boost or b_{trees}-boost) were much more similar to each other than to the MLEs from the full-model.

ii) For survival, all three predictive methods yielded the same estimates: a survival probability of 0.48-0.5 during the flood years ($t\!=\!3,4$) and no sex-effect.

iii) For capture probability, the model-average estimates suggested a slight sex effect of about 1.5 probability units, whereas both boosted models shrunk the capture-probability to a constant; in contrast, the MLEs varied wildly.

iv) Regarding the regularization pathways, the linear b_{PLS}-boosted estimates converged very quickly (within 25% of the gradient decent) to their final estimates; whereas the movement of the non-linear b_{Trees}-boosted estimates moved much more gradually.

3.3. Simulation 2: high-dimensional example

Over the 30 simulations, the Pearson correlation between the true and estimated survival had the following descriptive statistics: mean of 0.979, minimum of 0.955, $Q_{0.05}$ of 0.959, and a maximum of 0.997. For capture probability, the same statistics were: 0.961, 0.708, 0.911 and 0.998. The slope statistic, a measure of bias between estimated and true survival, had the following statistics: mean of 0.778, minimum of 0.618, $Q_{0.05}$ of 0.647, and maximum of 1.018. For capture probabilities, these slope statistics were: 0.782, 0.404, 0.542, 0.962. Figure 7 shows the results of one example simulation to demonstrate the high-precision and slight-bias that is characteristic of boosting algorithms and other ℓ_1 regularizers.

Regarding the stability selection results, Figure 8 shows an example of the stability paths over m (for the same simulations shown in Figure 7). Readers can view an online animated GIF which shows the stability

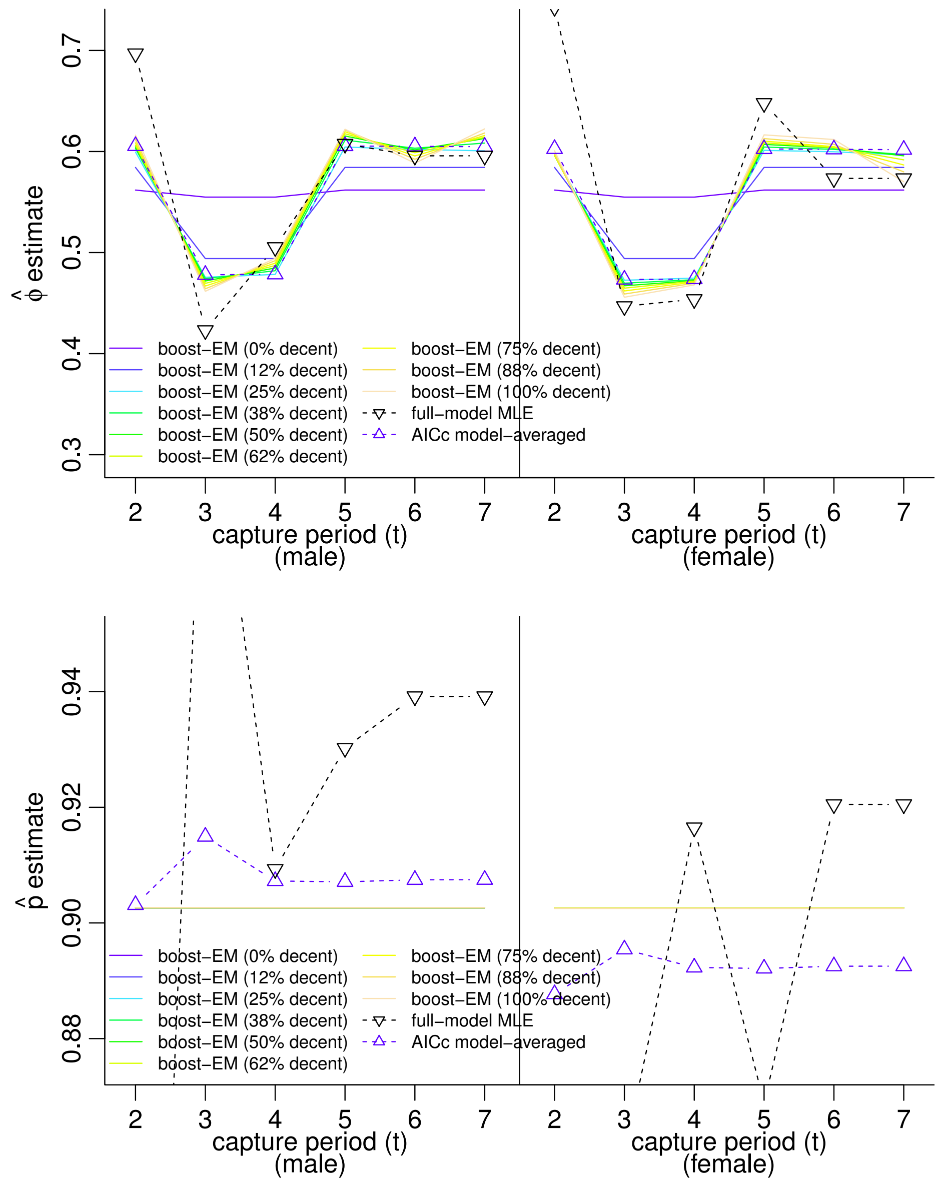

Figure 5: Dipper analysis of survival ϕ and capture probability p by CJSboost-EM using least-squares base-learners, plus comparison with AICc model-averaging and MLE $\big(\hat{\phi}(t{\times}\text{sex})\hat{p}(t{\times}\text{sex})\big)$. The regularization pathway of the estimates is visualized with the spectrum-coloured lines, starting at the intercept-only model (0% decent) and growing more complex as the gradient descent algorithm proceeds. The final estimates are achieved at 100% of the descent, when the boosting iteration m_{CV} is reached.

paths for all 30 simulations, at `http://github.com/faraway1nspace/HMMboost/` and in the Supplementary Material. The results can be summarized as:

i) The stability paths of the truly influential covariates (in *red*, Figure 8) were visually very different from the rest of the non-influential covariates (*grey*). In particular, the truly influential covariates reached high stability selection probabilities S for small values of m. For ϕ, they reach $S_{\phi}^{(m_{\text{CV}})}=1$ by the optimal m_{CV} in all simulations; while for p, 94.6% of the covariates reached $S_{p}^{(m_{\text{CV}})}=1$ by m_{CV}. On average,

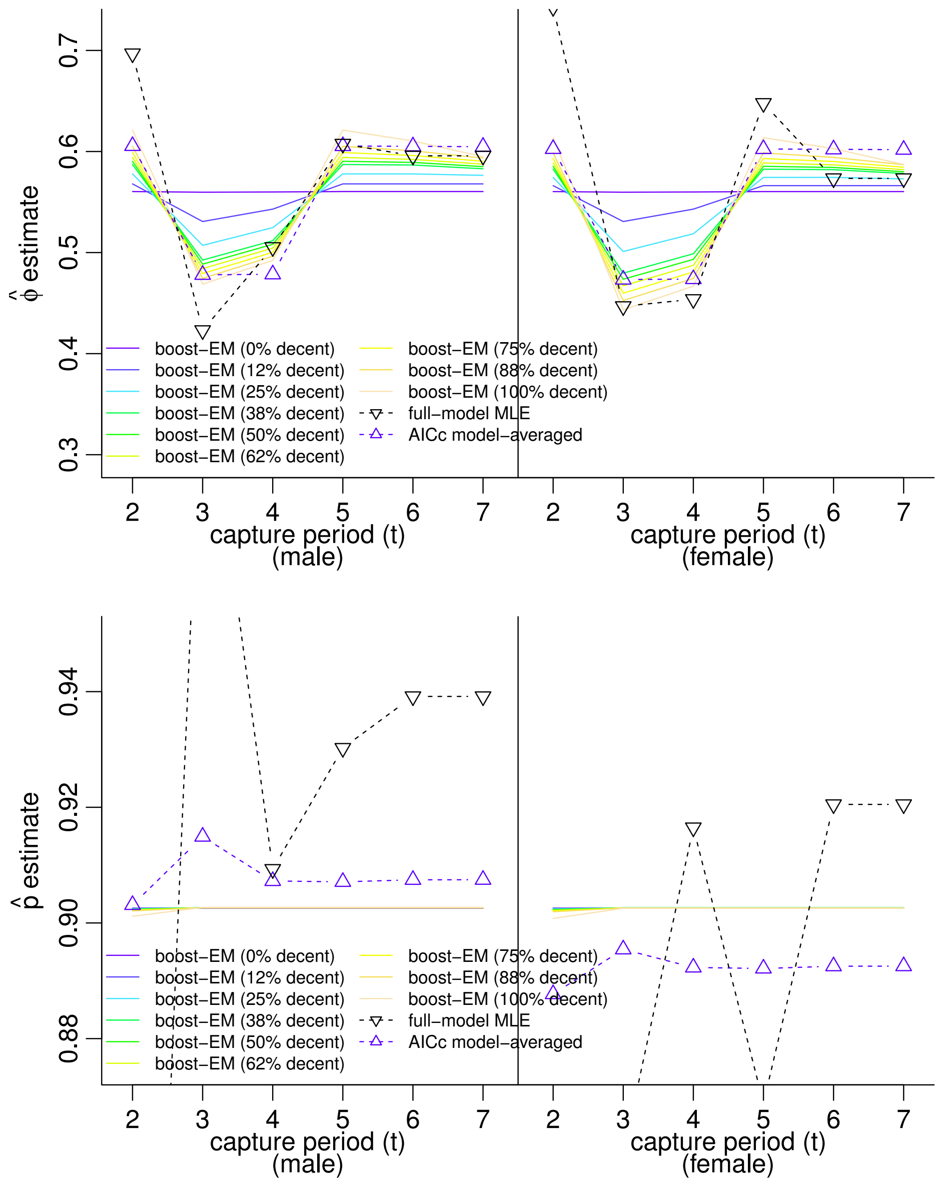

Figure 6: Dipper analysis of survival ϕ and capture probability p by CJSboost-EM using non-linear base-learners (CART-like trees), plus comparison with AICc model-averaging and MLE $\left(\hat{\phi}(t{\times}\text{sex})\hat{p}(t{\times}\text{sex})\right)$. The regularization pathway of the estimates is visualized with the spectrum-coloured lines, starting at the intercept-only model (0% decent) and growing more complex as the gradient descent algorithm proceeds. The final estimates are achieved at 100% of the descent, when the boosting iteration m_{cv} is reached.

