Subsampling for Cost Efficiency with Spatially Explicit Capture-Recapture Models

Nick Gondek

Submitted under the supervision of Dr. John Fieberg to the University Honors Program at the University of Minnesota-Twin Cities in partial fulfillment of the requirements for the degree of Bachelor of Science, magna cum laude, in Fisheries, Wildlife and Conservation Biology.

May 14, 2016

Abstract

Genetic mark-recapture studies estimate animal abundance using non-invasive DNA identification methods to "capture" and subsequently "recapture" individuals that leave genetic material at trap sites. Due to the cost of genotypic analysis, researchers often choose to process only a subsample of this genetic material. Traditional (non-spatial) mark-recapture estimators of abundance have been shown to be biased in this case, especially when the study population displays a behavioral trap response that varies at the individual level. Less is known about the influence of subsampling genetic mark-recapture data, randomly or non-randomly, when using spatially explicit capture-recapture (SECR) models to estimate abundance. We analyzed hair-snare data obtained from a 2012 genetic mark-recapture study of black bears (*Ursus americanus*), where 1019 hair samples were successfully genotyped. We simulated the process of subsampling hair samples either randomly or non-randomly. Similar to non-spatial mark-recapture estimators, subsampling produced density estimates that were lower, on average, than the full data estimate; however, non-random subsampling had much less of an effect on estimator performance, particularly at small sample sizes. Thus, non-random subsampling may be preferable to random sampling, despite the inherent violations of SECR assumptions that may result.

Acknowledgements

I thank John Fieberg for guidance with both the analysis and writing portions of this project, as much of this would not have been possible without his mentorship. I thank Dave Garshelis and Karen Noyce for access to the hair snare data that was analyzed in this study, the Minnesota Chapter of the Wildlife Society for allowing me to present my work at the 2016 annual meeting in Mankato, MN, and the University (of Minnesota) Honors Program for providing me with the resources and guidance necessary to start this project. Lastly, I'd like to thank the Fieberg lab for critiquing my work, and the various members of the Fisheries and Wildlife Department that encouraged and assisted me along the way.

Non-technical Summary

Mark-recapture studies, in which animals are 'tagged' and released, are often used by wildlife managers to estimate animal abundance. Usually, this involves physically capturing the animal in a trap (net, snare, etc), and physically tagging the animal with a unique number combination, so that it can be identified later on. Alternatively, animal's DNA can be used as a unique identifier, which facilitates studies that are less invasive and less costly than tagging by hand. Most of these methods involve hair snare traps - a baited lure is placed inside of a barbed wire perimeter, so that when the animal investigates the bait, they leave some hair behind which can be idenitified to determine the bear's identity. However, contrary to previous methods where an individual's identity could be determined simply by reading numbers on a tag, the hair samples need to be analyzed in a genetics lab to determine the bear's identity - which can quickly increase the cost of conducting a study. So, most of the time, managers subsample their hair samples to reduce their costs.

This sort of subsampling has been shown to be problematic for getting reliable population estimates, especially in species that have strong 'personalities' - for example, if one bear loves the bait, and leaves hundreds of samples, he is very likely to be included in a subsample. If a bear hated the whole experience, and left only one sample, he's much less likely to be included. Unfortunately, that might not be the only problem. Most abundance estimators inflate counts of observed individuals by an estimate of an individual's capture probability, which is essentially how likely a single animal is to be captured (in this case, to leave hair). This is calculated by observing how frequently we capture the individual. When we subsample, we're much more likely to analyze samples

from bears that are captured over and over again, so we calculate a capture probability that is too high, because we miss out on the individuals that are hard to capture. We then use that capture probability to derive abundance - when the capture probability is too high, we tend to have population estimates that are too low.

While subsampling *usually* causes problems, some hair samples actually don't contribute any new information. Traditionally, we would say an animal is 'captured' if we see it anywhere in our study area in a given time frame, regardless of where it was. As a result, a lot of the data we analyze just confirms what we already knew - for example, we know bear X was in the area from May 10 to May 20 after we see one of his hair samples, and any more of his hair we analyze during that time frame is useless, because we already knew he was there. A relatively new estimator (spatially explicit capture recapture, or SECR) actually uses this previously 'useless' data to estimate movement characteristics of the animal, and hence, the animal's likelihood of leaving a hair sample at each trap. Because of much of this formerly useless data gets thrown out when subsampling, we had reason to believe that SECR estimates may change quite a bit after the data is subsampled.

To investigate this, we subsampled some real hair snare data from northern Minnesota bears in a few different ways, and fit SECR models to those subsamples. We found we could minimize the effect of subsampling by choosing to process samples from as many site and time period combinations as possible, even if that meant that we were no longer sampling in a completely random fashion. Doing so means that the subsampled data no longer *accurately* describes each animal's movement characteristics (as described above), but it means that we are more likely to analyze hair samples from new individuals and

individuals that leave very few samples. In this case, the trade-off was worth it.

While we can't make generalizations yet, because we don't know how many bears were truly out there when the hair samples were obtained, we argue that subsampling *is* problematic for SECR models. However, depending on the behavior of the animals being studied, the effects likely vary - if animals have a tendency to leave many samples at a single site, we predict that subsampling may be less of an issue. Regardless, this problem has not been given much attention yet, and our results can provide some initial qualitative guidance to wildlife managers making decisions about how (or if) they should subsample their data in anticipation of using SECR models to estimate population size.

Contents

Introduction

Mark-recapture studies are routinely used by wildlife managers to estimate animal abundance; especially in the case of endangered species and game animals, abundance and its associated temporal trends are of critical importance for making informed management decisions (Borchers *et al.* 2002; McCrea & Morgan 2014). In many cases, however, abundance estimates can be difficult to interpret without some understanding of the effective area sampled; extrapolation to a regional scale by way of density is almost always desired, if not necessary (Borchers *et al.* 2002; Royle *et al.* 2013). As such, abundance estimates without associated reliable density estimates may be of limited use to managers.

Density estimates have often been obtained from abundance estimates using ad-hoc methods that range widely in their biological relevance (Royle *et al.* 2013). Spatially-explicit capture recapture methods (SECR), by contrast, link abundance estimators to their associated study area in a statistically rigorous way (Efford *et al.* 2005; Borchers 2012; Royle *et al.* 2013). SECR estimates of abundance scale directly with a given sample area, potentially allowing for more standardization of estimates across space or time. SECR models also make better use of the capture-recapture data. In particular, SECR models utilize the information in the spatial capture histories to model detection probabilities as a function of the distance between each animal's activity center (AC) and the trapping grid. By accounting for each individual's movement tendencies, SECR provides a way to account for an important source of individual heterogeneity in capture probabilities that is unaccounted for in traditional (non-spatial) mark-recapture estimators.

We explored a popular method for estimating abundance of mammal populations: genetic mark-recapture, where individuals are identified by sequencing the genome of residual genetic material left at a trap site (Boulanger *et al.* 2004; Petit & Valiere 2006; Gervasi *et al.* 2008; Buckworth & Territory 2012). Many studies aim to obtain hair samples, often by the use of baited traps and hair-snares, which provide a relatively non-invasive and efficient way to ascertain the identity of detected animals when compared to live capture. However, due to the costs of genotypic analysis, managers are often forced to subsample their hair clusters (Boulanger *et al.* 2004; Petit & Valiere 2006; Gervasi *et al.* 2008; Settlage *et al.* 2008).

Subsampling has been shown to be problematic for non-spatial mark-recapture estimators, especially when individuals exhibit a behavioral response to having been previously captured, and this behavioral response is not consistent across individuals (Tredick *et al.* 2007; Ebert *et al.* 2010; Augustine *et al.* 2014). In this case, individuals that leave many hair clusters are likely to be identified in a subsample, whereas individuals that leave few clusters are often excluded. In other words, clusters selected in a subsample are likely to come from individuals that are repeatedly captured. As a result, estimates of capture probability are biased high and abundance estimates are biased low; this effect is more pronounced as subsample size decreases (Augustine *et al.* 2014).

A fundamental difference between SECR and non-spatial capture-recapture models relates to how they treat multiple captures of the same individual during the same time period. Whereas non-spatial models collapse multiple captures at different traps (same time period) into a single capture event, these multiple captures would be used

to inform individual movement characteristics in a SECR model (Borchers 2012; Royle et al. 2013). Thus, samples that are redundant in a non-spatial model (captures at > 1 trap location in a single period) are of critical importance to SECR models. As a result, the effects of subsampling may be even more problematic for SECR models when compared to non-spatial capture-recapture models.

In response to the knowledge that multiple samples from one individual in the same period are redundant in non-spatial models, managers often avoid taking multiple samples from a single trap under the assumption that they are likely to be from the same individual, and thus not informative and not worth the cost of genotypic analysis. This strategy is effective for use with non-spatial models; analyzing several samples from the same site often leads to diminishing returns in precision and accuracy (Dreher *et al.* 2009). Instead, preference is given to samples from novel site by session combinations (hereafter referred to as site-sessions), because they are more likely to represent new individuals. This non-random sampling could cause issues in SECR models by violating the assumption that the observed samples reflect the movement characteristics of the study population. A natural solution to this violation is to avoid preference for novel site-sessions (ie, simple random sampling), at the cost of identifying fewer unique individuals and analyzing more redundant data.

Various simulation and empirical studies have addressed the issue of subsampling on non-spatial mark-recapture estimators, with a general conclusion that subsampling genetic mark-recapture data results in estimators that are biased low, but that the magnitude of this bias depends on the actual data at hand and several assumptions

such as erroneous genotyping rate and trap spacing (Tredick *et al.* 2007; Dreher *et al.* 2009; Augustine *et al.* 2014). The tradeoff between potential redundancy (using simple random sampling) and non-representation of movement characteristics (using non-random sampling) has not been investigated in the context of SECR models. **In this study, our objective was to use northern Minnesota genetic mark-recapture dataset with 1019 successful samples to compare abundance and density estimates using (1) various subsampling strategies utilized by wildlife managers and researchers and (2) various subsampling rates reflective of different budgetary constraints.** Using these results, we provide guidance for genetic mark-recapture estimates when budget constraints limit effective sample size.

Methods

The subsections below describe the nature of the data analyzed and the procedure followed in this study. The resultant code can be found in the Appendix under subsection "Code Documentation".

Data

The data analyzed in this study come from a 2012 genetic mark-recapture study of American black bears (*Ursus americanus*) in northern Minnesota (Garshelis & Noyce 2013). Using stationary hair-snare traps baited with suspended bacon and scent lures, the authors collected 1642 hair clusters (groups of hair samples obtained from adjacent barbs) from 121 sites over six trapping sessions from May through July 2012, each of ten day length. Of these 1642 clusters, 1113 were sent to a genetics laboratory for

genotypic analysis, and 1019 samples were successfully linked to individual bears.

It is important to note that the 1113 samples analyzed were effectively a subsample to begin with; the authors initially chose samples from 377 unique site by session combinations (site-sessions), with the remaining 776 samples chosen randomly. Given that the majority of samples were analyzed (67.7%), and that the majority of these samples were chosen randomly from the original sample set, we will treat the dataset as if it is representative of the population of interest.

Camera traps fitted to a subset of the trap locations indicated that bears occasionally visited the same trap several times in a single session, often using several different entry locations to approach the bait hung at the center of the trap. This individual variation is evidenced in the number of samples left at a given site-session by individual bears; 47% of the time, bears left only a single sample at a given site-session, but some left as many as 11, and some may have left even more considering that 529 samples were not analyzed (Garshelis & Noyce 2013). Additionally, movement characteristics differed strongly between individuals; 53% of bears visited between one and five unique sites, but bears visited up to 24 unique sites over the 6 sample periods.

Subsampling

We chose to explore four subsampling strategies, but two are of primary importance: simple random sampling and a subsampling method that gives preference to unique site-sessions (Figure 2). These are described below:

- SimpleRandom - n samples are chosen at random from the entire data set, without respect to period or site. This type of sampling is, on average, most representative

Figure 1: A typical baited hair-snare trap in northern Minnesota, one of 121 sites in the study conducted by Garshelis and Noyce (2013). Between May and July 2012, black bears left a total of 1642 hair clusters at these sites.

