MOOSE ABUNDANCE ESTIMATION USING FINITE POPULATION BLOCK KRIGING ON TOGIAK NATIONAL WILDLIFE REFUGE, ALASKA

By

Graham G. Frye

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

Master of Science
in
Statistics

University of Alaska Fairbanks
December 2016

Abstract

Monitoring the size and demographic characteristics of animal populations is fundamental to the fields of wildlife ecology and wildlife management. A diverse suite of population monitoring methods have been developed and employed during the past century, but challenges in obtaining rigorous population estimates remain. I used simulation to address survey design issues for monitoring a moose population at Togiak National Wildlife Refuge in southwestern Alaska using finite population block kriging. In the first chapter, I compared the bias in the Geospatial Population Estimator (GSPE; which uses finite population block kriging to estimate animal abundance) between two survey unit configurations. After finding that substantial bias was induced through the use of the historic survey unit configuration, I concluded that the "standard" unit configuration was preferable because it allowed unbiased estimation. In the second chapter, I examined the effect of sampling intensity on performance of the GSPE. I concluded that bias and confidence interval coverage were unaffected by sampling intensity, whereas the coefficient of variation (CV) and root mean squared error (RMSE) decreased with increasing sampling intensity. In the final chapter, I examined the effect of spatial clustering by moose on model performance. Highly clustered moose distributions induced a small amount of positive bias, confidence interval coverage lower than the nominal rate, higher CV, and higher RMSE. Some of these issues were ameliorated by increasing sampling intensity, but if highly clustered distributions of moose are expected, then substantially greater sampling intensities than those examined here may be required.

Table of Contents

Introduction

Monitoring the size and demographic characteristics of animal populations is fundamental to the fields of wildlife ecology and wildlife management [Williams et al., 2002]. A diverse suite of population monitoring methods have been developed and employed during the past century. Early efforts relied heavily on the use of uncalibrated indices obtained through attempted censuses to approximate the states and dynamics of wildlife populations (i.e., raw counts of individuals across the entire area of interest). However, modern approaches commonly employ more rigorous estimation methods with data obtained through sampling (e.g., capture-mark-recapture [McCrea and Morgan, 2015]; distance sampling [Buckland et al., 1993]; repeat-visit point counts, [Royle, 2004]). Additionally, the use of spatial statistics to improve wildlife population estimation has become more popular during the past decade, as new techniques have been developed and associated software made available (e.g., spatial capture-recapture [Royle et al., 2014]).

Moose (*Alces alces*) populations in North America provide an illustrative example of the evolution of population monitoring methods over time. Moose are ecologically important herbivores in boreal systems [Kielland and Bryant, 1998], and are culturally valued as a source of food. As such, the development of reliable monitoring techniques has received considerable attention. Early moose management relied on aerial censuses, in which biologists attempted to count all individuals in a given region [Timmerman, 1974]. Such approaches were largely replaced by design-based probabilistic sampling efforts (e.g., [Gasaway et al., 1986]). More recently, model-based approaches, assuming a spatial stochastic process as the data-generating mechanism, have been developed. In Alaska and some other regions of North America, the most widely used population estimation method for moose at present is finite population block kriging (FPBK), developed by Ver Hoef [2002]. Advantages of FPBK over design-based methods include: increased precision, the ability to perform small-area estimation (i.e., estimation for a specific subset of units in the sample frame), and the ability to employ non-random sampling designs [Ver Hoef, 2008].

Overview of Finite Population Block Kriging

Generally, kriging can be described as a geostatistical method that combines information on spatial trend and spatial correlation structure for the purpose of prediction [Cressie, 1991]. Unlike some other interpolation methods, kriging provides optimal interpolation based on mean squared prediction

error (MSPE). The following overview of kriging procedures closely follows the treatments and notation of [Ver Hoef, 2002] and [Ver Hoef, 2008], and partially those of [Cressie, 1991] and [Banerjee et al., 2015].

Variogram estimation[1]

Fundamental to the kriging process is selection and estimation of a variogram model, which approximates the spatial correlation structure in a given system. Common assumptions in the use of variogram models are second-order stationarity and isotropy. More explicitly:

(1) The process mean is the same at all points in the spatial region of interest: $\mathbb{E}[Z(\mathbf{s})] = \mu$, where $Z(\mathbf{s})$ is a random variable at location $\mathbf{s}$, and $\mathbf{s}$ is a two-dimensional vector of coordinates.

(2) The spatial covariance depends only on the distances, $|\mathbf{h}|$, between points, and not on the exact locations of points:

$$C(\mathbf{h}) = C[Z(\mathbf{s}), Z(\mathbf{s}+\mathbf{h})]. \tag{1}$$

(3) Similarly, the covariance depends only on $|\mathbf{h}|$ and does not depend upon the directional relationship between points.

A variety of standard variogram models exist, including exponential, spherical, Gaussian, and Matern. Choice of variogram models is important and can affect spatial predictions [Mazzella and Mazzella, 2013]. An essential characteristic of selected variogram models is that they must yield a valid variance-covariance matrix. The standard theoretical models (e.g., exponential, spherical) guarantee that this condition is met. An example of an exponential semivariogram fit to fictitious data, with the empirical semivariogram overlaid is depicted in Figure 1. Converting between covariogram and semivariogram formulations of a variogram model is accomplished through the relationship

$$\gamma(\mathbf{h}) = C(0) - C(\mathbf{h}), \tag{2}$$

where $\gamma(\mathbf{h})$ is the semivariance at distance $= |\mathbf{h}|$, $C(0)$ is the covariance at distance $= 0$, and $C(\mathbf{h})$ is the covariance at distance $= |\mathbf{h}|$.

[1] I use the term "variogram" here to refer generically to the group of models that can be fit as covariograms and semivariograms. When refering to one specific formulation (e.g., semivariogram), I use the more specific term. Usage of these terms varies somewhat among authors.

Variogram parameter estimation is often conducted with restricted maximum likelihood (REML; [Patterson and Thompson, 1974]), which is reported to be less biased in spatial analyses than full maximum likelihood [Mardia and Marshall, 1984]. Weighted least squares is another commonly used estimation approach.

Block kriging

Generally speaking, ordinary and universal kriging use information from sampled points to interpolate values of the variable of interest at unsampled points. Block kriging employs the same principle, but the kriging units are areal, rather than points. For simplification, distances between units are often quantified as the distances between block centroids. Following Cressie [1991] and Ver Hoef [2008], we begin with a linear model for our data, $\mathbf{z}$:

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\varepsilon}, \tag{3}$$

where $\boldsymbol{\mu} = \mathbf{X}\boldsymbol{\beta}$ if covariates are to be included (i.e., "universal" block kriging), $\mathbf{X}$ is the typical $n \times p$ dimensional design matrix and $\boldsymbol{\beta}$ is a $p \times 1$ dimensional parameter vector. The random errors, $\boldsymbol{\varepsilon}$, exhibit second-order stationarity, as defined above, with $\mathbb{E}[\varepsilon(\mathbf{s})] = 0$, and $\text{Var}(\boldsymbol{\varepsilon}) = \Sigma$. For standard block kriging, we average the value of interest from a continuous spatial process, $Z(\mathbf{s})$, over the area of the block, B:

$$Z(B) \equiv \int_B Z(\mathbf{s})ds/|B|, \tag{4}$$

where $|B|$ is the area of B and the expectation of the process is

$$\mathbb{E}[Z(B)] \equiv \mu(B) \equiv \int_B \mu(\mathbf{s})ds/|B|. \tag{5}$$

Using standard block kriging, we minimize the MSPE,

$$\mathbb{E}[\boldsymbol{\lambda}'\mathbf{z} - Z(B)]^2, \tag{6}$$

to solve for the kriging weights, $\boldsymbol{\lambda}$. If we sample at n sites and consider p predictors, the estimator for the mean of this spatial process over B is

$$\hat{Z}(B) = \boldsymbol{\lambda}'\mathbf{z} = \mathbf{c}_B'\Sigma^{-1}(\mathbf{z} - \hat{\boldsymbol{\mu}}) + \hat{\mu}_B, \tag{7}$$

where:

$\mathbf{c}_B = [c_1(B), c_2(B), ..., c_n(B)]'$ with $c_i(B) \equiv \int_B C(\mathbf{s} - \mathbf{s}_i)d\mathbf{s}/|B|$ for i = 1,2,...,n,

$\hat{\boldsymbol{\mu}} = \mathbf{X}\hat{\boldsymbol{\beta}}$,

$\hat{\mu}_B = \mathbf{x}'_B\hat{\boldsymbol{\beta}}$,

$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\Sigma^{-1}\mathbf{X})^{-1}\mathbf{X}'\Sigma^{-1}\mathbf{z}$, i.e., the generalized least squares estimator of $\boldsymbol{\beta}$,

$\mathbf{x}_B = [x_1(B), x_2(B), ..., x_n(B)]'$, with $x_j(B) \equiv \int_B x_j(\mathbf{s})d\mathbf{s}/|B|$ for j = 1,2,...,p.

The corresponding MSPE is

$$\text{Var}[\hat{Z}(B)] = \mathbb{E}[\boldsymbol{\lambda}'\mathbf{z} - Z(B)]^2 = \sigma_B^2 - \mathbf{c}\Sigma^{-1}\mathbf{c}_B + \mathbf{d}'_B(\mathbf{X}'\Sigma^{-1}\mathbf{X})^{-1}\mathbf{d}_B, \qquad (8)$$

where σ_B^2 is the variance within B and $\mathbf{d}_B = (\mathbf{x}_B - \mathbf{X}'\Sigma^{-1}\mathbf{c}_B)$.

Block kriging for infinite populations and variants of the kriging equations in general are discussed in greater depth by [Journel and Huijbregts, 1978] and [Cressie, 1991], among others.

Finite population block kriging

Finite-population methods are frequently used in wildlife management and related fields. The goal with finite-population inference is to estimate the value of a particular realization of some stochastic process when the sample frame is composed of a finite and countable number of experimental units. In the case of plot-based sampling, this means that sampled plots (m) are selected from a known number of plots composing the sample frame (M). The information on the proportion of plots sampled ($\frac{m}{M}$) is used to adjust variance estimates. Ver Hoef [2002] proposed a finite population version of block kriging (FPBK), which is appropriate for plot-based sampling situations in which inference is limited to a particular realization of a given process on a finite spatial lattice. The fundamental difference between block kriging and FPBK is that the focus in the former is on estimation of some unknown parameter (e.g., the population mean) of the data-generating process. In contrast, the focus in FPBK is on predicting the actual values (or some function thereof) that were realized by the data-generating process within the sample frame.

In FPBK, we start by considering a basic linear model similar to that in Equation 3,

$$\begin{pmatrix} \mathbf{z}_s \\ \mathbf{z}_u \end{pmatrix} = \begin{pmatrix} \mathbf{X}_s \\ \mathbf{X}_u \end{pmatrix} \boldsymbol{\beta} + \begin{pmatrix} \boldsymbol{\varepsilon}_s \\ \boldsymbol{\varepsilon}_u, \end{pmatrix} \tag{9}$$

where the data vector, $\mathbf{z}$, is now divided into sampled, $\mathbf{z}_s$, and unsampled, $\mathbf{z}_u$, components. Similarly, the design matrix, $\mathbf{X}$, is composed of sampled and unsampled components ($\mathbf{X}_s$ and $\mathbf{X}_u$, respectively), as is the random error term, $\boldsymbol{\varepsilon}$ ($\boldsymbol{\varepsilon}_s$ and $\boldsymbol{\varepsilon}_u$, respectively). $\mathbf{z}_s$ and $\boldsymbol{\varepsilon}_s$ are $n \times 1$ vectors, whereas $\mathbf{z}_u$ and $\boldsymbol{\varepsilon}_u$ are $(N-n) \times 1$ vectors. Similarly, $\mathbf{X}_s$ and $\mathbf{X}_u$ are $n \times p$ and $(N-n) \times p$ matrices, respectively. The key difference thus far is that we know the dimension of the unsampled portion of the spatial lattice, whereas in standard block kriging we know only how many blocks are sampled.

Analagous to the previous model,

$$\mathbb{E}\begin{pmatrix} \boldsymbol{\varepsilon}_s \\ \boldsymbol{\varepsilon}_u \end{pmatrix} = 0 \quad \text{and} \quad \text{Var}\begin{pmatrix} \boldsymbol{\varepsilon}_s \\ \boldsymbol{\varepsilon}_u \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Sigma}_{ss} & \boldsymbol{\Sigma}_{su} \\ \boldsymbol{\Sigma}_{us} & \boldsymbol{\Sigma}_{uu} \end{pmatrix},$$

We define a $M \times 1$ vector, $\mathbf{b} = \{\mathbf{b}_s, \mathbf{b}_u\}'$, that will weight the data vector, $\mathbf{z}$, to provide the form of plot-level predictions in which we are interested. In the case of animal abundance, we would define $\mathbf{b} = \{1, 1, ..., 1\}'$, which will yield predictions of the number of individuals in each cell and, ultimately the total number of individuals in the sample frame.

The FPBK abundance predictor ($\hat{N}$) is then found by minimizing the MSPE, $\mathbb{E}[\boldsymbol{\lambda}'\mathbf{z}_s - \mathbf{b}'\mathbf{z}]^2$, yielding

$$\hat{N} = \mathbf{b}_s'\mathbf{z}_s + \mathbf{b}_u'\hat{\mathbf{z}}_u. \tag{10}$$

The unknown component of this expression is predicted with

$$\hat{\mathbf{z}}_u = \boldsymbol{\Sigma}_{su}\boldsymbol{\Sigma}_{su}^{-1}(\mathbf{z}_s - \hat{\boldsymbol{\mu}}_s) + \hat{\boldsymbol{\mu}}_u, \tag{11}$$

where:

$\hat{\boldsymbol{\mu}}_s = \mathbf{X}_s\hat{\boldsymbol{\beta}}$, when covariates are included,

$\hat{\boldsymbol{\mu}}_u = \mathbf{X}_u\hat{\boldsymbol{\beta}}$, when covariates are included,

$\hat{\boldsymbol{\beta}} = (\mathbf{X}_s'\Sigma_{ss}^{-1}\mathbf{X}_s)^{-1}\mathbf{X}_s'\Sigma_{ss}^{-1}\mathbf{z}_s$, i.e., the generalized least squares estimator of $\boldsymbol{\beta}$.

The MSPE for $\hat{N}$ is

$$\text{Var}(\hat{N}) = \mathbb{E}[\boldsymbol{\lambda}'\mathbf{z}_s - \mathbf{b}'\mathbf{z}]^2 = \mathbf{b}'\boldsymbol{\Sigma}\mathbf{b} - \mathbf{c}_b'\boldsymbol{\Sigma}_{ss}^{-1}\mathbf{c}_b + \mathbf{d}_b'(\mathbf{X}_s'\boldsymbol{\Sigma}_{ss}^{-1}\mathbf{X}_s)^{-1}\mathbf{d}_b, \tag{12}$$

where:

$$\mathbf{c}_b = \mathbf{\Sigma}_{ss}\mathbf{b}_s + \mathbf{\Sigma}_{ss}\mathbf{b}_u,$$

$$\mathbf{d}_b = \mathbf{X}'_s\mathbf{b}_s + \mathbf{X}'_u\mathbf{b}_u - \mathbf{X}'_s\mathbf{\Sigma}_{ss}^{-1}\mathbf{c}_b.$$

Ver Hoef [2008] discusses FPBK and its derivation in greater depth.

The Geospatial Population Estimator

Ver Hoef (2008) developed an abundance estimator based on FPBK, which is widely known as the Geospatial Population Estimator (GSPE; [Kellie and Delong, 2006]). Throughout this thesis, I use the term "GSPE" to refer to the specific implementation of FPBK developed for animal abundance estimation by Ver Hoef (Appendix 4; see moose example in [Ver Hoef, 2008]) and adopted by the Alaska Department of Fish and Game (ADF&G) as the standard analysis procedure for moose survey data. The GSPE applies FPBK to counts from sampled survey units in a specified spatial region to predict abundance in unsampled survey units, thereby facilitating an estimate of total abundance across the spatial region of interest. The centroids of each unit are used to quantify proximity of units for fitting an exponential semivariogram:

$$\gamma(\mathbf{h}) = c_0 + c_e[1 - \exp(-||\mathbf{h}||/a_e)], \tag{13}$$

where $\gamma(\mathbf{h})$ is the semivariance, $\mathbf{h}$ is a vector of distance lags, $c_0 \geq 0$ is the nugget parameter, $c_e \geq 0$ is the partial sill parameter, and $a_e \geq 0$ is the range parameter [Cressie, 1991]. Semivariogram parameters are estimated using REML.

Density-based stratification is a component of the GSPE. Generally, geostatistical procedures assume that the spatial correlation structure is constant throughout the region of interest (stationarity). In systems where there may be mutliple spatial processes generating the data (as is possible with non-uniform animal distributions), stratification can help to ensure that this assumption is sufficiently satisfied. In the case of the GSPE, units are partitioned into two strata prior to sampling on the basis of anticipated density and each stratum is assumed to have an independent spatial stochastic data-generating process. Predictions are made separately within each stratum (with separately fit semivariograms) and stratum-specific predictions are combined in a manner equivalent to that used in stratified random sampling when there is no spatial correlation structure [Ver Hoef, 2008]; also see [Scheaffer et al., 1996].

Cross-correlation between strata can also be incorporated into the MSPE, but Ver Hoef [2008] concluded that such cross-correlation is either non-existent or trivially small with moose abundance data.

Assumptions of the GSPE

Several assumptions are required for valid inference from the GSPE. As previously stated, isotropy and second-order stationarity are assumed to exist in the spatial region of interest. Additionally, it is assumed that the theoretical exponential semivariogram (Equation 13) model is an accurate representation of the spatial correlation structure in the system. Another important assumption is that the moose in each sampled survey unit are perfectly enumerated, as inaccurate counts will yield inaccurate interpolations in unsampled units via mis-specified semivariograms. This assumption is easily satisfied in simulation studies, but may not be adequately met in many field survey scenarios. I assume perfect enumeration here, and do not address complications associated with imperfect detection of animals under field conditions. For detailed discussion of sightability models for dealing with imperfect detection, see [Ver Hoef, 2009], [Christ, 2011], and [Seaton, 2014].

An application of FPBK

The Togiak National Wildlife Refuge (TNWR) is located in southwestern Alaska, adjacent to Bristol Bay (Figure 2). Moose numbers on TNWR were historically low, with fewer than 30 occurring on the refuge in the early 1980s, and 84 counted during surveys in 1994. Since that time, numbers have increased dramatically, with the most recent survey (2011) resulting in a count of 1,626. The United States Fish and Wildlife Service (USFWS), which administers the refuge, has attempted to conduct an aerial census of the refuge annually since 1995. With approximately 1.7 million acres encompassed by TNWR, censuses require substantial effort and are seldom successfully completed due to time, weather, and manpower constraints. Consequently, the USFWS has identified a need to replace the annual census with a sampling-based approach for estimating moose population size, so that surveys can be completed annually with a reasonable amount of time and effort. Given its success in population estimation for moose elsewhere in Alaska, application of the GSPE to moose counts from a sample of survey units was identified as a reasonable alternative to annual censuses.

Objectives

The purpose of this study is to assess the performance of the GSPE on TNWR under a variety of simulated scenarios. I have three specific objectives:

(1) Assess the performance of the GSPE for moose abundance estimation under two alternative survey unit configurations at TNWR.

(2) Assess the influence of sampling intensity on the performance of the GSPE at TNWR.

(3) Assess the effect of clustered moose distributions on the performance of the GSPE at TNWR.

Figures

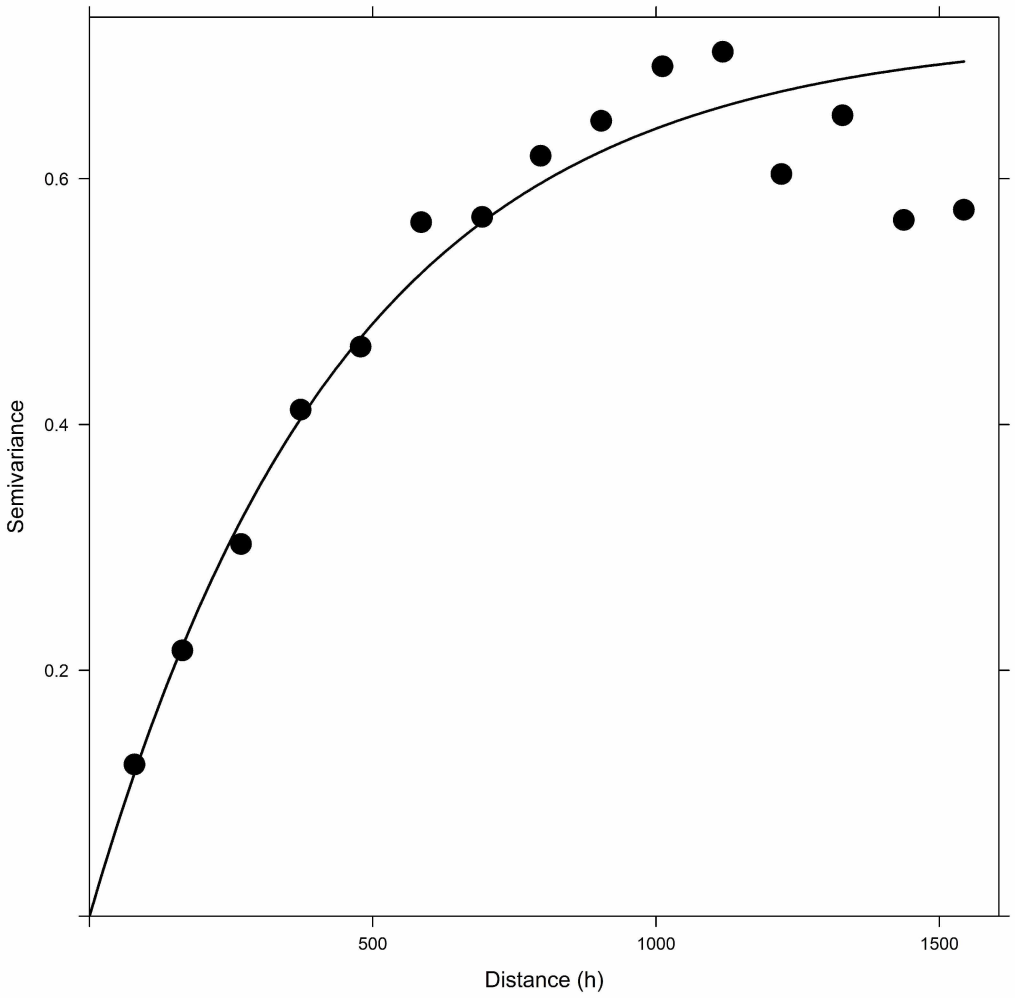


Figure 1: A theoretical exponential semivariogram (line) fit to fictitious data. The dots are the associated empirical semivariogram.

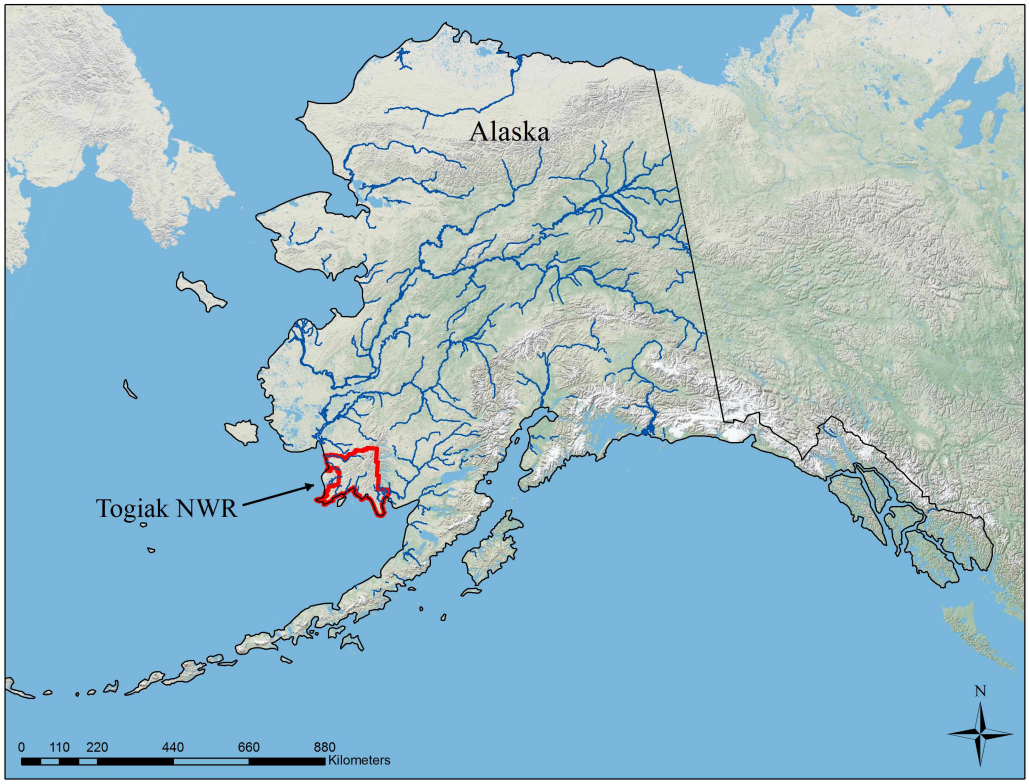


Figure 2: Location of Togiak National Wildlife Refuge.