their posterior inclusion probabilities (over all m) were 0.98 and 0.96 for ϕ and p, respectively.

ii) For the non-influential covariates, the stability selection probabilities at m_{CV} were low, $S^{(m_{\text{CV}})} \lesssim 0.5$, and rarely achieved a $S^{(m_{\text{CV}})} > 0.8$ by m_{CV}. Only 1.2% of such covariates achieved $S^{(m_{\text{CV}})} \geq 0.95$ by m_{CV}, for both ϕ and p. On average, their inclusion probabilities were 0.32 for ϕ and 0.38 for p.

iii) The stability path of the time-as-a-categorical-variable (a.k.a $\theta(t)$, in *blue*, Figure 8) showed a greater tendency for inclusion and achieved high stability selection probabilities, particularly for p. For p, it

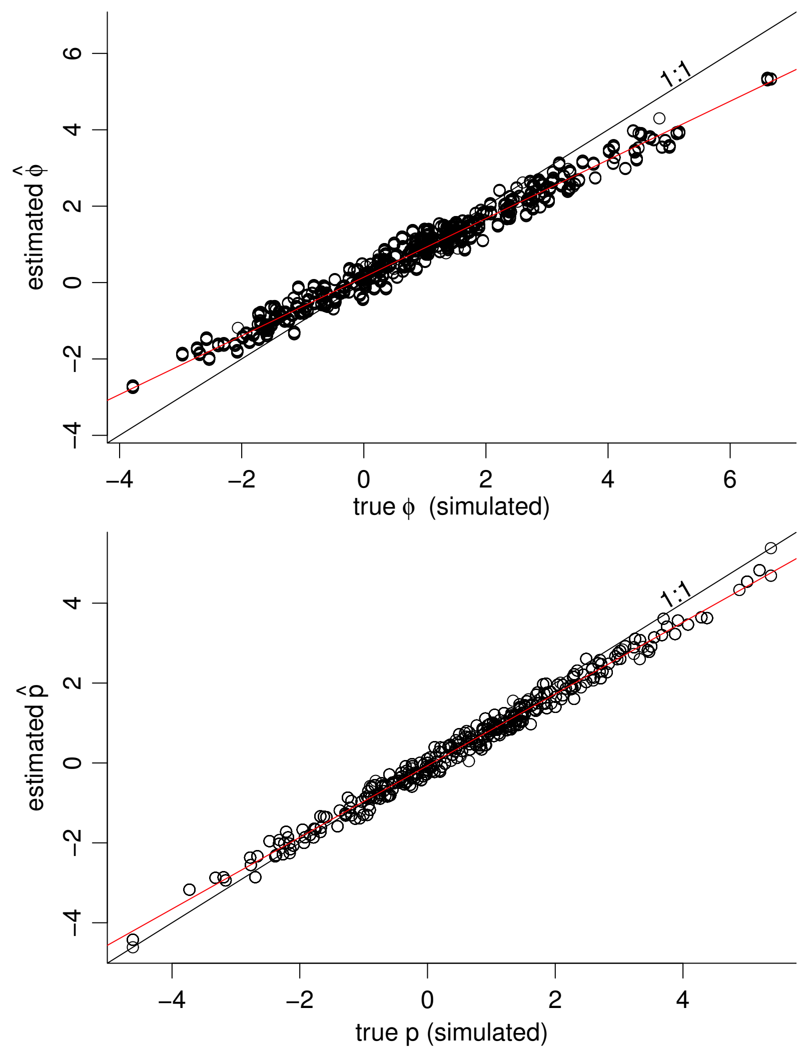

Figure 7: Comparing true vs estimated survival $\phi_{i,t}$ and capture probability $p_{i,t}$ for individuals i at capture-period t. Boosted estimates incur some downward bias (evident in the difference between the 1:1 line and the estimates' red trend-line) due to shrinkage of coefficients to the intercept-only model.

achieved $S_p^{(m_{\text{CV}})} \geq 0.95$ by m_{CV} in 60% of simulations (in which it was not truly influential). Its inclusion probabilities were 0.49 for ϕ and 0.75 for p, averaged over all simulations. This has important implications for model-selection consistency (or lack thereof). This may explain the anecdotal experience that, to have good-fitting capture-recapture models, one must usually incorporate time-varying capture-probabilities.

iv) The stability paths of covariates which were important in one parameter (like ϕ) but unimportant in the other parameter (like p) seemed to achieve higher inclusions probabilities (in *pink*, Figure 8), more so than the other non-influential covariates in grey. For p, such covariates achieved $S_p^{(m_{\text{CV}})} \geq 0.95$ in 10% of simulations, and in 3.3% of simulations for ϕ. This suggests an underlying structural correlation and may have implications for model-selection consistency.

Table 1 shows the coefficients of a prediction-optimal CJSboost model for one simulation (same as Figures

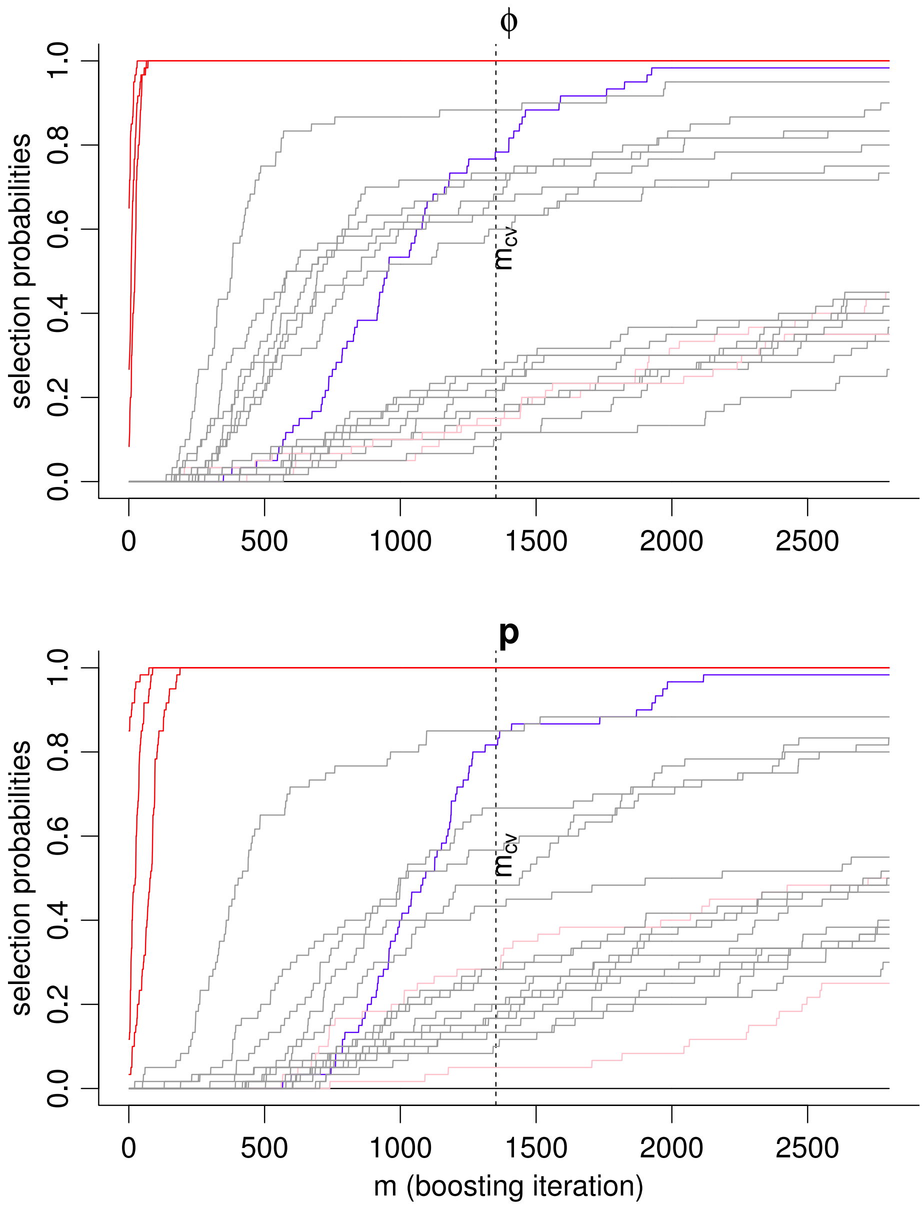

Figure 8: Demonstration of stability selection probabilities for one high-dimensional simulation. As the boosting iteration (m) gets large, regularization gets weaker, and all covariates have a higher selection probability S (estimated from a bootstrap). Lines in **red** are truly influential covariates. Lines in **gray** are non-influential covariates. Lines in **pink** are not-influential for θ, but are influential in the other parameter $\neg\theta$. Lines in **blue** represent the time-as-a-categorical-variable base-learner, a.k.a $\theta(t)$, which in this simulation was non-influential.