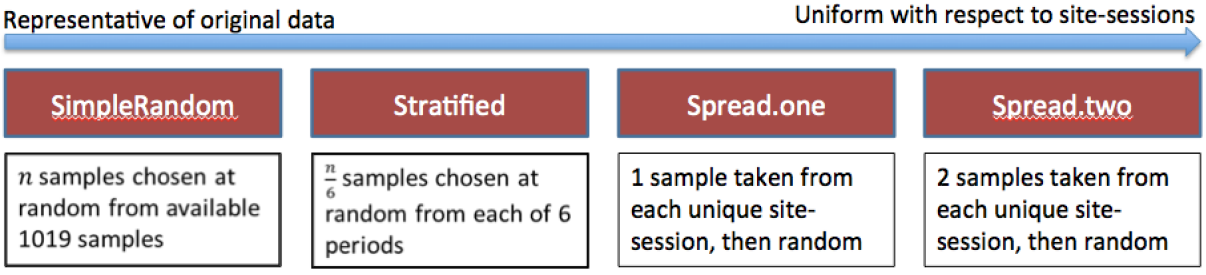

Figure 2: Subsampling types used to determine the effect of subsampling strategies on Spatially Explicit Capture Recapture models using hair snare data from northern Minnesota.

of the original data set; for example, site-sessions that have a large number of samples would have a the largest number of samples in the subsample, and site-sessions with only one sample are unlikely to be chosen.

- Stratified - n/6 samples are chosen at random from each of 6 sessions, without respect to site. It is important to note that this differs from a stratified random sample with proportional allocation, i.e., where the number of samples chosen from each session would be weighted by the number of samples in each session relative to the number of samples overall.
- Spread.one - one sample is chosen from each unique site-session. After this, samples are chosen randomly until n samples are selected. At $n < 377$, this strategy is idenitical to Spread.two.
- Spread.two - two samples are chosen from each unique site-session. After this, samples are chosen randomly until n samples are selected. This method is least representative of the original data, and effectively maximizes the number of site-sessions (and thus, unique individuals) included in analysis; for example, each unique site-session with only one or two samples have both of those samples chosen, and site-sessions with large amounts of samples are under-represented relative to the original dataset. At $n < 377$, this strategy is idenitical to Spread.one.

Model Structure

A SECR model is unique from other mark-recapture models in that an animal's capture probability is derived using the animal's activity center (AC) (Borchers *et al.* 2002; Royle *et al.* 2013). Though many curves are used to characterize how detection probabilities change as a function of distance between an animal's activity center and a

trap location, a common and readily understood choice is a half-normal curve, using two parameters: $g0$ and σ. $g0$ represents the probability of detecting an animal whose activity center is located exactly at the trap location. σ represents the rate at which this probability decreases as an animal's homerange center moves further away from the trap.

Similar to other population models, these parameters can be allowed to vary by sex (Sex), time (t), and they may be allowed to change following an initial capture event (i.e., a behavioral trap response (b)). In figure 3, $g0$ was modeled as a function of time and sex ($g0 \sim$ t + Sex) and σ varied by sex only ($\sigma \sim$ Sex); as such, each time and sex combination has its own intercept and capture probabilities decrease at different rates for males and females as a function of distance. Time effects may also be modelled in a structured, linear fashion (eg, $g0 \sim T$ instead of $g0 \sim t$), meaning that there is a trend in the change of intercept detection probability; this would be appropriate if detection probabilites decreased or increased monotonically over time.

Model Fitting

We fit 5 SECR models, each with a different combination of explanatory variables, to each subsampled dataset. In each case, we assumed σ varied only by sex. We explored five models with different covariates influencing $g0$ (Table 1).

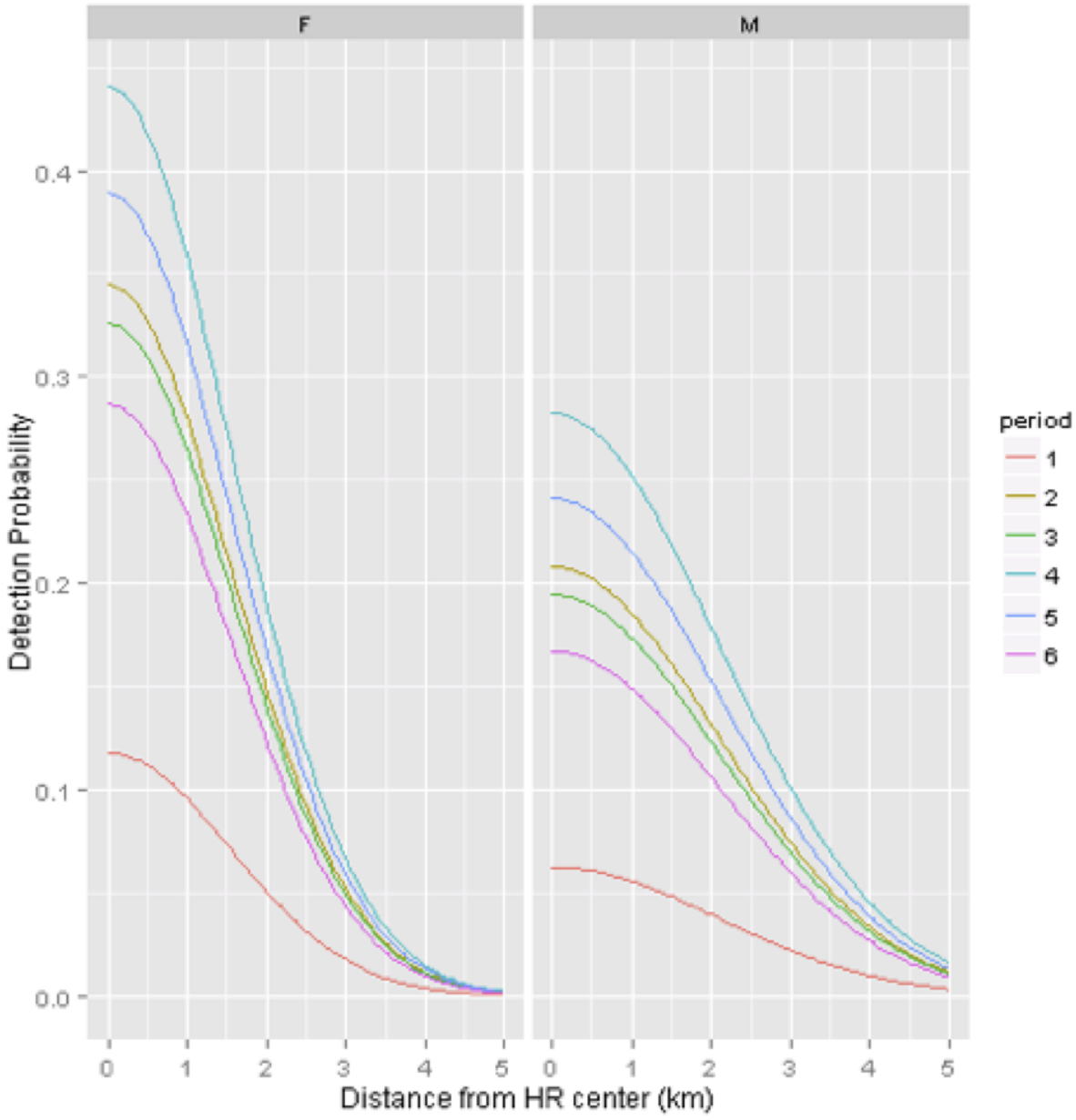

Figure 3: Estimated half-normal detection curves for a SECR model fit to all 1019 genetic hair samples, where $g0$ varied by time and sex ($g0 \sim$ t + Sex) and σ varied by sex only ($\sigma \sim$ Sex).

Table 1: Density estimates and Aikaike Information Criterion (AICc) scores associated with spatial-mark recapture (SECR) models fit to 1019 hair samples taken from black bears in Minnesota from May through July 2012. $\Delta AICc$ represents the difference from the lowest scoring model and the compared model. All models were fit using the R programming language and package secr. 'b' represents a behavior covariate, 't' represents a non-linear time covariate, and 'Sex' a sex covariate.

$g0$ Covariates	AICc	$\Delta AICc$	Bears/100 sq mile(*95% Confidence Interval*)
b + t + Sex	3492	0	13.50 (*9.86,18.48*)
t + Sex	3496	4	12.67 (*9.30,17.26*)
b + t	3502	10	13.60 (*9.93,19.64*)
t	3507	15	12.55 (*9.22,17.08*)
b	3515	23	14.23 (*10.41,19.46*)

Models were fit using the R programming language (R Core Team 2015), package 'secr' for the fitting of the SECR models, and packages 'foreach' and 'doParallel' for optimization of model fitting using parallel processing (Analytics & Weston 2014, 2015; Efford 2015). The main function for fitting models to subsampled data is secr.fit in package 'secr'. This function requires a capture history and a trapping grid to arrive at a derived density estimate and estimated parameters describing the effect of time, sex, and/or behavior on capture probabilities (Royle *et al.* 2013).

Simulation

In this study, a single 'simulation' can be broadly defined as subsampling the full dataset using one of the subsampling strategies (Figure 2), fitting of five SECR models with varying covariates (Table 1), and saving model output for later comparison (Figure 4). Initially we varied the subsampling strategy while holding sample size constant at n = 450 hair samples (Figure 2, 4B). This allowed us to evaluate the influence of subsampling strategy on the derived density estimate of each SECR model; these simulations are referred to as 'subsampling trials' in this document as well as in the accompanying code documentation.

We then considered two subsampling types (SimpleRandom and Spread.one) for further study. In a second set of 'size trials', we explored these two approaches across a range of sample sizes (n = 250 to 950, by 100) (Figure 4A).

Notation

Let $\hat{D}_{data,model}$ refer to a density estimate from the fit of a SECR model to dataset of type *data* (either full, so, or, sr), and *model* used to indicate the set of covariates for $g0$. For example, $\hat{D}_{full,b}$ represents the density estimate using the full dataset and behavior as the sole covariate for $g0$. $\hat{D}_{sr,b+t}$ represents a density estimate using a simple random sample and both behavior and time as covariates for $g0$; $\hat{D}_{so,t}$ represents a density estimate using the spread.one subsampling method and time as a covariate for $g0$

Let M_{model} refer to a SECR model with *model* again used to indicate the set of covariates for $g0$. For example, $M_{b+t+Sex}$ refers to a model with behavior, time and sex as covariates for $g0$. .

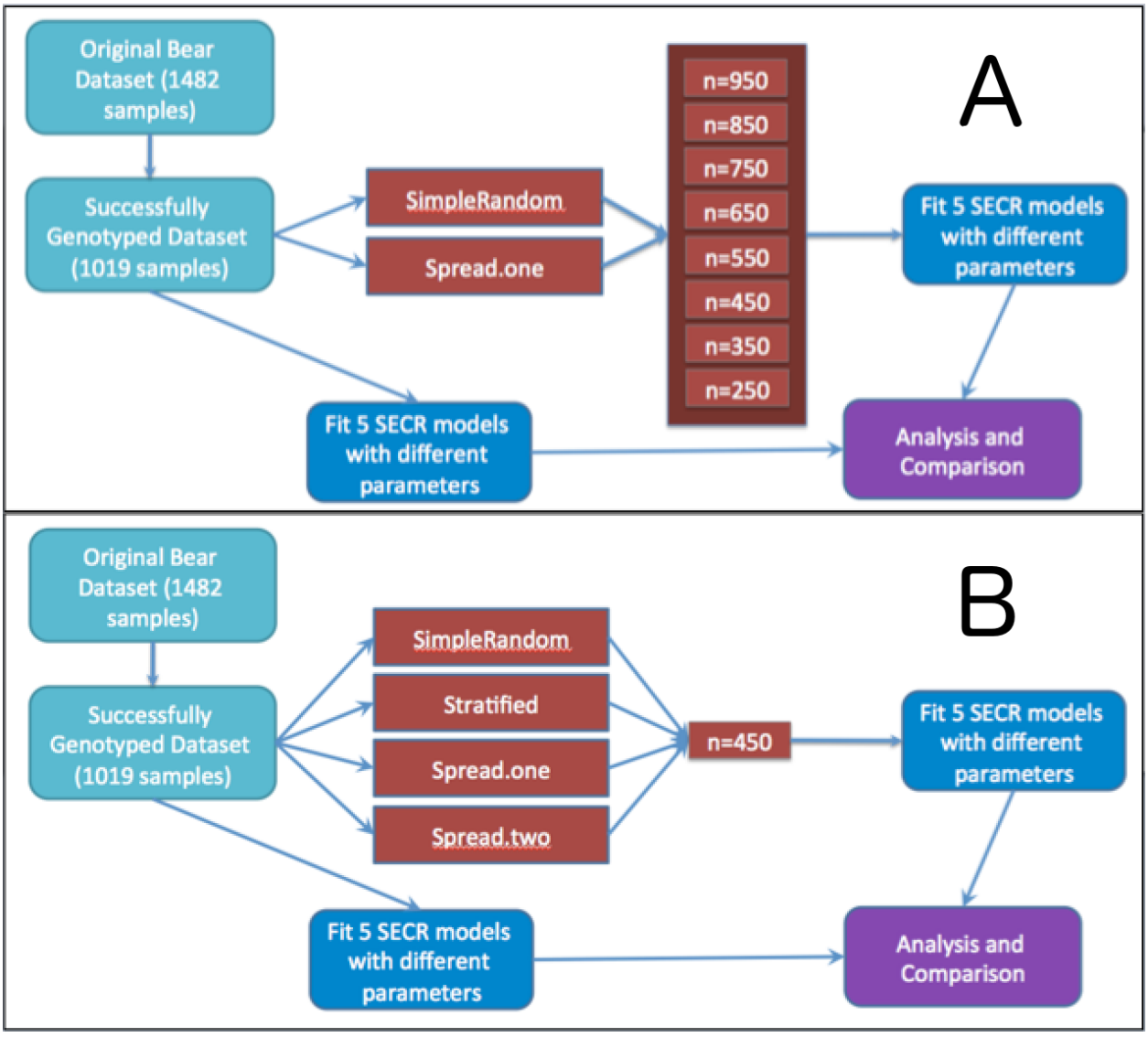

Figure 4: Visual representation of simulations performed in this study; We subsampled 1019 hair snare samples using a number of subsampling methods and size combinations for the purpose of comparing density estimates and associated biases.