Chapter 1: Comparing bias of the Geospatial Population Estimator between survey unit configurations at Togiak National Wildlife Refuge

Introduction

The Alaska Department of Fish and Game (ADF&G), for whom the GSPE was originally developed, established a standard grid of units for moose surveys in Alaska and northern Canada (Figure 3). The dimensions of these survey units are 2' latitude by 5' longitude, which yields unit areas ranging from approximately 13.5 km^2 in northern Alaska to 20.1 km^2 in west-central British Columbia (differences in area of the units is the result of convergence of longitudinal axes at the earth's poles). Latitudinal change in area of standard units is gradual, so unit areas are relatively uniform within specific study sites. This grid is used as the basis for moose surveys throughout most of Alaska, on both state and federal lands. Consequently, most development and assessment efforts involving application of the GSPE to moose population estimation have relied on these standard unit configurations. However, on TNWR, survey units that were used historically differ dramatically from the standard units in both shape and size (Figure 4). The historic unit boundaries follow topographical features and exclude elevations above 1000 ft as well as large bodies of water. The resulting sample frame is composed of units with diverse shapes and areas ranging from 8.8 km^2 to 433.8 km^2. In contrast, the standard units on TNWR are approximately equal in size (range: 17.3 km^2 - 17.9 km^2) and shape (Figure 3). Wildlife biologists from the USFWS were interested in maintaining their historic survey unit configuration for consistency and because of their familiarity with the historic units, but they wanted to be sure that this non-standard configuration would not cause estimates to be unreliable. To address this concern, I developed a spatially explicit simulation approach designed to compare the bias of the GSPE using the ADF&G standard grid to that using the historic survey unit configuration on TNWR.

Methods

Sample frame

For the purpose of this simulation study, I used the most recently (2011) surveyed subset of historic units to delineate the sample frame (Figure 5). The

primary reason for delineating the sample frame in this way is that previous counts could be used as *a priori* knowledge of the expected moose densities in each unit, which is useful for stratification (discussed subsequently). In order to make the standard and historic unit configurations comparable, I clipped the standard survey unit grid to match the bounds of the sample frame (Figure 6). Clipping the standard grid in this way resulted in additional heterogeneity in the size of some standard units, but significantly less than that in the historic unit configuration.

Stratification

Moose densities often vary substantially among survey units within specific study areas. Typically, there are a large number of survey units with no or few moose and a smaller number of units with higher moose densities. An effective way of dealing with this variation in counts is to partition the sample frame into "high" and "low" density strata. Previous survey data or pilot studies can be used as the basis for stratification prior to conducting surveys. The current recommendation from ADF&G is to assign units with densities lower than approximately $0.2/km^2$ to the low density stratum and units with densities higher than that to the high density stratum [Kellie and Delong, 2006], although this stratification cut-point will vary with study site. For this simulation study, I used the most recent survey data from TNWR (2011) as the basis for stratification. Because available survey data were grouped by historic sample units only, I delineated strata using the historic units (Figure 7). For the standard unit configuration, I defined all standard units that overlapped high-stratum historic units to be high-stratum standard units and all others to be low-stratum standard units (Figure 8). Re-defining the strata for the standard configuration was necessary because it is important that each unit occurs in only one stratum. Because the true strata were delineated using historic unit boundaries, this re-delineation of stratum bounds for the standard grid necessarily results in some high-stratum standard units that overlap the true stratum boundaries. Thus the high stratum is slightly larger in the standard unit configuration than in the historic unit configuration.

Simulations

I used a spatially explicit approach to repeatedly generate spatial distributions of simulated moose populations within the sample frame. I separated each historic stratum into a separate polygon layer using GIS applications in R ([R Development Core Team, 2015]; also see packages listed below). Within

each of these stratum polygons (high, low), I generated a specified number of randomly distributed moose locations (Figure 9). Each individual location had a set of coordinates associated with it and fell within the bounds of the appropriate stratum polygon. A new layer of polygons corresponding to the individual survey unit boundaries was then overlaid on this simulated moose population and used to tally the number of moose occurring within each individual unit, which is analogous to the counting of moose that occurs aerially during field surveys (Figure 9). Centroid coordinates were then computed for each survey unit polygon. After tallying moose abundance in each unit and computing centroid coordinates, I randomly selected a specified number of units from each stratum to serve as the sample. For each simulated moose population, this procedure was conducted once with the standard survey grid and once with the historic grid. The number of sampled units in each configuration was selected in such a way that the sampled areas per stratum were comparable across unit configurations (i.e., the area sampled in the low stratum of the standard configuration approximately matched that sampled in the low stratum of the historic configuration, and the area sampled in the high stratum of the standard configuration approximately matched that sampled in the high stratum of the historic configuration). Ultimately, for each simulated moose population, this procedure yielded one dataset for each of the two unit configurations. Each dataset contained the following data for each unit: (1) a unit identifier, (2) number of moose counted, (3) area of unit, (4) stratum of unit, (5) latitude of unit centroid, (6) longitude of unit centroid, (7) binary indicator of the unit being sampled or not, (8) binary indicator of the unit being included in the final abundance estimate. FPBK was then conducted with each dataset, providing an estimate of abundance and an associated confidence interval for each unit configuration.

This entire procedure was repeated for a specified number of iterations using a for loop. The results from each iteration of the loop were stored in matrix objects in R, then used to compute bias of the GSPE with the different unit configurations following completion of the loop. The bias for each configuration was estimated as:

$$\widehat{B(\hat{N})} = \frac{1}{k}\sum_{i=1}^{k}\hat{N}_i - N \tag{14}$$

where k is the number of simulated populations, $\hat{N}_i$ is the GSPE estimate for the i^{th} simulated population, and N is the true abundance of the population. The bias estimate was then converted to relative bias:

$$\text{Relative bias} = \frac{\widehat{B(\hat{N})}}{N} \tag{15}$$

I simulated a range of moose densities to evaluate the bias of the GSPE in each unit configuration under a variety of abundance scenarios (Table 1). Simulated total abundance ranged from 500 to 14000, with a range of 50 to 5000 in the low stratum and 450 to 9000 in the high stratum (Table 1).

To determine the number of iterations to use in each simulation loop, I computed abundance estimates for two populations at the extremes of the range of simulated abundances (500 and 14000). I used loops with 50, 100, 500, 1000, 1500, 2000, 5000, and 10000 iterations with each unit configuration and plotted the mean abundance estimate for each. I visually assessed convergence of estimates towards a stable value to select the final number of iterations for the simulation study.

Simulations and spatial data manipulation were conducted using R with several packages developed for spatial analysis (maptools [Bivand and Lewin-Koh, 2015], rgdal [Bivand et al., 2015], rgeos [Bivand and Rundel, 2015], sp [Roger S. Bivand, 2013], spatstat [Baddeley and Turner, 2005]). ArcGIS 10.0 [ESRI, 2011] was used for visual presentation of spatial data. For each simulated population, the GSPE was implemented using code written in R by Jay ver Hoef for ADF&G (Appendix 4). Variograms were fit using restricted maximum likelihood estimation (REML, [Patterson and Thompson, 1974]). The R script for the simulation loops is available in Appendix 1.

Results

Plots of mean abundance estimates suggested that results stabilized within 1000 iterations (Figure 10), so I chose to use loops with 1000 iterations.

Simulation results indicated that there was substantial systematic bias in the GSPE when using the historic survey unit configuration from TNWR (Table 1, Figure 11). The magnitude of the bias was so great that the true values of abundance did not even overlap the range of GSPE estimates, with one exception (Figure 11). In the single case where the true value did fall within the range of GSPE estimates, it was at the extreme lower tail of the distribution. In contrast, the estimator appeared to be unbiased when used with the standard survey unit configuration (Table 1, Figure 12).

Bias with the historic unit configuration was consistently positive, with a range of 102.5 to 2163.0. Bias estimates ranged from and -9.1 to 36.8 with the standard configuration. Expressed as a percentage of the true value (relative

bias), bias ranged from 14.4% to 21.7% with the historic configuration, and -0.2% to 0.4% with the standard configuration. 99.3-100% of estimates were higher than the true estimate using the historic configuration, whereas only 47.4-53.3% of estimates were higher than the true estimate with the standard configuration (Table 1).

Discussion

Clipping the grid of standard sample units to match the historic sample units resulted in partial units being included in the standard grid (Figure 6). Thus, some size heterogeneity was neccessarily created in the standard grid by matching the extent of the unit configurations. This heterogeneity was small relative to that in the historic unit configuration and did not appear to induce bias (Table 1, Figure 12).

Another issue arising from attempting to match the characteristics of unit configurations is that some standard units classified as high-stratum actually overlapped the true stratum boundary. Thus, there is some stratum classification error in the standard grid. The effect of mis-classifying units to the wrong stratum in this case would be primarily to reduce the precision of estimates by contaminating the high stratum with some area that truly belongs to the low stratum. It is possible that this mis-classification inflated the standard errors of our standard-unit estimates slightly, but the mis-classified area is relatively small, so the effect should be minimal.

Although area of the individual survey units is an input used when implementing the estimator, it appears that it is only used to provide an approximate correction for the slight variation in standard unit areas attributable to latitudinal changes within a given survey area (see lines 140-143 and 162-163 in Appendix 4), rather than accounting for major size differences among units. In fact, artificially assigning uniform areas to the historic units when implementing the GSPE appears to eliminate the bias observed in this study, which verifies that the source of bias is heterogeneity in unit size.

Another issue that could cause problems when using the historic survey unit configuration at TNWR is the relatively small number of units in each stratum. For the GSPE, a minimum of 20 survey units are required in each stratum in order to estimate the covariance structure within strata, with $\geq$ 30 strongly recommended by current practitioners [Kellie and Delong, 2006]. However, using the most recent data to delineate strata yielded only 25 units in the high stratum and 129 in the low stratum. Some historic units were excluded from the simulation sample frame because they were not surveyed in 2011 (see Figures 4 and 5), but even when including them, a relatively small

number of units would be available in each stratum.

Given these problems with applying the GSPE to the historic unit configuration and the comparatively good performance of the estimator with the standard grid, my recomendation is that TNWR switch to using the standard grid for future surveys. Although it is possible that the GSPE could be extended to accommodate heterogeneity in unit size, the issue of having too few units per stratum in the historic configuration would persist. Moreover, if biologists wish to compare past survey results for particular groups of historic units to those of new surveys conducted under the standard configuration, this can be accomplished via small area estimation with the GSPE [Ver Hoef, 2008]. This would simply require the investigator to specify the extent of the reduced estimation area when implementing the GSPE.

Future Work

Refining the GSPE to accommodate greater heterogeneity in survey unit size would be a useful extension of this method. This would entail converting the moose counts in each survey unit to densities prior to performing the kriging, and then back transforming to counts while accounting for the area of each individual unit when the interpolation is complete. This modification will have to be included in the point estimate and MSPE components of the current implementation.

The GSPE is currently implemented using only the exponential semivariogram. Although this form of spatial correlation appears to perform well (in terms of bias) when estimating moose abundance under the simulated scenarios, it would be informative to examine the performance of alternative theoretical semivariogram forms (e.g., spherical, Gaussian) relative to that of the exponential. Selection of an appropriate semivariogram model is a potentially important component of kriging-based spatial analyses [Van Groenigan, 2000, Mazzella and Mazzella, 2013].

A common issue in wildlife field surveys is imperfect detection. The GSPE does not address this problem, but rather assumes moose are perfectly enumerated in all sampled units. This assumption is easily met in the context of simulation exercises, but it much less realistic in the context of field surveys. Extensions of the GSPE are presently being developed and tested to address this problem. Specifically, researchers are presently working to incorporate sightability models into the GSPE framework that will correct estimates of abundance and precision for imperfect detection using separately derived estimates of detection error [Ver Hoef, 2009, Christ, 2011, Seaton, 2014].

Tables

Table 1: Bias of the Geospatial Population Estimator for different simulated moose abundances in standard and historic survey unit configurations at Togiak National Wildlife Refuge. For each abundance level, 1000 simulated populations were generated for each survey unit configuration. The unit of measurement is number of moose. Percent greater than N represents the proportion of the 1000 estimates that were greater than the true value.

Total	*High Stratum*	*Low Stratum*	*Standard Bias*	*Historic Bias*	*Standard % > N*	*Historic % > N*
500	450	50	0.1	108.5	50.6%	100%
700	450	250	0.3	102.5	49.8%	99.3%
1000	900	100	-2.2	216.7	49.1%	100%
1400	900	500	4.0	204.4	46.7%	100%
2000	1800	200	-3.4	432.5	49.5%	100%
2800	1800	1000	-3.7	404.5	51.9%	100%
5000	4500	500	-9.1	1080.4	47.4%	100%
7000	4500	2500	-5.3	1009.6	48.6%	100%
10000	9000	1000	36.8	2163.0	53.3%	100%
14000	9000	5000	-6.8	2016.2	49.1%	100%

Figures

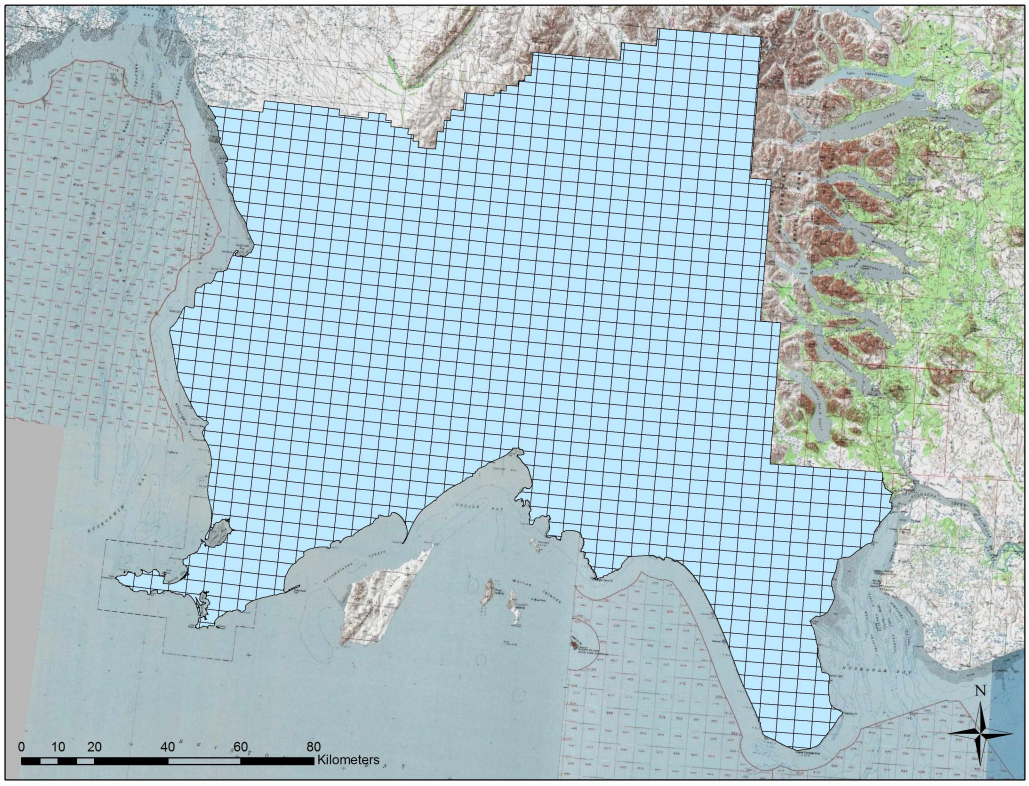


Figure 3: Standard survey unit configuration on Togiak National Wildlife Refuge.

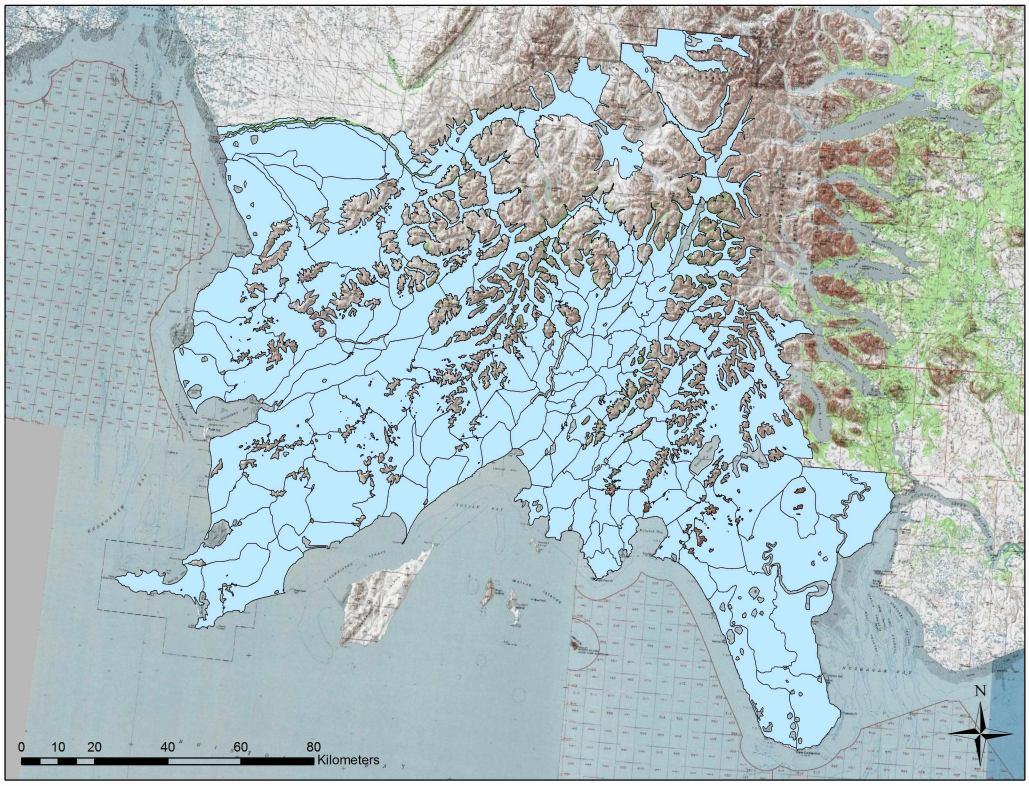


Figure 4: Historic survey unit configuration on Togiak National Wildlife Refuge.

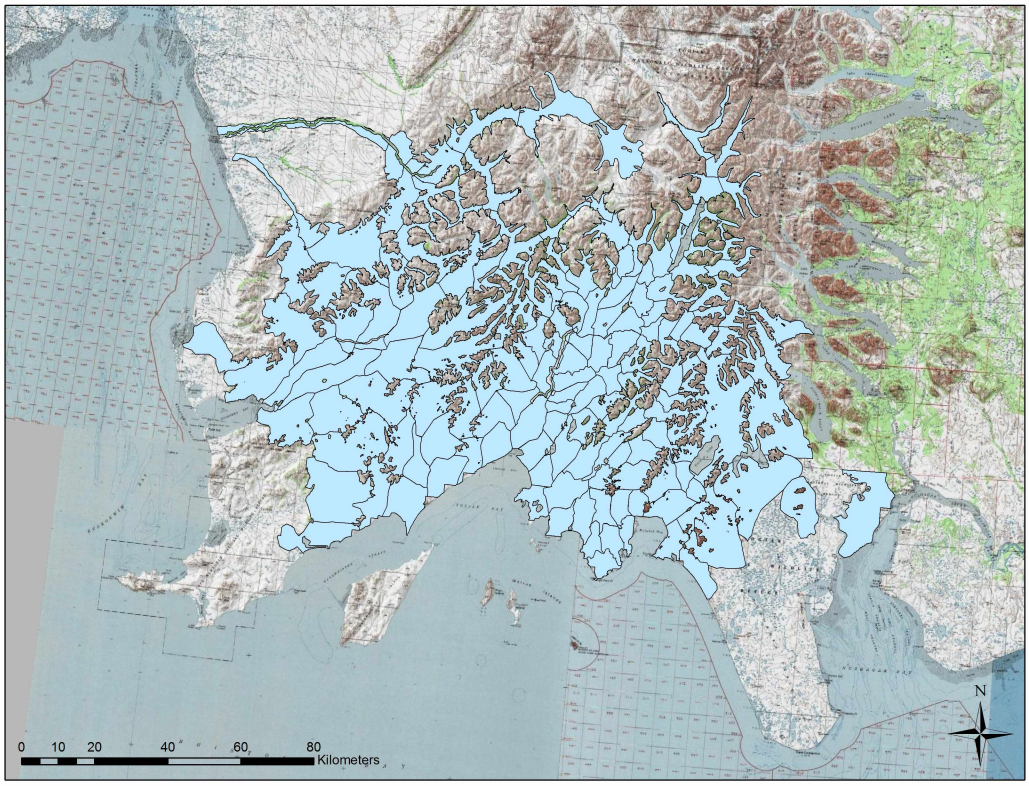


Figure 5: Subset of historic survey units used to delineate the simulation sample frame at Togiak National Wildlife Refuge.

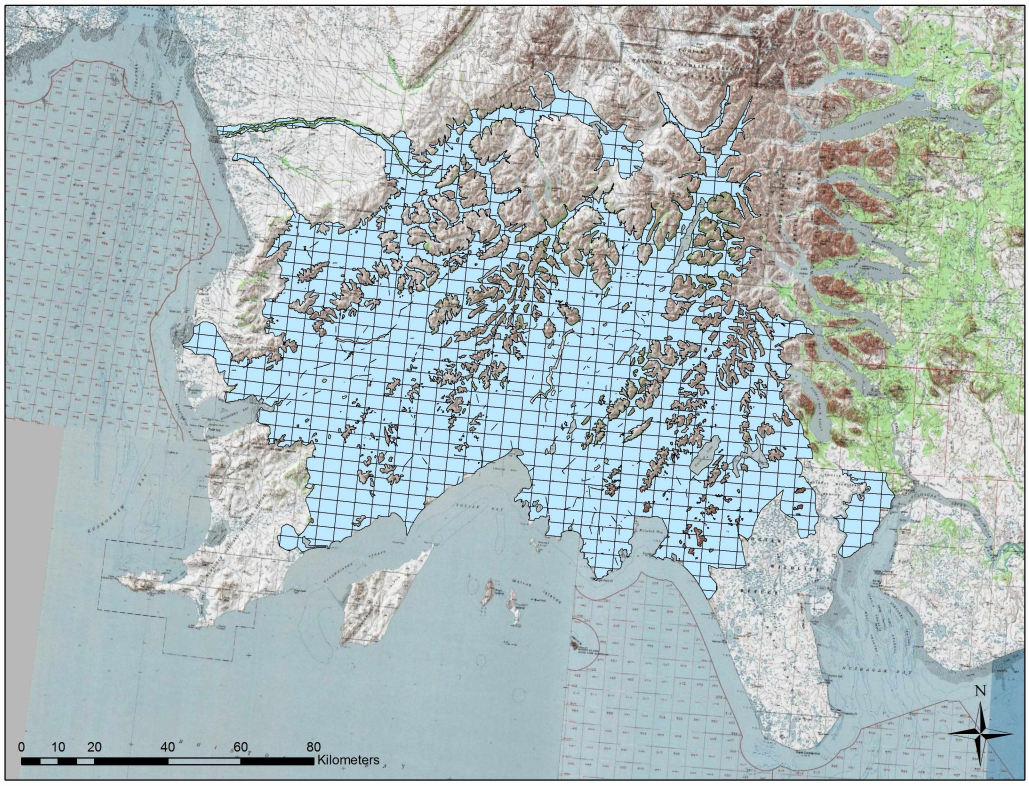


Figure 6: Standard survey units clipped to match the simulation sample frame at Togiak National Wildlife Refuge.

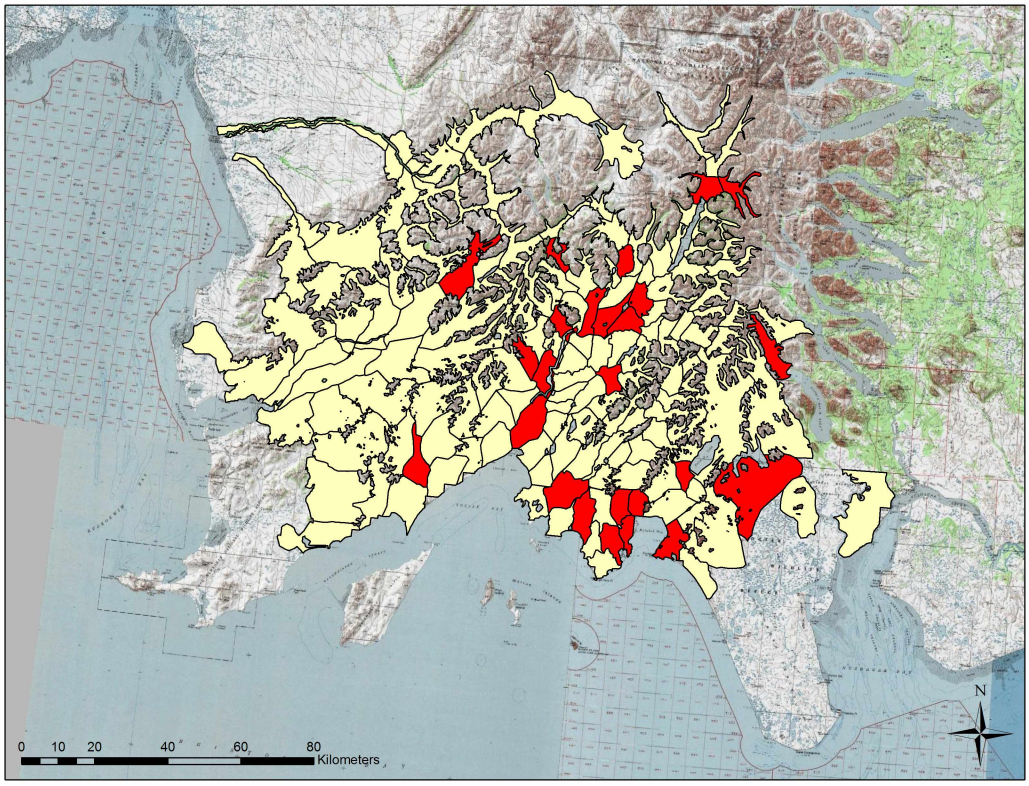


Figure 7: Historic survey units stratified by expected moose density at Togiak National Wildlife Refuge. Red units are in the high density stratum. Yellow units are in the low density stratum.