7 and 8). As expected, the regularized regression coefficients placed highest weight on the 6 truly influential covariates, albeit with a downward bias that is characteristic of ℓ_1 regularization (the true values were $\|\beta_k\| = 1$). The model shrunk the remaining non-influential coefficients to low values, but not to zero, incurring a False Discovery Rate of 0.34. Table 1 also demonstrates the effects of coefficient hard-thresholding using the posterior inclusion probabilities estimated in Figure 8. At higher thresholds (0.80-0.95), the model succeeds in having a FDR and FRR of zero, as well as accurate unbiased estimates of the coefficients (seemingly an

Table 1: **Estimates of coefficients from CJSboost, for one high-dimensional model-selection problem, under different degrees of hard-thresholding**

Parameter	Prediction Optimal†	Inclusion Probability Threshold‡								MLE Oracle§	SE Oracle
		0.55	0.65	0.75	0.8	0.85	0.9	0.95	0.99		
Survival Φ											
$\hat\beta_\phi$(time:1)	-0.002	-0.01	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(time:2)	-0.041	-0.238	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(time:3)	-0.036	-0.271	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(time:4)	-0.026	-0.285	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(time:5)	0.017	0.205	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(time:6)	0.006	-0.005	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(time:7)	0.015	0.124	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(time:8)	0.022	0.196	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(time:9)	0.025	0.264	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(time:10)	-0.001	-0.091	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(a)	-0.083	-0.173	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(b)	0.828	0.982	1.064	1.045	**1.067**	**1.067**	**1.067**	**1.067**	1.074	1.068	0.143
$\hat\beta_\phi$(c)	-0.021	0	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(d)	-0.761	-0.93	-0.991	-0.983	**-0.965**	**-0.965**	**-0.965**	**-0.965**	-0.919	-0.967	0.123
$\hat\beta_\phi$(e)	0.175	0.262	0.288	0.303	0	0	0	0	0	0	0
$\hat\beta_\phi$(f)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(g)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(h)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(i)	-0.051	-0.107	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(j)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(k)	-0.717	-0.838	-0.975	-0.968	**-0.953**	**-0.953**	**-0.953**	**-0.953**	-0.868	-0.955	0.119
$\hat\beta_\phi$(l)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(m)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(n)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(o)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(p)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(q)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(r)	-0.048	-0.151	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(s:1)	-0.034	-0.109	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(s:2)	0.028	0.093	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(t:1)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(t:2)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(u:1)	-0.061	-0.165	0	0	0	0	0	0	0	0	0
$\hat\beta_\phi$(u:2)	0.059	0.166	0	0	0	0	0	0	0	0	0
Capture Probability p											
$\hat\beta_p$(time:1)	0	0.002	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(time:2)	0	0.266	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(time:3)	0	-0.23	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(time:4)	0	-0.041	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(time:5)	0	-0.098	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(time:6)	0	0.159	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(time:7)	0	-0.04	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(time:8)	0	0.123	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(time:9)	0	-0.056	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(time:10)	0	-0.062	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(a)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(b)	0.942	1.129	1.149	1.184	**1.176**	**1.176**	**1.176**	**1.176**	0.846	1.178	0.144
$\hat\beta_p$(c)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(d)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(e)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(f)	-0.933	-1.142	-1.181	-1.189	**-1.186**	**-1.186**	**-1.186**	**-1.186**	-0.856	-1.189	0.135
$\hat\beta_p$(g)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(h)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(i)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(j)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(k)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(l)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(m)	0.042	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(n)	0.01	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(o)	0.81	0.993	1.033	1.047	**1.059**	**1.059**	**1.059**	**1.059**	0	1.061	0.124
$\hat\beta_p$(p)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(q)	-0.027	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(r)	-0.063	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(s:1)	-0.15	-0.202	-0.243	0	0	0	0	0	0	0	0
$\hat\beta_p$(s:2)	0.116	0.161	0.197	0	0	0	0	0	0	0	0
$\hat\beta_p$(t:1)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(t:2)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(u:1)	0	0	0	0	0	0	0	0	0	0	0
$\hat\beta_p$(u:2)	0	0	0	0	0	0	0	0	0	0	0
False Discovery Rate:	0.342	0.237	0.053	0.026	0	0	0	0	0		
False Rejection Rate:	0	0	0	0	0	0	0	0	0.167		

Bold coefficients show oracle-properties.

Covariates *a*-*r* are continuous; covariates *s*-*u* are categorical; β(time:t) is equivalent to a $\theta(t)$ sub-model.

† CJSboost-EM model with m_{stop} tuned by bootstrap-validation.

‡ Debiased CJSboost-EM model (unregularized; $m \to \infty$) after discarding covariates with inclusion probabilities below a threshold.

§ MLEs when the true model is known in advance.

"oracle", for this one simulation). The optimal threshold seems to be in the of 0.80-0.95, similar to the threshold suggested by Bach (2008) and Meinshausen & Bühlmann (2010). After "debiasing" (Murphy, 2012c; here, meaning running $m \to \infty$ after hard-thresholding), the CJSboost estimates become nearly equal to the oracle MLEs (a benchmark model run with 100% foresight about the true model). Thresholding at low values (< 0.8) and debiasing added too much weight on some non-influential covariates (i.e., no shrinkage), whereas thresholding at extremely high values (> 0.95) incurred a False Rejection.

4. Discussion

This study presents a boosted ensemble method for the Cormack-Jolly-Seber capture-recapture model, called CJSboost. I compared its estimates to AICc model-averaging. While univariate boosting is well-known in applied ecology (Elith et al., 2008; Hothorn et al., 2010; Tyne et al., 2015), the naive application of boosting for capture-recapture was not possible because of the serially-dependent nature of capture-histories. In response to this challenge, this paper presents two modifications to the Component-wise Boosting procedure, one based on Expectation-Maximization (first suggested in the appendix of Ward et al., 2009) and another based on Monte-Carlo imputation of HMM latent states. Both lead to equivalent inferences (up to an approximation error) and serve to validate each other. Code and a tutorial are available on the Github site `http://github.com/faraway1nspace/HMMboost`. The framework can be easily extended to other capture-recapture systems, thereby introducing new machine-learning techniques to capture-recapture practitioners, such as CART-like trees, splines and kernels.

The motivation for boosted capture-recapture models are many:

1. automatic variable selection and step-wise multimodel inference (without the sometimes-impossible task of fitting all possible fixed-effects models, as in AIC-based model averaging);
2. regularization and sparse estimation, which deflate the influence of unimportant covariates;
3. shrinkage of estimates away from extreme values and inadmissible values (e.g., $\phi = 1$);
4. a smoother way to address parameter non-identifiability issues, via regularization and step-wise estimation, rather than arbitrary constraints (e.g., fixing $\phi_T = \phi_{T-1}$);
5. highly extensible (see the wide variety of base-learners available under the `mboost` package, Bühlmann & Hothorn, 2007; Hofner et al., 2012);
6. inference based on predictive performance.

Through simulation and an analysis of the European Dipper dataset (Lebreton et al., 1992), this study is primarily concerned with comparisons of CJSboost to AICc model-averaging. This is not because of theoretical connections between the two (although some do exist); rather, AIC model-selection and model-averaging are the incumbent multimodel inference techniques in capture-recapture practise. It is therefore very reassuring that estimates from CJSboost and AICc model-averaging are qualitatively comparable, revealing strikingly similar patterns. This was apparent among simple least-squares base-learners as well as purely-algorithmic base-learners like CART. One distinction was that the CJSboost models were slightly more conservative and had more shrinkage on coefficients. This is desirable, especially during the current crisis of reproducibility (Simmons et al., 2011; Yaffe, 2015), because the AIC is thought to be overly permissive (Shao, 1993, 1997; Burnham & Anderson, 2004; Vrieze, 2012; Hooten & Hobbs, 2015).

Secondly, the AIC serves as a useful conceptual bridge for introducing practitioners to the notion of regularization and predictive performance. For instance, the AIC is itself a specific type of regularized objective function (fixed-penalty ℓ_0 regularizer) nested within a more general class of regularizers, within which Component-wise Boosting is generally considered a ℓ_1 regularizer (Efron et al., 2004; Bühlmann & Hothorn, 2007). The AIC also has a cross-validation interpretation (Stone, 1977; Shao, 1993, 1997). Therefore, capture-recapture practitioners, who are already (perhaps unwittingly) using predictive-performance and regularization, should expand their concept of "model parsimony" and multi-model inference to include boosting. There has been a call for ecologists to embrace algorithmic means of inference (Oppel et al., 2009), and now this is available to capture-recapture practitioners.

4.1. Inference under boosting

One potential problem of boosted capture-recapture models is the new thinking required to understand what it is and how to describe its results. With origins in machine-learning, such algorithmic inference procedures may seem incomprehensible to ecologists: they may begrudge the lack of familiar inference tools like *p*-values and 95%CI (although, these are frequently misused; Hoekstra et al., 2014; Morey et al., 2016) or AIC weights. I offer two ways to understand boosting: comparison with other regularizers, and as a type of multi-model inference optimized for prediction.