Let $g0_{full}$ refer to the estimate of $g0$ using the full dataset, and let $g0_{sr}$ and $g0_{so}$ refer to $g0$ estimates obtained when subsampling using SimpleRandom and Spread.one, respectively. Further, let $g0b$ represent the difference between intial and recapture probabilities. Again, we will use subscripts to refer to the dataset and model used to estimate this parameter.

Results

Full Dataset Estimates

$M_{b+t+Sex}$ had the lowest AIC score of the five models investigated (Table 1). Estimates that included behavior as a parameter for $g0$ (ie, $\hat{D}_{full,b+t+Sex}$, $\hat{D}_{full,b+t}$, and $\hat{D}_{full,b}$) resulted in larger density estimates than models that excluded behavior ($\hat{D}_{full,t+Sex}$, and $\hat{D}_{full,t}$), with $\hat{D}_{full,b}$ resulting in the largest density estimate (14.23) and $\hat{D}_{full,t}$ resulting in the smallest density estimate (12.55) (Table 1).

Simulation Study I: Subsampling Strategy

All distributions of $\hat{D}_{sr,model}$ were shifted towards lower values compared to $\hat{D}_{full,model}$, regardless of the model analyzed. Further, in all cases, sampling distributions of $\hat{D}_{sr}$ were more variable than sampling distributions of $\hat{D}_{so}$ (Figure 5).

With n=450, sampling distributions of $\hat{D}_{so,b+t+Sex}$, $\hat{D}_{so,b+t}$, and $\hat{D}_{so,b}$ were centered on the corresponding estimates of the full dataset ($\hat{D}_{full,b+t+Sex}$, $\hat{D}_{full,b+t}$, and $\hat{D}_{full,b}$; Figure 5), whereas the sampling distributions for $\hat{D}_{full,t+Sex}$ and $\hat{D}_{full,t}$ were shifted

towards lower values (Figure 5, bottom row).

Simulation Study II: Sample Size

As expected, similar to non-spatial mark-recapture estimates, smaller sample sizes produced more variable estimates, with greater discrepancies between $\hat{D}_{sr,model}$ and $\hat{D}_{full,model}$. However, models $M_{b+t+Sex}$, M_{b+t}, and M_b resulted in sampling distributions of $\hat{D}_{so,model}$ that were centered at the corresponding estimate of $\hat{D}_{full}$, regardless of sample size (Figure 6, bottom row). Similar to the previously discussed trends, distributions of $\hat{D}_{so}$ were also closer to $\hat{D}_{full}$ when a behavioral effect was not modelled (M_{t+Sex} and M_t).

To better understand these results, we investigated the effect of subsampling on intial capture and recapture probabilities across different sample sizes using $M_{b+t+Sex}$. Similar to the effects of subsampling on non-spatial mark-recpature parameter estimators, estimates of both $g0_{sr}$ and $g0_{so}$ were were, on average, larger than $g0_{full}$. Further, estimates of both $g0b_{sr}$ and $g0b_{so}$ were, on average, larger in absolute magnitude relative to $g0b_{full}$. These effects were more pronounced as subsample size decreased, and were also more pronounced when using SimpleRandom subsampling than when using Spread.one (Figure 7).

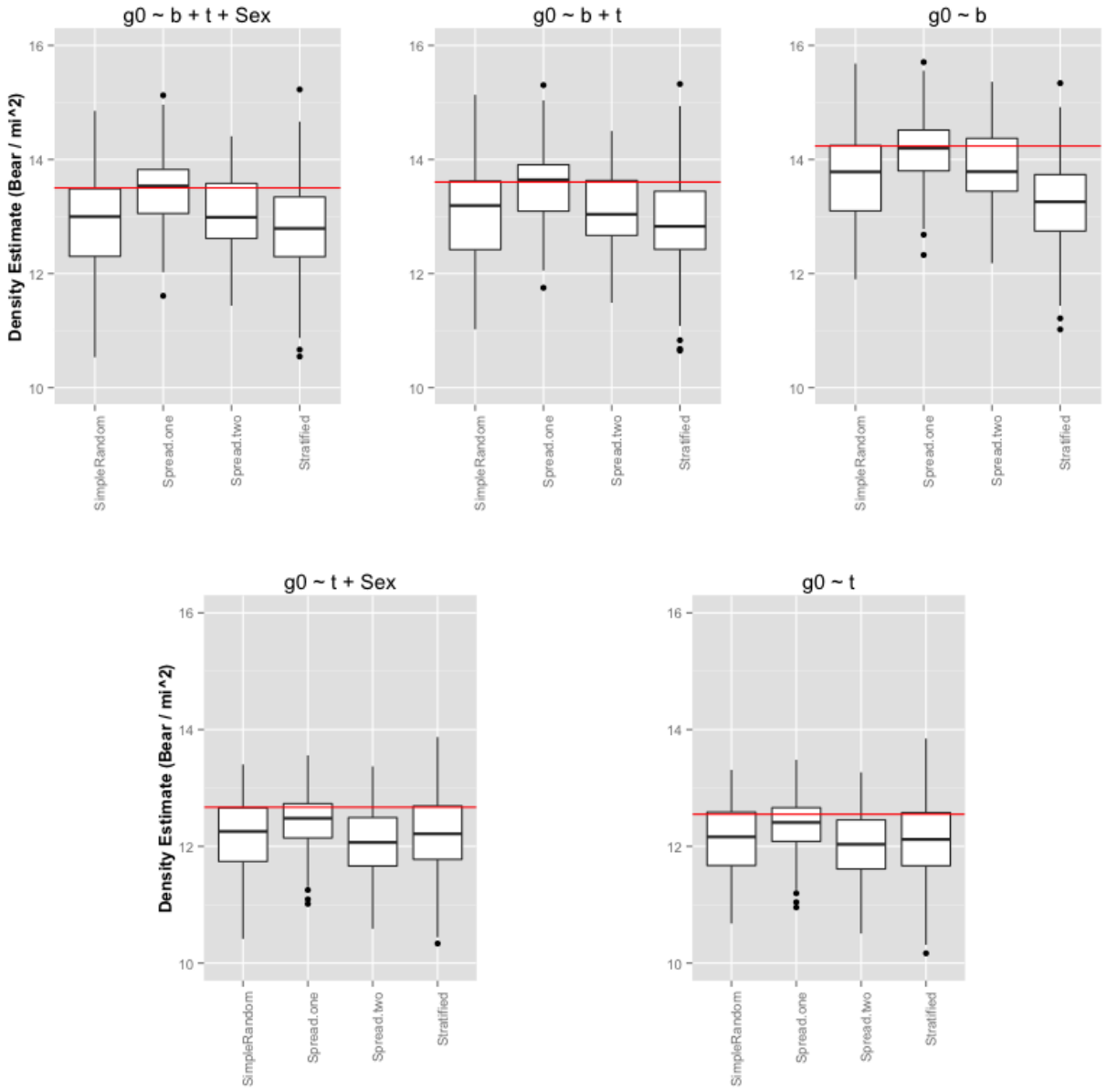

Figure 5: Derived density estimates of a spatially explicit capture-recapture model fitted to n=450 subsamples of 1019 hair clusters using the four subsampling methods described in Figure 2. The red horizontal line represents the full dataset estimate (n=1019) for each model.

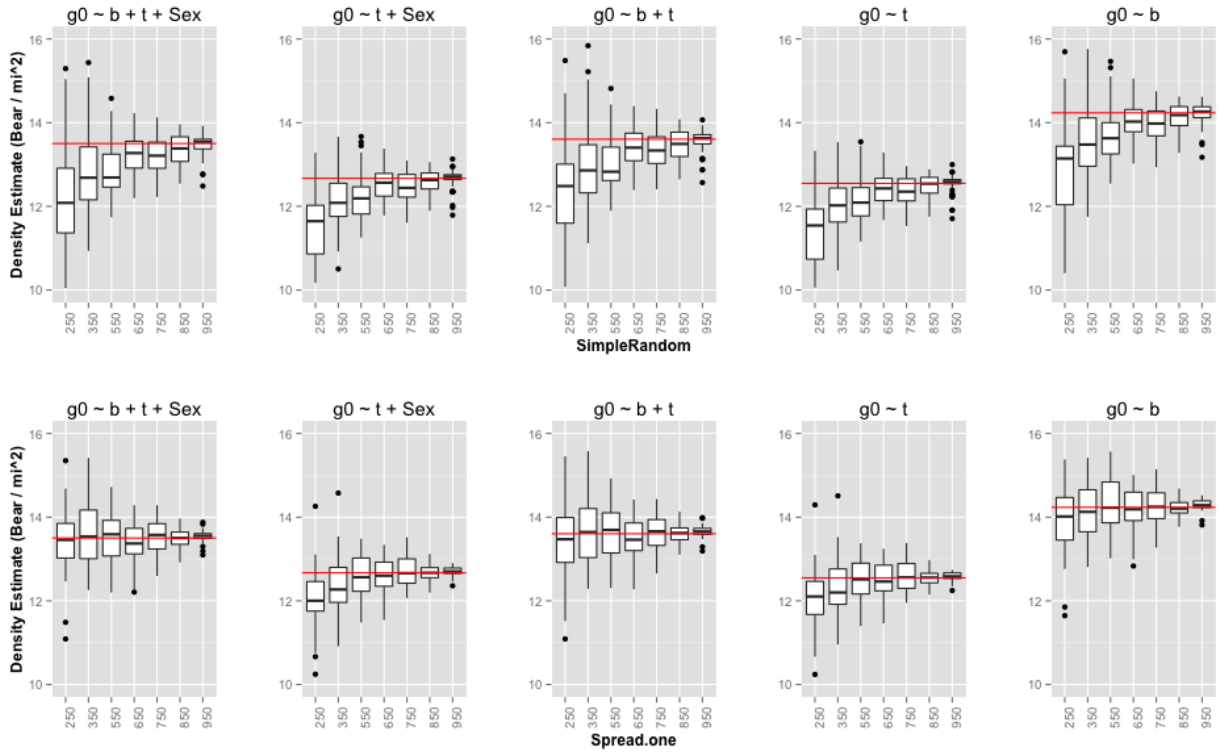

Figure 6: Derived density estimates of a SECR model fitted to various size subsamples (250 to 950 by 100) of 1019 hair clusters using one of two subsampling methods (SimpleRandom and Spread.one) described in Figure 2. The horizontal red line indicates the density estimate when fitting the given model to the full dataset.