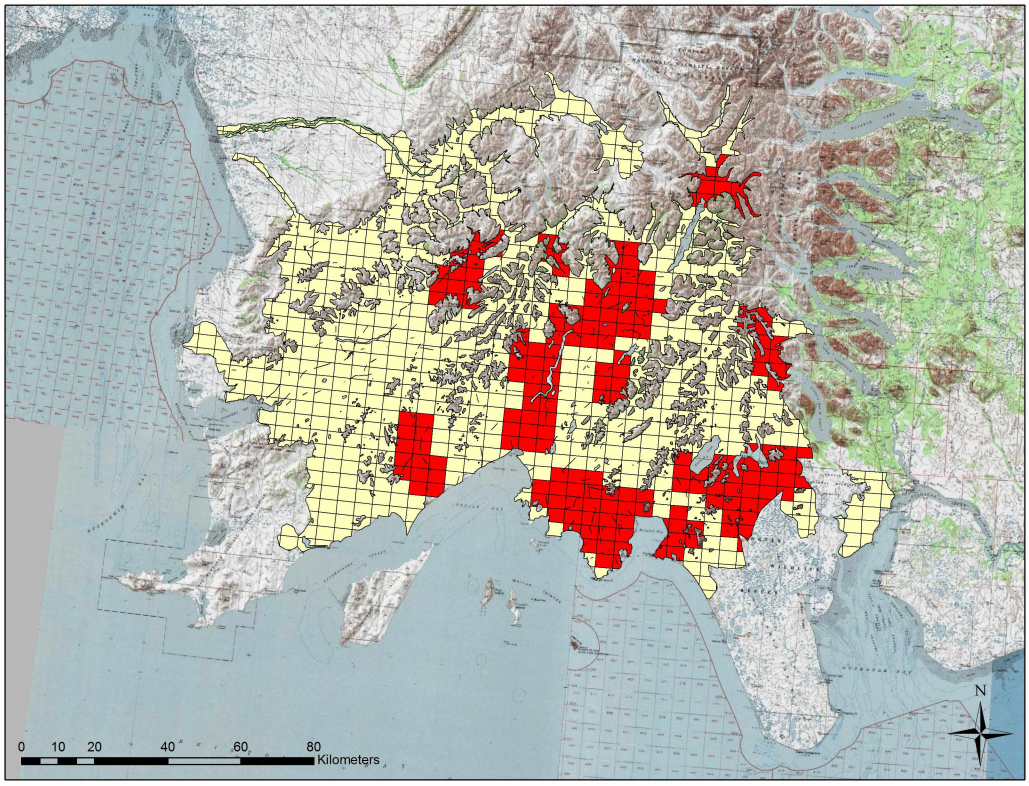


Figure 8: Standard survey units stratified by expected moose density at Togiak National Wildlife Refuge. Red units are in the high density stratum. Yellow units are in the low density stratum.

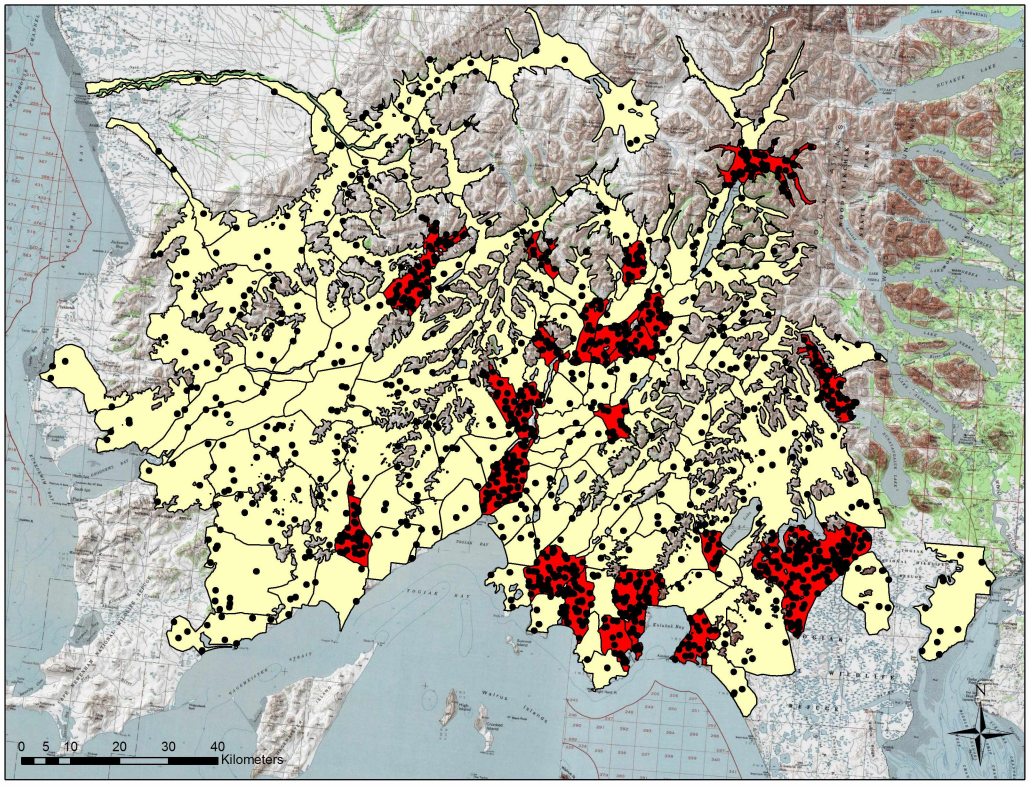


Figure 9: Example of a simulated moose population at Togiak National Wildlife Refuge. Red units are in the high density stratum. Yellow units are in the low density stratum. Each point represents an individual moose location (800 in the high stratum, 500 in the low stratum).

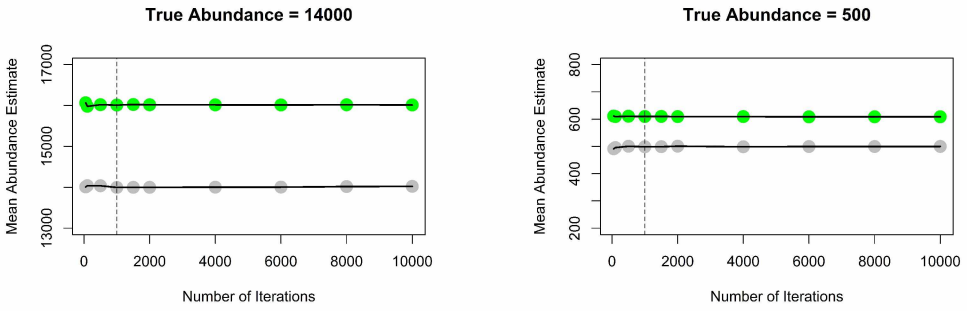


Figure 10: Mean GSPE abundance estimates as a function of number of simulation iterations for two levels of simulated abundance. Green dots represent estimates from the historic unit configuration, and gray dots represent estimates from the standard configuration. The dashed line, at 1000 iterations, depicts the point a which stabilization was inferred.

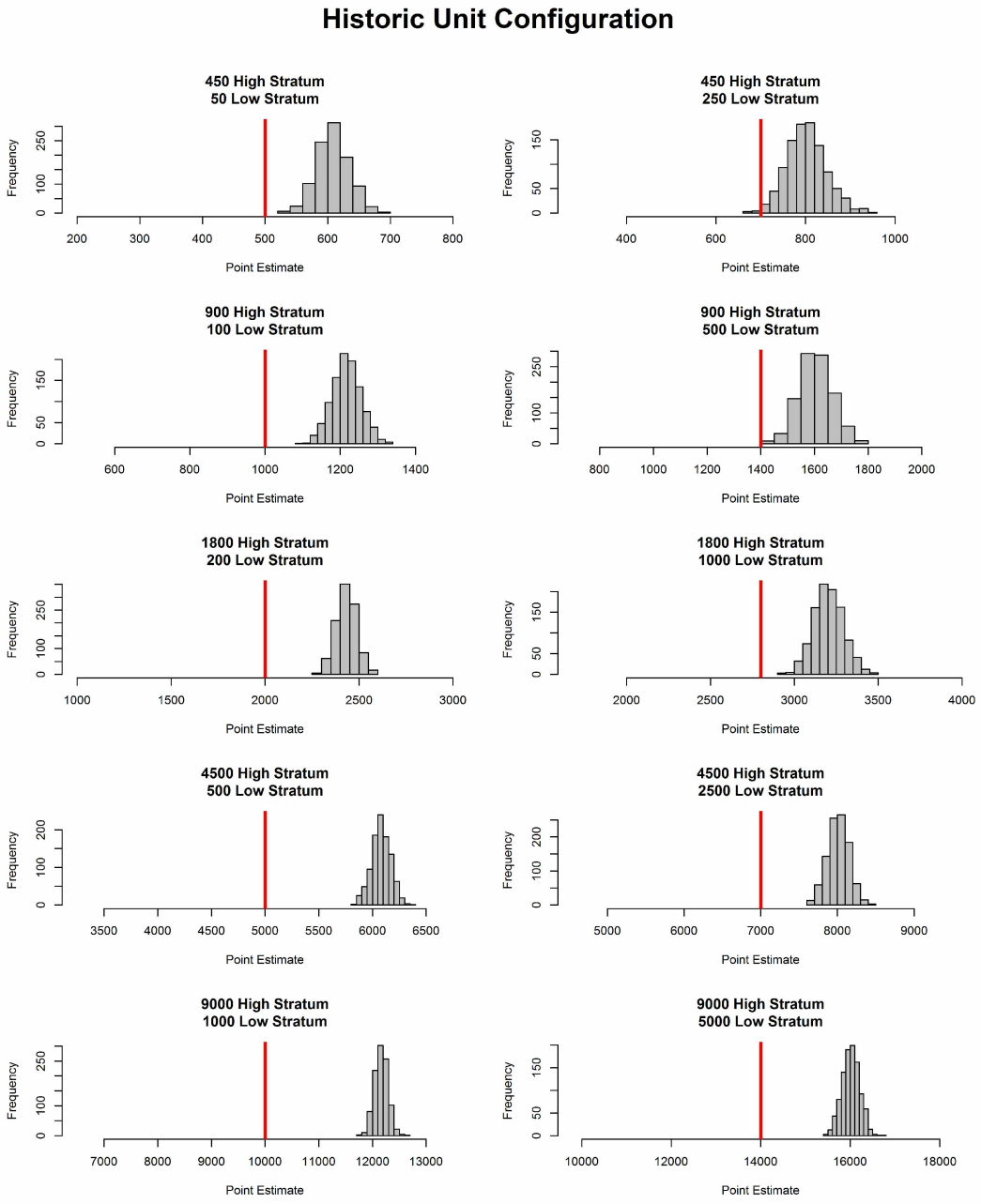


Figure 11: Histograms of moose population estimates from the Geospatial Population Estimator applied to the historic survey unit configuration at Togiak National Wildlife Refuge. Using simulated populations of the specified size per stratum, 1000 estimates were generated for each abundance scenario. The vertical red line denotes the true population size.

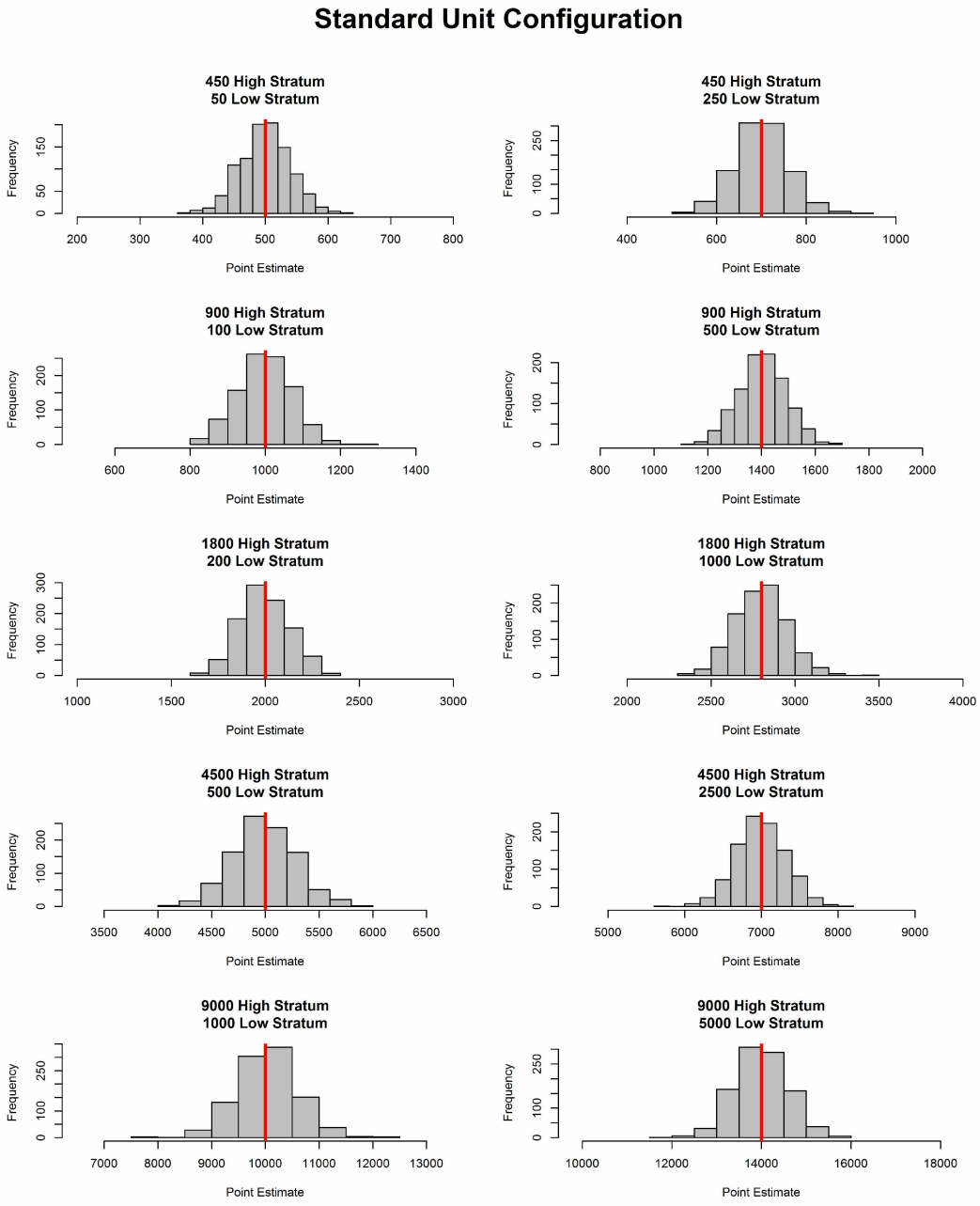


Figure 12: Histograms of moose population estimates from the Geospatial Population Estimator applied to the standard survey unit configuration at Togiak National Wildlife Refuge. Using simulated populations of the specified size per stratum, 1000 estimates were generated for each abundance scenario. The vertical red line denotes the true population size.

Chapter 2: Effect of sampling intensity on performance of the Geospatial Population Estimator at Togiak National Wildlife Refuge

Introduction

For plot-based sampling, it is reasonable to expect that the performance of an estimator is related to sampling intensity (i.e., the proportion of plots sampled). Because of the limited number of available units (M) in the historic Togiak survey unit configuration, a fixed number of units was sampled (m) in each simulation iteration for comparing survey unit configurations in Chapter 1. However, it is reasonable to expect the performance of the GSPE to be related to the number of units sampled. For example, one would expect the precision of estimates to increase as a greater proportion of the available (finite) units are sampled (i.e., as the $\frac{m}{M}$ ratio increases, a greater number of survey unit values are known with certainty). Here, I examine the influence of sampling intensity on measures of estimator performance in the context of moose abundance estimation with the GSPE on TNWR.

Methods

Sample frame

As previously, I used the most recently (2011) surveyed region of TNWR to delineate the sample frame. However, given the conclusion in Chapter 1 that the GSPE exhibits substantial bias with the historic unit configuration, I used only the standard configuration in this simulation study. Additionally, since this study does not include a comparison between unit configurations, I used a sample frame composed of whole standard units, rather than standard units clipped by the bounds of the historic configuration (Figure 13).

Stratification

The stratification scheme used in this simulation study was the same as that in Chapter 1. Specifically, moose densities often vary substantially among survey units, and an effective way of dealing with this variation in counts is to partition the sample frame into "high" and "low" density strata. The current recommendation from ADF&G is to assign units with densities lower than approximately 0.2/km^2 to the low density stratum and units with densities higher

than that to the high density stratum [Kellie and Delong, 2006], although this stratification cut-point will vary with study site. Because available survey data were grouped by historic sample units only, I delineated strata using the historic units (Figure 14). I defined all standard units that overlapped high-stratum historic units to be high-stratum standard units and all others to be low-stratum standard units. Unlike Chapter 1, these simulated distributions were based on the delineation of standard unit strata, rather than historic unit strata (Figure 13). Thus, the small amount of stratification error present in the standard grid in Chapter 1 was eliminated in this analysis.

Simulations

Similar to Chapter 1, I repeatedly generated spatial distributions of simulated moose populations within the sample frame. I separated each stratum into a separate polygon layer using GIS applications in R ([R Development Core Team, 2015]; also see packages listed below). Within each of these stratum polygons (high, low), I generated a specified number of randomly distributed moose locations (Figure 15). Each individual location had a set of coordinates associated with it and fell within the bounds of the appropriate stratum polygon. A new layer of polygons corresponding to the individual survey unit boundaries was then overlaid on this simulated moose population and used to tally the number of moose occurring within each individual unit, which is analogous to the counting of moose that occurs aerially during field surveys (Figure 15). Centroid coordinates were then computed for each survey unit polygon. After tallying moose abundance in each unit and computing centroid coordinates, I randomly selected a specified number of units from each stratum to serve as the sample. The number of units sampled ranged from 20-209 per stratum (discussed in more detail subsequently). Ultimately, one dataset for each of the varied levels of sampling intensity contained the following data for each sampled survey unit: (1) a unit identifier, (2) number of moose counted, (3) area of unit, (4) stratum of unit, (5) latitude of unit centroid, (6) longitude of unit centroid, (7) binary indicator of the unit being sampled or not, (8) binary indicator of the unit being included in the final abundance estimate. The GSPE was then implemented with each individual dataset. This entire procedure was repeated using a loop, and the results from each iteration of the loop were stored in matrix objects in R. These stored results were then used in computing model-performance metrics.

Measures of model performance

After completing simulation loops, I used the stored results to compute model-performance metrics. The bias, coefficient of variation, confidence interval coverage, and root mean square error were estimated for each level of sampling intensity. The bias was estimated as:

$$\widehat{B(\hat{N})} = \frac{1}{k}\sum_{i=1}^{k}\hat{N}_i - N, \tag{16}$$

where k is the number of simulated populations, $\hat{N}_i$ is the GSPE estimate for the ith simulated population, and N is the true abundance of the population. Bias was then converted to relative bias:

$$\text{Relative bias} = \frac{\widehat{B(\hat{N})}}{N}. \tag{17}$$

The coefficient of variation was computed as:

$$\text{CV} = \frac{\hat{SE}}{\hat{N}}, \tag{18}$$

where $\hat{SE}$ is the mean value of the standard error for 1000 simulated populations. True confidence interval coverage for the nominal rate of 95% was estimated as:

$$\text{True coverage} = \left[\frac{\sum_{i=1}^{k}\mathbf{1}(LB_i \leq N \leq UB_i)}{k}\right] \times 100, \tag{19}$$

where k is the number of simulated populations, $\mathbf{1}$ is an indicator function equal to 1 if the parenthetical logical statement is true and 0 if it is false, $\hat{N}_i$ is the estimate of N from the i^{th} simulated population, LB_i is the lower bound of the 95% confidence interval associated with the estimate of N from the i^{th} simulated population, and UB_i is the upper bound of the 95% confidence interval associated with the estimate of N from the i^{th} simulated population.

Simulated sampling intensities ranged from 20 units (9.5%) to 209 units (100%) for the high stratum and 30 units (2.7%) to 200 units (26.7%) in the low stratum. As in Chapter 1, I used 1000 iterations of the simulation loop for each sampling scenario. For each sampling scenario, the true number of moose in the population was held constant at 1144 in the high stratum and 482 in the low stratum (i.e., the number counted during the most recent survey of TNWR in 2011).

Simulations and spatial data manipulation were conducted using R with several packages developed for spatial analysis (maptools [Bivand and Lewin-Koh, 2015], rgdal [Bivand et al., 2015], rgeos [Bivand and Rundel, 2015], sp [Roger S. Bivand, 2013], spatstat [Baddeley and Turner, 2005]). ArcGIS 10.0 [ESRI, 2011] was used for visual presentation of spatial data. For each simulated population, the GSPE was implemented using code written in R by Jay ver Hoef for ADF&G (Appendix 4). Semivariograms were fit using REML [Patterson and Thompson, 1974]. The R script for the Chapter 2 simulation loops is available in Appendix 2.

Results

As expected, some aspects of estimator performance improved as sampling intensity increased. In particular, precision of the GSPE increased as the number of units sampled increased, as evidenced by decreases in CV with increasing sampling intensity (Figure 17). The CV decreased with increasing sampling intensity in both the high and low strata. In contrast, estimator bias did not appear to be related to sampling intensity for the range of sampling intensities examined (Figure 16). Rather, the GSPE appeared to be approximately unbiased regardless of the number of units sampled. The RMSE exhibited substantial decreases as sampling intensity increased (Figure 19). This pattern was evident for increased sampling intensity in both the low and high strata. Confidence interval coverage did not appear to be related to sampling intensity, but was slightly lower than the nominal rate in most cases (Figure 18).

Discussion

Results from these simulations suggest that greater sampling intensity generally improves model performance. However, this theme is only evident in the CV and RMSE metrics. The CV results clearly indicate that the uncertainty in GSPE estimates decreases as more units are sampled (Figure 17). Kellie and Delong [2006] suggested that increasing sampling intensity in the low stratum has minimal influence on the precision of GSPE estimates. In contrast, results from these simulations suggest that precision in GSPE estimates increases when sampling intensity increases within either stratum. This is a logical result given that more information is available when more units are sampled. In other words, greater sampling intensity requires fewer unit-specific abundance predictions and therefore less uncertainty in the total number of moose present. Nevertheless, with lower moose densities in the low stratum, variance should generally be lower than in the high stratum. So, as Kellie and

Delong [2006] suggest, the influence of increased sampling intensity in the low stratum may be of lesser magnitude in many scenarios than the influence of increased sampling intensity in the high stratum.

Estimates of relative bias did not indicate an influence of sampling intensity on bias of the GSPE (Figure 16). In fact, estimated bias was low for all simulated scenarios. This suggests that inducing bias is not a concern when sampling relatively small numbers of units (20-30). Rather, reduced precision appears to be the main cost of low-intensity sampling.

Estimates of RMSE are consistent with the pattern observed in CV and relative bias for the simulated scenarios. Since the GSPE appears to be unbiased across the range of sampling intensities examined, the RMSE is almost entirely a function of the variance, and thus decreases in concert with CV. More explicitly, RMSE = $[\text{Bias}(\hat{N})^2 + \text{Var}(\hat{N})]^{1/2}$, so when $\text{Bias}(\hat{N}) \approx 0$, RMSE $\approx$ $[\text{Var}(\hat{N})]^{1/2}$.

No relationship between sampling intensity and confidence interval coverage was apparent from the simulation results (Figure 18). True coverage was reasonably close to the nominal rate in all cases. Interestingly, deviations from the nominal rate (95%) were almost all in the negative direction.

As previously, several assumptions have been made in this analysis and should be explicitly acknowledged here. First, isotropy and second-order stationarity are assumed for our underlying stochastic process (which we know to be true in this case, because of the way in which moose locations were simulated). Additionally, we assume that the theoretical exponential semivariogram (Equation 13) model is a reasonable representation of the spatial correlation structure in this system. And, once again, we assume perfect enumeration of moose within sampled survey units. This is not an issue in these simulated populations, but it can pose substantial difficulties when applying this technique under field conditions.

It is important to consider the results from these simulations in the context of these assumptions and the specific conditions simulated. In particular, the distribution of moose within strata was simulated randomly, which is often not the case for real moose populations. The implications of non-random moose distributions are explored further in Chapter 3.