In univariate analyses, boosting has some relationships to other procedures (see Meir & Rätsch, 2003, for an overview). For linear-models with Gaussian error, component-wise boosting is generally equivalent to the Lasso (Efron et al., 2004; Bühlmann & Hothorn, 2007). The Lasso can be viewed as simultaneously optimizing a goodness-of-fit term (i.e., a loss function) and a *penalty* on model complexity (the ℓ_1-norm on regression coefficients). This form should be immediately familiar to most ecologists: the AIC also has a goodness-of-fit term and a fixed-penalty on model complexity ($-2\ell_0$-norm of regression coefficients). Hooten & Hobbs (2015) unify these ideas in a Bayesian framework: regularization is merely a strong prior disfavouring model complexity; more formally, regularized risk minimization is equivalent to Bayesian Maximum A-posteriori Probability (MAP) estimation (Murphy, 2012a), when the loss function is the negative log-likelihood. This is a helpful perspective, because inasmuch as capture-recapture practitioners are turning to Bayesian solutions under sparse data (Schofield et al., 2009; Schofield & Barker, 2011; Rankin et al., 2014, 2016), the CJSboost framework is allied and should be seriously considered. The above equivalences are more difficult to motivate using quixotic base-learners like CART-like trees, but which otherwise have great empirical performance under complex interactions and non-linear associations.

A second view of boosting is as an ensemble of many small models, like model-averaging. The terminology of a "learner" hails from its machine-learning origins, but base-learners are really just familiar analytic techniques commonly used for standalone modelling, like Ordinary Least Squares regression or CART. The influence of any one model is *weighted* according to the step-wise gradient descent procedure known as Boosting. Consider the case of Ordinary Least Squares base-learners: under extreme regularization ($m=1$), the boosted estimates are the MLE of a simple intercept model $\big(\text{e.g. } \hat{\phi}(\cdot)\hat{p}(\cdot)\big)$. At weaker regularization $m \to \infty$, the estimates tend to the MLEs of the fully-saturated model (Mayr et al., 2012). In between these extremes, at $m_{\text{stop}} = m_{\text{CV}}$, the estimates are *shrunken*, and somewhat qualitatively similar to AICc model-averaging. The size of the ensemble and its complexity is governed by predictive performance (through cross-validation or bootstrap-validation). Thus, the resulting multimodel prediction function is that which minimizes the expected loss, and is therefore constrained from over-fitting. Unsurprisingly, the estimates have a slight downward bias but are more stable across outliers and different realizations of the data (i.e. favouring low-variance in the classic "bias-variance" trade-off; Bühlmann & Yu, 2003; Murphy, 2012a).

But what can one say about "significance" or "biological importance"? The answer is the interpretation of the additive coefficients (assuming they are similarly scaled): coefficients with the largest absolute values are the most influential on survival or capture probability. Using bootstrap stability-selection, we can also use approximate posterior inclusion probabilities as a type of uncertainty statistic: covariates/base-learners with high inclusion probabilities are probably more important; covariates with low inclusion probabilities (< 0.5) are probably not that important. Probabilities lead to straight-forward inference. The stability paths (Figure 8) may also help visually discriminate between important covariates and noisy non-influential covariates, as suggested by Meinshausen & Bühlmann (2010): they notice a visual pattern whereby the true-model covariates enter the ensemble earlier and peal away from the unimportant covariates.

The above interpretations are hardly more difficult than understanding the AIC and model-averaging. In the applied ecological literature, there are few authors who formally justify a preference for the AIC versus other regularization and prediction techniques. Neither do ecologists seem to weigh in on philosophical arguments in favour of a prediction-optimal model versus a sparse model. Such matters are confused by a literature that is unclear about the underlying justification for AIC weighting and averaging (compare, for example, statements by Burnham & Anderson, 2004, vs Raftery, 1995 and Hooten & Hobbs, 2015, about AIC weights as model probabilities). Commonly, ecologists cite "model parsimony" and Kullback-Leibler divergence as a justification for the AIC. This particular view of parsimony, however, favours certain outcomes.

Burnham & Anderson (2004) offer a formal defence of the AIC and AIC model-averaging based on a notion of covariate "tapering": the view that a response variable should theoretically have many small influences, possibly infinite, and our analyses should increasingly reveal more of these minor influences as we collect more data. They argue that natural phenomena are not "sparse", unlike the systems studied by computer scientists, nor is there ever a "true model" (an oxymoron). This view is echoed by Vrieze (2012). The tapered worldview seems compelling for analyzing complex biological systems, where everything influences everything else. It is also, conveniently, the scenario in which the AIC and LOOCV are asymptotically prediction optimal and model-selection consistent (Shao, 1993, 1997; Burnham & Anderson, 2004; Vrieze, 2012).

4.2. Tapering vs sparsity

Nonetheless, I offer four arguments for capture-recapture methods to be more conservative. First, in an era of "Big Data" (geo-spatial, genetic, bio-logging, etc.) analysts increasingly have access to dozens of inventive potential covariates, many of which are different operationalizations of the same physical phenomena (e.g., consider the many ways one can measure Sea-Surface Temperature at multiple space-time scales). This Big Data deluge requires sparser discrimination among covariates, and if not, may encourage fishing for significance. Second, in an era of questionable scientific reproducibility (Simmons et al., 2011; Yaffe, 2015), we need better control on False Discoveries (among other things). This is a huge challenge, because from an optimal-prediction perspective, a False Rejection is much more costly to the expected loss than a shrunken False Discovery (Shao, 1993), thus making procedures overly liberal, including both the AIC and ℓ_1 regularizers. Third, there may be structural correlations in capture-recapture procedures that strongly favour certain outcomes, and which may preclude any hope for sparse, model-selection consistent estimates. I offer no theory to back this claim, but based on high-dimensional simulations, this study reveals high posterior inclusions probabilities for $p(t)$ models (even when it is not the true model), as well as for covariates which are significant in one component, but not the other. This is likely not a feature of CJSboost, but a more widespread capture-recapture phenomenon (see Bailey et al., 2010 and Rankin et al., 2016, for problems of partial-identifiability of parameter estimates in capture-recapture). It can be expected to be more severe under low-detection probabilities. Fourth, in the author's experience, the AIC/AICc seems to favour over-parametrized models that would be inadmissible under a Bayesian or a prediction paradigm, such as 100% survival and (the more ambiguous) 100% capture probability. Here, shrinkage on extreme values under regularization is similar to a Bayesian weak prior against boundary values.

To be clear, prediction-optimal ℓ_1 regularization, like L2boosting and the Lasso, are not very sparse, nor are they model-selection consistent (Meinshausen & Bühlmann, 2006; Zou, 2006; Bühlmann & Hothorn, 2010). They do, however, have more shrinkage on complexity than the AICc (Shao, 1997; Bühlmann & Hothorn, 2007) and AICc model-averaging, which is demonstrated in this study through simulation and an analysis of a real dataset. For more sparse model selection, the technique of bootstrapped stability selection (Meinshausen & Bühlmann, 2010; Murphy, 2012c) can be used to hard-threshold covariates which have low posterior inclusion probabilities ($\lesssim 0.8-0.95$).

4.3. Multimodel inference: build-up or post-hoc?

A boosted ensemble is built from the simplest intercept model and then "grows" more complex in a step-wise manner. This is the reverse of many multimodel inference techniques that do *post-hoc* weighting of models, such as AIC model-averaging and Bayesian model-averaging. However, the *post-hoc* approach becomes unmanageable with just a few covariates and parameters, given the combinatorial explosion in the number of plausible fixed-effect models. There is a risk that well-intentioned researchers will take short-cuts, such as a step-wise search strategy (Pérez-Jorge et al., 2016; Taylor et al., 2016), which may be susceptible to local-minima.

In conventional boosting, use of a convex loss function ensures that the gradient descent does not get stuck in a local minima. For non-convex problems, such as gamboostLSS (Mayr et al., 2012) and CJSboost, forced weakness/constraints on base-learners makes the problem more defined, but inevitably the start-values will dictate the direction of the gradient descent. However, for CJS and most capture-recaptures models, there is usually a well-defined intercept-only model that can serve as a principled way to initialize the predictions, such that if a unique MLE exists for the fully-saturated model, the boosting algorithm will reach it as $m \to \infty$. If there are parameter non-identifiability issues (such as for $\{\phi_T, p_T\}$), early stopping will ensure that the shrinkage is in the direction of the intercept-only model. Or, classic constraints can be imposed within the base-learners, such as fixing $\phi_T = \phi_{T-1}$.

4.4. Extensions and future considerations

This study is merely the first step in developing and introducing boosting for HMM and capture-recapture. Many of the properties which hold for univariate Component-wise Boosting will need theoretical and empirical validation. Many questions arise, for example, how do the selection properties vary by sample-size, especially in reference to BIC and AIC model-averaging? How sensitive are the results to low detection probabilities? Does the EM technique and/or the MC technique generalize to multi-state models? How important is tuning both hyperparameters m and ν? Does the algorithm always reach the MLE of the fully-saturated model as $m \to \infty$ and under what conditions does it fail? Is CJSboost and AICc-selection minimax optimal for mark-recapture?

By validating the boosting technique for a simple open-population model, this study paves the way for more popular capture-recapture models, such as POPAN and the PCRD, which have more model parameters in the likelihood function, like temporary-migration processes. With more parameters, the boosting algorithms will require more efficient ways of tuning hyperparameters. See Appendix B.2 for ideas in this regard.