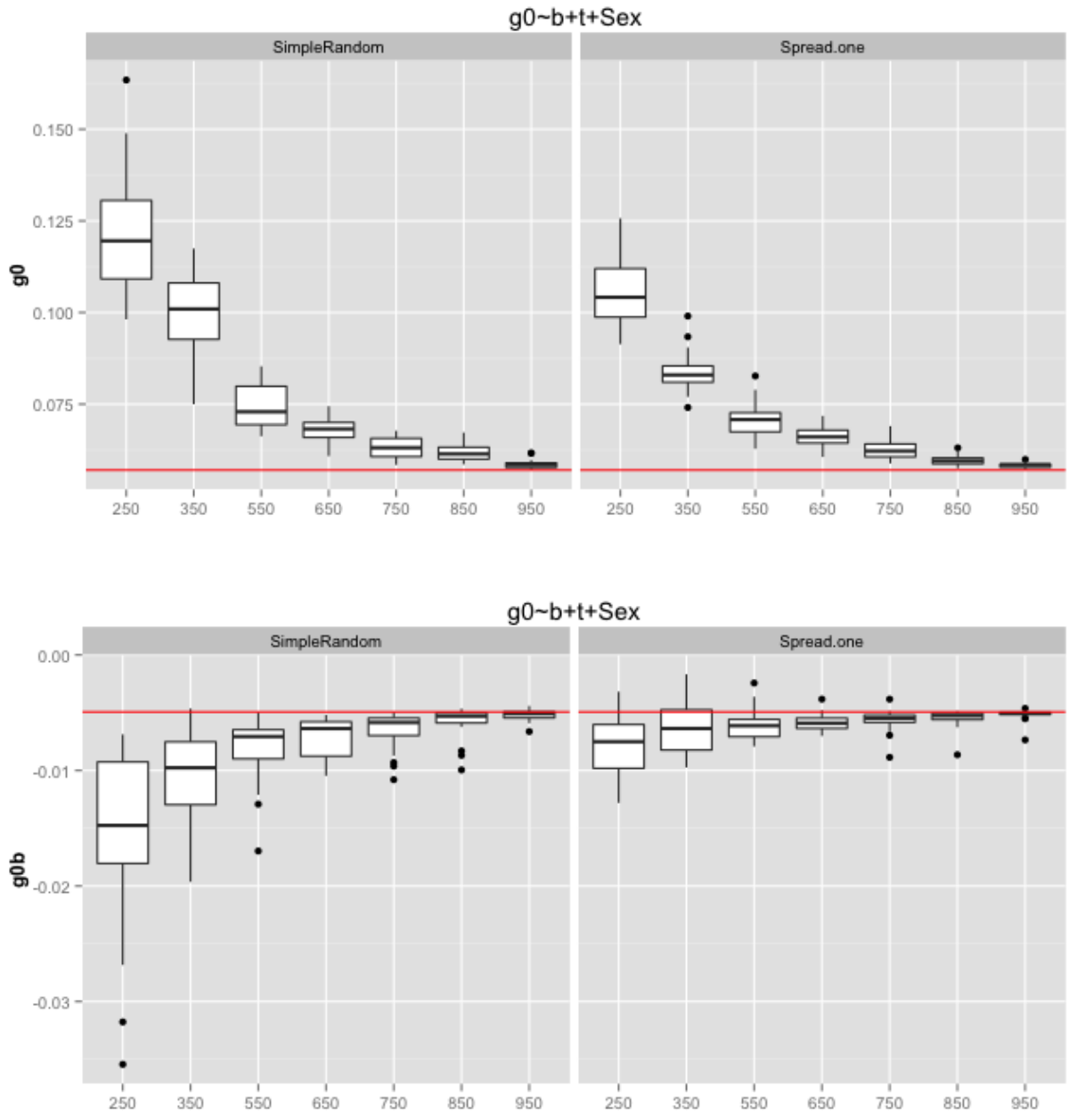

Figure 7: Estimated $g0$ (initial capture probability) and $g0b$ (difference between initial and recapture probability) for different sample sizes of two subsampling types, SimpleRandom and Spread.one (described in Figure 2). The horizontal red line indicates the value when fitting this model ($g0 \sim$ b + t + Sex) on the full dataset. The other four models displayed similar effects with respect to $g0$. The other two models which included a behavioral covariate for $g0$ displayed similar trends with respect to $g0b$.

Discussion

Subsampling Performance

Our simulation results suggest that SECR density estimators, like their non-spatial counterparts, are biased low when applied to subsampled data (Tredick *et al.* 2007; Augustine *et al.* 2014). These effects were minimal, however, when using a subsampling method that gives preference to unique site-sessions (Spread.one), provided that a behavioral effect was included in the model. Distributions of $\hat{D}_{so}$ were closer to $\hat{D}_{full}$ at most subsample sizes for each model (Figure 6), and similarly, estimates of $g0_{so}$ and $g0b_{so}$ were closer to their full dataset counterparts (Figure 7). Non-random sampling increased the number of uniquely identified individuals (especially those that left only one or two samples), which outweighed the inherent cost of violating SECR's assumption that the capture history provides an accurate description of individual movement characteristics.

Bears in this study exhibited considerable heterogeneity in their behavioral response (47% of the time, bears left only a single sample at a given site-session, but some left as many as 11; Garshelis & Noyce 2013). As a result, subsampling the data using simple random sampling was much more likely to select redundant data from bears that tended to leave several samples at a single site-session. Conversely, non-random sampling likely performed well because much of the data it excluded from the full dataset was redundant (same bear, same site, same session), and thus it increased the likelihood of including novel samples.

Key assumptions in this study include the treatment of the full dataset as a

representative sample from the population - Noyce and Garshelis (2013) successfully subsampled 67.7% of samples, so there is likely an unmodelled behavioral effect from the samples that were not originally analyzed when the study took place. Further, these analyzed hair samples were already non-random to a degree (377 non-random with preference to unique site-sessions and 776 completely random), so our analysis may be missing novel individuals or known individuals' locations that were similarly not identified in the original genotypic analysis. However, it is also likely that many of these samples were redundant due to the strong individual heterogeneity noted in this study. Additionally, we analyzed only successful samples, and thus ignored the possibility of failure to indentify individuals from a submitted sample; however, because our dataset was not simulated, there may have been false indentification of individuals due to allelic dropout or false amplification (Dreher *et al.* 2009), or the 'shadow effect' - erroneously treating a novel individual as a recapture due to similarity in their genotype (Mills *et al.* 2000).

Future Analysis

The work presented here indentifies an issue that has not previously been given much attention; that subsampling may be more problematic for SECR models than non-spatial mark-recapture estimates. Our simulations suggest that using a strategy to maximize site-sessions within an analyzed subsample may ameliorate these problems, but the optimal strategy likely depends on the characteristics of the observed study population. Further simulation is needed before we can make more general conclusions and recommendations regarding the effect of subsampling methodology on SECR estimates; this could be accomplished by simulating genetic mark-recapture data with

varying degrees of behavioral responses and various genotypic error rates (both positive bias inducing, in the case of shadow effects, or negative, in the case of allelic dropout and false amplification) to see when the trade-offs between the subsampling methods presented here begin to favor random sampling. Yet, the insights we have gained through our simulations allow us to make several predictions.

In a simulated case with no redundant samples (eg, each bear that enters a trap leaves exactly one sample and leaves, and $g0b = 0$), one might expect random sampling to perform better (less biased, more precise) than a non-random sample, as the former would be more representative of the actual biological processes affecting movement patterns, and thus better model the heterogeneity in capture probability that arises from those patterns. As redundant samples are introduced (eg, each bear that enters a trap leaves more than one sample and leaves), one might expect that preference to unique site-sessions would once again perform best, as it did in our simulations using empirical data.

References

Analytics, R. & Weston, S. (2014). *DoParallel: Foreach parallel adaptor for the parallel package*.

Analytics, R. & Weston, S. (2015). *Foreach: Provides foreach looping construct for r*.

Augustine, B.C., Tredick, C.A. & Bonner, S.J. (2014). Accounting for behavioural response to capture when estimating population size from hair snare studies with

missing data. *Methods Ecol Evol*, **5**, 1154–1161.

Borchers, D. (2012). A non-technical overview of spatially explicit capture–recapture models. *Journal of Ornithology*, **152**, S435–S444.

Borchers, D., Buckland, S. & Zucchini, W. (2002). *Estimating animal abundance: Closed populations*. Springer-Verlag London.

Boulanger, J., Himmer, S. & Swan, C. (2004). Monitoring of grizzly bear population trends and demography using dNA mark-recapture methods in the owikeno lake area of british columbia. *Canadian Journal of Zoology*, **82**, 1267–1277.

Buckworth, R. & Territory, N. (2012). GENETAG: genetic mark-recapture for real-time harvest rate monitoring: Pilot studies in northern australian spanish mackerel fish. *Fisheries Research and Development Corporation*, **78**, 1–17.

Dreher, B., Rosa, G. & Lukacs, P. (2009). Subsampling hair samples affects accuracy and precision of DNA Based population estimates. *The Journal of Wildlife Management*, **73**, 1184–1188.

Ebert, C., Knauer, F., Storch, I. & Hohmann, U. (2010). Individual heterogeneity as a pitfall in population estimates based on non-invasive genetic sampling: A review and recommendations. *Wildlife Biology*, **16**, 225–240.

Efford, M. (2015). *Secr: Spatially explicit capture-recapture models*.

Efford, M., Warburton, B., Coleman, M. & Barker, R. (2005). A field test of two methods for density estimation. *Wildlife Society Bulletin*, **33**, 731–738.

Garshelis, D. & Noyce, K. (2013). Capture heterogeneity in hair-trapping of bears. *Summary of Research Findings*, 71–85.

Gervasi, V., Ciucci, P., Boulanger, J., Posillico, M. & Sulli, C. (2008). A preliminary estimate of the apennine brown bear population size based on hair-snag sampling and

multiple data source mark-recapture huggins models. *Ursus*, **19**, 105–121.

McCrea, R.S. & Morgan, B.J.T. (2014). *Analysis of capture-recapture data*. Chapman; Hall, CRC.

Mills, S.L., Citta, J.J., Lair, K.P., Schwartz, M.K. & Tallmon, D.A. (2000). Estimating animal abundance using noninvasive DNA sampling: Promise and pitfalls. *Ecological Applications*, **10**, 283–294.

Petit, E. & Valiere, N. (2006). Estimating population size with noninvasive capture mark recapture data. *Conservation Biology*, **20**, 1062–1073.

R Core Team. (2015). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria.

Royle, J., Chandler, R., Sollmann, R. & Gardner, B. (2013). *Spatial capture-recapture*. Academic Press.

Settlage, K.E., T, M.F., Clark, J.D. & King, T.L. (2008). Challenges of DNA based mark recapture studies of american black bears. *The Journal of Wildlife Management*, **72**, 1035–1042.

Tredick, C., Vaughan, M., Stauffer, D. & Simek, S. (2007). Sub-sampling genetic data to estimate black bear population size: A case study. *Ursus*, **18**, 179–188.

Code Documentation

This section is meant to document and explain various functions and scripts that were created in my (Nick Gondek) spring 2016 thesis project with John Fieberg and Dave Garshelis.

Generally, the process is broken down into three steps - subsampling (BearSubsample), data transformation (BearSumFunc), and model fitting/storage (BearParallelModelFit). Three functions perform these three steps, and one function aggregates them into a single overarching function (BearParallelSubSECR). Two scripts use this function in a loop to perform many iterations of this process, one varying the subsampling type, and one varying subsampling type while also varying subsample size (FloridaScript.R and MankatoScript.R, respectively).

In the document below, various chunks will represent different R scripts and functions, and will be annotated in order to explain how each contributes to the final product.

Table of Contents

Appendix

X) Example SECR Simulation

XI) Other Scripts and Functions

a) BearModelFit.R and BearSubSECR.R
b) Synthesisv2.R, Synthesisv3.R, and PilotFitSummary.Rmd
c) VerificationSECR.rmd and VerificationSECR_corrected.Rmd
d) Data Exploration.R and Data ExplorationUpdated.R
e) Data Summary.R
f) huggins.R and HuggyBear.R
g) SecrAll1.R and SecrAll2.R

Subsampling Function

I. BearSubsample.R

a. Summary

This function takes in a sample observation data frame (which is constructed both in DataExplorationUpdated.R and the eventual funtion BearParallelSubSECR.R) and subsamples it according to the four methods that were chosen, explained in greater detail below. The input is a large data frame of sample observations (~1000), and the output is a subsampled data frame of sample observations (Usually ~450, but this varies later on).

b. Arguments

data - A sample observation data frame to be subsampled. This is constructed both in DataExplorationUpdated.R and the eventual funtion BearParallelSubSECR.R.

type - The subsampling type to be used ("SimpleRandom", "Stratified", "Spread.one", "Spread.two").

a) SimpleRandom - takes a random sample of size n from the entire data set, without any respect to sites or sessions. This is done by using mosaic's sample function, which works on data frames.

b) Stratified - a stratified sample taking n/6 random samples from each of the 6 periods. Note the difference between a **true stratified random sample** in that the samples are taken disproportionately from each period - instead of weighting each period based on their number of samples, a constant n/6 are taken from each period. This is performed using a for loop which iterates over the 6 periods and using the same mosaic sample function as above.

c) Spread.one - takes a single sample from each site x period combination. This is done using mutate, by creating a new variable called uniqueID which is a numeric of site x period. The data is then subsetted using a logical vector (!duplicated(uniqueID)) to take every unique combination, and then filled out with a random sample of the leftovers.

d) Spread.two - takes two samples from each site x period combination. This is done using mutate, by creating a new variable called uniqueID which is a numeric of site x period. The data is then subsetted twice in a for loop using a logical vector (!duplicated(uniqueID)) to take every unique combination, and then filled out with a random sample of the leftovers.