Figures

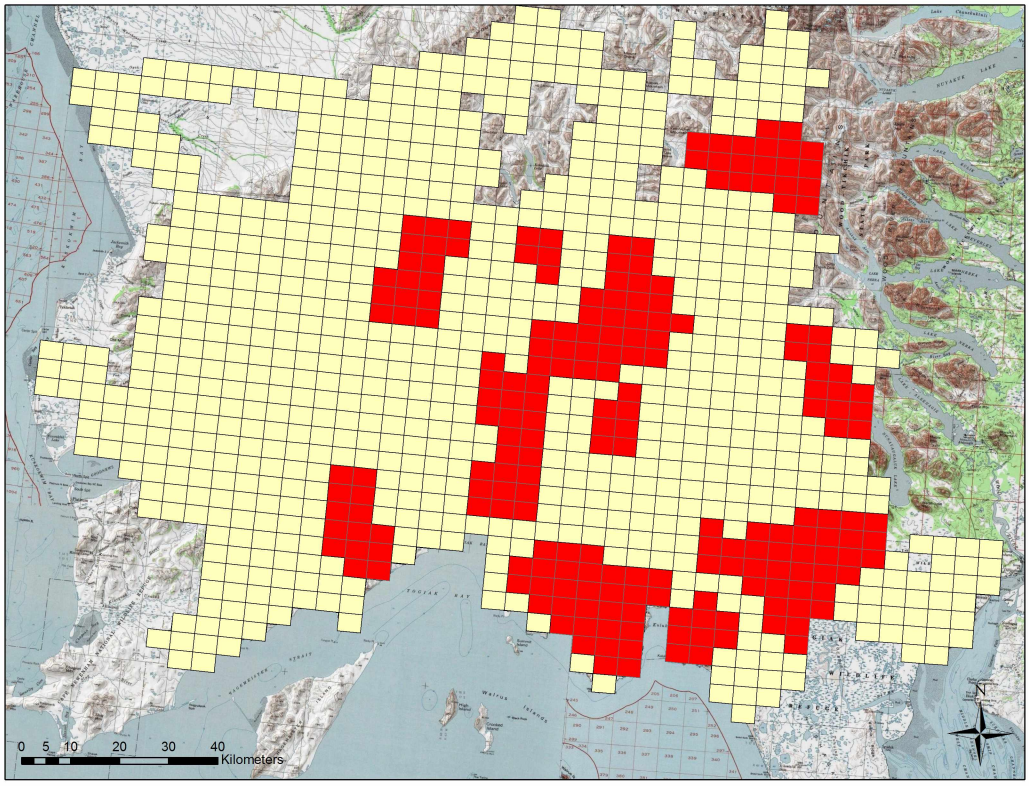


Figure 13: Sample frame used for simulations examining the effect of sampling intensity on performance of the Geospatial Population Estimator at Togiak National Wildlife Refuge, Alaska. Red sample units are in the high density stratum. Yellow units are in the low density stratum.

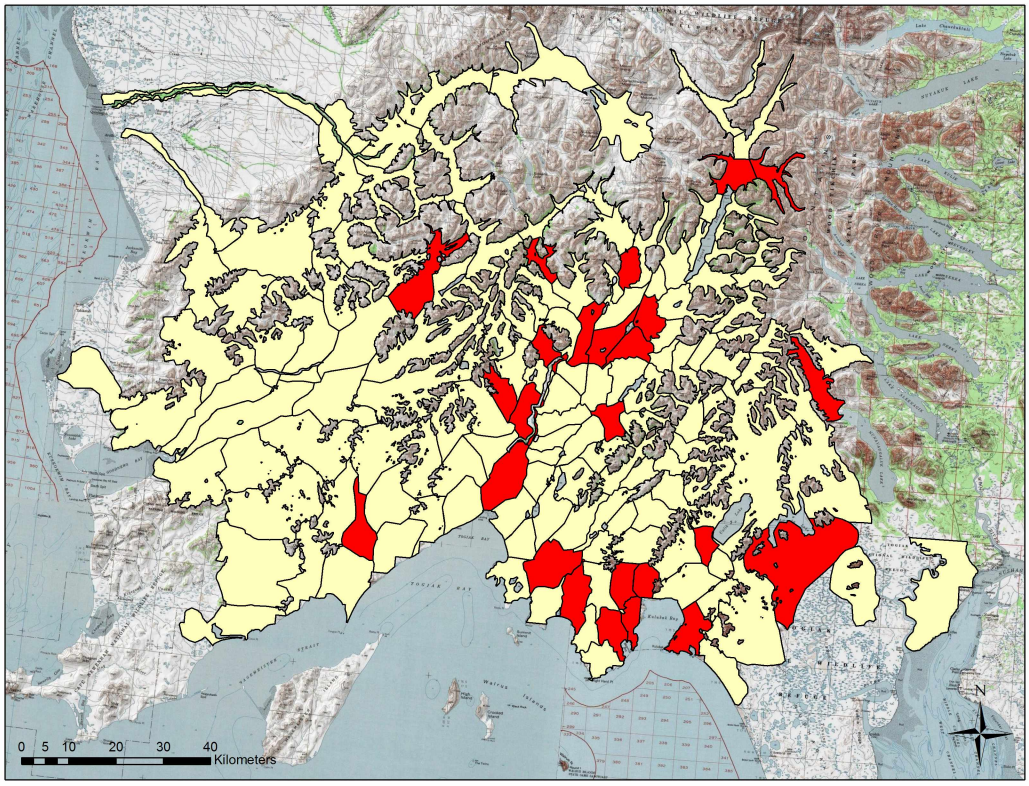


Figure 14: Historic survey units stratified by expected moose density at Togiak National Wildlife Refuge. Red units are in the high density stratum. Yellow units are in the low density stratum.

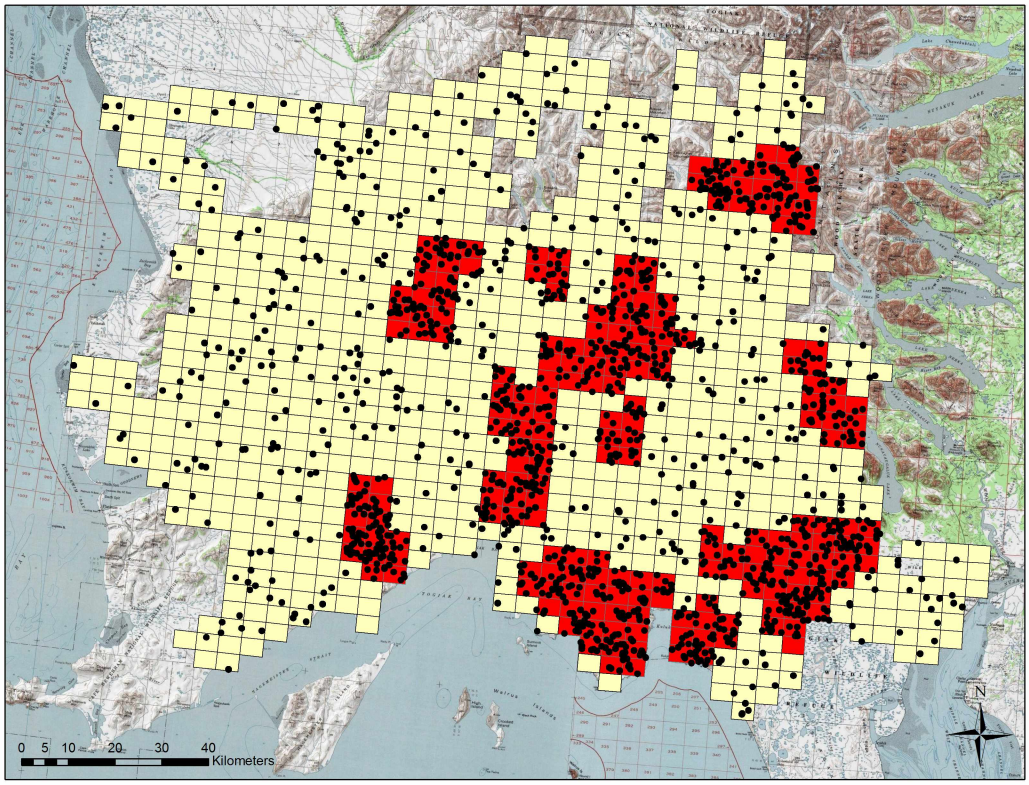


Figure 15: An example of a simulated moose population within the sample frame. Black dots represent individual moose. Red units are in the high density stratum. Yellow units are in the low density stratum.

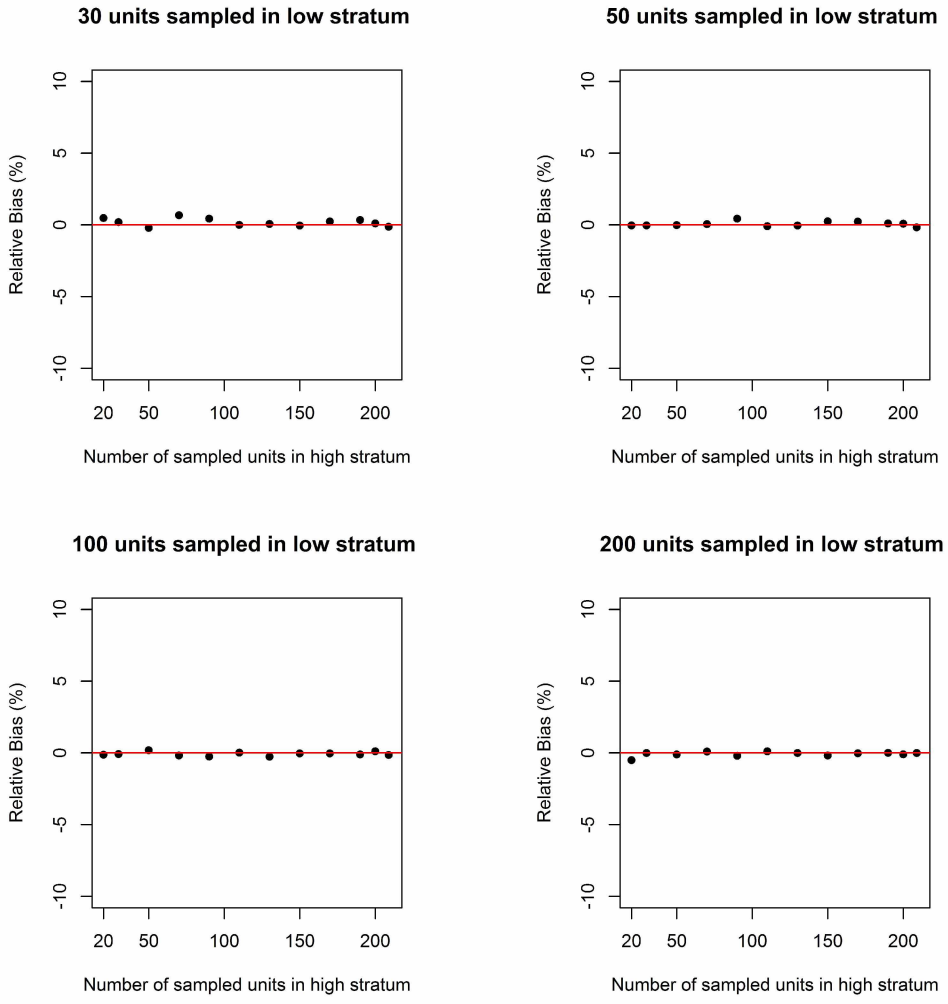


Figure 16: Relative bias (%) as a function of the number of units sampled in the high stratum for each of four levels of sampling intensity in the low stratum.

Figure 17: Coefficient of variation (CV) as a function of the number of units sampled in the high stratum for each of four levels of sampling intensity in the low stratum.

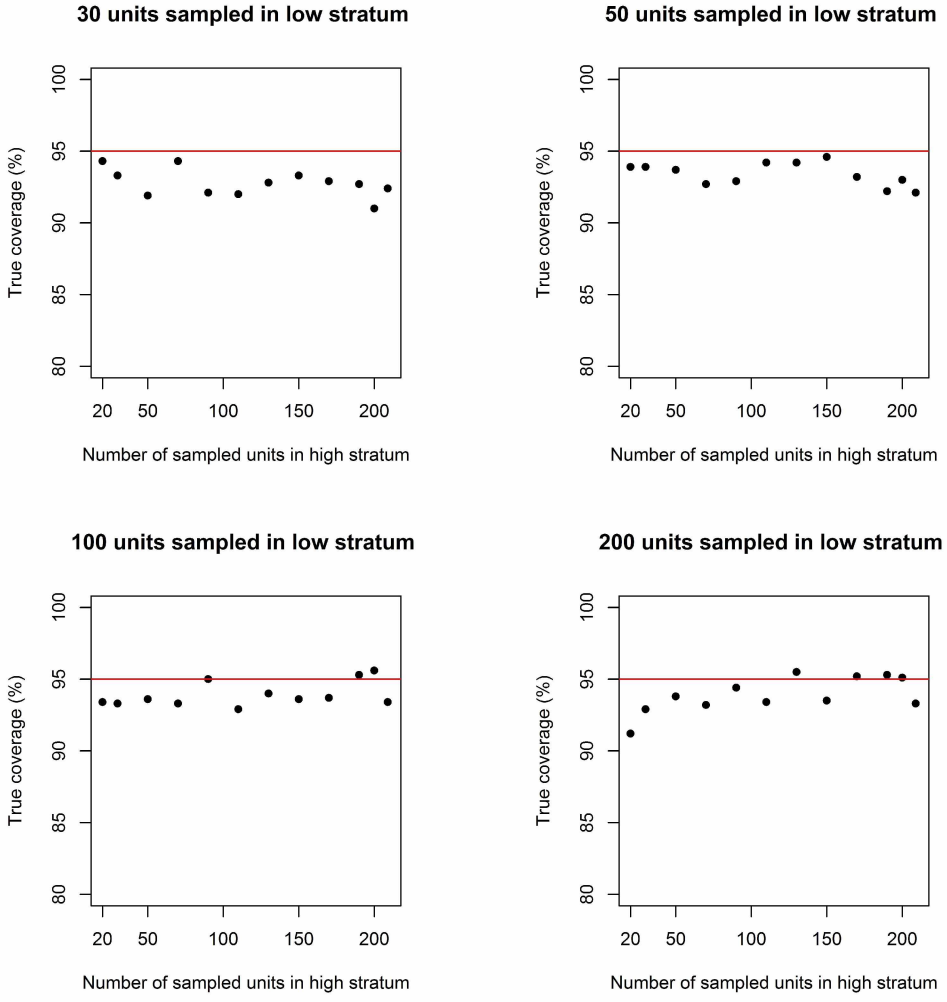


Figure 18: True coverage (%) as a function of the number of units sampled in the high stratum for each of four levels of sampling intensity in the low stratum. The red line indicates the nominal coverage rate (95%).

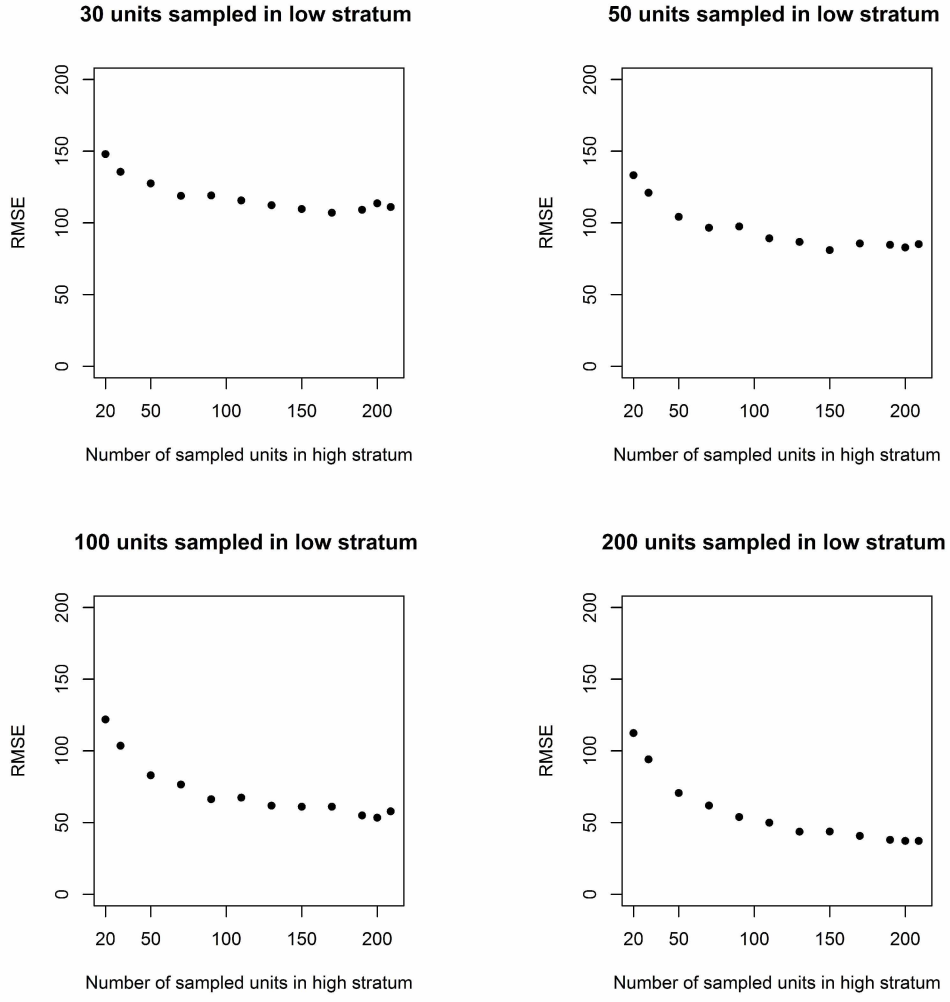


Figure 19: Root mean square error (RMSE) as a function of the number of units sampled in the high stratum for each of four levels of sampling intensity in the low stratum.

Chapter 3: Effect of clustered moose distributions on precision of the Geospatial Population Estimator at Togiak National Wildlife Refuge

Introduction

In previous chapters, I investigated the influence of survey unit configuration and sampling intensity on GSPE performance at TNWR when moose were distributed randomly within density-strata. However, moose are often not distributed randomly upon real landscapes. Often moose occur in clustered distributions to varied degrees. Clustering can increase the inter-unit heterogeneity in moose abundance, and thus impact the performance of abundance estimators. Here, I investigate the influence of clustering on performance of the GSPE at TNWR.

Methods

Sample frame

As in previous chapters, I used the most recently (2011) surveyed region of TNWR to delineate the sample frame, which facilitated the identification of realistic density strata. I used only the standard survey unit configuration composed of whole standard units (Figure 20) in examining the effects of spatial clustering.

Stratification

The stratification scheme used in this simulation study was the same as that in previous chapters. Specifically, moose densities often vary substantially among survey units, and an effective way of dealing with this variation in counts is to partition the sample frame into "high" and "low" density strata. The current recommendation from ADF&G is to assign units with densities lower than approximately $0.2/km^2$ to the low density stratum and units with densities higher than that to the high density stratum [Kellie and Delong, 2006], although this stratification cut-point will vary with study site. Because available survey data were grouped by historic sample units only, I delineated strata using the historic units (Figure 21). I defined all standard units that overlapped high-stratum historic units to be high-stratum standard units and all others to be low-stratum standard units (Figure 20).

Simulations

I used the same general approach to repeatedly generating spatial distributions of simulated moose populations within the sample frame as that used in previous chapters. However, the types of spatial distributions that I generated varied from previous chapters. As before, I separated each stratum into a separate polygon layer using GIS applications in R ([R Development Core Team, 2015]; also see packages listed below) and within each of these stratum polygons (high, low), I generated a specified number of moose locations. However, for this simulation study the random moose locations were generated with varied degrees of clustering. Specifically, I created a cluster index ranging from 0 to 1, which specified the degree of clustering. This cluster index was used to scale the "nclusters" argument in the spsample function from the sp package in R [Roger S. Bivand, 2013], which generates points from a Poisson cluster process. Small values of the cluster index (close to 0) yielded highly clustered distributions, whereas large values (close to 1) yielded relatively unclustered distributions. The theoretical (although highly unrealistic) minimum of the cluster index is all moose occurring in a single group. The maximum is analogous to each individual occurring in its own group, which yields a random distribution of individual moose, which is equivalent to the distributions used in Chapters 1 and 2. Figure 22 visually depicts changes in the degree of spatial clustering as the cluster index increases.

Each individual location had a set of coordinates associated with it and fell within the bounds of the appropriate stratum polygon. A new layer of polygons corresponding to the individual survey unit boundaries was then overlaid on this simulated moose population and used to tally the number of moose occurring within each individual unit, which is analogous to the counting of moose that occurs aerially during field surveys (Figure 23). Centroid coordinates were then computed for each survey unit polygon. After tallying moose abundance in each unit and computing centroid coordinates, I randomly selected a specified number of units from each stratum to serve as the sample. The number of units sampled ranged from 30-50 for the low stratum and 30-80 for the high stratum. Ultimately, one dataset for each of four levels of sampling intensity (detailed subsequently) contained the following data for each sampled survey unit: (1) a unit identifier, (2) number of moose counted, (3) area of unit, (4) stratum of unit, (5) latitude of unit centroid, (6) longitude of unit centroid, (7) binary indicator of the unit being sampled or not, (8) binary indicator of the unit being included in the final abundance estimate. The GSPE was then implemented with each dataset, providing an estimate of abundance and associated measures of model performance. This entire proce-

dure was repeated using a loop, and the results from each iteration of the loop were stored in matrix objects in R.

I ran 1000 iterations of this simulation loop for each of 20 levels of the cluster index (values between 0.05 and 1.0 in increments of 0.05). These 20 levels of clustering with 1000 iterations per level were repeated for each of four sampling-intensity scenarios: 30 high-density and 30 low-density units, 50 high-density and 30 low-density units, 50 high-density and 50 low-density units, and 80 high-density and 50 low-density units.

Simulations and spatial data manipulation were conducted using R with several packages developed for spatial analysis (maptools [Bivand and Lewin-Koh, 2015], rgdal [Bivand et al., 2015], rgeos [Bivand and Rundel, 2015], sp [Roger S. Bivand, 2013], spatstat [Baddeley and Turner, 2005]). ArcGIS 10.0 [ESRI, 2011] was used for visual presentation of spatial data. For each simulated population, the GSPE was implemented using code written in R by Jay ver Hoef for ADF&G (Appendix 4). Variograms were fit using REML [Patterson and Thompson, 1974]. The R script for simulation loops with clustered distributions is available in Appendix 3.

Measures of model performance

The same measures of model performance used in Chapter 2 were also used to assess the influence of clustering. Specifically,

Bias:

$$\widehat{B(\hat{N})} = \frac{1}{k}\sum_{i=1}^{k}\hat{N}_i - N \tag{20}$$

where k is the number of simulated populations, $\hat{N}_i$ is the GSPE estimate for the ith simulated population, and N is the true abundance of the population.

Relative bias:

$$\text{Relative bias} = \frac{\widehat{B(\hat{N})}}{N} \tag{21}$$

Coeffiencient of variation:

$$\text{CV} = \frac{\hat{SE}}{\hat{N}} \tag{22}$$

where $\hat{SE}$ is the mean value of the standard error for 1000 simulated populations.

True confidence interval coverage for the nominal 95% rate:

$$\text{True coverage} = \left[\frac{\sum_{i=1}^{k} \mathbf{1}(LB_i \leq N \leq UB_i)}{k} \right] * 100 \tag{23}$$

where k is the number of simulated populations, $\mathbf{1}$ is an indicator function equal to 1 if the parenthetical logical statement is true and 0 if it is false, $\hat{N}_i$ is the estimate of N from the i^{th} simulated population, LB_i is the lower bound of the 95% confidence interval associated with the estimate of N from the i^{th} simulated population, and UB_i is the upper bound of the 95% confidence interval associated with the estimate of N from the i^{th} simulated population.

Results

In general, variation in the degree of clustering had a notable impact on model performance. The most dramatic influences were on the precision of estimates (CV; Figure 25) and the RMSE (Figure 27). With extremely clustered distributions (i.e., low cluster index), a small amount of positive bias was observed for lower sampling intensities (Figure 24). For moderate and low degrees of clustering, bias did not appear to be a problem. Confidence interval coverage was slightly below the nominal rate for most scenarios (Figure 26). However, when distributions were highly clustered, coverage was well below the nominal rate for lower sampling intensities.

Discussion

In general, results from these simulations suggest that increased spatial clustering of moose reduces model performance. However, much of this effect appears to be ameliorated by increasing sampling intensity. Issues with bias and confidence interval coverage were only evident for extremely clustered distributions. In those cases, a slight positive bias appeared and coverage dropped slightly (Figures 24 and 26). The drop in coverage may be related to the increase in bias, since confidence intervals will be pulled farther from the true value as bias increases, resulting in fewer intervals containing the true

value. For moderate and low levels of clustering the estimator appeared to be unbiased (Figure 24). Similarly, at moderate and low levels of clustering, confidence interval coverage did not appear to be affected (Figure 26). Interestingly, as observed in Chapter 2, coverage was almost always slightly below the nominal rate.

The precision of estimates decreased and the RMSE increased as clustering increased (Figures 25 and 27). Most of the change in RMSE with clustering was driven by changes in CV, although the small amount of bias observed for highly clustered distributions contributed to the highest values of RMSE when sampling intensities were low (Figures 24 and 27).

It is noteworthy that moderate increases in sampling intensity largely ameliorated the issues of bias and reduced coverage (with the possible exception of coverage for the most clustered distribution examined; Figure 26). Similarly, for moderate to high degrees of clustering, the CV (Figure 25) and RMSE (Figure 27) were substantially improved with moderate increases in sampling intensity. Together, these results suggest that when moose distributions are clustered, greater sampling intensity may be desirable in order to improve estimator performance.