One major benefit of the CJSboost framework is its extensibility. It can easily accommodate phenomena such as individual heterogeneity, spatial capture-recapture and cyclic-splines. These are possible because the CJSboost code is written for compatibility with the `mboost` family of R packages, and leverages their impressive variety of base-learners (Bühlmann & Hothorn, 2007; Hofner et al., 2012). For example, the `brandom` base-learner can accommodate individual random effects for addressing individual heterogeneity in a manner similar to Bayesian Hierarchical models (Rankin et al., 2016). Kernels (`brad`) and spatial splines (`bspatial`) can be used for smooth spatial effects (Kneib et al., 2009; Hothorn et al., 2010; Tyne et al., 2015) offering an entirely new framework for spatial capture-recapture. The largest advantage is that users can add these extensions via the R formula interface, rather than having to modify deep-level code. CJSboost, therefore, offers a unified framework for many types of capture-recapture ideas that would otherwise require many different analytical paradigms to study the same suite of phenomena.

5. Conclusions

1. Boosting, the regularized gradient-descent and ensemble algorithm from machine learning, can be applied to capture-recapture by reformulating the models as Hidden Markov Models, and interweaving an Expectation-Maximization E-step within each boosting iteration. An alternative boosting algorithm, based on stochastic imputation of HMM latent states, yields approximately equivalent estimates.
2. Boosting negotiates the "bias-variance" trade-off (for minimizing an expected loss) by incurring a slight bias in all coefficients, but yields estimates that are more stable to outliers and over-fitting, across multiple realizations of the data. In contrast, Maximum Likelihood estimates are unbiased, but are highly variable.
3. CJSboost allows for powerful learners, such as recursive-partitioning trees (e.g., CART) for automatic variable-selection, interaction detection, and non-linearity. This flexibility seems to come at a cost of slightly more conservative estimates (if the underlying true model is linear).
4. Both AICc model-selection and boosting are motivated by good predictive performance: minimizing an expected loss, or generalization error. When using least-squares or CART-like base-learners, the estimates from CJSboost are qualitatively similar to AICc model-averaging, but with increased shrinkage on coefficients.
5. CJSboost seems to perform very well in high-dimensional model selection problems, with an ability to recover a small set of influential covariates. Typically, there is a small and non-zero weight on some unimportant covariates (especially $p(t)$ base-learners). This pattern is consistent with the performance of univariate component-wise boosting and other ℓ_1 regularizers.
6. If the goal of a capture-recapture analysis is not prediction, but to recover a sparse "true model", then CJSboosted models can be hard-thresholded via stability-selection. Hard-thresholded CJSboost models show some promise towards model-selection consistency and oracle-properties, but there may be some structural correlations in capture-recapture likelihoods that make this generally untrue.

6. Acknowledgements

I would like to thank Professor Sayan Mukherjee for inspiration during the Duke University course Probabilistic Machine Learning and giving this project an initial "thumbs up".

7. Works Cited

Akaike, H. (1974). A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, *19*, 716 – 723. doi:10.1109/TAC.1974.1100705.

Anderson, D. R., Burnham, K. P., & Thompson, W. L. (2000). Null hypothesis testing: problems, prevalence, and an alternative. *Journal of Wildlife Management*, *64*, 912–923. doi:10.2307/3803199.

Bach, F. R. (2008). Bolasso: model consistent Lasso estimation through the bootstrap. In *Proceedings of the 25th International Conference on Machine Learning* ICML '08 (pp. 33–40). New York, NY, USA: ACM. doi:10.1145/1390156.1390161.

Bailey, L. L., Converse, S. J., & Kendall, W. L. (2010). Bias, precision, and parameter redundancy in complex multistate models with unobservable states. *Ecology*, *91*, 1598–1604. doi:10.1890/09-1633.1.

Breiman, L. (1998). Arcing classifier (with discussion and a rejoinder by the author). *The Annals of Statistics*, *26*, 801–849. doi:10.1214/aos/1024691079.

Breiman, L. (1999). Prediction games and Arcing algorithms. *Neural Computation*, *11*, 1493–1517. doi:10.1162/089976699300016106.

Buckland, S. T., Burnham, K. P., & Augustin, N. H. (1997). Model selection: an integral part of inference. *Biometrics*, *53*, 603–618. doi:10.2307/2533961.

Burnham, K., & Anderson, D. (2014). P values are only an index to evidence: 20th-vs. 21st-century statistical science. *Ecology*, *95*, 627–630. doi:10.1890/13-1066.1.

Burnham, K. P., & Anderson, D. R. (2004). Multimodel inference: understanding AIC and BIC in model selection. *Sociological Methods & Research*, *33*, 261–304. doi:10.1177/0049124104268644.

Burnham, K. P., & Overton, W. S. (1978). Estimation of the size of a closed population when capture probabilities vary among animals. *Biometrika*, *65*, 625 – 633. doi:10.1093/biomet/65.3.625.

Bühlmann, P., & Hothorn, T. (2007). Boosting algorithms: regularization, prediction and model fitting. *Statistical Science*, *22*, 477–505. doi:10.1214/07-STS242.

Bühlmann, P., & Hothorn, T. (2010). Twin Boosting: improved feature selection and prediction. *Statistics and Computing*, *20*, 119–138. doi:10.1007/s11222-009-9148-5.

Bühlmann, P., & Yu, B. (2003). Boosting with the L2 loss: regression and classification. *Journal of the American Statistical Association*, *98*, 324–339. doi:10.1198/016214503000125.

Carothers, A. (1973). The effects of unequal catchability on Jolly-Seber estimates. *Biometrics*, *29*, 79–100. doi:10.2307/2529678.

David Draper (2010). Discussion: Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *72*, 461–462. doi:10.1111/j.1467-9868.2010.00740.x.

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, *32*, 407–499. doi:10.1214/009053604000000067.

Elith, J., Leathwick, J. R., & Hastie, T. (2008). A working guide to boosted regression trees. *The Journal of Animal Ecology*, *77*, 802–813. doi:10.1111/j.1365-2656.2008.01390.x.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion). *The Annals of Statistics*, *28*, 337–374. doi:10.1214/aos/1016218223.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, *29*, 1189–1232.

Hand, D. J., & Vinciotti, V. (2003). Local versus global models for classification problems: fitting models where it matters. *The American Statistician*, *57*, 124–131. doi:10.1198/0003130031423.

Hoekstra, R., Morey, R. D., Rouder, J. N., & Wagenmakers, E.-J. (2014). Robust misinterpretation of confidence intervals. *Psychonomic Bulletin & Review*, *21*, 1157–1164. doi:10.3758/s13423-013-0572-3.

Hofner, B., Kneib, T., & Hothorn, T. (2014). A unified framework of constrained regression. *Statistics and Computing*, *26*, 1–14. doi:10.1007/s11222-014-9520-y.

Hofner, B., Mayr, A., Robinzonov, N., & Schmid, M. (2012). *Model-based Boosting in R: A Hands-on Tutorial Using the R Package mboost*. Technical Report 120 Department of Statistics, Ludwig-Maximilians-Universität Munich. URL: http://epub.ub.uni-muenchen.de/12754/.

Hooten, M., & Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists. *Ecological Monographs*, *85*, 3–28. doi:10.1890/14-0661.1.

Hothorn, T., Hornik, K., & Zeileis, A. (2006). Unbiased recursive partitioning: a conditional inference framework. *Journal of Computational and Graphical Statistics*, *15*, 651–674. doi:10.1198/106186006X133933.

Hothorn, T., Müller, J., Schröder, B., Kneib, T., & Brandl, R. (2010). Decomposing environmental, spatial, and spatiotemporal components of species distributions. *Ecological Monographs*, *81*, 329–347. doi:10.1890/10-0602.1.

Hunt, T., Bejder, L., Allen, S. J., Rankin, R. W., Hanf, D., & Parra, G. J. (2016). Demographic characteristics of Australian humpback dolphins reveal important habitat toward the south-western limit of their range. *Endangered Species Research*, *in review*.

Hutchinson, R., Liu, L., & Dietterich, T. (2011). Incorporating boosted regression trees into ecological latent variable models. In W. Burgard, & D. Roth (Eds.), *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence* (pp. 1343–1348). Association for the Advancement of Artificial Intelligence.

Johnson, J. B., & Omland, K. S. (2004). Model selection in ecology and evolution. *Trends in Ecology & Evolution*, *19*, 101–108. doi:10.1016/j.tree.2003.10.013.

Kearns, M., & Valiant, L. (1994). Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM*, *41*, 67–95. doi:10.1145/174644.174647.

Kneib, T., Hothorn, T., & Tutz, G. (2009). Variable selection and model choice in geoadditive regression models. *Biometrics*, *65*, 626–634. doi:10.1111/j.1541-0420.2008.01112.x.

Laake, J. L. (2013). *RMark: An R Interface for Analysis of Capture-Recapture Data with MARK*. AFSC Processed Report 2013-01 Alaska Fisheries Science Center, NOAA, National Marine Fisheries Service Seattle, WA, USA.

Lebreton, J.-D., Burnham, K. P., Clobert, J., & Anderson, D. R. (1992). Modeling survival and testing biological hypotheses using marked animals: a unified approach with case studies. *Ecological Monographs*, *62*, 67–118. doi:10.2307/2937171.

Mayr, A., Fenske, N., Hofner, B., Kneib, T., & Schmid, M. (2012). Generalized additive models for location, scale and shape for high dimensional data—a flexible approach based on boosting. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *61*, 403–427. doi:10.1111/j.1467-9876.2011.01033.x.

Meinshausen, N., & Bühlmann, P. (2006). High-dimensional graphs and variable selection with the Lasso. *The Annals of Statistics*, *34*, 1436–1462. doi:10.1214/009053606000000281.

Meinshausen, N., & Bühlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *72*, 417–473. doi:10.1111/j.1467-9868.2010.00740.x.