n - The number of observations to be 'genotyped' - for the initial subsampling types, this was fixed at 450.

```
BearSubsample<-function(data, type, n){
  require(dplyr)
  if (type=="SimpleRandom"){ ##Simple random (all n taken at random with no regard
                             ##to period or site)
     selected<-mosaic::sample(data, size=n)
  }


  if (type=="Stratified"){##For each of the six periods
    selected<-NULL
    for (i in (1:6)){
      new<-mosaic::sample(data[data$Period==i,], size = n/6)
      selected<-rbind(selected, new)
      selected<-filter(selected, !(is.na(Period)))
    }
  }

  if (type=="Spread.one"){##First take 1 from each site X period, then random
    selected<-NULL
    droprow<-NULL
    data<-mutate(data, uniqueID=(site*as.numeric(Period)))
    data<-sample(data)##Mix them up, then take the first one from each uniqueID

    selected<-data[!duplicated(data$uniqueID),]
    droprow<-(!duplicated(data$uniqueID)) * c(1:nrow(data))
    data<- data[-droprow,]

    selected<-rbind(selected, mosaic::sample(data, size=(n-nrow(selected))))
  }

  if (type=="Spread.two"){##First take 2 from each site X period, then random
    selected<-NULL
    droprow<-NULL
    data<-mutate(data, uniqueID=(site*as.numeric(Period)))
    data<-sample(data)##Mix them up, then take the first one from each uniqueID

    for (s in (1:2)){ ##Do it twice this time
    selected<-rbind(data[!duplicated(data$uniqueID),], selected)
    droprow<-(!duplicated(data$uniqueID)) * c(1:nrow(data))
    data<- data[-droprow,]
      }


    selected<-rbind(selected, mosaic::sample(data, size=(n-nrow(selected))))
  }

  return(selected)
  }
```

SECR Functions

II. BearSumFunc.R

a. Summary

This function takes in a data frame of sample observations (subsampled or not, it does not matter) and transforms it into a SECR friendly proximity detector capture history file, to be used in the next step of fitting the SECR models and saving their resultant objects and outputs.

b. Arguments

data - a csv file of sample observations, which may be the full sample set or the subsampled sample set, saved as a temp file

summary - if TRUE, function outputs various metrics related to the input sample observations - these are largely from John's DataExplorationUpdated.R. This argument defaults to FALSE, to avoid cluttering the console when scripting.

output.csv - if NULL, function ouputs the proximity detector itself as a data frame. If a file path is set, function outputs a csv of this same data frame.

type - default is "proximity", and this is unchanged over the course of these initial iterations (Pre-April 2016). John's initial proposal had expressed interest in multi detectors ("multi"), so I incorporated this functionality for ease of future use. Any other type inputted results in an error.

```
BearSumFunc <- function(data, summary=FALSE, output.csv = NULL, type="proximity"){
  library(dplyr)
  library(mosaic)##Required Libraries
  sampobs<-read.csv(data) ##Reading in data

  #Gender: Sex = 204.25 -> Male, Sex= 250.25 ->Female
  sampobs$Group<-rep("M",nrow(sampobs))
  sampobs$Group[sampobs$Sex==250.25]<-"F"

  #Create a data set that contains a count of the number of times each bear
  #was seen for each unique site x period combination.  Also determine
  #sex of each individual and tabulate the number of males and femlaes
  caphist<- sampobs%>%group_by(Individual, site, Period)%>%
    summarize(Count= n())
  sexid<-unique(select(sampobs, Individual, Group))
  table(sexid$Group)
  caphist2<-merge(caphist, sexid, all=FALSE)


  ##optional summary output
  if(summary==TRUE){
    sum.out<-matrix(NA, 3, 4)
    rownames(sum.out)<- c("Same bear, site, period",
              "n detections per site & period", "Unique bears at each site")
    colnames(sum.out)<- c("Max", "Mean", "sd", "n")

    capnums<-caphist%>%group_by(site, Period)%>% summarize(ncap=n() )
    timecap<-caphist%>%group_by(Period)%>% summarize(nbears=n_distinct(Individual) )
    sitecap<-caphist%>%group_by(site)%>% summarize(nbears=n_distinct(Individual) )

    #same bear, site, period
    sum.out[1,]<-as.double(favstats(caphist$Count)[5:8])
     #n detections per site&period
    sum.out[2,]<-as.double(favstats(capnums$ncap)[5:8])
    #n detections per site&period
```

```
    sum.out[3,]<-as.double(favstats(sitecap$nbears)[5:8])

    outputsummary<-list(Captures.by.Period=timecap, Summary=sum.out)
    print(outputsummary)
  }

  #Write out proximity detector (if a path is set, write csv, otherwise
  #just output the data frame itself)
  bearCH<-data.frame(Session="BearMR", ID=caphist$Individual,
  Occassion=caphist2$Period, Detector=caphist2$site,
    Sex=caphist2$Group)

  bearCHP<-data.frame(Session="BearMR", ID=sampobs$Individual,
            Occassion=sampobs$Period,
            Detector=sampobs$site, Sex=sampobs$Group)
  # names(bearCH)<-c("Session", "ID", "Occasion",
  # "Detector", "Count", "Sex") ##
   if (!(is.null(output.csv))){
              if (type=="proximity"){
               write.table(bearCH, file=output.csv,
                   row.names=FALSE, col.names=FALSE, sep=",")
              } else if (type=="multi"){
                write.table(bearCHP, file=output.csv,
                     row.names=FALSE, col.names=FALSE, sep=",")
              } else print("Unrecognized detection type")
   }

    if (is.null(output.csv)){
            if (type=="proximity"){
              return(bearCH)
            } else if (type=="multi"){
              return(bearCHP)
            } else print("Unrecognized detection type")
  }
}
```

III. BearParallelModelFit.R

a. Summary

This function runs the computationally intensive process of secr model fitting independently on each available processor to cut down on the processing time dramatically. There are some minor nuances that are different from a typical for loop, and they are explained in comments below. Following the model fitting, the resultant output are saved in the form of a single csv, which is appended as needed with new entries, and RDS files of the models themselves are saved into their corresponding folder, as seen in the directory image below (or within the SpatialMR folder itself).

Also, there is a density conversion function tucked at the top of this script - I didn't feel it was substantial enough to warrant its own script.

b. Arguments

trapgrid - the trapgrid file that is to be used for the secr fitting. For the purposes of this initial analysis, we use the trapgrid from the field, but I integrated this to allow use of a simulated trapgrid.

subtype - the subtype used in the previous step of subsampling - note that no subsampling actually happens in this step, but it is needed to provide the correct path to save the model output.

iteration - the iteration of the loop, not for use interactively, but to save the model fit rds file with a new number.

size - another argument to save models in the correct folders - if size = 450, the output gets put into the SubsamplingData folder, else, it gets put into the SizeData folder.

```
desnconv<-function(x){
  100*x/3861.022
}
library(secr)
library(secrdesign)
library(scrbook)

BearParallelModelFit<-function(trapgrid="../data/detectorfileScaled.csv",
                               subtype=NULL, iteration=1, size=450) {
  output<-subtype
library(foreach)
#library(doMC)
library(doParallel)
#proxdata path and caphist
if(size==450){proxdata<-paste("../data/SubsamplingData/", subtype,
                              "/TempCaphist.csv", sep="")}
if(size!=450){proxdata<-paste("../data/SizeData/", subtype,
                              "/TempCaphist.csv", sep="")}

    bearCH<-read.capthist(captfile = proxdata, trapgrid,
                          detector= 'proximity', covnames="Sex")

#models to run within the foreach loop
models<-list(list(g0~b+t+Sex, sigma~Sex),
             list(g0~t+Sex, sigma~Sex),
             list(g0~b+t, sigma~Sex),
             list(g0~t, sigma~Sex),
             list(g0~b, sigma~Sex))
SecrFits<-vector("list", 5)
whichMod<-c("Model A", "Model B", "Model C", "Model D", "Model E")
names(SecrFits)<-whichMod

#setup parallel backend to use all processors -
#note that this is not generally recommended for
#computers that are actually in use, but since this
#was run primarily on a server and/or broken
#laptop, I opted to use all cores, because I didn't
#need any cores to run other tasks.

 #make a processing cluster of ALL available cores
cl<-makeCluster(detectCores())

#tell foreach loop that we are using these cores
registerDoParallel(cl)

#start time
```

```
strt<-Sys.time()

#foreach loop, parallel (%do% operator runs
#systematically, %dopar% is parallel)

foreach(d=1:5) %dopar% {
  #import packages, need to be
  #loaded on each processor independently
  library(foreach)
  library(doParallel)
  library(secr)
  library(secrdesign)
  library(scrbook)
  source('~/Google Drive/spatialMR/Rscripts/BearParallelModelFit.R')
  secrNew<-NULL
  modelEval<-models[[d]] #which model this core is actually going to process
  secrNew <- secr.fit(bearCH, model=modelEval, buffer = 10, trace = FALSE, CL=TRUE)

  if (!(is.null(secrNew)))
  { print(dN<-derived(secrNew))

    if(size==450){ #if the size is exactly 450,
      #must be a subsampling trial
  pathNew<-("../data/SubsamplingData/FinalBearEst.csv")
  nrowSub<- nrow(read.csv("../data/SubsamplingData/SubsampleLabID.csv"))
    }

    if(size!=450){ #if the size is not exactly 450,
      #must be a size trial
  pathNew<-("../data/SizeData/FinalBearEst.csv")
  nrowSub<- nrow(read.csv("../data/SizeData/SubsampleLabID.csv"))
    }


  densNew<-desnconv(dN[2,c(1,3,4)]) #converting density


  if(size==450){
    write.table(c(nrowSub, densNew, subtype, whichMod[d]),
                file =pathNew, append = TRUE, col.names = FALSE,
                row.names = FALSE, sep = ",")
  pathNew<-paste("../data/SubsamplingData/", output,"/",
                 whichMod[d],"/secr", iteration, ".rds", sep="")
  }

  if(size!=450){
    write.table(c(nrowSub, densNew, subtype, whichMod[d], size),
                file =pathNew, append = TRUE, col.names = FALSE,
                row.names = FALSE, sep = ",")
    pathNew<-paste("../data/SizeData/", output,"/", whichMod[d],
                   "/secr", iteration, ".rds", sep="")
  }
```

```
  saveRDS(secrNew, file=pathNew)   }
}

#end time
print(Sys.time()-strt) #print how long the fitting took
print(Sys.time()) #print the current time (to check for stalling)
stopCluster(cl) #stop processing cluster
}
```

c. File Structure Files not referenced below are either remnants of previous iterations of this process, or are temporary files that are used in the model fitting process (these are named as such).

- Green - This is the final bear estimates csv using SECR, which contains bear estimates and associated confidence intervals, model letter (A-E), and the subsample ID. The rds files, further along in their respective Subtype folder and model letter folder, are the output lists of the secr model fitting. For each subtype for both SizeData and SubsamplingData, there are model letter folders where these rds files are kept.
- Red - These are conflict files created by google drive - to be honest, I am not sure what causes them, but they do not seem to affect the above FinalBearEst.csv in any way. To be safe, I have not deleted them.
- Yellow - This is the final bear estimates csv using Huggins p and c, which is in progress now (Mar 25, 2016).
- Purple - This is the csv containing the LabID numbers for each of the subsamples, so that they can be processed with huggins, and the resultant outputs can be directly compared using the same subsamples.