Figures

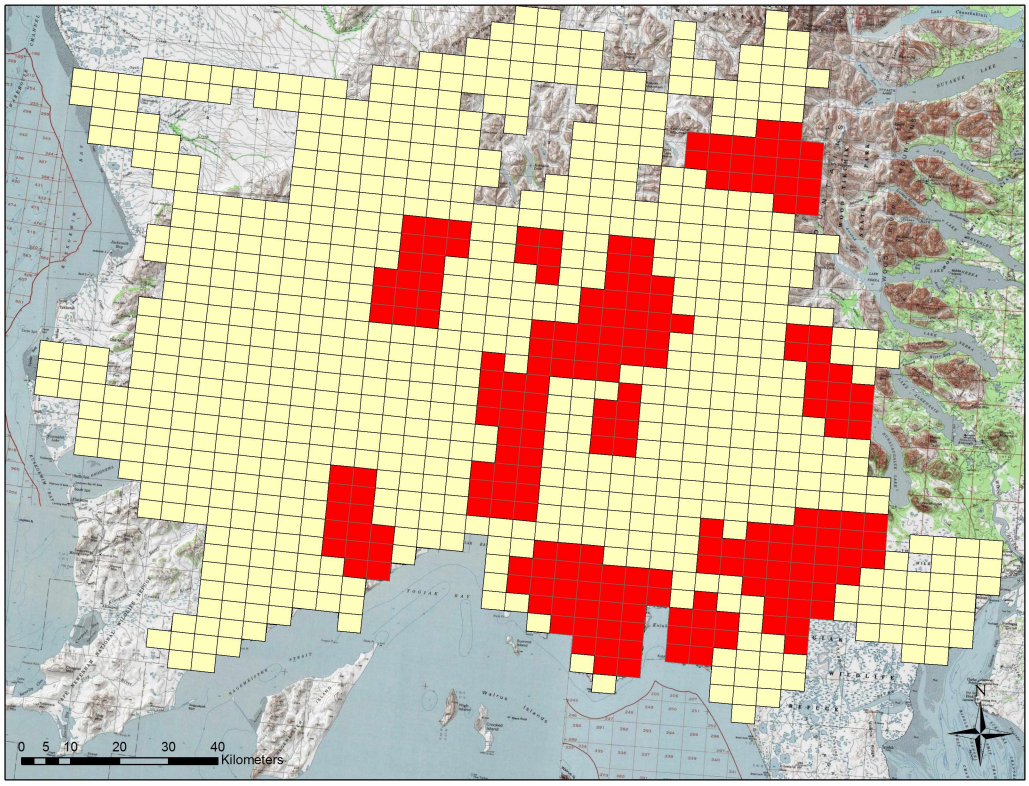


Figure 20: Sample frame used for simulations examining the effect of clustered moose distributions on performance of the Geospatial Population Estimator at Togiak National Wildlife Refuge, Alaska

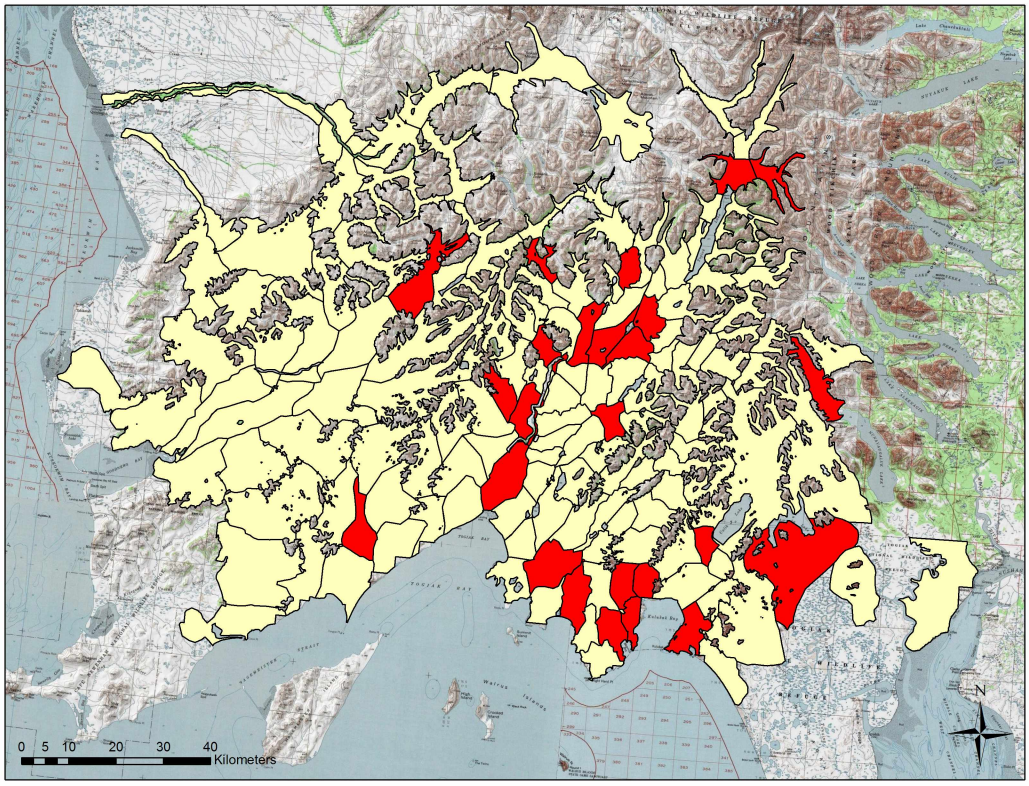
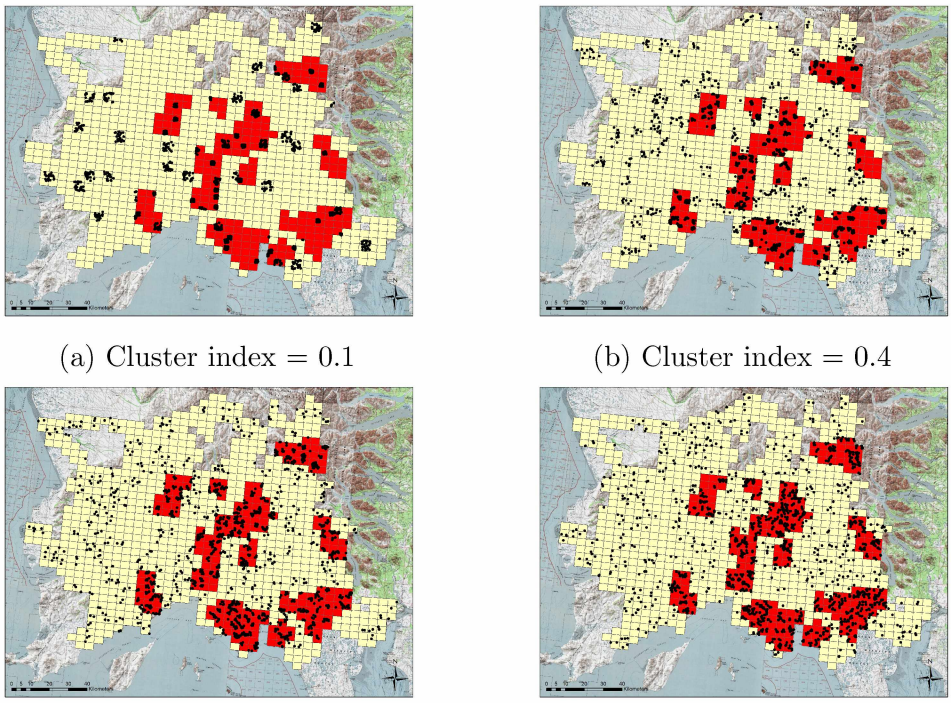


Figure 21: Historic survey units used as the basis for stratifying standard survey units at Togiak National Wildlife Refuge. Red units depict the high density stratum. Yellow units depict the low density stratum.



(a) Cluster index = 0.1

(b) Cluster index = 0.4

(c) Cluster index = 0.7

(d) Cluster index = 1.0

Figure 22: Examples of simulated moose distributions with different degrees of clustering at Togiak National Wildlife Refuge, Alaska. Red boxes represent survey units in the high density stratum. Yellow boxes represent survey units in the low density stratum. Black dots represent the locations of individual moose. A total of 1,626 (1,144 in the high stratum, 482 in the low stratum) moose occur in each of these four example populations.

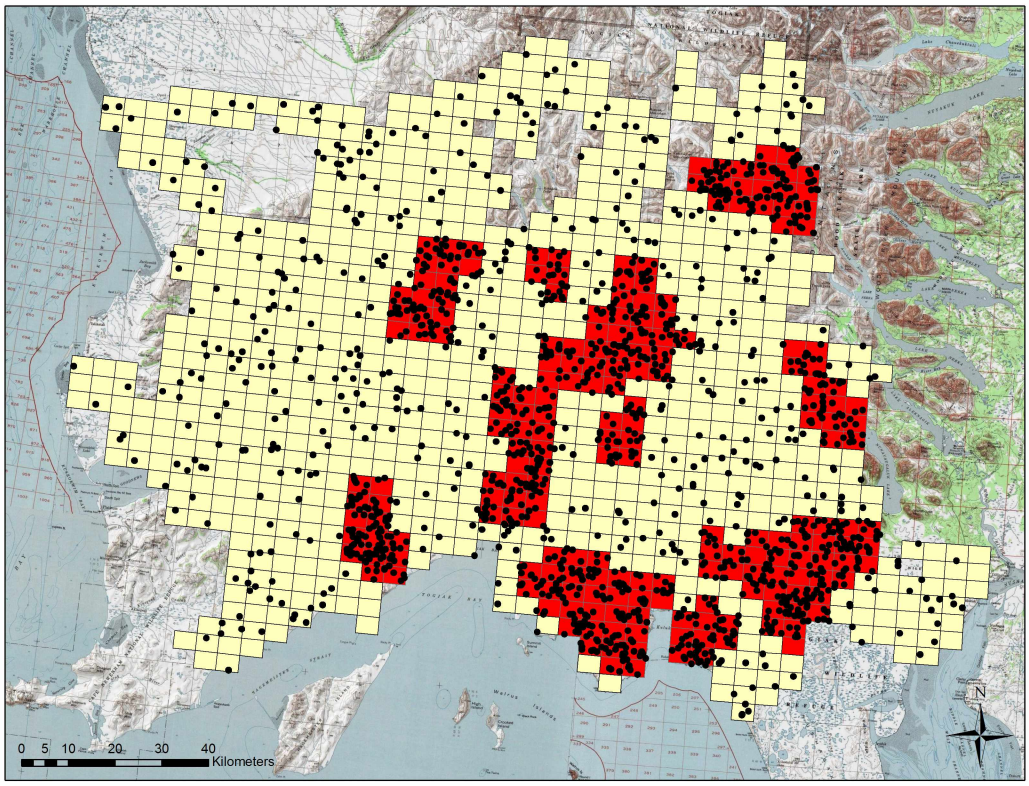


Figure 23: An example of a simulated moose population within the sample frame (cluster index = 1.0) with survey unit boundaries overlaid. Black dots represent individual moose. Red units are in the high density stratum. Yellow units are in the low density stratum.

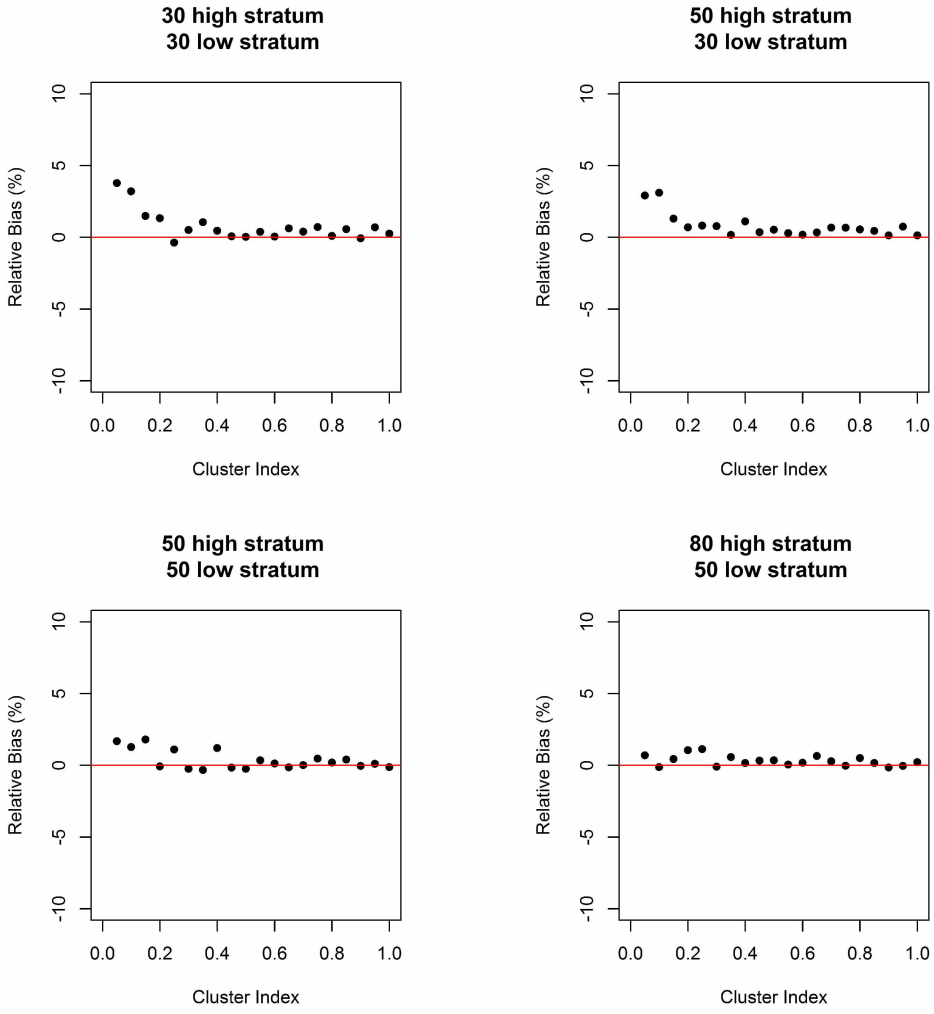


Figure 24: Relative bias (%) as a function of simulated moose clustering for each of four levels of sampling intensity. The cluster index represents the degree of clustering. Low values correspond to more highly clustered distributions and high values correspond to less clustered distributions. Figure 22 provides a visual representation of clutering for a range of cluster index values.

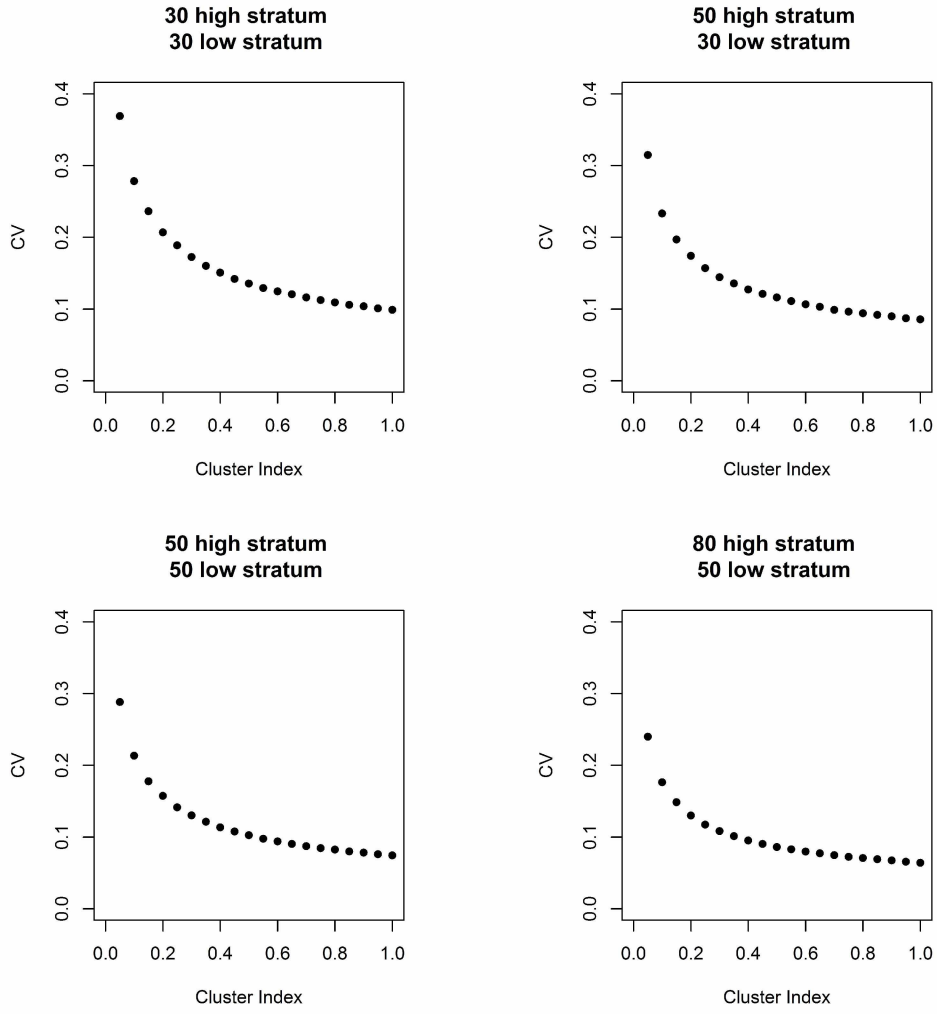


Figure 25: Coefficient of variation (CV) as a function of simulated moose clustering for each of four levels of sampling intensity. The cluster index represents the degree of clustering. Low values correspond to more highly clustered distributions and high values correspond to less clustered distributions. Figure 22 provides a visual representation of clutering for a range of cluster index values.

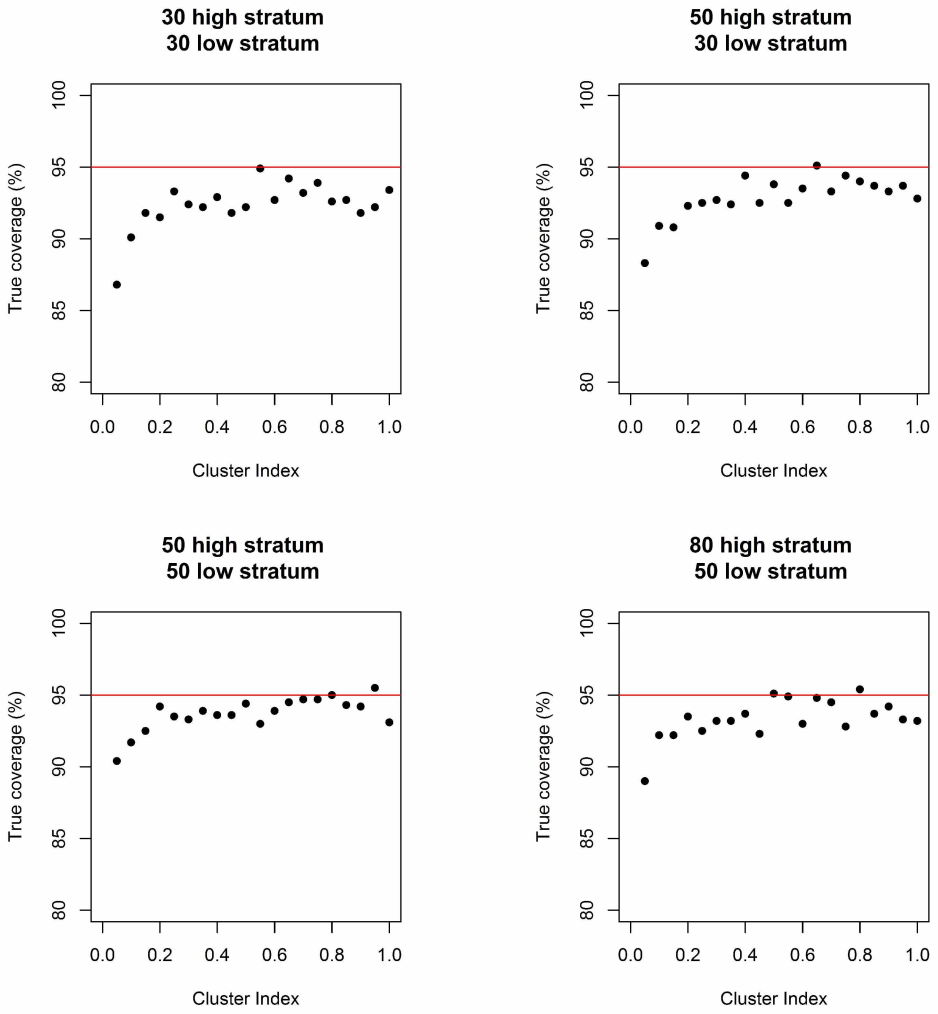


Figure 26: True (estimated) 95% confidence interval coverage as a function of simulated moose clustering for each of four levels of sampling intensity. The horizontal red line depicts the nominal 95% coverage rate. The cluster index represents the degree of clustering. Low values correspond to more highly clustered distributions and high values correspond to less clustered distributions. Figure 22 provides a visual representation of clutering for a range of cluster index values.

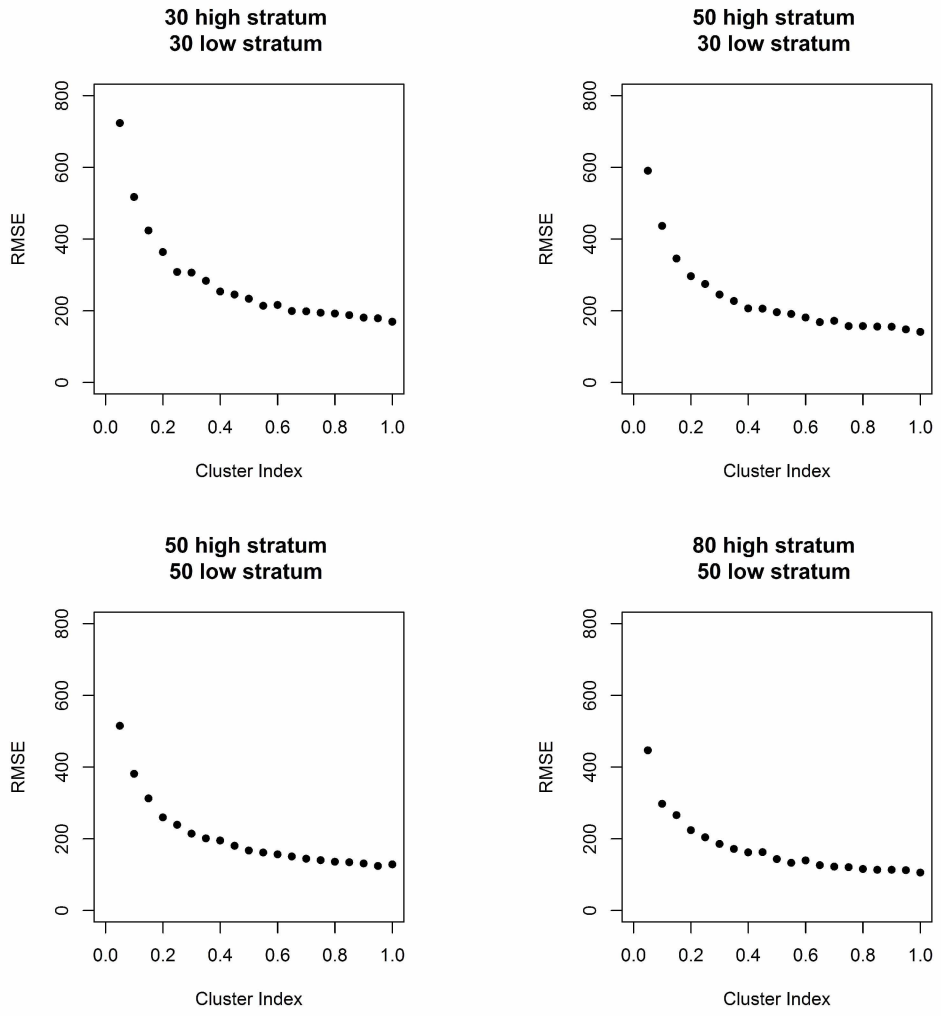


Figure 27: Root mean square error (RMSE) as a function of simulated moose clustering for each of four levels of sampling intensity. The cluster index represents the degree of clustering. Low values correspond to more highly clustered distributions and high values correspond to less clustered distributions. Figure 22 provides a visual representation of clutering for a range of cluster index values.

General summary

The Geospatial Population Estimator (GSPE) is widely used to estimate moose abundance via finite population block kriging. In Chapter 1, I compared GSPE bias between two survey unit configurations at Togiak National Wildlife Refuge (TNWR), Alaska. One configuration was composed of historic survey units with heterogeneous shapes and sizes. The other configuration was composed of survey units from the "standard grid" [Kellie and Delong, 2006], which are approximately the same shape and size. Substantial bias was observed with the historic configuration for all population sizes examined. The bias appears to result from the fact that the GSPE does not account for large amounts of heterogeneity in unit size. Unless extensions of the GSPE that accommodate unit-size heterogeneity are developed, future moose surveys at TNWR should use a survey configuration with homogeneous unit sizes to avoid bias in the GSPE. Even if such extensions are developed, the standard units are preferable for TNWR because the greater number of survey units available per stratum enables better estimation of the spatial covariance structure.

In Chapter 2, I assessed the performance of the GSPE at TNWR under varied levels of sampling intensity. Given the results of Chapter 1, only the standard survey grid was used. Results suggested that bias and confidence interval coverage were not problems even with low sampling intensities, but that precision of estimates increased substantially when sampling intensity increased in both the low- and high-density strata. These results are conditional on random moose distributions within strata.

In Chapters 1 and 2, simulated moose distributions in TNWR were generated randomly. However, moose distributions often exhibit some degree of clustering on natural landscapes. In Chapter 3, I simulated moose distributions with varied degrees of clustering to assess the influence of clustering on model performance. In general, model performance decreased as the degree of clustering increased. Clustering increased bias and root mean square error, and decreased precision and confidence interval coverage. However, moderate increases in sampling intensity helped to ameliorate the effects of clustering on most aspects of model performance.