Meir, R., & Rätsch, G. (2003). An introduction to boosting and leveraging. *Lecture Notes in Computer Science*, *2600*, 118–183. doi:10.1007/3-540-36434-X_4.

Morey, R. D., Hoekstra, R., Rouder, J. N., Lee, M. D., & Wagenmakers, E.-J. (2016). The fallacy of placing confidence in confidence intervals. *Psychonomic Bulletin & Review*, *23*, 103–123. doi:10.3758/s13423-015-0947-8.

Mukherjee, S., Rifkin, R., & Poggio, T. (2003). Regression and Classification with Regularization. In D. D. Denison, M. H. Hansen, C. C. Holmes, B. Mallick, & B. Yu (Eds.), *Nonlinear estimation and classification* Lecture Notes in Statistics (pp. 111–128). New York, NY, USA: Springer.

Murphy, K. P. (2012a). Frequentist Statistics. In *Machine Learning: A Probabilistic Approach* Adaptive Computation and Machine Learning Series (pp. 191–216). Cambridge, MA, USA: MIT Press.

Murphy, K. P. (2012b). Markov and hidden Markov models. In *Machine Learning: A Probabilistic Approach* Adaptive Computation and Machine Learning Series (pp. 589–630). Cambridge, MA, USA: MIT Press.

Murphy, K. P. (2012c). Sparse Linear Models. In *Machine Learning: A Probabilistic Approach* Adaptive Computation and Machine Learning Series (pp. 421–478). Cambridge, MA, USA: MIT Press.

Oppel, S., Strobl, C., & Huettmann, F. (2009). *Alternative methods to quantify variable importance in ecology*. Technical Report 65 Department of Statistics, Ludwig-Maximilians-Universität Munich.

Pérez-Jorge, S., Gomes, I., Hayes, K., Corti, G., Louzao, M., Genovart, M., & Oro, D. (2016). Effects of nature-based tourism and environmental drivers on the demography of a small dolphin population. *Biological Conservation*, *197*, 200–208. doi:10.1016/j.biocon.2016.03.006.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*, 257–286. doi:10.1109/5.18626.

Raftery, A. E. (1995). Bayesian model selection in social research. *Sociological methodology*, *25*, 111–164.

Rankin, R. W., Maldini, D., & Kaufman, G. D. (2014). Bayesian estimate of Australian humpback whale calving intervals under sparse resighting rates: 1985 – 2009. *Journal of Cetacean Research and Management*, *13*, 109–121.

Rankin, R. W., Nicholson, K. E., Allen, S. J., Krützen, M., Bejder, L., & Pollock, K. H. (2016). A full-capture Hierarchical Bayesian model of Pollock's Closed Robust Design and application to dolphins. *Frontiers in Marine Science*, *3*. doi:10.3389/fmars.2016.00025.

Robinzonov, N. (2013). *Advances in boosting of temporal and spatial models*. Doctoral Thesis LMU München: Fakultät für Mathematik, Informatik und Statistik Munich. URL: http://nbn-resolving.de/urn:nbn:de:bvb:19-153382.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, *5*, 197–227. doi:10.1023/A:1022648800760.

Schmid, M., & Hothorn, T. (2008a). Boosting additive models using component-wise P-Splines. *Computational Statistics & Data Analysis*, *53*, 298–311. doi:10.1016/j.csda.2008.09.009.

Schmid, M., & Hothorn, T. (2008b). Flexible boosting of accelerated failure time models. *BMC Bioinformatics*, *9*, 269. doi:10.1186/1471-2105-9-269.

Schmid, M., Potapov, S., Pfahlberg, A., & Hothorn, T. (2010). Estimation and regularization techniques for regression models with multidimensional prediction functions. *Statistics and Computing*, *20*, 139–150. doi:10.1007/s11222-009-9162-7.

Schmid, M., Wickler, F., Maloney, K. O., Mitchell, R., Fenske, N., & Mayr, A. (2013). Boosted Beta Regression. *PLoS ONE*, *8*. doi:10.1371/journal.pone.0061623.

Schofield, M. R., & Barker, R. J. (2011). Full open population capture–recapture models with individual covariates. *Journal of Agricultural, Biological, and Environmental Statistics*, *16*, 253–268. doi:10.1007/s13253-010-0052-4.

Schofield, M. R., Barker, R. J., & Mackenzie, D. I. (2009). Flexible hierarchical mark-recapture modeling for open populations using WinBUGS. *Environmental and Ecological Statistics*, *16*, 369–387. doi:10.1007/s10651-007-0069-1.

Shao, J. (1993). Linear Model selection by cross-validation. *Journal of the American Statistical Association*, *88*, 486–494. doi:10.1080/01621459.1993.10476299.

Shao, J. (1997). An asymptotic theory for linear model selection. *Statistica Sinica*, *7*, 221–242.

Simmons, J. P., Nelson, L. D., & Simonsohn, U. (2011). False-positive psychology undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological Science*, *22*, 1359–1366. doi:10.1177/0956797611417632.

Stone, M. (1977). An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39*, 44–47.

Taylor, A. R., Schacke, J. H., Speakman, T. R., Castleberry, S. B., & Chandler, R. B. (2016). Factors related to common bottlenose dolphin (Tursiops truncatus) seasonal migration along South Carolina and Georgia coasts, USA. *Animal Migration*, *3*. doi:10.1515/ami-2016-0002.

Tibshirani, R. (2011). Regression shrinkage and selection via the lasso: a retrospective: Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *73*, 273–282. doi:10.1111/j.1467-9868.2011.00771.x.

Tyne, J. A., Johnston, D. W., Rankin, R., Loneragan, N. R., & Bejder, L. (2015). The importance of spinner dolphin (*Stenella longirostris*) resting habitat: implications for management. *Journal of Applied Ecology*, *52*, 621–630. doi:10.1111/1365-2664.12434.

Vrieze, S. I. (2012). Model selection and psychological theory: A discussion of the differences between the Akaike information criterion (AIC) and the Bayesian information criterion (BIC). *Psychological Methods*, *17*, 228–243. doi:10.1037/a0027127.

Ward, G., Hastie, T., Barry, S., Elith, J., & Leathwick, J. R. (2009). Presence-only data and the EM algorithm. *Biometrics*, *65*, 554–563. doi:10.1111/j.1541-0420.2008.01116.x.

White, G. C., & Burnham, K. P. (1999). Program MARK: survival estimation from populations of marked animals. *Bird Study*, *46*, S120–S139. doi:10.1080/00063659909477239.

Yaffe, M. B. (2015). Reproducibility in science. *Science Signaling*, *8*, EG5. doi:10.1126/scisignal.aaa5764.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, *101*, 1418–1429. doi:10.1198/016214506000000735.

APPENDICES

Appendix A. Algorithms for Filtering and Sampling HMM Latent States

The CJSboost algorithm depends on conditional independence of data pairs $(y_{i,t}, X_{i,t})$ for individuals i in capture period t, in order to estimate the negative-gradient in the descent algorithm. This is possible if we impute information about the latent state sequences z for pairs of capture periods at t and $t-1$. The two CJSboost algorithms, CJSboost-EM and CJSboost-MC, achieve this same idea with two different, but related, techniques. In both cases, we will use a classic "forwards-backwards" messaging algorithm to gain information about the probability distribution of the latent state sequences. In CJSboost-EM, we calculate the *two-slice marginal probabilities* $p(z_{t-1}=u, z_t=v|\mathbf{y}_{1:T}, \phi, p)$, per boosting iteration; in CJSboost-MC, we will *sample* $\mathbf{z}$ from its posterior distribution $\pi(\mathbf{z}_{1:T}|\mathbf{y}_{1:T}, \phi, p)$. See Rabiner (1989) and Murphy (2012b) for accessible tutorials.

Both algorithms use a forwards algorithm and backwards algorithm. We will drop the indices i, and focus on the capture history of a single individual. $\mathbf{y}$ is the time-series of binary outcomes of length T. $\mathbf{z}$ is a vector of latent states $z \in \{\text{dead}, \text{alive}\}$. We condition on an individual's first capture at time $t = t^0$, and are only concerned with the sequence $\mathbf{z}_{t^0:T}$. Survival from step $t-1$ to t is ϕ_t. Conditional on z_t, the capture probabilities are $p(y_t=1|\text{alive}) = p_t$, and $p(y_t=1|\text{dead}) = 0$. In HMM notation, the CJS processes can be presented as the following column-stochastic matrices:

$$\boldsymbol{\Phi}_t = \begin{array}{c} \\ \text{dead} \\ \text{alive} \end{array} \begin{array}{c} \begin{array}{cc} \text{dead} & \text{alive} \end{array} \\ \begin{pmatrix} 1 & 1-\phi_t \\ 0 & \phi_t \end{pmatrix} \end{array} \quad \boldsymbol{\Psi}_t = \begin{array}{c} \\ \text{no capture} \\ \text{capture} \end{array} \begin{array}{c} \begin{array}{cc} \text{dead} & \text{alive} \end{array} \\ \begin{pmatrix} 1 & 1-p_t \\ 0 & p_t \end{pmatrix} \end{array} \tag{A.1}$$

In HMM parlance, Φ is the Markovian transition process; we denote the probability $p(z_t=u|z_{t-1}=u)$ as $\Phi_t(u, v)$. Ψ is the emission process governing capture probabilities; we denote the probability $p(y_t=1|z_t=v)$ as $\Psi_t(v)$.