IV. BearSubParallelSECR.R

a. Summary

This function was created to complete the entire process of subsampling and model fitting in one line, so that it may be used in a script later on. As such, there isn't much else going on that is not covered by the above three functions. The first half of the function is from John's DataExplorationUpdated.R, cleaning up the samples data frame and filtering to only use successful samples.

b. Arguments

subtype - the desired subsampling type - both passed onto BearSubsample, to use the correct subsampling type, as well as to BearParallelModelFit, in order to save the models in the correct directory.

iterNo - the iteration of the loop, not for use interactively, but to save the model fit rds file with a new number - this is passed onto BearParallelModelFit.

size - the desired number of samples obtained via subsampling. Also used to save models in the correct folders - if size = 450, the output gets put into the SubsamplingData folder, else, it gets put into the SizeData folder.

```
BearSubParallelSECR<- function(subtype, iterNo, size=450) {
  source('~/Google Drive/spatialMR/Rscripts/BearParallelModelFit.R')
  source('~/Google Drive/spatialMR/Rscripts/BearSubsample.R')
  source('~/Google Drive/spatialMR/Rscripts/BearSumFunc.R')

  #'Loading necessary libraries.
```

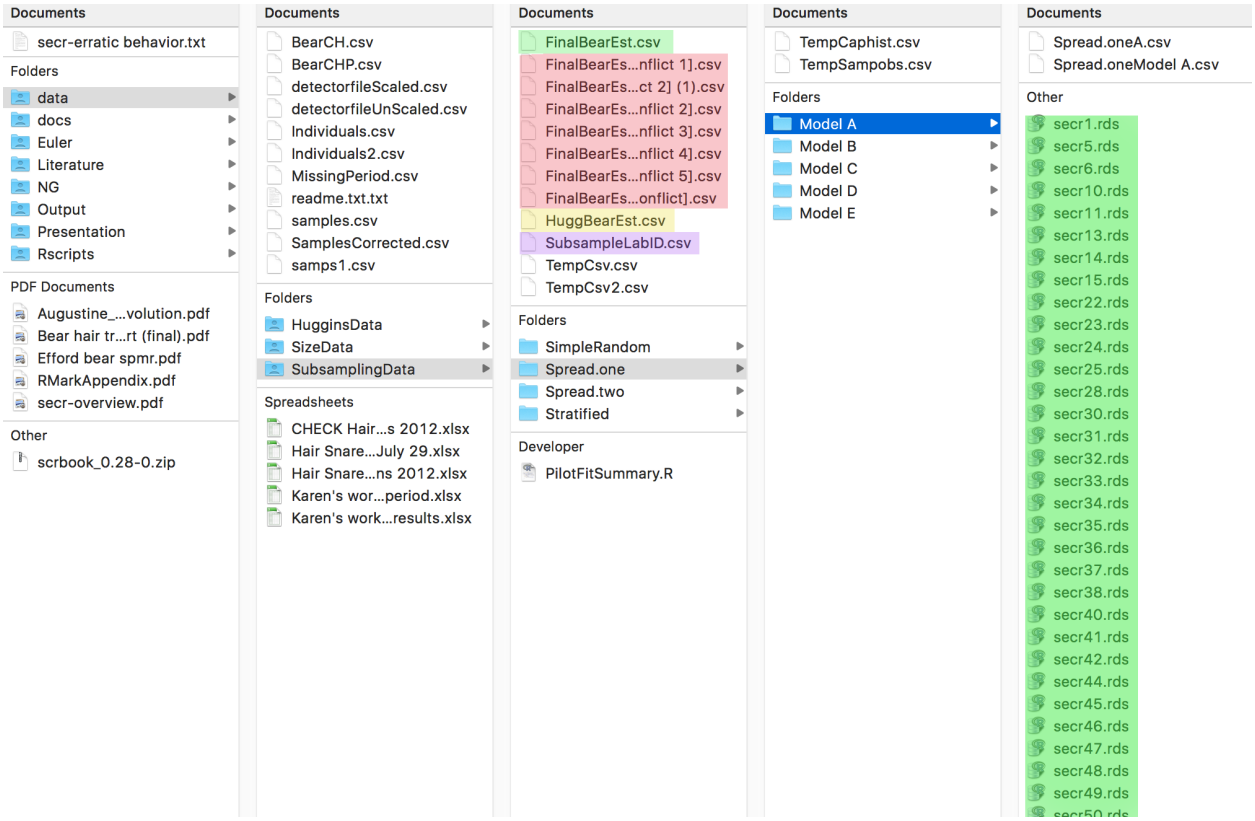

Figure 1: DirectoryFormat

```r
library(dplyr)
library(secr)
library(secrdesign)
library(scrbook)
library(mosaic)

#' Read in raw data on samples and individuals tab
ind<-read.csv("../data/Individuals2.csv")
samps<-read.csv("../data/samplesCorrected.csv")

#' Note: on July 23, David sent a file with corrections made
#' to a few observations that were missing period
#' or had period="6?".  This program uses the corrected
#' "samples" tab from that spreadsheet.  This is the
#' main difference from the DataExploration.R file.
#'

#' Look at event IDs & Periods. Use strsplit to determine
#' the site associated with each sample

events<-unlist(strsplit(as.character(samps$Event.ID), "-"))
samps$site<-as.numeric(sapply(strsplit( +
  as.character(samps$Event.ID), "-"), `[`, 1))
suppressWarnings(samps$cid<-as.numeric(sapply(strsplit( +
  as.character(samps$Event.ID), "-"), `[`, 2)))
table(samps$site)

#' Focus only on observations that were sampled successfully.
sampobs<-filter(samps, Class=="sample")

#' Fix the 6? entries, turn this variable into a numeric variable
sampobs$Period[sampobs$Period=="6?"]<-"6"

#' For now, drop observations that are missing Period.  Eventually,
#' however, we will want to reconcile two of these
#' observations that were highlighted earlier. The remaining 7 of
#' these 9 observations correspond to observations from collared bears.
sampobs<-filter(sampobs, Period!="")

#' Subsampling of the sampobs data.
sampobs2 <- BearSubsample(sampobs, subtype, size)

#' NEW 1/14/16 - saving the subsample labID for later
#' that way the same subsample can be reused.
if (size==450){
write.table(matrix(sampobs2$Lab.ID, 1, size),
            file = "../data/SubsamplingData/SubsampleLabID.csv", sep=",",
            append = TRUE, col.names = FALSE, row.names = FALSE )
}
if (size!=450){
  write.table(matrix(sampobs2$Lab.ID, 1, size),
            file = "../data/SizeData/SubsampleLabID.csv", sep=",",
            append = TRUE, col.names = FALSE, row.names = FALSE )
```

```
  }
  #' Now, turn Period into a numeric variable
  #sampobs2$Period<-droplevels(sampobs2$Period)
  sampobs2$Period<-as.numeric(sampobs2$Period)
  table(sampobs2$Period)

  #' Drop several of the columns that we will not need.
  sampobs3<-sampobs2[,c(1:9, 16,18,42:44)]

  #' Setting csv pathways, Write this file out
  if (size==450){
  pathcsv1<-paste("../data/SubsamplingData/", subtype, "/TempSampobs.csv", sep = "")
  pathcsv2<-paste("../data/SubsamplingData/", subtype, "/TempCaphist.csv", sep = "")
  write.csv(sampobs3, file=pathcsv1, row.names=FALSE)
      }

  if (size!=450){
    pathcsv1<-paste("../data/SizeData/", subtype, "/TempSampobs.csv", sep = "")
    pathcsv2<-paste("../data/SizeData/", subtype, "/TempCaphist.csv", sep = "")
    write.csv(sampobs3, file=pathcsv1, row.names=FALSE)
      }

  #' Apply the capture history transformation necessary for secr models
  BearSumFunc(data=pathcsv1, summary = FALSE, output=pathcsv2)

  #' Run the model fitting and storage function.
  BearParallelModelFit(subtype = subtype, iteration=iterNo, size= size)
  print(paste(subtype,iterNo, "done!"))
}
```

Huggins Function

V. BearHugginsParallel.R

This is still in progress - more later, when it is complete. Essentially, it will function similarly to BearParallelModelFit, using the huggins code from John's HuggyBear.R, and roughly analagous parameters to Models A-E in the SECR trials above.

Looping Scripts

VI. FloridaScript.R

A very straightforward script meant to run 'subsampling trials' - a for loop that performs a secr model fitting on each of the four subsampling types, and then repeats. The upper bound of the for loop is fairly arbitrary, as this was run almost continously throughout the month of January 2016.

```
source('~/Google Drive/spatialMR/Rscripts/BearSubParallelSECR.R')
for (iter in 150:1000){ ## This lower bound is changed interactively
  BearSubParallelSECR("Stratified", iterNo=iter)
  BearSubParallelSECR("Spread.one", iterNo=iter)
  BearSubParallelSECR("Spread.two", iterNo=iter)
  BearSubParallelSECR("SimpleRandom", iterNo=iter)
}
```

VII. MankatoScript.R

A script designed to run 'size trials' - for loop that executes two different subsampling types (chosen based on results of the previous script) with varying sizes. Try functionality added because at very low n, secr occasianally fails (not entirely sure why yet). Again, the upper bound of the outer loop is fairly arbitrary.

```
source('~/Google Drive/spatialMR/Rscripts/BearSubParallelSECR.R')
for (s in 130:1000) ## This lower bound is changed interactively
  {
testSizes<-seq(400, 240, -10)
    for (x in 1:17){
      try({BearSubParallelSECR("Spread.one", iterNo=s, size = testSizes[x])})
      try({BearSubParallelSECR("SimpleRandom", iterNo=s, size = testSizes[x])})
    }
  }
```

Analysis Scripts

VIII. EstimateSummary.R

These are the graphs and summaries that were used for my presentation on Feb 9 2016 at the Wildlife Society Conference in Mankato. The script reads in the final bear estimate (FinalBearEst.csv) from the SubsamplingData folder and compares them using box and whisker plots in gg, with a red line indicating what the 'full model' (all ~1000 samples) put out. Then, bar graphs representing the residual between this full model estimate and the avg estimate for each subtype. Lastly, anova to see whether the density estimates are significantly different from one another, holding model constant.

```
#'This script will read in the existing Bear Est data and compare them, using a bwplot.
#'(1/21/16)
library(mosaic)

BearEst<-read.csv("../data/SubsamplingData/FinalBearEst.csv")
colnames(BearEst)<-c("SubID", "densEstimate", "ucl", "lcl", "Subtype", "Model")

#'Number of Observations for each subtype X model. As of 1/21, there are ~16 for each.
#'As of 1/22, almost 25 for each. 100 should be attainable by around the 30th of Jan.
tally(Subtype~Model, data=BearEst)

#'Full model values from VerificationSECR.Rmd
fullModel<-c(14,15,14,13,14)
names(fullModel)<-c("Model A", "Model B", "Model C", "Model D", "Model E")

#'A box-whisker plot function to streamline the process for each model.
BearBWplot<-function(model="Model A"){
  ggplot(filter(BearEst, Model==model), aes(x=Subtype, y=densEstimate))+
    geom_boxplot() +
    ylab("Density Estimate (Bears per Sq Mile)")+
    ggtitle(model) +
    geom_hline(yintercept=fullModel[model], col="red")
    }
#'Treating all models the same, how does the subtype affect estimate
ggplot(BearEst, aes(x=Subtype, y=densEstimate))+
  geom_boxplot() +
  ylab("Density Estimate (Bears per Sq Mile)")+
```

```
  ggtitle("All Models")

#'Constructing a bwplot to see the difference between the different subtypes for each model
BearBWplot("Model A")
BearBWplot("Model B")
BearBWplot("Model C")
BearBWplot("Model D")
BearBWplot("Model E")

#'Look at the residuals between the full model and the means of the subtypes.
BearResid<-function(model="Model A"){
  resids<-mosaic::mean(densEstimate~Subtype, data=filter(BearEst, Model==model))-fullModel[model]
  barplot(resids, ylim=c(-1,1))
}
BearResid("Model A")
BearResid("Model B")
BearResid("Model C")
BearResid("Model D")
BearResid("Model E")

#'Another useful look at the data: ANOVA to see if the estimates are significantly
#'different from one another
bearmod<-lm(densEstimate~Subtype, data=BearEst)
Anova(bearmod)

#'Now by model - Interesting, they are only significantly different in A, C, E (as of
#'1/22/16) and those are the three that have behavior as a parameter for g0!
Anova(lm(densEstimate~Subtype, data=filter(BearEst, Model=="Model A")))
Anova(lm(densEstimate~Subtype, data=filter(BearEst, Model=="Model B")))
Anova(lm(densEstimate~Subtype, data=filter(BearEst, Model=="Model C")))
Anova(lm(densEstimate~Subtype, data=filter(BearEst, Model=="Model D")))
Anova(lm(densEstimate~Subtype, data=filter(BearEst, Model=="Model E")))

#'Spinning into md document
#'   spin("EstimateSummary.R")
```