One aspect of these simulations worth considering is that survey units were selected as a simple random sample within each density stratum. Because the GSPE is a model-based approach to inference, random sample selection is not a requirement. In fact, non-random sampling schemes (e.g., systematic sampling) are often preferable to random sampling in model-based approaches to inference [Cressie, 1991, Van Groenigan, 2000, Ver Hoef, 2002]. In other words, the GSPE models the covariance among sample units explicitly, so it

does not rely on random sampling schemes. With that in mind, it can be beneficial to strategically sample units in such a way that avoids large areas of unsampled space, which can inflate prediction variance. This may improve the precision of population estimates even when faced with small samples or highly clustered moose distributions. Similarly, intentional placement of some sample units in close proximity to one another can aid in estimation of nugget and range parameters in variogram-based approaches. Future efforts should consider the effects of non-random sample selection on estimator performance.

As previously mentioned, the GSPE is currently implemented using only the exponential semivariogram (Equation 13). Selection of an appropriate variogram model is a potentially important component of kriging-based spatial analyses [Van Groenigan, 2000, Mazzella and Mazzella, 2013]. The exponential semivariogram appeared to perform well in these simulations, but other variogram forms were not examined. Future investigators should explore the performance of alternative variogram models relative to that of the exponential. Similarly, it would be valuable to examine the robustness of different semivariograms to mis-specification in the context of moose abundance surveys. This could be easily accomplished by simulating distributions under a given covariance structure, and then analyzing the data using a different covariance model.

Acknowledgments

The U.S. Fish and Wildlife Service initiated and funded this project. Anna-Marie Benson initiated, arranged funding for, and provided input and direction on the focus of this project. Hilmar Maier and McCrea Cobb provided GIS data. Pat Walsh and Andy Aderman, at Togiak National Wildlife Refuge, provided count data from previous moose surveys.

Ron Barry and Jay Ver Hoef provided insight on the possible sources of bias when using the historic Togiak survey units.

Code for implementing the Geospatial Population Estimator in R (Appendix 4) was written by Jay Ver Hoef and provided by the Alaska Department of Fish and Game.

Literature Cited

[Baddeley and Turner, 2005] Baddeley, A. and Turner, R. (2005). spatstat: An r package for analyzing spatial point patterns. *Journal of Statistical Software*, 12:1–42.

[Banerjee et al., 2015] Banerjee, S., Carlin, B. P., and Gelfand, A. E. (2015). *Hierarchical modeling and analysis for spatial data.* CRC Press, Boca Raton, Florida.

[Bivand et al., 2015] Bivand, R., Keitt, T., and Rowlingson, B. (2015). *rgdal: Bindings for the Geospatial Data Abstraction Library.* R package version 1.0-7.

[Bivand and Lewin-Koh, 2015] Bivand, R. and Lewin-Koh, N. (2015). *maptools: Tools for Reading and Handling Spatial Objects.* R package version 0.8-37.

[Bivand and Rundel, 2015] Bivand, R. and Rundel, C. (2015). *rgeos: Interface to Geometry Engine - Open Source (GEOS).* R package version 0.3-13.

[Buckland et al., 1993] Buckland, S., Anderson, D., Burnham, K., and Laake, J. (1993). *Distance sampling: estimating abundance of biological populations.* Chapman and Hall, London, UK.

[Christ, 2011] Christ, A. (2011). *Sightability correction for moose population surveys.* Alaska Department of Fish and Game, Anchorage, Alaska.

[Cressie, 1991] Cressie, N. (1991). *Spatial statistics.* John Wiley and Sons, Inc., New York, New York.

[ESRI, 2011] ESRI (2011). Arcgis desktop: Release 10.0. Environmental Systems Research Institute, Redlands, CA.

[Gasaway et al., 1986] Gasaway, W. C., Dubois, S. D., Reed, D. J., and Harbo, S. J. (1986). Estimating moose population parameters from aerial surveys. *Biological papers of the University of Alaska*, 22:1–108.

[Journel and Huijbregts, 1978] Journel, A. and Huijbregts, C. (1978). *Mining geostatistics.* Academic Press, London, UK.

[Kellie and Delong, 2006] Kellie, K. A. and Delong, R. A. (2006). *Geospatial survey operations manual.* Alaska Department of Fish and Game, Fairbanks, Alaska.

[Kielland and Bryant, 1998] Kielland, K. and Bryant, J. P. (1998). Moose herbivory in taiga: effects on biogeochemistry and vegetation dynamics in primary succession. *Oikos*, 82:377–383.

[Mardia and Marshall, 1984] Mardia, K. V. and Marshall, R. J. (1984). Maximum likelihood estimation of models for residual covariance in spatial regression. *Biometrika*, pages 135–146.

[Mazzella and Mazzella, 2013] Mazzella, A. and Mazzella, A. (2013). The importance of model choice for experimental semivariogram modeling and its consequence in evaluation process. *Journal of Engineering*, pages 1–10.

[McCrea and Morgan, 2015] McCrea, R. S. and Morgan, B. J. T. (2015). *Analysis of capture-recapture data.* CRC Press, Boca Raton, Florida.

[Patterson and Thompson, 1974] Patterson, H. D. and Thompson, R. (1974). Maximum likelihood estimation of components of variance. *Proceedings of the 8th international biometric conference*, pages 197–207.

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

[Roger S. Bivand, 2013] Roger S. Bivand, Edzer Pebesma, V. G.-R. (2013). *Applied spatial data analysis with R, Second edition.* Springer, New York, NY.

[Royle, 2004] Royle, J. A. (2004). N-mixture models for estimating population size from spatially replicated counts. *Biometrics*, 60:108–115.

[Royle et al., 2014] Royle, J. A., Chandler, R. B., Sollmann, R., and Gardner, B. (2014). *Spatial capture-recapture.* Academic Press, Waltham, MA.

[Scheaffer et al., 1996] Scheaffer, R., Mendenhall, W., and Ott, R. (1996). *Elementary survey sampling,* 5^{th} *ed.* Duxbury Press, Belmont, CA.

[Seaton, 2014] Seaton, K. (2014). *Evaluating options for improving GSPE performance and developing a sightability correction factor.* Alaska Department of Fish and Game, Fairbanks, Alaska.

[Timmerman, 1974] Timmerman, H. R. (1974). Moose inventory methods: a review. *Naturaliste Canadien*, 101:615–629.

[Van Groenigan, 2000] Van Groenigan, J. (2000). The influence of variogram parameters on optimal sampling schemes for mapping kriging. *Geoderma*, 97:223–236.

[Ver Hoef, 2002] Ver Hoef, J. M. (2002). Sampling and geostatistics for spatial data. *Ecoscience*, 9:152–161.

[Ver Hoef, 2008] Ver Hoef, J. M. (2008). Spatial methods for plot-based sampling of wildlife populations. *Environmental and Ecological Statistics*, 15:3–13.

[Ver Hoef, 2009] Ver Hoef, J. M. (2009). Finite population block kriging with detection functions. *Unpublished manuscript.*

[Williams et al., 2002] Williams, B. K., Nichols, J. D., and Conroy, M. J. (2002). *Analysis and management of animal populations: modeling, estimation, and decision making.* Academic Press, San Diego, California.

Appendix 1: R code for Chapter 1 simulations

```
#############################################################
#####~~~ Unit Configuration Simulations, Chapter 1 ~~~#####
#####~~~ Author: G.G. Frye, 2016                  ~~~#####
#############################################################

##### Start with clean slate
rm(list=ls())
dev.off()

##### Working directory
workdir <- ""
setwd(workdir)
getwd()
dir()

##### Date
Date <- ""
RunNo <- ""

##### Packages
library(rgdal)
library(maptools)
library(spatstat)
library(sp)
library(rgeos)

##### Analysis Area: "HIGH", "LOW", or "ALL"
#####     (This specifies the area for which a population
#####      estimate is desired: high/low stratum only
#####      or whole survey region)

ANAREA <- "ALL"

#=============================================================
# Load shapefiles for: (1) Standard units (STD),
#     (2) High stratum Togiak units (HIGH), and
#     (3) Low stratum Togiak units (LOW)
#=============================================================

HIGH <- readOGR(dsn="", layer = "")
LOW <- readOGR(dsn="", layer = "")
STD <- readOGR(dsn="", layer = "")
proj4string(STD) = proj4string(HIGH) #Define the projection

#=============================================================
```

```
# Load GSPE functions
#=============================================================

source("GSPE_Functions.r")

#=============================================================
# Choose number of units to sample from each configuration
#    and for each stratum
#=============================================================

sizeHS <- 100 # Standard units, high stratum -- 209 available
sizeLS <- 80 # Standard units, low stratum -- 748 available

sizeHG <- 20  # Togiak units, high stratum -- 25 available
sizeLG <- 30  # Togiak units, low stratum -- 129 available

#=============================================================
# Choose number of iterations for the loop
#=============================================================

NumIter <- 1000 # Number of iterations for the loop

#=============================================================
# Choose true population size for each stratum
#=============================================================

NumHigh <- 450
NumLow <- 50
NumTotal <- NumHigh + NumLow

#*************************************************************
#=============================================================
# START LOOP HERE
#=============================================================
#*************************************************************

PtEstTog <- matrix(rep(NA, NumIter))
PtEstStd <- matrix(rep(NA, NumIter))

SETog <- matrix(rep(NA, NumIter))
SEStd <- matrix(rep(NA, NumIter))

AreaTog <- matrix(rep(NA, NumIter))
AreaStd <- matrix(rep(NA, NumIter))

ConfInt95Tog <- matrix(NA, NumIter, 2)
ConfInt95Std <- matrix(NA, NumIter, 2)

```

```
94 GassSemVarHi <- matrix(NA, nrow = NumIter, ncol = 3)
95 GassSemVarLo <- matrix(NA, nrow = NumIter, ncol = 3)
96
97 StdSemVarHi <- matrix(NA, nrow = NumIter, ncol = 3)
98 StdSemVarLo <- matrix(NA, nrow = NumIter, ncol = 3)
99
100 for(j in 1:NumIter){
101
102 #=============================================================
103 # Generate random points within each stratum layer, with each
104 #    point representing an individual moose location
105 #=============================================================
106
107 #Random points, high stratum
108 HIGH_PTS <- spsample(HIGH, n = NumHigh, "random")
109
110 #Random points, low stratum
111 LOW_PTS <- spsample(LOW, n = NumLow, "random")
112
113 #=============================================================
114 # Use 'over' functions for point-in-polygon analysis
115 #    to assign the points to appropriate survey units
116 #=============================================================
117
118 ##### HIGH stratum historic units
119
120 HIGH_PTS$GassUnit <- over(HIGH_PTS, HIGH)$UnitID
121
122 ##### LOW stratum historic units
123
124 LOW_PTS$GassUnit <- over(LOW_PTS, LOW)$UnitID
125
126 ##### HIGH stratum standard units
127
128 HIGH_PTS$SUS_ID <- over(HIGH_PTS, STD)$SUS_ID
129 HIGH_PTS$SUS_ID_1 <- over(HIGH_PTS, STD)$SUS_ID_1
130 HIGH_PTS$StdUnit <- HIGH_PTS$SUS_ID + HIGH_PTS$SUS_ID_1
131
132 ##### LOW stratum standard units
133
134 LOW_PTS$SUS_ID <- over(LOW_PTS, STD)$SUS_ID
135 LOW_PTS$SUS_ID_1 <- over(LOW_PTS, STD)$SUS_ID_1
136 LOW_PTS$StdUnit <- LOW_PTS$SUS_ID + LOW_PTS$SUS_ID_1
137
138 #=============================================================
139 # Tally number of simulated moose points in each unit
140 #=============================================================
141
```

```
LOW_PTS <- data.frame(LOW_PTS) # SpatialPointsDataframe to regular
    # dataframe
LOW_PTS$count <- 1 # Add column of 1's to represent a count of 1
    # for each simulated moose
HIGH_PTS <- data.frame(HIGH_PTS) # SpatialPointsDataframe to
    # regular dataframe
HIGH_PTS$count <- 1 # Add column of 1's to represent a count of 1
    # for each simulated moose

byGassUnitLow <- aggregate(LOW_PTS$count, list(LOW_PTS$GassUnit)
    ,sum) # low counts for each Togiak unit
byGassUnitLow$Stratum <- "LOW"

byGassUnitHigh <- aggregate(HIGH_PTS$count, list(HIGH_PTS$
    GassUnit), sum) # high counts for each Togiak unit
byGassUnitHigh$Stratum <- "HIGH"

byGassUnit <- rbind(byGassUnitLow, byGassUnitHigh) # combined
    # counts for each Togiak unit, both strata
byGassUnit <- byGassUnit[order(byGassUnit$Group.1),]

byStdUnitLow <- aggregate(LOW_PTS$count, list(LOW_PTS$StdUnit),
    sum) # low counts for each standard unit

byStdUnitHigh <- aggregate(HIGH_PTS$count, list(HIGH_PTS$
    StdUnit), sum) # high counts for each standard unit

byStdUnitBind <- rbind(byStdUnitLow, byStdUnitHigh) # combined
    # counts for each standard unit
byStdUnit <- aggregate(byStdUnitBind$x, list(byStdUnitBind$
    Group.1), sum)   # Aggregate again -- not unique units
byStdUnit <- byStdUnit[order(byStdUnit$Group.1),] # sort by
    # unit number

#=============================================================
# Add missing 0-count units back to compiled vector of counts
#    for historic Units
#=============================================================

lowtest <- data.frame(LOW)
lowtest <- data.frame(lowtest$UnitID)
lowtest$Stratum <- "LOW"
lowtest$x <- 0
colnames(lowtest) <- c("Group.1", "Stratum", "x")
lowtest <- data.frame(lowtest$Group.1, lowtest$x, lowtest$Stratum)
colnames(lowtest) <- c("Group.1", "x", "Stratum")

hightest <- data.frame(HIGH)
```

```
hightest <- data.frame(hightest$UnitID)
hightest$Stratum <- "HIGH"
hightest$x <- 0
colnames(hightest) <- c("Group.1", "Stratum", "x")
hightest <- data.frame(hightest$Group.1, hightest$x, hightest$
    Stratum)
colnames(hightest) <- c("Group.1", "x", "Stratum")

bothtest <- rbind(lowtest, hightest)
bothtest <- bothtest[order(bothtest$Group.1),] # sort by
    # unit number
GassAll <- rbind(bothtest, byGassUnit)
GassCounts <- aggregate(GassAll$x, list(GassAll$Group.1), sum)
GassCounts <- GassCounts[order(GassCounts$Group.1),] # sort by
    # unit number
GassCounts <- data.frame(GassCounts$Group.1, GassCounts$x,
    # bothtest$Stratum)
colnames(GassCounts) <- c("Group.1", "x", "Stratum")

#=============================================================
# Add missing 0-count units back to compiled vector of counts
#    for standard units
#=============================================================

stdtest <- data.frame(STD)
stdtest <- data.frame(stdtest$SUS_ID, stdtest$SUS_ID_1)
stdtest$Group.1 <- stdtest$stdtest.SUS_ID +
    stdtest$stdtest.SUS_ID_1
stdtest <- data.frame(stdtest$Group.1)
stdtest$x <- 0
colnames(stdtest) <- c("Group.1", "x")

StdAll <- rbind(stdtest, byStdUnit)
StdCounts <- aggregate(StdAll$x, list(StdAll$Group.1), sum)
colnames(StdCounts) <- c("UnitID", "Moose_Count")

#=============================================================
# Use 'over' functions for point-in-polygon analysis
#    to assign the points to appropriate survey units
#=============================================================

STD$Stratum <- NA
STD$UnitID <- STD$SUS_ID + STD$SUS_ID_1
HIGH$Strat <- "HIGH"

#If a standard unit overlaps a high unit, then the standard unit
    #gets "HIGH" in STD$Stratum; else "LOW"
StdStrat <- data.frame(over(STD, HIGH, returnList = FALSE), STD$
```

```
      UnitID)
StdStrat <- data.frame(StdStrat$STD.UnitID, StdStrat$Stratum)
StdStrat$Stratum[StdStrat$StdStrat.Stratum == "HIGH"] <- "HIGH"
StdStrat$Stratum[is.na(StdStrat$StdStrat.Stratum)] <- "LOW"
StdStrat <- data.frame(StdStrat$StdStrat.STD.UnitID, StdStrat$
      Stratum)
colnames(StdStrat) <- c("UnitID", "Stratum")
StdStrat <- StdStrat[order(StdStrat$UnitID),]

#=============================================================
# Set up standard unit dataframe for FPBK
#=============================================================

colnames(StdCounts) <- c("UnitID", "Moose_Count")
StdCounts <- StdCounts[order(StdCounts$UnitID),]
Stdmerge1 <- merge(StdCounts, StdStrat, by = "UnitID")

STDdat <- data.frame(STD)
STDdat$Lat <- STDdat$CENTRLAT + STDdat$CENTRLAT_1
STDdat$Long <- STDdat$CENTRLON + STDdat$CENTRLON_1
STDdat$Counted <- 0
STDdat2 <- STDdat[,c(39,40,41,42)]
STDdat2 <- STDdat2[order(STDdat2$UnitID),]
Stdmerge2 <- merge(Stdmerge1, STDdat2)
Stdmerge2 <- Stdmerge2[,c(1,4,5,2,3,6)]
colnames(Stdmerge2) <- c("UnitID","CentrLat","CentrLong",
      "Moose_Count","Stratum","Counted")
StdData <- Stdmerge2

#=============================================================
# Randomly select standard sample units
#=============================================================

##### High Stratum

StdHigh <- subset(StdData, StdData$Stratum == "HIGH")
StdHigh <- data.matrix(StdHigh)
nH <- length(StdHigh[,1])
HighIndex <- sample(nH, size=sizeHS ,replace = FALSE)
for(i in 1:sizeHS){
      StdHigh[HighIndex[i],6] <- 1
      }
ZStdHigh <- data.frame(StdHigh)

##### Low Stratum

StdLow <- subset(StdData, StdData$Stratum == "LOW")
StdLow <- data.matrix(StdLow)
```

```
nL <- length(StdLow[,1])
LowIndex <- sample(nL, size=sizeLS ,replace = FALSE)
for(i in 1:sizeLS){
      StdLow[LowIndex[i],6] <- 1
      }
ZStdLow <- data.frame(StdLow)

ZStd <- rbind(ZStdHigh, ZStdLow)

#=============================================================
# Add additional required columns to ZStd
#=============================================================

STD <- STD[order(STD$UnitID),]
ZStd <- ZStd[order(ZStd$UnitID),]
ZStd$AreaMi <- STD$AREAMI + STD$AREAMI_1
ZStd$surveyid <- 77
ZStd$columnpred <- NA
ZStd$Stratum[ZStd$Stratum == 2] <- "LOW"
ZStd$Stratum[ZStd$Stratum == 1] <- "HIGH"

#=============================================================
# Define analysis area based on choice at start of script
#=============================================================

if(ANAREA=="HIGH"){
      for(i2 in 1:length(ZStd$columnpred)){
            if(ZStd$Stratum[i2] == "HIGH"){ZStd$columnpred[i2] <- 1
            }else{ZStd$columnpred[i2] <- 0}
      }
}
if(ANAREA=="LOW"){
      for(i2 in 1:length(ZStd$columnpred)){
            if(ZStd$Stratum[i2] == "LOW"){ZStd$columnpred[i2] <- 1
            }else{ZStd$columnpred[i2] <- 0}
      }
}
if(ANAREA=="ALL"){ZStd$columnpred <- 1}

##### Reorganize columns

ZStd <- ZStd[,c(8,1,6,7,4,5,2,3,9)]

#=============================================================
# Perform block kriging with functions written by Jay Ver Hoef
#     on standard units
#     (see Appendix 4 for GSPE functions)
#=============================================================
```

```
334
335 data <- ZStd
336 column.pred <- "columnpred"
337 column.ana <- "Moose_Count"
338 column.unitid <- "UnitID"
339 column.ana.formula <- "[UNKNOWN]"
340 strat <- "Stratum"
341 area <- "AreaMi"
342 column.lat <- "CentrLat"
343 column.lon <- "CentrLong"
344 sampled <- "Counted"
345 column.surveyid <- "surveyid"
346 Stdcalc.out <- geo.moosepop(column.ana = column.ana, strat =
347       strat, data = data, sampled = sampled, area = area,
348       column.pred = column.pred, column.lat=column.lat,
349       column.lon=column.lon)
350 inpt.parms <- list(column.pred=column.pred,column.ana=
351       column.ana,column.ana.formula=column.ana.formula,
352       strat=strat,area=area,sampled=sampled)
353
354 PtEstStd[j,] <- Stdcalc.out$estimate.total
355 SEStd[j,] <- Stdcalc.out$estimate.standard.error
356 ConfInt95Std[j,] <- Stdcalc.out$conf.int.95
357 StdCI90i <- Stdcalc.out$ci90
358 StdCI80i <- Stdcalc.out$ci80
359 AreaStd[j,] <- Stdcalc.out$sampled.area[3,2]
360 StdSemVarHi[j,1] <- Stdcalc.out$parmest1[1,1]
361 StdSemVarHi[j,2] <- Stdcalc.out$parmest1[1,2]
362 StdSemVarHi[j,3] <- Stdcalc.out$parmest1[1,3]
363 StdSemVarLo[j,1] <- Stdcalc.out$parmest2[1,1]
364 StdSemVarLo[j,2] <- Stdcalc.out$parmest2[1,2]
365 StdSemVarLo[j,3] <- Stdcalc.out$parmest2[1,3]
366
367
368 #=============================================================
369 # Set up historic unit dataframe for block kriging
370 #=============================================================
371
372 GassCentr <- read.csv("Togiak_centroids.csv", header=TRUE)
373
374 length(GassCounts[,1])
375 colnames(GassCounts) <- c("UnitID", "Moose_Count", "Stratum")
376 GassCounts <- GassCounts[order(GassCounts$UnitID),]
377
378 GassData <- merge(GassCentr, GassCounts, by = "UnitID")
379 GassData <- GassData[,c(1,7,6,8,9)]
380 colnames(GassData) <- c("UnitID","CentrLat","CentrLong",
381       "Moose_Count","Stratum")
```

```
382
383 GassData$Counted <- 0
384
385 ###### High Stratum sampling
386
387 GassHigh <- subset(GassData, GassData$Stratum == "HIGH")
388 GassHigh <- data.matrix(GassHigh)
389 nH <- length(GassHigh[,1])
390 HighIndex <- sample(nH, size=sizeHG ,replace = FALSE)
391 for(i in 1:sizeHG)
392 {
393       GassHigh[HighIndex[i],6] <- 1
394 }
395 ZGassHigh <- data.frame(GassHigh)
396
397 ##### Low Stratum sampling
398
399 GassLow <- subset(GassData, GassData$Stratum == "LOW")
400 GassLow <- data.matrix(GassLow)
401 nL <- length(GassLow[,1])
402 LowIndex <- sample(nL, size=sizeLG ,replace = FALSE)
403 for(i in 1:sizeLG)
404 {
405       GassLow[LowIndex[i],6] <- 1
406 }
407 ZGassLow <- data.frame(GassLow)
408
409 ZGass <- rbind(ZGassHigh, ZGassLow)
410
411
412 ##### Add additional required columns to ZGass
413
414 LowDF <- data.frame(LOW)
415 LowArea <- data.frame(LowDF$UnitID, LowDF$unit_area_)
416 colnames(LowArea) <- c("UnitID", "AreaKM")
417 HighDF <- data.frame(HIGH)
418 HighArea <- data.frame(HighDF$UnitID, HighDF$unit_area_)
419 colnames(HighArea) <- c("UnitID", "AreaKM")
420
421 GassL_H <- rbind(LowArea, HighArea)
422 GassL_H <- GassL_H[order(GassL_H$UnitID),]
423 ZGass <- ZGass[order(ZGass$UnitID),]
424 ZGass$AreaMi <- GassL_H$AreaKM * 0.386102159
425
426 ##### Add final columns and convert numeric stratum labels to
427 ##### names
428
429 ZGass$surveyid <- 77
```

```
ZGass$columnpred <- NA
ZGass <- ZGass[,c(7,1,6,9,4,5,2,3,8)]
ZGass$Stratum[ZGass$Stratum == 1] <- "LOW"
ZGass$Stratum[ZGass$Stratum == 2] <- "HIGH"

##### Define analysis area based on choice at start of script

if(ANAREA=="HIGH"){
      for(i2 in 1:length(ZGass$columnpred)){
            if(ZGass$Stratum[i2] == "HIGH"){ZGass$columnpred[i2] <-
               1
            }else{ZGass$columnpred[i2] <- 0}
      }
}
if(ANAREA=="LOW"){
      for(i2 in 1:length(ZGass$columnpred)){
            if(ZGass$Stratum[i2] == "LOW"){ZGass$columnpred[i2] <- 1
            }else{ZGass$columnpred[i2] <- 0}
      }
}
if(ANAREA=="ALL"){ZGass$columnpred <- 1}

#=============================================================
# Perform block kriging with functions written by Jay Ver Hoef
#     on historic units
#     (see Appendix 4 for GSPE functions)
#=============================================================

data <- ZGass # Assign appropriate data frame
column.pred <- "columnpred"
column.ana <- "Moose_Count"
column.unitid <- "UnitID"
column.ana.formula <- "[UNKNOWN]"
strat <- "Stratum"
area <- "AreaMi"
column.lat <- "CentrLat"
column.lon <- "CentrLong"
sampled <- "Counted"
column.surveyid <- "surveyid"
Gasscalc.out<-geo.moosepop(column.ana = column.ana, strat =
      strat,data = data, sampled = sampled, area = area,
      column.pred = column.pred, column.lat=column.lat,
      column.lon=column.lon)
inpt.parms<-list(column.pred=column.pred,column.ana=column.ana,
      column.ana.formula=column.ana.formula,strat=strat,
      area=area,sampled=sampled)

PtEstTog[j,] <- Gasscalc.out$estimate.total
```

```
SETog[j,] <- Gasscalc.out$estimate.standard.error
ConfInt95Tog[j,] <- Gasscalc.out$conf.int.95
GassCI90i <- Gasscalc.out$ci90
GassCI80i <- Gasscalc.out$ci80
AreaTog[j,] <- Gasscalc.out$sampled.area[3,2]
GassSemVarHi[j,1] <- Gasscalc.out$parmest1[1,1]
GassSemVarHi[j,2] <- Gasscalc.out$parmest1[1,2]
GassSemVarHi[j,3] <- Gasscalc.out$parmest1[1,3]
GassSemVarLo[j,1] <- Gasscalc.out$parmest2[1,1]
GassSemVarLo[j,2] <- Gasscalc.out$parmest2[1,2]
GassSemVarLo[j,3] <- Gasscalc.out$parmest2[1,3]

##### Keep track of loop progress

print(paste("Loop #", j, " -- ", j/NumIter*100, "% complete"))
flush.console()

#*****************************************************************
#=================================================================
# END LOOP HERE
#=================================================================
#*****************************************************************
}

#=================================================================
# Confidence interval coverage
#=================================================================

if(ANAREA == "ALL"){
#Historic Coverage:
CI95TogDF <- data.frame(ConfInt95Tog)
colnames(CI95TogDF) <- c("Lower", "Upper")
CI95TogDF$in_interval <- 0
CI95Tog <- as.matrix(CI95TogDF)
for (i in 1:NumIter){
      if (CI95Tog[i,1] < NumTotal & CI95Tog[i,2] > NumTotal){
           CI95Tog[i,3] <- 1}
      }
(CoverageTog <- sum(CI95Tog[,3]) / nrow(CI95Tog))
}

if(ANAREA == "HIGH"){
#Historic Coverage:
CI95TogDF <- data.frame(ConfInt95Tog)
colnames(CI95TogDF) <- c("Lower", "Upper")
CI95TogDF$in_interval <- 0
CI95Tog <- as.matrix(CI95TogDF)
for (i in 1:NumIter){
```

```
525       if (CI95Tog[i,1] < NumHigh & CI95Tog[i,2] > NumHigh){
526            CI95Tog[i,3] <- 1}
527       }
528 (CoverageTog <- sum(CI95Tog[,3]) / nrow(CI95Tog))
529 }
530
531 if(ANAREA == "LOW"){
532 #Historic Coverage:
533 CI95TogDF <- data.frame(ConfInt95Tog)
534 colnames(CI95TogDF) <- c("Lower", "Upper")
535 CI95TogDF$in_interval <- 0
536 CI95Tog <- as.matrix(CI95TogDF)
537 for (i in 1:NumIter){
538       if (CI95Tog[i,1] < NumLow & CI95Tog[i,2] > NumLow){
539            CI95Tog[i,3] <- 1}
540       }
541 (CoverageTog <- sum(CI95Tog[,3]) / nrow(CI95Tog))
542 }
543
544 if(ANAREA == "ALL"){
545 #Std Unit Coverage:
546 CI95StdDF <- data.frame(ConfInt95Std)
547 colnames(CI95StdDF) <- c("Lower", "Upper")
548 CI95StdDF$in_interval <- 0
549 CI95Std <- as.matrix(CI95StdDF)
550 for (i in 1:NumIter){
551       if (CI95Std[i,1] < NumTotal & CI95Std[i,2] > NumTotal){
552            CI95Std[i,3] <- 1}
553       }
554 (CoverageStd <- sum(CI95Std[,3]) / nrow(CI95Std))
555 }
556
557 if(ANAREA == "HIGH"){
558 #Std Unit Coverage:
559 CI95StdDF <- data.frame(ConfInt95Std)
560 colnames(CI95StdDF) <- c("Lower", "Upper")
561 CI95StdDF$in_interval <- 0
562 CI95Std <- as.matrix(CI95StdDF)
563 for (i in 1:NumIter){
564       if (CI95Std[i,1] < NumHigh & CI95Std[i,2] > NumHigh){
565            CI95Std[i,3] <- 1}
566       }
567 (CoverageStd <- sum(CI95Std[,3]) / nrow(CI95Std))
568 }
569
570 if(ANAREA == "LOW"){
571 #Std Unit Coverage:
572 CI95StdDF <- data.frame(ConfInt95Std)
```

```
colnames(CI95StdDF) <- c("Lower", "Upper")
CI95StdDF$in_interval <- 0
CI95Std <- as.matrix(CI95StdDF)
for (i in 1:NumIter){
      if (CI95Std[i,1] < NumLow & CI95Std[i,2] > NumLow){
            CI95Std[i,3] <- 1}
      }
(CoverageStd <- sum(CI95Std[,3]) / nrow(CI95Std))
}

#=============================================================
# Write results to a file
#=============================================================

workdir <- ""
setwd(workdir)
getwd()
dir()

##### File with raw results from each iteration

sink(paste(Date,RunNo,ANAREA,"_",NumHigh,"H_",NumLow,"L_",
      NumIter,"Iter","Results.txt", sep=""))
cat("\n")
cat(paste(Date,RunNo,ANAREA,"_",NumHigh,"H_",NumLow,"L_",
      NumIter,"Iter","Results", sep=""))

cat("\n\nNumber of iterations =")
NumIter

cat("\nNumber of low stratum moose =")
NumLow

cat("\nNumber of high stratum moose =")
NumHigh

cat("\nTotal number of moose =")
NumTotal

cat("\nVector of Togiak Grid Point Estimates =")
data.frame(PtEstTog)

cat("\nVector of Standard Grid Point Estimates =")
data.frame(PtEstStd)

cat("\nVector of Togiak Grid SEs =")
data.frame(SETog)