Appendix A.1. Forward-algorithm

The forward messaging algorithm involves the recursive calculation of $\alpha_t(v)$, per time t and state $z_t=v$. α_t is the *filtered belief state* of z_t given all the observed information in $\mathbf{y}$ from first capture t^0 until t. Notice, that for clarity, we drop the notation for conditioning on ϕ and p, but these are always implied.

$$\begin{aligned} a_t(v) &:= p(z_t=v|\mathbf{y}_{t^0:t}) \\ &= \frac{1}{\mathcal{Z}_t} p(y_t|z_t=v) p(z_t=v|\mathbf{y}_{t^0:\,t-1}) \\ &= \frac{1}{\mathcal{Z}_t} p(y_t|z_t=v) \sum_u p(z_t=v|z_{t-1}=u) p(z_{t-1}=u|\mathbf{y}_{t^0:\,t-1}) \\ &= \frac{1}{\mathcal{Z}_t} \Psi_t(v) \sum_u \Phi(u,v) \alpha_{t-1}(u) \end{aligned} \tag{A.2}$$

$$\mathcal{Z}_t = \sum_v \left(\Psi_t(v) \sum_u \Phi(u,v) \alpha_{t-1}(u) \right), \; \sum_v \alpha_t(v) = 1$$

The algorithm is initialized at time t^0 (an individual's first capture) with $\alpha_{t^0}(\text{alive}) = 1$. Conditional on the values of $\alpha_t(v)$ for all v, one can proceed to calculate the next values of $\alpha_{t+1}(v)$, and so on, until $t=T$.

Appendix A.2. Backwards-algorithm

Messages are passed backwards in a recursive algorithm starting at $t = T$ and moving backwards until $t = t^0$, the first-capture period, while updating entries in $\beta_t(v)$.

$$\begin{aligned}
\beta_{t-1}(u) &:= p(\mathbf{y}_{t:T}|z_{t-1} = u) \\
&= \sum_v p(\mathbf{y}_{t+1:T}|z_t = v)p(y_t|z_t = v)p(z_t = v|z_{t-1} = u) \\
&= \sum_v \beta_t(v)\Psi_t(v)\Phi_t(u, v)
\end{aligned} \tag{A.3}$$

The algorithm is initialized $\beta_T(\cdot) = 1$ for all states v (notice that the entries do not need to sum to 1). Having calculated the backwards and forwards messages, we can now proceed to characterize the latent state distributions.

Appendix A.3. Two-slice marginal probabilities for Expectation-Maximization

Expectation-Maximization is an iterative technique for maximizing a difficult objective function by working with an easy "complete-data" objective function $\log p(y, z|\theta)$. EM works by cycling through an M-step and an E-step. In boosting-EM, the M-step corresponds to the usual update of the prediction vectors $F_\theta^{(m)} = F_\theta^{(m-1)} + \nu_\theta \hat{f}$ (conditional on z), and are used to estimate $\hat{\theta}$. The E-step imputes expectations of the latent states z, conditional on the data and current estimates of $\hat{\theta}^{(m)}$.

In the CJSboost-EM algorithm, we require expectations for the joint states (z_{t-1}, z_t). We substitute in the two-slice marginal probabilities $p(z_{t-1}, z_t|\mathbf{y}_{t^0:T}, \phi, p)$. These can be easily evaluated for a capture history $\mathbf{y}_i$ using the outputs (α, β) from the forward-backwards algorithm.

$$\begin{aligned}
w_t(u, v) &:= p(z_{t-1} = u, z_t = v|\mathbf{y}_{t^0:T}) \\
&= \frac{1}{\xi_t} p(z_{t-1}|\mathbf{y}_{t^0:t-1})p(z_t|z_{t-1}, \mathbf{y}_{t:T}) \\
&= \frac{1}{\xi_t} p(z_{t-1}|\mathbf{y}_{t^0:t-1})p(y_t|z_t)p(\mathbf{y}_{t+1:T}|z_t)p(z_t|z_{t-1}) \\
&= \frac{1}{\xi_t} \alpha_{t-1}(u)\Psi_t(v)\beta_t(v)\Phi_t(u, v)
\end{aligned} \tag{A.4}$$

$$\xi_t = \sum_u \sum_v \alpha_{t-1}(u)\Psi_t(v)\beta_t(v)\Phi_t(u, v), \; \sum_u \sum_v w_t(u, v) = 1$$

The E-step is completed after evaluating the set $\{w_{i,t}(\text{alive}, \text{alive}), w_{i,t}(\text{alive}, \text{dead}), w_{i,t}(\text{dead}, \text{dead})\}$, for each capture period $t > t_i^0$ and for each individual capture history $\{\mathbf{y}_i\}_{i=1}^n$. This is an expensive operation; computational time can be saved by re-evaluating the expectations every second or third boosting iteration m, which, for large $m_{\text{stop}} > 100$ and small ν, will have a negligible approximation error.

Appendix A.4. Sampling state-sequences from their posterior

For the CJSboost Monte-Carlo algorithm, we sample a latent state sequence $\mathbf{z}_i$ from the posterior $\pi(\mathbf{z}_{1:T}|\mathbf{y}_{1:T}, \phi, p)$, for each individual i per boosting step. Conditional on the latent states, the negative-gradients are easily evaluated and we can proceed to boost the estimates and descend the risk gradient. However, because the algorithm is stochastic, we must avoid getting trapped in a local minima by sampling many sequences (e.g., $S \approx 10-20$), thereby approximating the full posterior distribution of $\mathbf{z}$. Over all S samples, the average gradient will *probably* be in the direction of the global minima. For large m and small ν, the approximation error is small.

The algorithm uses backwards-sampling of the posterior under the chain rule:

$$p(\mathbf{z}_{t^0:T}|\mathbf{y}_{t^0:T}) = p(z_T|\mathbf{y}_{t^0:T}) \prod_{t=T-1}^{t^0} p(z_t|z_{t+1}, \mathbf{y}_{t^0:T}) \tag{A.5}$$

We start with a draw at time $t = T$, $z_T^{(s)} \sim p(z_T = v|\mathbf{y}_{t^0:T}) = \alpha_T(v)$, and condition earlier states on knowing the next-step-ahead state, proceeding backwards until $t = t^0$.

$$\begin{aligned}
z_t^{(s)} &\sim p(z_t = u|z_{t+1} = v, \mathbf{y}_{t^0:\,t}) \\
&= \frac{p(z_t, z_{t+1}|\mathbf{y}_{t^0:\,t+1})}{p(z_{t+1}|\mathbf{y}_{t^0:\,t+1})} \\
&\propto \frac{p(y_{t+1}|z_{t+1})p(z_t, z_{t+1}|\mathbf{y}_{t^0:\,t})}{p(z_{t+1}|\mathbf{y}_{t^0:\,t+1})} \\
&= \frac{p(y_{t+1}|z_{t+1})p(z_{t+1}|z_t)p(z_t|\mathbf{y}_{t^0:\,t})}{p(z_{t+1}|\mathbf{y}_{t^0:\,t+1})} \\
&= \frac{\Psi_{t+1}(v)\Phi_{t+1}(u, v)\alpha_t(u)}{\alpha_{t+1}(v)}
\end{aligned} \tag{A.6}$$

Thus, knowing α, β, Φ and Ψ, we can easily generate random samples of $\mathbf{z}$ that are drawn from its posterior distribution. The backwards sampling step is repeated for each $t > t_i^0$ capture period, for each s sequence, for each i capture history, for each m boosting iteration.

Appendix B. Tuning Hyperparameters m and ν

This section will present a simple work-flow for finding approximately optimal values of m_{stop}, ν_ϕ and ν_p. Our objective is to minimize the expected loss $\mathcal{L}$, or generalization error. We estimate $\mathcal{L}$ through B-times bootstrap-validation. For each b bootstrap, we create a CJSboost prediction function, $G^{(b)}(X; m, \nu_\phi, \nu_p)$ which is trained on the bootstrapped data and is a function of the hyperparameters ν_ϕ, ν_p and m. We calculate the holdout-out risk using the out-of-bootstrap b^c capture-histories and covariate data, $(\mathbf{Y}^{(b^c)}, \mathbf{X}^{(b^c)})$. The average hold-out risk over B bootstraps, L_{cv}, is our objective to minimize.

$$\mathcal{L} \approx L_{cv} = \underset{m,\nu_\phi,\nu_p}{\operatorname{argmin}} \frac{1}{B} \sum_{b=1}^{B} L\left(\mathbf{Y}^{(b^c)}, G^{(b)}(\mathbf{X}^{(b^c)}; m, \nu_\phi, \nu_p)\right)$$

For a given ν_ϕ and ν_p, the hold-out risk can be monitored internally to the boosting algorithm for each step m. Therefore, a single B-bootstrap run is all that is necessary to find the optimal m, given ν_ϕ and ν_p. But since ν_ϕ and ν_p are continuous, one must discretize the range of possible values and re-run separate B-bootstrap-validation exercises per combination of ν_ϕ and ν_p. This is very expensive, and one must accept some approximation error.

Appendix B.1. Algorithm 1 for tuning ν

For just two parameters, the pertinent quantity to optimize is the ratio $\lambda = \frac{\nu_p}{\nu_\phi}$, for a fixed mean $\nu_\mu = \frac{1}{2}(\nu_\phi + \nu_p)$. Therefore, a univariate discrete set of $\Lambda = \left\{\lambda^{(1)}, \lambda^{(2)}, ..., \lambda^{(J)}\right\}$ can be searched for the smallest $L_{\text{cv}}(\lambda)$.