IX. SizeSummary.R

Size summary data that was also presented on Feb 9 2016 in Mankato. First, this script enumerates the number of models that have been completed for each size x subtype combination, then performs a summary on a lm between density estimate and size, for models D and E, for subtypes SimpleRandom and Spread.one. Lastly, the relationship between sd and sample size is examined in a graph.

```
sizesum<-read.csv("../data/SizeData/FinalBearEst.csv")
library(mosaic)
library(ggplot2)
colnames(sizesum)<-c("SubsampleID", "densEstimate", "lcl", "ucl", "Subtype", "Model", "Size")
tally(Subtype~Size + Model, data=sizesum)


#less significant slope for simple random
summary(lm(densEstimate~Size, data=filter(sizesum, Model=="Model D",
                              Subtype=="SimpleRandom")))
```

```
xyplot(densEstimate~Size, data=filter(sizesum, Model=="Model D",
                                Subtype=="SimpleRandom"), type=c("p","r"))
mdD<-sd(densEstimate~Size, data=filter(sizesum, Model=="Model D",
                                 Subtype=="SimpleRandom"))
plot(mdD~names(mdD))

summary(lm(densEstimate~Size, data=filter(sizesum, Model=="Model E",
                               Subtype=="SimpleRandom")))
xyplot(densEstimate~Size, data=filter(sizesum, Model=="Model E",
                                Subtype=="SimpleRandom"), type=c("p","r"))
mdE<-sd(densEstimate~Size, data=filter(sizesum, Model=="Model E",
                                 Subtype=="SimpleRandom"))
plot(mdE~names(mdE))

#more significant slope for spread.one
summary(lm(densEstimate~Size, data=filter(sizesum, Model=="Model D",
                                Subtype=="Spread.one")))
xyplot(densEstimate~Size, data=filter(sizesum, Model=="Model D",
                                  Subtype=="Spread.one"), type=c("p","r"))
mdD<-sd(densEstimate~Size, data=filter(sizesum, Model=="Model D",
                                   Subtype=="Spread.one"))
plot(mdD~names(mdD))

summary(lm(densEstimate~Size, data=filter(sizesum, Model=="Model E",
                                  Subtype=="Spread.one")))
xyplot(densEstimate~Size, data=filter(sizesum, Model=="Model E",
                                   Subtype=="Spread.one"), type=c("p","r"))
mdE<-sd(densEstimate~Size, data=filter(sizesum, Model=="Model E",
                                    Subtype=="Spread.one"))
plot(mdE~names(mdE))

#Output for slide - four linear models showing the effect of size of the sample on the density estimate
par(mfrow=c(2,2))

  d1<-ggplot(data=filter(sizesum, Model=="Model D", Subtype=="Spread.one"),
             aes(x=Size, y=densEstimate))
  geom_point() +
  geom_smooth(method=lm, se=TRUE, col="red", size=1) +
  ggtitle("Model D - Spread") +
  ylab("DensEstimate")+
  xlab("Size") +
  theme(plot.title = element_text(size=22))

  d2<-ggplot(data=filter(sizesum, Model=="Model D", Subtype=="SimpleRandom"),
             aes(x=Size, y=densEstimate))
  geom_point() +
  geom_smooth(method=lm, se=TRUE, col="red", size=1) +
  ggtitle("Model D - Simple Random") +
  ylab("DensEstimate")+
  xlab("Size") +
  theme(plot.title = element_text(size=22))

  e1<-ggplot(data=filter(sizesum, Model=="Model E", Subtype=="Spread.one"),
```

```
            aes(x=Size, y=densEstimate))
  geom_point() +
  geom_smooth(method=lm, se=TRUE, col="red", size=1) +
  ggtitle("Model E - Spread") +
  ylab("DensEstimate")+
  xlab("Size") +
  theme(plot.title = element_text(size=22))

e2<-ggplot(data=filter(sizesum, Model=="Model E", Subtype=="SimpleRandom"),
            aes(x=Size, y=densEstimate))
  geom_point() +
  geom_smooth(method=lm, se=TRUE, col="red", size=1) +
  ggtitle("Model E - Simple Random") +
  ylab("DensEstimate")+
  xlab("Size") +
  theme(plot.title = element_text(size=22))

library(cowplot)
plot_grid(d1, d2, e1, e2, ncol = 2, nrow = 2)
```

Appendix

X. Example SECR Simulation

Here is an example of a single iteration of a SECR subsamping simualation. Note that this just the code from the inside of BearSubParallelSECR, with some minor tweaks to save time in the knitting of this document. Before each function, the input is shown, and after each function, the resultant output.

First, set some objects that would have been the arguments for BearSubParallelSECR.

```
subtype<-"SimpleRandom"
size<-450
iterNo<-999

#'This is where BearSubParallelSECR starts
source('~/Google Drive/spatialMR/Rscripts/BearParallelModelFit.R')
```

```
## Warning: package 'secr' was built under R version 3.2.3
```

```
source('~/Google Drive/spatialMR/Rscripts/BearSubsample.R')
source('~/Google Drive/spatialMR/Rscripts/BearSumFunc.R')

#'Loading necessary libraries.

library(dplyr)
library(secr)
library(secrdesign)
library(scrbook)
library(mosaic)
```

```
#' Read in raw data on samples and individuals tab (saved as .csv files)
ind<-read.csv("../data/Individuals2.csv")
```

```
samps<-read.csv("../data/samplesCorrected.csv")

#' Note: on July 23, David sent a file with corrections made to a few
#' observations that were missing period
#' or had period="6?".  This program uses the corrected "samples" tab
#' from that spreadsheet.  This is the
#' main difference from the DataExploration.R file.
#'

head(ind) ## data frame organized by individuals genotyped
```

```
##   Individual Sex no_Samples Loci   CXX20     G1D    G10M   MSUT2    MU50
## 1         56   F         65   24 123.137 176.186 214.218 205.207 134.136
## 2       1435   F          1   24 123.123 172.186 208.214 205.209 132.136
## 3       2213   F        117   24 135.139 174.184 206.216 197.205 120.132
## 4       3002   F         26   24 135.145 174.186 206.206 197.205 120.140
## 5       3101   M          1   24 123.123 172.182 208.214 197.205 120.132
## 6       3103   F          1   24 123.123 172.180 208.214 197.209 120.132
##      G10X    G10P REN144.A06     G1A    G10C     D1A    G10J    G10L
## 1 133.147 163.163    129.129 194.194 209.215 157.179 207.211 137.159
## 2 147.149 163.163    125.129 194.196 215.215 167.179 187.211 137.141
## 3 141.145 153.157    121.121 198.198 205.213 157.163 187.203 137.137
## 4 141.151 153.155    119.121 196.198 205.213 163.165 203.205 137.149
## 5 141.147 151.163    119.129 194.194 215.215 175.179 187.191 137.153
## 6 141.149 157.163    119.125 194.198 213.215 167.179 187.187 141.153
##      CPH9  CXX110    D123    MU26    G10U    MU59 REN145.P07    G10B
## 1 143.149 155.159 141.145 183.185 179.181 233.237    157.157 156.158
## 2 143.149 155.159 143.145 185.199 175.179 233.243    157.167 156.158
## 3 143.149 137.159 145.155 185.195 179.181 231.241    157.159 156.160
## 4 143.149 137.153 155.155 185.185 177.181 231.237    157.157 158.160
## 5 149.151 157.159 141.145 185.185 175.179 233.239    159.167 156.164
## 6 143.149 157.159 141.145 185.185 173.179 239.243    159.167 156.156
##      MU23    G10H Sex.1
## 1 203.207 239.241     F
## 2 187.203 241.253     F
## 3 189.203 241.253     F
## 4 189.199 241.253     F
## 5 187.205 241.241     M
## 6 203.203 241.253     F
```

```
head(samps) ## data frame organized by samples
```

```
##       Sample Lab.ID  Class Individual Bundle.No. Period Event.ID Priority
## 1    120-1-1    885 sample       0056        130      1   120-01        4
## 2 380-120-03    732 sample       0056        380      1   120-03        3
## 3    120-1-6   1004 sample       0056        254      1   120-06        4
## 4 380-120-07    731 sample       0056        380      1   120-07        2
## 5  20-120-09     20 sample       0056         20      1   120-09        2
## 6  20-120-10     87 sample       0056         20      1   120-10        3
##   Selected Bear.No.  X X.G X.U Left U.L     Comments X.1 Loci   CXX20
## 1        1       NA NA      30    A   L 1st envelope  NA    7 123.137
## 2        1       NA NA  10  NA    A   L              NA    7 123.137
```

```
## 3        2       NA NA  10  NA    A   L 1st envelope  NA    7 123.137
## 4        1       NA NA  10  NA    A   L               NA    7 123.137
## 5        3       NA NA      30    A   L 1st envelope  NA   15 123.137
## 6        1       NA NA  10  NA    A   U               NA   12 123.137
##       G1D    G10M   MSUT2    MU50    G10X    G10P REN144.A06     G1A G10C
## 1 176.186 214.218 205.207 134.136 133.147                NA      NA   NA
## 2 176.186 214.218 205.207 134.136 133.147                NA      NA   NA
## 3 176.186 214.218 205.207 134.136 133.147                NA      NA   NA
## 4 176.186 214.218 205.207 134.136 133.147                NA      NA   NA
## 5 176.186 214.218 205.207 134.136 133.147           129.129 194.194   NA
## 6 176.186 214.218 205.207 134.136 133.147 163.163        NA      NA   NA
##       D1A    G10J    G10L CPH9 CXX110    D123 MU26 G10U    MU59 REN145.P07
## 1      NA                  NA     NA      NA   NA   NA                  NA
## 2      NA                  NA     NA      NA   NA   NA                  NA
## 3      NA                  NA     NA      NA   NA   NA                  NA
## 4      NA                  NA     NA      NA   NA   NA                  NA
## 5 157.179 207.211 137.159   NA     NA 141.145   NA   NA                  NA
## 6      NA 207.211 137.159   NA     NA      NA   NA   NA 233.237         NA
##   G10B    MU23    G10H    Sex
## 1   NA      NA      NA 250.25
## 2   NA      NA      NA 250.25
## 3   NA      NA      NA 250.25
## 4   NA      NA      NA 250.25
## 5   NA 203.207 239.241 250.25
## 6   NA      NA 239.241 250.25
```

Take samples and add site id, filter for just successful samples, do a little cleanup.

```
  #' Look at event IDs & Periods. Use strsplit to determine the site associated with each sample
  events<-unlist(strsplit(as.character(samps$Event.ID), "-"))
  samps$site<-as.numeric(sapply(strsplit(
                as.character(samps$Event.ID), "-"), `[`, 1))
  suppressWarnings(samps$cid<-as.numeric(sapply(strsplit(
                as.character(samps$Event.ID), "-"), `[`, 2)))
  table(samps$site)
```

```
##
##   1   2   3   4   5   7   8   9  10  11  12  13  14  15  16  17  18  19
##  11  18   2  14  15   2  10   1  27  21   1   1  28  29  28   8  18   1
##  20  21  22  23  24  25  26  27  28  29  32  33  34  35  36  38  39  40
##   7  20  29  13   9   7  33  25  17  13   4  25  34  18  26   6   8   1
##  41  42  45  46  48  49  50  51  52  53  54  55  56  57  59  60  61  62
##  12  16  18  35   9   6  23   4   8   8   1  21  22   7  10  30   4  29
##  63  64  66  67  69  70  71  72  73  74  75  77  78  80  81  82  83  84
##   1  11  12  20  17   1  26  20  12   9  26  11   7   1   4   2  21   6
##  85  86  87  88  89  90  91  92  93  94  95  96  98  99 100 101 102 103
##  11  20  17  12  20  20  16  17  10   7   5  21  14  15  24  10   7  18
## 104 105 106 107 108 109 113 114 115 116 117 118 119 120 121
##  10  12  22   4  12   9   9  23   5   1   6   2   8  38  18
```

```
  #' Focus only on observations that were sampled successfully.
  sampobs<-filter(samps, Class=="sample")
```

```
#' Fix the 6? entries, turn this variable into a numeric variable
sampobs$Period[sampobs$Period=="6?"]<-"6"

#' For now, drop observations that are missing Period.  Eventually,
#' however, we will want to reconcile two of these observations that
#' were highlighted earlier. The remaining 7 of these 9 observations
#' correspond to observations from collared bears.
sampobs<-filter(sampobs, Period!="")
head(sampobs)
```

```
##       Sample Lab.ID  Class Individual Bundle.No. Period Event.ID Priority
## 1    120-1-1    885 sample       0056        130      1   120-01        4
## 2 380-120-03    732 sample       0056        380      1   120-03        3
## 3    120-1-6   1004 sample       0056        254      1   120-06        4
## 4 380-120-07    731 sample       0056        380      1   120-07        2
## 5  20-120-09     20 sample       0056         20      1   120-09        2
## 6  20-120-10     87 sample       0056         20      1   120-10        3
##   Selected Bear.No.  X X.G X.U Left U.L     Comments X.1 Loci   CXX20
## 1        1       NA NA      30    A   L 1st envelope  NA    7 123.137
## 2        1       NA NA  10  NA    A   L               NA    7 123.137
## 3        2       NA NA  10  NA    A   L 1st envelope  NA    7 123.137
## 4        1       NA NA  10  NA    A   L               NA    7 123.137
## 5        3       NA NA      30    A   L 1st envelope  NA   15 123.137
## 6        1       NA NA  10  NA    A   U               NA   12 123.137
##       G1D    G10M   MSUT2    MU50    G10X    G10P REN144.A06     G1A G10C
## 1 176.186 214.218 205.207 134.136 133.147                 NA      NA   NA
## 2 176.186 214.218 205.207 134.136 133.147                 NA      NA   NA
## 3 176.186 214.218 205.207 134.136 133.147                 NA      NA   NA
## 4 176.186 214.218 205.207 134.136 133.147                 NA      NA   NA
## 5 176.186 214.218 205.207 134.136 133.147            129.129 194.194   NA
## 6 176.186 214.218 205.207 134.136 133.147 163.163         NA      NA   NA
##       D1A    G10J    G10L CPH9 CXX110    D123 MU26 G10U    MU59 REN145.P07
## 1      NA                 NA     NA      NA   NA   NA                  NA
## 2      NA                 NA     NA      NA   NA   NA                  NA
## 3      NA                 NA     NA      NA   NA   NA                  NA
## 4      NA                 NA     NA      NA   NA   NA                  NA
## 5 157.179 207.211 137.159   NA     NA 141.145   NA   NA                  NA
## 6      NA 207.211 137.159   NA     NA      NA   NA   NA 233.237         NA
##   G10B    MU23    G10H    Sex site cid
## 1   NA      NA      NA 250.25  120   1
## 2   NA      NA      NA 250.25  120   3
## 3   NA      NA      NA 250.25  120   6
## 4   NA      NA      NA 250.25  120   7
## 5   NA 203.207 239.241 250.25  120   9
## 6   NA      NA 239.241 250.25  120  10
```