```

```
621 cat("\nVector of Standard Grid SEs =")
622 data.frame(SEStd)
623
624 cat("\nVector of Togiak Grid Sampled Areas (sq miles) =")
625 data.frame(AreaTog)
626
627 cat("\nVector of Standard Grid Sampled Areas (sq miles) = \n\n")
628 data.frame(AreaStd)
629
630 cat("\nTog 95% CI and 1/0 Pt Inclusion =")
631 CI95Tog
632
633 cat("\nStd 95% CI and 1/0 Pt Inclusion =")
634 CI95Std
635
636 cat("\nTogiak Fitted Semi-variograms --- High =")
637 GassSemVarHi
638 cat("\nTogiak Fitted Semi-variograms --- Low =")
639 GassSemVarLo
640
641 cat("\nStandard Fitted Semi-variograms --- High =")
642 StdSemVarHi
643 cat("\nStandard Fitted Semi-variograms --- Low =")
644 StdSemVarLo
645
646 sink()
647
648 ##### File with summarized results
649
650 sink(paste(Date,RunNo,ANAREA,"_",NumHigh,"H_",NumLow,"L_",NumIter,
651       "Iter","Summary.txt", sep=""))
652 cat("\n")
653 cat(paste(Date,RunNo,ANAREA,"_",NumHigh,"H_",NumLow,"L_",NumIter,
654       "Iter","Summary.txt", sep=""))
655
656 cat("\n\nNumber of iterations =")
657 NumIter
658
659 cat("\nNumber of low stratum moose =")
660 NumLow
661
662 cat("\nNumber of high stratum moose =")
663 NumHigh
664
665 cat("\nTotal number of moose =")
666 NumTotal
667
668 cat("\nMean(PtEstTog) =")
```

```
669 mean(PtEstTog)
670
671 cat("\nMean(PtEstStd) =")
672 mean(PtEstStd)
673
674 cat("\nMean SE(SETog) =")
675 mean(SETog)
676
677 cat("\nMean SE(SEStd) =")
678 mean(SEStd)
679
680 if(ANAREA == "ALL"){
681 cat("\nTogiak Unit Bias =")
682 mean(PtEstTog) - NumTotal
683 }
684
685 if(ANAREA == "LOW"){
686 cat("\nTogiak Unit Bias =")
687 mean(PtEstTog) - NumLow
688 }
689
690 if(ANAREA == "HIGH"){
691 cat("\nTogiak Unit Bias =")
692 mean(PtEstTog) - NumHigh
693 }
694
695 if(ANAREA == "ALL"){
696 cat("\nStandard Unit Bias =")
697 mean(PtEstStd) - NumTotal
698 }
699
700 if(ANAREA == "LOW"){
701 cat("\nStandard Unit Bias =")
702 mean(PtEstStd) - NumLow
703 }
704
705 if(ANAREA == "HIGH"){
706 cat("\nStandard Unit Bias =")
707 mean(PtEstStd) - NumHigh
708 }
709 cat("\nTogiak Coverage -- 95% CI =")
710 CoverageTog
711
712 cat("\nStd Coverage -- 95% CI =")
713 CoverageStd
714
715 cat("\nMean of Togiak Grid Sampled Area (sq miles) =")
716 mean(AreaTog)
```

```
717
718 cat("\nMean of Standard Grid Sampled Area (sq miles) = \n\n")
719 mean(AreaStd)
720
721 sink()
```

./Config_Simulations_Neat.R

Appendix 2: R code for Chapter 2 simulations

```
#############################################################
#####~~~ Sampling Intensity Simulations, Chapter 2 ~~~#####
#####~~~ Author: G.G. Frye, 2016                  ~~~#####
#############################################################

##### Start with clean slate
rm(list=ls())
 dev.off()

##### Working directory
workdir <- ""
setwd(workdir)
getwd()
dir()

##### Date
Date <- ""
RunNo <- ""

##### Packages
library(rgdal)
library(maptools)
library(spatstat)
library(sp)
library(rgeos)

##### Analysis Area: "HIGH", "LOW", or "ALL"
#####     (This specifies the area for which a population
#####      estimate is desired: high/low stratum only
#####      or whole survey region)

ANAREA <- "ALL"

#=============================================================
# Load shapefiles for whole standard units with strata
#      delineated on the basis of 2011 survey
#=============================================================

AllUnits <- readOGR(dsn="", layer = "")
plot(AllUnits)
AllUnits@data$ID <- AllUnits@data$SUS_ID + AllUnits@data$
      SUS_ID_1 + AllUnits@data$SUS_ID_12 + AllUnits@data$
      SUS_ID__13 + AllUnits@data$SUS_ID__14 + AllUnits@data$
      SUS_ID__15
AllUnits$UnitID <- AllUnits$SUS_ID_1 + AllUnits$SUS_ID_12
```

```
46 head(AllUnits@data)
47
48 STDHIGH <- AllUnits[which(AllUnits$Stratum == "HIGH"),]
49 plot(STDHIGH)
50 proj4string(STDHIGH)
51 str(STDHIGH@data)
52 head(STDHIGH@data)
53
54 STDLOW <- AllUnits[which(AllUnits$Stratum == "LOW"),]
55 plot(STDLOW)
56 proj4string(STDLOW)
57 str(STDLOW@data)
58 head(STDLOW@data)
59
60 #=============================================================
61 # Load shapefiles for whole standard units with strata
62 #      delineated on the basis of 2011 survey
63 #=============================================================
64
65 source("GSPE_Functions.r")
66
67 #=============================================================
68 # Choose number of units to sample from low stratum
69 #    (NOTE: >= 20 required for each stratum)
70 #=============================================================
71
72 ##### Low stratum fixed, high stratum varying with p-loop below
73
74 sizeLS <- 30 # Standard units, low stratum -- 748 available
75
76 #=============================================================
77 # Choose number of iterations for the loop
78 #=============================================================
79
80 NumIter <- 1000 # Number of iterations for the loop
81
82 #=============================================================
83 # Choose true population size within each stratum
84 #=============================================================
85
86 ##### Using the 2011 counts
87
88 NumHigh <- 1144
89 NumLow <- 482
90 NumTotal <- NumHigh + NumLow
91
92 #=============================================================
93 # Fill HighVector with high stratum units to be sampled
```

```
#=============================================================

HighVector <- c(20, 30, 50, 100, 200)

#*************************************************************
#=============================================================
# START LOOPS HERE
#=============================================================
#*************************************************************

for(p in 1:length(HighVector)){ # Loop through HighVector

sizeHS <- HighVector[p]

PtEstStd <- matrix(rep(NA, NumIter))
colnames(PtEstStd) <- c("PtEst")

SEStd <- matrix(rep(NA, NumIter))
colnames(SEStd) <- c("SE")

UnitsSamp <- matrix(NA, nrow = NumIter, ncol = 3)
colnames(UnitsSamp) <- c("HIGH", "LOW", "TOTAL")

TotalSamp <- matrix(NA, nrow = NumIter, ncol = 3)
colnames(TotalSamp) <- c("HIGH", "LOW", "TOTAL")

MooseCount <- matrix(NA, nrow = NumIter, ncol = 3)
colnames(MooseCount) <- c("HIGH", "LOW", "TOTAL")

SampAreaStd <- matrix(NA, nrow = NumIter, ncol = 3)
colnames(SampAreaStd) <- c("HIGH", "LOW", "TOTAL")
TotalAreaStd <- matrix(NA, nrow = NumIter, ncol = 3)
colnames(TotalAreaStd) <- c("HIGH", "LOW", "TOTAL")

ConfInt95Std <- matrix(NA, NumIter, 2)
colnames(ConfInt95Std) <- c("Lower95CL", "Upper95CL")

ConfInt80Std <- matrix(NA, NumIter, 2)
colnames(ConfInt80Std) <- c("Lower80CL", "Upper80CL")

ConfInt90Std <- matrix(NA, NumIter, 2)
colnames(ConfInt90Std) <- c("Lower90CL", "Upper90CL")

CIpropMean95 <- matrix(NA, NumIter)
colnames(CIpropMean95) <- c("CIpropMean95")

CIpropMean80 <- matrix(NA, NumIter)
colnames(CIpropMean80) <- c("CIpropMean80")
```

```
142
143 CIpropMean90 <- matrix(NA, NumIter)
144 colnames(CIpropMean90) <- c("CIpropMean90")
145
146 StdSemVarHi <- matrix(NA, nrow = NumIter, ncol = 3)
147 colnames(StdSemVarHi) <- c("Nugget", "Sill", "Range")
148
149 StdSemVarLo <- matrix(NA, nrow = NumIter, ncol = 3)
150 colnames(StdSemVarLo) <- c("Nugget", "Sill", "Range")
151
152 for(j in 1:NumIter){ # Loop through specified number of
153 #    simulated populations
154
155 #=============================================================
156 # Generate random points within each stratum layer, with each
157 #    point representing an individual moose location
158 #=============================================================
159
160 ##### Random points, high stratum
161
162 HIGH_PTS <- spsample(STDHIGH, n = NumHigh, "random")
163
164 ##### Random points, low stratum
165
166 LOW_PTS <- spsample(STDLOW, n = NumLow, "random")
167
168 #=============================================================
169 # Use 'over' functions for point-in-polygon analysis
170 #    to assign the random points to appropriate survey units
171 #=============================================================
172
173 ##### HIGH stratum points to units
174
175 HIGH_PTS$SUS_ID_1 <- over(HIGH_PTS, AllUnits)$SUS_ID_1
176 HIGH_PTS$SUS_ID_12 <- over(HIGH_PTS, AllUnits)$SUS_ID_12
177 HIGH_PTS$STDSTRATUnit <- HIGH_PTS$SUS_ID_1 + HIGH_PTS$SUS_ID_12
178
179 ##### LOW stratum points to units
180
181 LOW_PTS$SUS_ID_1 <- over(LOW_PTS, AllUnits)$SUS_ID_1
182 LOW_PTS$SUS_ID_12 <- over(LOW_PTS, AllUnits)$SUS_ID_12
183 LOW_PTS$STDSTRATUnit <- LOW_PTS$SUS_ID_1 + LOW_PTS$SUS_ID_12
184
185 #=============================================================
186 # Tally number of simulated moose points in each unit
187 #=============================================================
188
189 LOW_PTS <- data.frame(LOW_PTS) #Change SpatialPointsDataframe in
```

```
190       # a regular dataframe
191 LOW_PTS$count <- 1 #Add column of 1's to represent a count of 1
192       # for each simulated moose
193 HIGH_PTS <- data.frame(HIGH_PTS) #Change SpatialPointsDataframe
194       # in a regular dataframe
195 HIGH_PTS$count <- 1 #Add column of 1's to represent a count of
196       # 1 for each simulated moose
197
198 byStdUnitLow <- aggregate(LOW_PTS$count, list(LOW_PTS$
199       STDSTRATUnit), sum) # low counts for each standard unit
200
201 byStdUnitHigh <- aggregate(HIGH_PTS$count, list(HIGH_PTS$
202       STDSTRATUnit), sum) # high counts for each standard unit
203
204 byStdUnitBind <- rbind(byStdUnitLow, byStdUnitHigh) # combined
205       # counts for each standard unit
206 byStdUnit <- aggregate(byStdUnitBind$x, list(byStdUnitBind$
207       Group.1), sum)   # Aggregate again because these aren't
208             # unique units
209 byStdUnit <- byStdUnit[order(byStdUnit$Group.1),] # sort by unit
210       # number
211
212 #=============================================================
213 # Add missing 0-count units back to compiled vector of counts
214 #=============================================================
215
216 stdtest <- data.frame(AllUnits)
217 stdtest <- data.frame(stdtest$SUS_ID_1, stdtest$SUS_ID_12)
218 stdtest$Group.1 <- stdtest$stdtest.SUS_ID_1 + stdtest$
219       stdtest.SUS_ID_12
220 stdtest <- data.frame(stdtest$Group.1)
221 stdtest$x <- 0
222 colnames(stdtest) <- c("Group.1", "x")
223
224 StdAll <- rbind(stdtest, byStdUnit)
225 StdCounts <- aggregate(StdAll$x, list(StdAll$Group.1), sum)
226 colnames(StdCounts) <- c("UnitID", "Moose_Count")
227
228 SSTRAT <- data.frame(AllUnits)
229 SSTRAT$UnitID <- SSTRAT$SUS_ID_1 + SSTRAT$SUS_ID_12
230 StdStrat <- SSTRAT[order(SSTRAT$UnitID),]
231
232
233 #=============================================================
234 # Set up final dataframe for FPBK
235 #=============================================================
236
237 colnames(StdCounts) <- c("UnitID", "Moose_Count")
```

```
StdCounts <- StdCounts[order(StdCounts$UnitID),]
Stdmerge1 <- merge(StdCounts, StdStrat, by = "UnitID")

STDdat <- data.frame(AllUnits)
STDdat$UnitID <- STDdat$SUS_ID_1 + STDdat$SUS_ID_12

STDdat$Counted <- 0

STDdat2 <- STDdat[order(STDdat$UnitID),]
Stdmerge2 <- merge(Stdmerge1, STDdat2, by = "UnitID")
Stdmerge3 <- Stdmerge2[,c("UnitID","Latitude.x","Longitude.x",
      "Moose_Count","Stratum.x","Counted")]
colnames(Stdmerge3) <- c("UnitID","CentrLat","CentrLong",
      "Moose_Count","Stratum","Counted")
StdData <- Stdmerge3


#=============================================================
# Randomly select sample units
#=============================================================


##### High Stratum

StdHigh <- subset(StdData, StdData$Stratum == "HIGH")
StdHigh <- data.matrix(StdHigh)
nH <- length(StdHigh[,1])
HighIndex <- sample(nH, size=sizeHS ,replace = FALSE)
for(i in 1:sizeHS){
      StdHigh[HighIndex[i],6] <- 1
      }
ZStdHigh <- data.frame(StdHigh)

##### Low Stratum

StdLow <- subset(StdData, StdData$Stratum == "LOW")
StdLow <- data.matrix(StdLow)
nL <- length(StdLow[,1])
LowIndex <- sample(nL, size=sizeLS ,replace = FALSE)
for(i in 1:sizeLS){
      StdLow[LowIndex[i],6] <- 1
      }
ZStdLow <- data.frame(StdLow)

ZStd <- rbind(ZStdHigh, ZStdLow)

##### Add additional required columns to ZStd