This is less daunting than it may seem, because the range of λ is practically bounded. For example, if $m_{\text{stop}} = 1000$ and $\lambda = 100$, ϕ is effectively shrunk to its intercept starting value, and higher values of λ have little effect. Also, Bühlmann & Yu (2003) suggest that the generalization error has a very shallow minima around the optimal values of m, which means that our hyperparameters need only get within the vicinity of their optimal values, rather than strict numerical convergence. Finally, $L_{\text{cv}}(\lambda)$ is typically convex for varying λ (so long as the same bootstrap-weights are recycled for all new estimates of $L_{\text{cv}}(\lambda)$). Therefore, we can employ any convex optimization algorithm for non-differentiable functions to iteratively search for the optimal λ. The thrust of any such algorithm is a multiplicative "stepping-out" procedure to quickly find the correct order of magnitude for λ. For example, starting a $\lambda^{(0)}=1$, we need only 7 doubling steps to grow λ to $128 \times \lambda^{(0)}$; further refinements will have little practical impact on the final model estimates.

An example algorithm is the following.

1. set $\nu_\mu = 0.01$ and $\lambda^{(0)} = 1$; generate the B bootstrap samples; initialize the set $\Lambda = \{\lambda^{(0)}, \frac{1}{2}\lambda^{(0)}\}$;
2. for each λ in Λ, estimate $L_{cv}(\lambda)$ and store the values in the list $\text{Ł} = \{L^{(0)}, ...\}$;
3. for j in $1\!:\!J$, do:
   (a) get the current best value for the ratio $\lambda_{\text{min}} = \underset{\lambda\in\Lambda}{\text{argmin}} L_{cv}(\lambda)$
   (b) estimate a new candidate λ^*:
   if $\lambda_{\text{min}} = \min(\Lambda)$, then $\lambda^* = \frac{1}{2}\min(\Lambda)$;
   else if $\lambda_{\text{min}} = \max(\Lambda)$, then $\lambda^* = 2 \cdot \max(\Lambda)$;
   else $\lambda^* = \lambda_{\text{min}} + k \cdot \alpha$, where k is the step direction and α is the step size.
   (c) calculate the shrinkage weights: $\nu_\phi^{(j)} = \frac{2\cdot\nu_\mu}{\lambda^*+1}$; $\nu_p^{(j)} = \lambda^* \cdot \nu_\phi^{(j)}$;
   (d) perform bootstrap-validation to estimate $L_{cv}^{(j)}(\lambda^*)$;
   (e) append $\Lambda \leftarrow \lambda^*$ and append $\text{Ł} \leftarrow L_{cv}^{(j)}$;

The algorithm continues until a pre-defined convergence criteria is met, or, practically, a maximum number of iterations is reached. The final values of ν_ϕ, ν_p, and m_{cv} are those which correspond to the minimum $L_{\text{cv}} \in \text{Ł}$.

There are various convex optimization algorithms that differ in how to calculate the k and α. In CJSboost, most of the optimization benefits occur during the "stepping-out" procedure, and so exact values of k and α are less important, so long as they guarantee convergence. I suggest the following sub-algorithm (nested within step 3b above), which convergences slowly but quickly rules out large chunks of bad values of λ.

1. Define the triplet set Γ composed of the current best estimate of λ_{min} as well as the values just to the left and right, such that $\lambda_{\text{min}}^{-1} < \lambda_{\text{min}} < \lambda_{\text{min}}^{+1}$;
2. Sort the entries of Γ according to the order $L_{cv}(\gamma^{(1)}) < L_{cv}(\gamma^{(2)}) < L_{cv}(\gamma^{(3)})$;
3. Estimate the step size and direction:

$$\text{if } \|\gamma^{(1)} - \gamma^{(2)}\| \geq \|\gamma^{(1)} - \gamma^{(3)}\|:$$

$$\text{then } \alpha = \tfrac{1}{2}\|\gamma^{(1)} - \gamma^{(2)}\| \text{ and } k = \text{sign}(\gamma^{(1)} - \gamma^{(2)});$$

$$\text{else } \alpha = \tfrac{1}{2}\|\gamma^{(1)} - \gamma^{(3)}\| \text{ and } k = \text{sign}(\gamma^{(1)} - \gamma^{(3)});$$

4. $\lambda^* = \lambda_{\text{min}} + k \cdot \alpha$

Typically seven or ten iterations are necessary in order to find suitable values of λ, ν_ϕ and ν_p. Unfortunately, this strategy is only for a two-parameter likelihood with a single ratio to optimize. For other capture-recapture models with more parameters (e.g., POPAN, PCRD), a different tuning strategy will be necessary.

Appendix B.2. Algorithm 2 for tuning ν

With more parameters in the capture-recapture likelihood, the number of necessary steps in algorithm 1 will increase exponentially. I suggest a second iterative algorithm whose number of iterations may only increase linearly with the number of parameters. The principle of this second algorithm is based on the observation that when the ratio $\frac{\nu_p}{\nu_\phi}$ is poorly optimized, then additional boosting steps along the gradient $\frac{\partial \ell}{\partial F_\theta}$ will result in *increases* in the holdout-risk, and will do so asymmetrically for F_ϕ vs F_p. When $\frac{\nu_p}{\nu_\phi}$ is optimized, the number of boosting steps which increase the hold-out risk will be roughly the same for p and ϕ, averaged over all bootstrap hold-out samples. I suggest using this ratio as an estimate of $\hat{\lambda} = \frac{\nu_p}{\nu_\phi}$.

Call $\Delta_\theta^{(m)}$ a boosting step along the partial derivative of $\frac{\partial \ell}{\partial F_\theta}$ which successfully reduces the holdout-risk.

$$\hat{\lambda}^{(j)} = \hat{\lambda}^{(j-1)} Q\left(\frac{\sum_{m=1}^{m_k} \Delta_p^{(m)}}{\sum_{m=1}^{m_k} \Delta_\phi^{(m)}}\right) \tag{B.1}$$

where Q is a robust measure of central tendency over all B bootstraps (median, trimmed-mean), and m_k is some boosting step $m_k > m_{\text{cv}}$. The first estimate $\hat{\lambda}^{(1)}$ is typically an underestimate, so the algorithm is iterated, each time using the previous $\hat{\lambda}^{(j-1)}$ for a current estimate of ν_p and ν_ϕ with which to perform a bootstrap-validation exercise, and then updating $\hat{\lambda}^{(j)}$ by (B.1). $\hat{\lambda}^{(J)}$ typically converges to a single value within approximately 10 iterations. $\hat{\lambda}^{(J)}$ is *not* the optimal λ as estimated by algorithm 1, but it is in the vicinity (Figure B.9).

Clearly, for just two parameters and one ratio, this second algorithm is not competitive with algorithm 1. But, when there are more than two parameters in the likelihood, this algorithm can simultaneously estimate all pertinent ratios. Further refinements will be necessary, but simulations demonstrate that there is information in the risk gradient trajectories that can help optimize the hyperparameters.

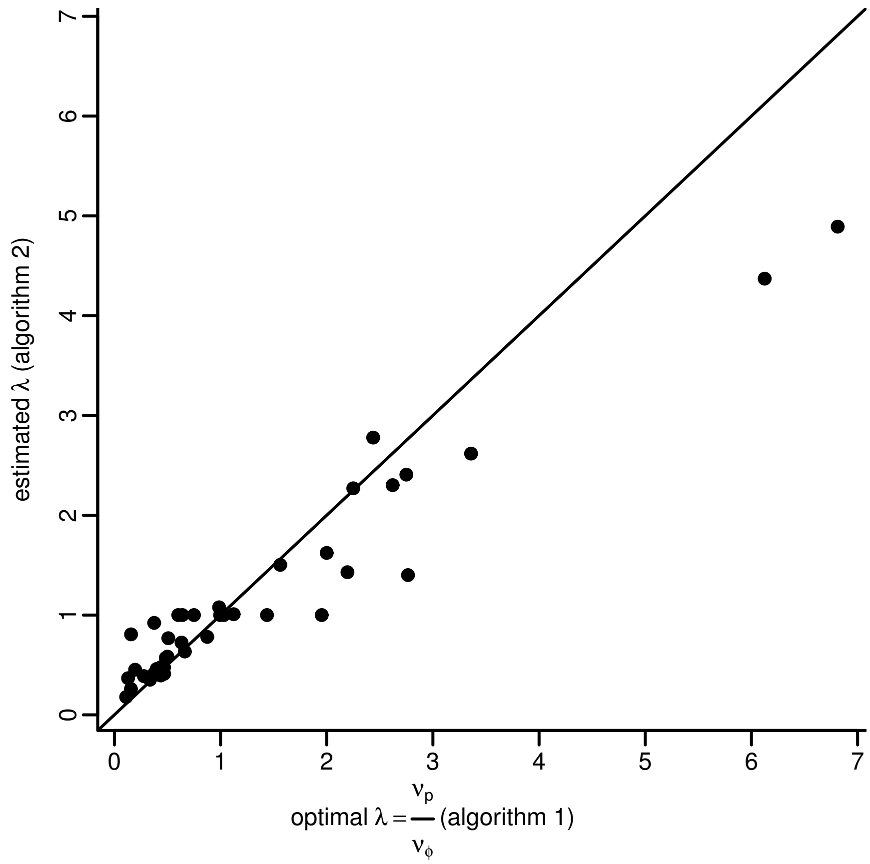

Figure B.9: Two algorithms for tuning the shrinkage weight hyperparameters ν_ϕ and ν_p, and their ratio λ, in order to minimize the expected loss (estimated via bootstrap-validation). Forty simulations compare the two algorithms, where algorithm 1 is considered optimal.