Take in sampobs (1019 samples), output a subsample with the chosen subtype, and the size specified.

```
#' Subsampling of the sampobs data.
sampobs2 <- BearSubsample(sampobs, subtype, size)
nrow(sampobs2) == size;size
```

```
## [1] TRUE
```

```
## [1] 450
```

```
head(sampobs2)
```

```
##        Sample Lab.ID  Class Individual Bundle.No. Period Event.ID Priority
## 128    60-4-1    943 sample       2213        191      4    60-01        4
## 411  13-27-05     13 sample   13-27-05         13      1    27-05        1
## 306  123-2-04    259 sample  12-106-02        123      3     2-04        2
## 73    26-1-12    932 sample       2213        180      1    26-12        4
## 350    16-5-2    935 sample  12-106-02        183      5    16-02        4
## 575 356-53-04    687 sample    17-2-02        356      6    53-04        1
##     Selected Bear.No.  X X.G X.U Left U.L     Comments X.1 Loci   CXX20
## 128        1       NA NA   3  30    A   L               NA    7 135.139
## 411        1       NA NA      30    C   U               NA   24 135.143
## 306        1       NA NA      30    B   L               NA    7 135.139
## 73         1       NA NA      30    C   L 1st envelope  NA    7 135.139
## 350        1       NA NA  10  NA    B   U               NA    7 135.139
## 575        1       NA NA      30    C   U               NA    7 133.135
##         G1D    G10M   MSUT2    MU50    G10X    G10P REN144.A06     G1A
## 128 174.184 206.216 197.205 120.132 141.145                 NA      NA
## 411 176.182 216.218 195.201 114.136 129.149 153.155    117.117 194.198
## 306 184.184 206.212 181.201 120.130 153.153                 NA      NA
## 73  174.184 206.216 197.205 120.132 141.145                 NA      NA
## 350 184.184 206.212 181.201 120.130 153.153                 NA      NA
## 575 172.186 210.212 191.197 128.132 135.141                 NA      NA
##        G10C     D1A    G10J    G10L    CPH9  CXX110    D123    MU26
## 128      NA      NA                     NA      NA      NA      NA
## 411 205.215 159.163 187.187 149.153 143.145 155.159 145.155 185.191
## 306      NA      NA                     NA      NA      NA      NA
## 73       NA      NA                     NA      NA      NA      NA
## 350      NA      NA                     NA      NA      NA      NA
## 575      NA      NA                     NA      NA      NA      NA
##        G10U    MU59 REN145.P07    G10B    MU23    G10H    Sex site cid
## 128      NA                 NA      NA      NA      NA 250.25   60   1
## 411 177.179 241.241    159.159 156.158 199.207 239.241 250.25   27   5
## 306      NA                 NA      NA      NA      NA 204.25    2   4
## 73       NA                 NA      NA      NA      NA 250.25   26  12
## 350      NA                 NA      NA      NA      NA 204.25   16   2
## 575      NA                 NA      NA      NA      NA 204.25   53   4
##     orig.ids
## 128      128
## 411      411
## 306      306
## 73        73
## 350      350
## 575      575
```

Normally, we would save the subsample Lab.ID's in the correct folder, but we'll skip this step because we won't actually be running the SECR model fitting in this example. Set file pathways to contain the temporary files needed to fit the secr model (sample data, tempSampobs.csv, and prox data, erroneously named tempCaphist.csv)

```
  #' Now, turn Period into a numeric variable
  #sampobs2$Period<-droplevels(sampobs2$Period)
  sampobs2$Period<-as.numeric(sampobs2$Period)
  table(sampobs2$Period)
```

```
##
##   1   2   3   4   5   6
##  37  88  76 112  81  56
```

```
  #' Drop several of the columns that we will not need.
  sampobs3<-sampobs2[,c(1:9, 16,18,42:44)]

  #' Setting csv pathways, Write this file out
  if (size==450){
  pathcsv1<-paste("../data/SubsamplingData/", subtype, "/TempSampobs.csv", sep = "")
  pathcsv2<-paste("../data/SubsamplingData/", subtype, "/TempCaphist.csv", sep = "")
  write.csv(sampobs3, file=pathcsv1, row.names=FALSE)
      }

  if (size!=450){
    pathcsv1<-paste("../data/SizeData/", subtype, "/TempSampobs.csv", sep = "")
    pathcsv2<-paste("../data/SizeData/", subtype, "/TempCaphist.csv", sep = "")
    write.csv(sampobs3, file=pathcsv1, row.names=FALSE)
      }
```

Takes in the sample observations (same as above) and output a prox data frame for BearParallelModelFit.

```
  #' Apply the capture history transformation necessary for secr models
  BearSumFunc(data=pathcsv1, summary = FALSE, output=pathcsv2)
  head(read.csv(pathcsv2))
```

```
##   BearMR X0056 X4 X81 F
## 1 BearMR  0056  3  92 F
## 2 BearMR  0056  4  93 F
## 3 BearMR  0056  2  96 F
## 4 BearMR  0056  6 102 F
## 5 BearMR  0056  4 103 F
## 6 BearMR  0056  5 103 F
```

Finally, actually run the model using the prox data output from above and the trapgrid that is used for every simulation. These are saved, as seen above, in FinalBearEst.csv and in Model folders as objects. Note that I am not actually running the model here, just for ease of knitting this document. Here is an examples of the csv line saved using model output.

```
  #' Run the model fitting and storage function.
  #BearParallelModelFit(subtype = subtype, iteration=iterNo, size= size)
  test<-read.csv("../data/SubsamplingData/FinalBearEst.csv")[30,]
  colnames(test)<-c("SubsampleID", "densEstimate","lcl", "ucl", "subtype", "Model")
  test
```

```
##    SubsampleID densEstimate      lcl      ucl    subtype   Model
## 30           6     12.27586 8.945905 16.84534 Spread.two Model D
```

And here is an example of the actual model object that is saved after each model fitting.

```
readRDS("../data/SubsamplingData/SimpleRandom/Model A/secr104.rds")
```

```
##
## secr.fit(capthist = bearCH, model = modelEval, buffer = 10, CL = TRUE,
##     trace = FALSE)
## secr 2.10.2, 20:05:38 28 Jan 2016
##
## Detector type     proximity
## Detector number   121
## Average spacing   1.078547 m
## x-range           -10.24721 8.170471 m
## y-range           -10.32148 9.661065 m
##
## N animals       :  38
## N detections    :  310
## N occasions     :  6
## Mask area       :  0.1224767 ha
##
## Model           :  g0~b + t + Sex sigma~Sex
## Fixed (real)    :  none
## Detection fn    :  halfnormal
## N parameters    :  10
## Log likelihood  :  -1203.684
## AIC             :  2427.368
## AICc            :  2435.516
##
## Beta parameters (coefficients)
##                  beta    SE.beta         lcl         ucl
## g0         -2.5405523 0.29033486 -3.10959814 -1.97150639
## g0.bTRUE    0.2040771 0.27502061 -0.33495343  0.74310756
## g0.t2       1.1651293 0.32054537  0.53687195  1.79338669
## g0.t3       0.9836750 0.36035138  0.27739923  1.68995068
## g0.t4       1.3624337 0.37795625  0.62165301  2.10321429
## g0.t5       1.0988556 0.38774081  0.33889758  1.85881364
## g0.t6       0.6827297 0.39761983 -0.09659084  1.46205026
## g0.SexM    -0.5222601 0.21702365 -0.94761867 -0.09690161
## sigma       0.9518281 0.06070928  0.83284006  1.07081607
## sigma.SexM  0.5751746 0.08457258  0.40941543  0.74093384
##
## Variance-covariance matrix of beta parameters
##                       g0      g0.bTRUE         g0.t2         g0.t3
## g0          0.084294332 -0.0103724219 -5.048276e-02 -0.0481124288
## g0.bTRUE   -0.010372422  0.0756363363 -4.175924e-02 -0.0603111481
## g0.t2      -0.050482756 -0.0417592404  1.027493e-01  0.0907332061
## g0.t3      -0.048112429 -0.0603111481  9.073321e-02  0.1298531158
## g0.t4      -0.046003546 -0.0707336318  9.651985e-02  0.1138666332
## g0.t5      -0.045428109 -0.0730439446  9.779063e-02  0.1157049606
## g0.t6      -0.047190793 -0.0739744850  9.822330e-02  0.1163708096
## g0.SexM    -0.032021469  0.0071982794 -5.053287e-03 -0.0066402786
## sigma      -0.006201258 -0.0013457914  3.993988e-04  0.0007653391
## sigma.SexM  0.006447876  0.0005965166 -9.270164e-05 -0.0003509171
##                     g0.t4         g0.t5         g0.t6      g0.SexM
```

```
## g0         -0.0460035460 -0.0454281092 -0.0471907933 -0.032021469
## g0.bTRUE   -0.0707336318 -0.0730439446 -0.0739744850  0.007198279
## g0.t2       0.0965198532  0.0977906285  0.0982232973 -0.005053287
## g0.t3       0.1138666332  0.1157049606  0.1163708096 -0.006640279
## g0.t4       0.1428509278  0.1258375599  0.1265791445 -0.008250145
## g0.t5       0.1258375599  0.1503429390  0.1288268960 -0.009013009
## g0.t6       0.1265791445  0.1288268960  0.1581015317 -0.006778850
## g0.SexM    -0.0082501450 -0.0090130094 -0.0067788502  0.047099263
## sigma       0.0008238526  0.0008053862  0.0013321246  0.006566707
## sigma.SexM -0.0003615833 -0.0002382688 -0.0007088851 -0.009316653
##                     sigma    sigma.SexM
## g0         -0.0062012583  6.447876e-03
## g0.bTRUE   -0.0013457914  5.965166e-04
## g0.t2       0.0003993988 -9.270164e-05
## g0.t3       0.0007653391 -3.509171e-04
## g0.t4       0.0008238526 -3.615833e-04
## g0.t5       0.0008053862 -2.382688e-04
## g0.t6       0.0013321246 -7.088851e-04
## g0.SexM     0.0065667075 -9.316653e-03
## sigma       0.0036856166 -3.670457e-03
## sigma.SexM -0.0036704574  7.152521e-03
##
## Fitted (real) parameters evaluated at base levels of covariates
##        link   estimate SE.estimate        lcl       ucl
## g0    logit 0.07306376  0.01966306 0.04271307 0.1222272
## sigma   log 2.59044083  0.15740881 2.29984117 2.9177596
```

```
  print(paste(subtype,iterNo, "done!"))
```

```
## [1] "SimpleRandom 999 done!"
```

XI. Other Scripts and Functions

a) BearModelFit.R and BearSubSECR.R

These are outdated versions of BearParallelModelFit.R and BearSubParallelSECR.R. As the names suggest, the newer versions of these functions employ parallel processing as a means to cut down on processing time.

b) Synthesisv2.R, Synthesisv3.R, and PilotFitSummary.Rmd

These are the first attempts of running all three of the functions one after the other, and can be thought of as precursors to BearSubParallelSECR.R. As such, they are also outdated.

c) VerificationSECR.rmd and VerificationSECR_corrected.Rmd

Originally meant to verify the model results that John produced for his progress report, this rmd file creates the 'full model' estimates using all 1019 samples for the 5 models I chose to analyze. The folder of the same name houses the csv with these estimates, as well as the RDS files of the model output for later use.

d) Data Exploration.R and Data ExplorationUpdated.R

These were written by John, and they analyze the original sample data. Much of this was incorporated into the very beginning of BearSubParallelSECR.R in order to make sure there were no discrepancies between how John and I aggregated the samples.

e) Data Summary.R

This script is as it sounds, is a summary of the bear data, mostly verified what Dave reported in the original bear study. Much of it is looking at the differences in individuals propensity to revisit traps and other individual heterogeneity.

f) huggins.R and HuggyBear.R

Huggins analysis of the bear data for the purpose of comparing estimates between closed pop models and secr. Much of HuggyBear.R was incorporated into BearHugginsParallel, with the only major difference being that the latter is a function that saves its model output and takes in subsamples.

g) SecrAll1.R and SecrAll2.R

These scripts run various SECR models for the purpose of inclusion in the progress report sent to Dave.