```

```
STD <- AllUnits[order(AllUnits$UnitID),]
ZStd <- ZStd[order(ZStd$UnitID),]
ZStd$AreaMi <- AllUnits$Area
ZStd$surveyid <- 77
ZStd$columnpred <- NA
ZStd$Stratum[ZStd$Stratum == 2] <- "LOW"
ZStd$Stratum[ZStd$Stratum == 1] <- "HIGH"

#=============================================================
# Define analysis area based on choice at start of script
#=============================================================

if(ANAREA=="HIGH"){
      for(i2 in 1:length(ZStd$columnpred)){
            if(ZStd$Stratum[i2] == "HIGH"){ZStd$columnpred[i2] <- 1
            }else{ZStd$columnpred[i2] <- 0}
      }
}
if(ANAREA=="LOW"){
      for(i2 in 1:length(ZStd$columnpred)){
            if(ZStd$Stratum[i2] == "LOW"){ZStd$columnpred[i2] <- 1
            }else{ZStd$columnpred[i2] <- 0}
      }
}
if(ANAREA=="ALL"){ZStd$columnpred <- 1}

#=============================================================
# Perform block kriging with functions written by Jay Ver Hoef
#     (see Appendix 4 for GSPE functions)
#=============================================================

##### Specify arguments for geomoosepop function

data <- ZStd
column.pred <- "columnpred"
column.ana <- "Moose_Count"
column.unitid <- "UnitID"
column.ana.formula <- "[UNKNOWN]"
strat <- "Stratum"
area <- "AreaMi"
column.lat <- "CentrLat"
column.lon <- "CentrLong"
sampled <- "Counted"
column.surveyid <- "surveyid"
Stdcalc.out <- geo.moosepop(column.ana = column.ana, strat =
      strat, data = data, sampled = sampled, area = area,
      column.pred = column.pred, column.lat=column.lat,
      column.lon=column.lon)
```

```
334 inpt.parms <- list(column.pred=column.pred,column.ana=
335       column.ana,column.ana.formula=column.ana.formula,
336       strat=strat,area=area,sampled=sampled)
337
338 PtEstStd[j,] <- Stdcalc.out$estimate.total
339 SEStd[j,] <- Stdcalc.out$estimate.standard.error
340 UnitsSamp[j,1] <- Stdcalc.out$sample.sizes[1,2]
341 UnitsSamp[j,2] <- Stdcalc.out$sample.sizes[2,2]
342 UnitsSamp[j,3] <- Stdcalc.out$sample.sizes[3,2]
343 TotalSamp[j,1] <- Stdcalc.out$total.samples[1,2]
344 TotalSamp[j,2] <- Stdcalc.out$total.samples[2,2]
345 TotalSamp[j,3] <- Stdcalc.out$total.samples[3,2]
346 MooseCount[j,1] <- Stdcalc.out$moose.counted[1,2]
347 MooseCount[j,2] <- Stdcalc.out$moose.counted[2,2]
348 MooseCount[j,3] <- Stdcalc.out$moose.counted[3,2]
349 ConfInt95Std[j,] <- Stdcalc.out$conf.int.95
350 ConfInt90Std[j,] <- Stdcalc.out$ci90
351 ConfInt80Std[j,] <- Stdcalc.out$ci80
352 SampAreaStd[j,1] <- Stdcalc.out$sampled.area[1,2]
353 SampAreaStd[j,2] <- Stdcalc.out$sampled.area[2,2]
354 SampAreaStd[j,3] <- Stdcalc.out$sampled.area[3,2]
355 TotalAreaStd[j,1] <- Stdcalc.out$total.area[1,2]
356 TotalAreaStd[j,2] <- Stdcalc.out$total.area[2,2]
357 TotalAreaStd[j,3] <- Stdcalc.out$total.area[3,2]
358 CIpropMean95[j,] <- Stdcalc.out$ci.prop.mean.95
359 CIpropMean90[j,] <- Stdcalc.out$ci.prop.mean.90
360 CIpropMean80[j,] <- Stdcalc.out$ci.prop.mean.80
361 StdSemVarHi[j,1] <- Stdcalc.out$parmest1[1,1]
362 StdSemVarHi[j,2] <- Stdcalc.out$parmest1[1,2]
363 StdSemVarHi[j,3] <- Stdcalc.out$parmest1[1,3]
364 StdSemVarLo[j,1] <- Stdcalc.out$parmest2[1,1]
365 StdSemVarLo[j,2] <- Stdcalc.out$parmest2[1,2]
366 StdSemVarLo[j,3] <- Stdcalc.out$parmest2[1,3]
367
368 ##### Keep track of loop progress
369
370 print(paste("Loop #", j, " -- ", j/NumIter*100, "% complete"))
371 flush.console()
372
373 #*************************************************************
374 #=============================================================
375 # END j LOOP HERE
376 #=============================================================
377 #*************************************************************
378
379 }
380
381 #=============================================================
```

```
# Coverage
#=============================================================

##### 95% #####

if(ANAREA == "ALL"){
CI95StdDF <- data.frame(ConfInt95Std)
colnames(CI95StdDF) <- c("Lower", "Upper")
CI95StdDF$in_interval <- 0
CI95Std <- as.matrix(CI95StdDF)
for (i in 1:NumIter){
      if (CI95Std[i,1] < NumTotal & CI95Std[i,2] > NumTotal){
            CI95Std[i,3] <- 1}
      }
(Coverage95Std <- sum(CI95Std[,3]) / nrow(CI95Std))
      }

if(ANAREA == "HIGH"){
CI95StdDF <- data.frame(ConfInt95Std)
colnames(CI95StdDF) <- c("Lower", "Upper")
CI95StdDF$in_interval <- 0
CI95Std <- as.matrix(CI95StdDF)
for (i in 1:NumIter){
      if (CI95Std[i,1] < NumHigh & CI95Std[i,2] > NumHigh) {
            CI95Std[i,3] <- 1}
      }
(Coverage95Std <- sum(CI95Std[,3]) / nrow(CI95Std))
      }

if(ANAREA == "LOW"){
CI95StdDF <- data.frame(ConfInt95Std)
colnames(CI95StdDF) <- c("Lower", "Upper")
CI95StdDF$in_interval <- 0
CI95Std <- as.matrix(CI95StdDF)
for (i in 1:NumIter){
      if (CI95Std[i,1] < NumLow & CI95Std[i,2] > NumLow){
            CI95Std[i,3] <- 1}
      }
(Coverage95Std <- sum(CI95Std[,3]) / nrow(CI95Std))
      }

##### 90% #####

if(ANAREA == "ALL"){
CI90StdDF <- data.frame(ConfInt90Std)
colnames(CI90StdDF) <- c("Lower", "Upper")
CI90StdDF$in_interval <- 0
CI90Std <- as.matrix(CI90StdDF)
```

```
for (i in 1:NumIter){
      if (CI90Std[i,1] < NumTotal & CI90Std[i,2] > NumTotal){
            CI90Std[i,3] <- 1}
      }
(Coverage90Std <- sum(CI90Std[,3]) / nrow(CI90Std))
      }

if(ANAREA == "HIGH"){
CI90StdDF <- data.frame(ConfInt90Std)
colnames(CI90StdDF) <- c("Lower", "Upper")
CI90StdDF$in_interval <- 0
CI90Std <- as.matrix(CI90StdDF)
for (i in 1:NumIter){
      if (CI90Std[i,1] < NumHigh & CI90Std[i,2] > NumHigh){
            CI90Std[i,3] <- 1}
      }
(Coverage90Std <- sum(CI90Std[,3]) / nrow(CI90Std))
      }

if(ANAREA == "LOW"){
CI90StdDF <- data.frame(ConfInt90Std)
colnames(CI90StdDF) <- c("Lower", "Upper")
CI90StdDF$in_interval <- 0
CI90Std <- as.matrix(CI90StdDF)
for (i in 1:NumIter){
      if (CI90Std[i,1] < NumLow & CI90Std[i,2] > NumLow){
            CI90Std[i,3] <- 1}
      }
(Coverage90Std <- sum(CI90Std[,3]) / nrow(CI90Std))
      }

##### 80% #####

if(ANAREA == "ALL"){
CI80StdDF <- data.frame(ConfInt80Std)
colnames(CI80StdDF) <- c("Lower", "Upper")
CI80StdDF$in_interval <- 0
CI80Std <- as.matrix(CI80StdDF)
for (i in 1:NumIter){
      if (CI80Std[i,1] < NumTotal & CI80Std[i,2] > NumTotal){
            CI80Std[i,3] <- 1}
      }
(Coverage80Std <- sum(CI80Std[,3]) / nrow(CI80Std))
}

if(ANAREA == "HIGH"){
CI80StdDF <- data.frame(ConfInt80Std)
colnames(CI80StdDF) <- c("Lower", "Upper")
```

```
478 CI80StdDF$in_interval <- 0
479 CI80Std <- as.matrix(CI80StdDF)
480 for (i in 1:NumIter){
481       if (CI80Std[i,1] < NumHigh & CI80Std[i,2] > NumHigh){
482             CI80Std[i,3] <- 1}
483       }
484 (Coverage80Std <- sum(CI80Std[,3]) / nrow(CI80Std))
485 }
486
487 if(ANAREA == "LOW"){
488 CI80StdDF <- data.frame(ConfInt80Std)
489 colnames(CI80StdDF) <- c("Lower", "Upper")
490 CI80StdDF$in_interval <- 0
491 CI80Std <- as.matrix(CI80StdDF)
492 for (i in 1:NumIter){
493       if (CI80Std[i,1] < NumLow & CI80Std[i,2] > NumLow){
494             CI80Std[i,3] <- 1}
495       }
496 (Coverage80Std <- sum(CI80Std[,3]) / nrow(CI80Std))
497 }
498
499 #=============================================================
500 # Write results to text files
501 #=============================================================
502
503 workdir <- ""
504 setwd(workdir)
505 getwd()
506 dir()
507
508 ##### File with raw results from each iteration
509
510 sink(paste("Togiak Intensity",Date,RunNo,ANAREA,"_",sizeHS,
511       "H_",sizeLS,"L_","Results.txt", sep=""))
512 cat("\n")
513 cat(paste("Togiak Intensity",Date,RunNo,ANAREA,"_",sizeHS,"H_",
514       sizeLS,"L_","Results", sep=""))
515
516 cat("\n\nNumber of iterations =")
517 print(NumIter)
518
519 cat("\nTrue number of low stratum moose =")
520 print(NumLow)
521
522 cat("\nTrue number of high stratum moose =")
523 print(NumHigh)
524
525 cat("\nTrue total number of moose =")
```

```
526 print(NumTotal)
527
528 cat("\nVector of Standard Grid Point Estimates =")
529 print(data.frame(PtEstStd))
530
531 cat("\nVector of Standard Grid SEs =")
532 print(data.frame(SEStd))
533
534 cat("\nStd 95% CI and 1/0 Pt Inclusion =")
535 print(CI95Std)
536
537 cat("\nStd 90% CI and 1/0 Pt Inclusion =")
538 print(CI90Std)
539
540 cat("\nStd 80% CI and 1/0 Pt Inclusion =")
541 print(CI80Std)
542
543 cat("\nMatrix of Standard Grid Sampled Areas (sq miles) = \n\n")
544 print(data.frame(SampAreaStd))
545
546 cat("\nMatrix of Standard Grid Total Areas (sq miles) = \n\n")
547 print(data.frame(TotalAreaStd))
548
549 cat("\nMatrix of Moose Counts = \n\n")
550 print(data.frame(MooseCount))
551
552 cat("\nVector of CIpropMean95 = \n\n")
553 print(data.frame(CIpropMean95))
554
555 cat("\nVector of CIpropMean90 = \n\n")
556 print(data.frame(CIpropMean90))
557
558 cat("\nVector of CIpropMean80 = \n\n")
559 print(data.frame(CIpropMean80))
560
561 cat("\nStandard Fitted Semi-variograms --- High =")
562 print(StdSemVarHi)
563 cat("\nStandard Fitted Semi-variograms --- Low =")
564 print(StdSemVarLo)
565
566 sink()
567
568 ##### File with summarized results
569
570 sink(paste("Togiak Intensity",Date,RunNo,ANAREA,"_",sizeHS,
571       "H_",sizeLS,"L_","Summary.txt", sep=""))
572 cat("\n")
573 cat(paste("Togiak Intensity",Date,RunNo,ANAREA,"_",sizeHS,
```

```
574        "H_",sizeLS,"L_","Summary.txt", sep=""))
575
576 cat("\n\nNumber of iterations =")
577 print(NumIter)
578
579 cat("\nTrue number of low stratum moose =")
580 print(NumLow)
581
582 cat("\nTrue number of high stratum moose =")
583 print(NumHigh)
584
585 cat("\nTrue total number of moose =")
586 print(NumTotal)
587
588 cat("\nNumber of low stratum units sampled =")
589 print(sizeLS)
590
591 cat("\nNumber of high stratum units sampled =")
592 print(sizeHS)
593
594 cat("\nTotal number of units sampled =")
595 print(sizeLS + sizeHS)
596
597 cat("\nTrue total number of moose =")
598 print(NumTotal)
599
600 cat("\nMean(PtEstStd) =")
601 print(mean(PtEstStd))
602
603 cat("\nMean SE(SEStd) =")
604 print(mean(SEStd))
605
606 cat("\nSD of SE(SEStd) = ")
607 print(sd(SEStd))
608
609 cat("\n95% Wald CI for SE(SEStd) = \n")
610 cat(paste("Lower 95% CL = ",mean(SEStd) - 1.96*sd(SEStd),"\n"))
611 cat(paste("Upper 95% CL = ",mean(SEStd) + 1.96*sd(SEStd),"\n"))
612
613 cat("\n90% Wald CI for SE(SEStd) = \n")
614 cat(paste("Lower 90% CL = ",mean(SEStd) - 1.645*sd(SEStd),"\n"))
615 cat(paste("Upper 90% CL = ",mean(SEStd) + 1.645*sd(SEStd),"\n"))
616
617 cat("\n80% Wald CI for SE(SEStd) = \n")
618 cat(paste("Lower 80% CL = ",mean(SEStd) - 1.28*sd(SEStd),"\n"))
619 cat(paste("Upper 80% CL = ",mean(SEStd) + 1.28*sd(SEStd),"\n"))
620
621 cat("\nStd Coverage -- 95% CI =")
```

```
622 print(Coverage95Std)
623
624 cat("\nStd Coverage -- 90% CI =")
625 print(Coverage90Std)
626
627 cat("\nStd Coverage -- 80% CI =")
628 print(Coverage80Std)
629
630 if(ANAREA == "ALL"){
631 cat("\nStandard Unit Bias =")
632 print(mean(PtEstStd) - NumTotal)
633 }
634
635 if(ANAREA == "LOW"){
636 cat("\nStandard Unit Bias =")
637 print(mean(PtEstStd) - NumLow)
638 }
639
640 if(ANAREA == "HIGH"){
641 cat("\nStandard Unit Bias =")
642 print(mean(PtEstStd) - NumHigh)
643 }
644
645 cat("\nMean of Standard Grid Sampled Area (sq miles) = \n\n")
646 print(mean(SampAreaStd))
647
648 cat("\nMean of Standard Grid Total Area (sq miles) = \n\n")
649 print(mean(TotalAreaStd))
650
651 cat("\n% Area Sampled = \n\n")
652 print(mean(SampAreaStd) / mean(TotalAreaStd))
653
654
655 sink()
656
657 #*****************************************************************
658 #================================================================
659 # END p LOOP HERE
660 #================================================================
661 #*****************************************************************
662
663 }
```

./Intensity_Simulations_Neat.R

Appendix 3: R code for Chapter 3 simulations

```
#################################################################
#####~~~ Simulating clustered distributions, Chapter 3 ~~~#####
#####~~~ Author: G.G. Frye, 2016                       ~~~#####
#################################################################


##### Start with clean slate
rm(list=ls())
dev.off()

##### Working directory
workdir <- ""
setwd(workdir)
getwd()
dir()

##### Date
Date <- ""
RunNo <- ""

##### Packages
library(rgdal)
library(maptools)
library(spatstat)
library(sp)
library(rgeos)

##### Analysis Area: "HIGH", "LOW", or "ALL"
#####     (This specifies the area for which a population
#####      estimate is desired: high/low stratum only
#####      or whole survey region)

ANAREA <- "ALL"

#=================================================================
# Load shapefiles for whole standard units with strata
#      delineated on the basis of 2011 survey
#=================================================================

AllUnits <- readOGR(dsn="", layer = "")
plot(AllUnits)
AllUnits@data$ID <- AllUnits@data$SUS_ID + AllUnits@data$SUS_ID_1
     + AllUnits@data$SUS_ID_12 + AllUnits@data$SUS_ID__13 +
     AllUnits@data$SUS_ID__14 + AllUnits@data$SUS_ID__15
AllUnits$UnitID <- AllUnits$SUS_ID_1 + AllUnits$SUS_ID_12
```

```
46 head(AllUnits@data)
47
48 STDHIGH <- AllUnits[which(AllUnits$Stratum == "HIGH"),]
49 plot(STDHIGH)
50 proj4string(STDHIGH)
51 str(STDHIGH@data)
52 head(STDHIGH@data)
53
54 STDLOW <- AllUnits[which(AllUnits$Stratum == "LOW"),]
55 plot(STDLOW)
56 proj4string(STDLOW)
57 str(STDLOW@data)
58 head(STDLOW@data)
59
60 #=============================================================
61 # Load GSPE functions (see Appendix 4)
62 #=============================================================
63
64 source("GSPE_Functions.r")
65
66 #=============================================================
67 # Choose number of units to sample from each stratum
68 #    (NOTE: >= 20 required for each stratum)
69 #=============================================================
70
71 sizeHS <- 50 # High stratum -- 209 available
72 sizeLS <- 50 # Low stratum -- 748 available
73
74 #=============================================================
75 # Choose number of iterations for the loop
76 #=============================================================
77
78 NumIter <- 1000 # Number of iterations for the loop
79
80 #=============================================================
81 # Choose true population size within each stratum
82 #=============================================================
83
84 ##### Using the 2011 counts
85
86 NumHigh <- 1144
87 NumLow <- 482
88 NumTotal <- NumHigh + NumLow
89
90 #=============================================================
91 # Suppress warnings (NOT errors) for entire script (lots of
92 #    irrelevant warnings related to version conflicts)
93 #   NOTE: Don't forget to reinstate original warnings setting
```

```
#     at end of script!!
#==========================================================

oldw <- getOption("warn")
options(warn = -1)

#==========================================================
# Fill cluster proportion vector (0-1; This scales "nclusters"
#     argument in spsample function.  Lower values create fewer
#     and denser clusters. 1.0 yields a random distribution of
#     individuals.)
#==========================================================

PropVector <- c(0.05,0.1,0.15,0.2,0.25,0.3,0.35,0.4,0.45,0.5,
      0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9,0.95,1)

#************************************************************
#==========================================================
# START LOOPS HERE
#==========================================================
#************************************************************

for(p in 1:length(PropVector)){ # Loop through PropVector

ClusterPropHigh <- ClusterPropLow <- PropVector[p]

PtEstStd <- matrix(rep(NA, NumIter))
colnames(PtEstStd) <- c("PtEst")

SEStd <- matrix(rep(NA, NumIter))
colnames(SEStd) <- c("SE")

UnitsSamp <- matrix(NA, nrow = NumIter, ncol = 3)
colnames(UnitsSamp) <- c("HIGH", "LOW", "TOTAL")

TotalSamp <- matrix(NA, nrow = NumIter, ncol = 3)
colnames(TotalSamp) <- c("HIGH", "LOW", "TOTAL")

MooseCount <- matrix(NA, nrow = NumIter, ncol = 3)
colnames(MooseCount) <- c("HIGH", "LOW", "TOTAL")

SampAreaStd <- matrix(NA, nrow = NumIter, ncol = 3)
colnames(SampAreaStd) <- c("HIGH", "LOW", "TOTAL")
TotalAreaStd <- matrix(NA, nrow = NumIter, ncol = 3)
colnames(TotalAreaStd) <- c("HIGH", "LOW", "TOTAL")

ConfInt95Std <- matrix(NA, NumIter, 2)
colnames(ConfInt95Std) <- c("Lower95CL", "Upper95CL")
```

```
142
143 ConfInt80Std <- matrix(NA, NumIter, 2)
144 colnames(ConfInt80Std) <- c("Lower80CL", "Upper80CL")
145
146 ConfInt90Std <- matrix(NA, NumIter, 2)
147 colnames(ConfInt90Std) <- c("Lower90CL", "Upper90CL")
148
149 CIpropMean95 <- matrix(NA, NumIter)
150 colnames(CIpropMean95) <- c("CIpropMean95")
151
152 CIpropMean80 <- matrix(NA, NumIter)
153 colnames(CIpropMean80) <- c("CIpropMean80")
154
155 CIpropMean90 <- matrix(NA, NumIter)
156 colnames(CIpropMean90) <- c("CIpropMean90")
157
158 StdSemVarHi <- matrix(NA, nrow = NumIter, ncol = 3)
159 colnames(StdSemVarHi) <- c("Nugget", "Sill", "Range")
160
161 StdSemVarLo <- matrix(NA, nrow = NumIter, ncol = 3)
162 colnames(StdSemVarLo) <- c("Nugget", "Sill", "Range")
163
164
165 for(j in 1:NumIter){ # Loop through specified number of
166 #    simulated populations
167
168 #=============================================================
169 # Generate random points within each stratum layer, with each
170 #    point representing an individual moose location
171 #=============================================================
172
173 ##### High stratum points
174
175 repeat{
176      X <- spsample(STDHIGH, n=NumHigh, type = "clustered",
177           nclusters=ceiling(NumHigh*ClusterPropHigh), iter=1000)
178      X$StdUnit <- over(X, STDHIGH)$UnitID
179
180      if(length(X@data$StdUnit) < NumHigh) {
181           X2 <- try((spsample(x=STDHIGH, n=(NumHigh-
182                length(X@data$StdUnit)), type = "clustered", iter=
183                1000, nclusters=ceiling((NumHigh-length(
184                X@data$StdUnit))*ClusterPropHigh))), silent=TRUE)
185           if('try-error' %in% class(X2)) next
186           else for(i in 1:2000){
187                if(length(X2) < NumHigh-length(X@data$StdUnit)){
188                     try(X2 <- spsample(STDHIGH, n=NumHigh-length(
189                     X@data$StdUnit), type = "clustered",
```

```
                        nclusters=ceiling((NumHigh-length(
                        X@data$StdUnit))*ClusterPropHigh)), silent=
                              TRUE)
                              if('try-error' %in% class(X2)) next
                        }
                  if(length(X2) > NumHigh-length(X@data$StdUnit)){
                        X2 <- X2[1:(NumHigh-length(X@data$StdUnit)),]
                        }
                        else break
                  }
            X2$StdUnit <- over(X2, STDHIGH)$UnitID
            row.names(X) <- c(1:length(X))
            try(row.names(X2) <- seq(from=(length(X)+1), to=NumHigh,
                  by=1), silent=TRUE)
                  if('try-error' %in% class(row.names(X2))) next
            X <- spRbind(X,X2)
            }

      if(length(X@data$StdUnit) > NumHigh){
            X <- X[1:NumHigh,]
            }

      if(length(X) == NumHigh) {break}
      }

length(X)
plot(STDHIGH)
points(X)

##### Low stratum points

repeat{
      Y <- spsample(STDLOW, n=NumLow, type = "clustered",
            nclusters=ceiling(NumLow*ClusterPropLow), iter=1000)
      Y$StdUnit <- over(Y, STDLOW)$UnitID

      if(length(Y@data$StdUnit) < NumLow) {
            Y2 <- try((spsample(x=STDLOW, n=(NumLow-length(Y@data$
                  StdUnit)), type = "clustered", iter=1000,
                  nclusters=ceiling((NumLow-length(Y@data$StdUnit))*
                  ClusterPropLow))), silent=TRUE)
                  if('try-error' %in% class(Y2)) next
            else for(i in 1:2000){
                  if(length(Y2) < NumLow-length(Y@data$StdUnit)){try(
                        Y2 <- spsample(STDLOW, n=NumLow-length(
                        Y@data$StdUnit), type = "clustered",
                        nclusters=ceiling((NumLow-length(
```

```
237                         Y@data$StdUnit))*ClusterPropLow)), silent=TRUE
                                )
238                         if('try-error' %in% class(Y2)) next
239                         }
240                   if(length(Y2) > NumLow-length(Y@data$StdUnit)){
241                         Y2 <- Y2[1:(NumLow-length(Y@data$StdUnit)),]
242                         }
243                         else break
244                   }
245             Y2$StdUnit <- over(Y2, STDLOW)$UnitID
246             row.names(Y) <- c(1:length(Y))
247             try(row.names(Y2) <- seq(from=(length(Y)+1), to=(length(
248                   Y) + length(Y2)), by=1), silent=TRUE)
249                   if('try-error' %in% class(row.names(Y2))) next
250             Y <- spRbind(Y,Y2)
251             }
252
253       if(length(Y@data$StdUnit) > NumLow) {
254             Y <- Y[1:NumLow,]
255             }
256
257       if(length(Y) == NumLow) {break}
258       }
259
260 length(Y)
261 plot(STDLOW)
262 points(Y)
263
264 HIGH_PTS <- X
265 LOW_PTS <- Y
266
267 #=============================================================
268 # Use 'over' functions for point-in-polygon analysis
269 #    to assign the random points to appropriate survey units
270 #=============================================================
271
272 ##### HIGH stratum points to units
273
274 HIGH_PTS$SUS_ID_1 <- over(HIGH_PTS, AllUnits)$SUS_ID_1
275 HIGH_PTS$SUS_ID_12 <- over(HIGH_PTS, AllUnits)$SUS_ID_12
276 HIGH_PTS$STDSTRATUnit <- HIGH_PTS$SUS_ID_1 + HIGH_PTS$SUS_ID_12
277
278 ###### LOW stratum points to units
279
280 LOW_PTS$SUS_ID_1 <- over(LOW_PTS, AllUnits)$SUS_ID_1
281 LOW_PTS$SUS_ID_12 <- over(LOW_PTS, AllUnits)$SUS_ID_12
282 LOW_PTS$STDSTRATUnit <- LOW_PTS$SUS_ID_1 + LOW_PTS$SUS_ID_12
283
```

```
#=============================================================
# Tally number of simulated moose points in each unit
#=============================================================

LOW_PTS <- data.frame(LOW_PTS) # SpatialPointsDataframe to
    # dataframe
LOW_PTS$count <- 1 #Add column of 1's to represent a count of 1
    # for each simulated moose
HIGH_PTS <- data.frame(HIGH_PTS) # SpatialPointsDataframe to
    # dataframe
HIGH_PTS$count <- 1 #Add column of 1's to represent a count of 1
    # for each simulated moose

byStdUnitLow <- aggregate(LOW_PTS$count, list(LOW_PTS$
    STDSTRATUnit), sum) # low counts for each standard unit

byStdUnitHigh <- aggregate(HIGH_PTS$count, list(HIGH_PTS$
    STDSTRATUnit), sum) # high counts for each standard unit

byStdUnitBind <- rbind(byStdUnitLow, byStdUnitHigh) # combined
    # counts for each standard unit
byStdUnit <- aggregate(byStdUnitBind$x, list(byStdUnitBind$
    Group.1), sum) # Aggregate again because these aren't unique
    # units
byStdUnit <- byStdUnit[order(byStdUnit$Group.1),] # sort by unit
    # number

#=============================================================
# Add missing 0-count units back to compiled vector of counts
#=============================================================

stdtest <- data.frame(AllUnits)
stdtest <- data.frame(stdtest$SUS_ID_1, stdtest$SUS_ID_12)
stdtest$Group.1 <- stdtest$stdtest.SUS_ID_1 + stdtest$stdtest.SUS_
    ID_12
stdtest <- data.frame(stdtest$Group.1)
stdtest$x <- 0
colnames(stdtest) <- c("Group.1", "x")

StdAll <- rbind(stdtest, byStdUnit)
StdCounts <- aggregate(StdAll$x, list(StdAll$Group.1), sum)
colnames(StdCounts) <- c("UnitID", "Moose_Count")

SSTRAT <- data.frame(AllUnits)
SSTRAT$UnitID <- SSTRAT$SUS_ID_1 + SSTRAT$SUS_ID_12
StdStrat <- SSTRAT[order(SSTRAT$UnitID),]

#=============================================================
```

```
# Set up final dataframe for FPBK
#=============================================================

colnames(StdCounts) <- c("UnitID", "Moose_Count")
StdCounts <- StdCounts[order(StdCounts$UnitID),]
Stdmerge1 <- merge(StdCounts, StdStrat, by = "UnitID")

STDdat <- data.frame(AllUnits)
STDdat$UnitID <- STDdat$SUS_ID_1 + STDdat$SUS_ID_12

STDdat$Counted <- 0

STDdat2 <- STDdat[order(STDdat$UnitID),]
Stdmerge2 <- merge(Stdmerge1, STDdat2, by = "UnitID")
Stdmerge3 <- Stdmerge2[,c("UnitID","Latitude.x","Longitude.x",
      "Moose_Count","Stratum.x","Counted")]
colnames(Stdmerge3) <- c("UnitID","CentrLat","CentrLong",
      "Moose_Count","Stratum","Counted")
StdData <- Stdmerge3

#=============================================================
# Randomly select standard sample units
#=============================================================

##### High Stratum

StdHigh <- subset(StdData, StdData$Stratum == "HIGH")
StdHigh <- data.matrix(StdHigh)
nH <- length(StdHigh[,1])
HighIndex <- sample(nH, size=sizeHS ,replace = FALSE)
for(i in 1:sizeHS){
      StdHigh[HighIndex[i],6] <- 1
      }
ZStdHigh <- data.frame(StdHigh)

##### Low Stratum

StdLow <- subset(StdData, StdData$Stratum == "LOW")
StdLow <- data.matrix(StdLow)
nL <- length(StdLow[,1])
LowIndex <- sample(nL, size=sizeLS ,replace = FALSE)
for(i in 1:sizeLS){
      StdLow[LowIndex[i],6] <- 1
      }
ZStdLow <- data.frame(StdLow)

ZStd <- rbind(ZStdHigh, ZStdLow)

```

```
##### Add additional required columns to ZStd

STD <- AllUnits[order(AllUnits$UnitID),]
ZStd <- ZStd[order(ZStd$UnitID),]
ZStd$AreaMi <- AllUnits$Area
ZStd$surveyid <- 77
ZStd$columnpred <- NA
ZStd$Stratum[ZStd$Stratum == 2] <- "LOW"
ZStd$Stratum[ZStd$Stratum == 1] <- "HIGH"

#============================================================
# Define analysis area based on choice at start of script
#============================================================

if(ANAREA=="HIGH"){
      for(i2 in 1:length(ZStd$columnpred))
            {
            if(ZStd$Stratum[i2] == "HIGH"){ZStd$columnpred[i2] <- 1
            }else{ZStd$columnpred[i2] <- 0}
            }
}
if(ANAREA=="LOW"){
      for(i2 in 1:length(ZStd$columnpred))
            {
            if(ZStd$Stratum[i2] == "LOW"){ZStd$columnpred[i2] <- 1
            }else{ZStd$columnpred[i2] <- 0}
            }
}
if(ANAREA=="ALL"){ZStd$columnpred <- 1}

#============================================================
# Perform block kriging with functions written by Jay Ver Hoef
#     (see Appendix 4 for GSPE functions)
#============================================================

##### Specify arguments for geomoosepop function

data <- ZStd
column.pred <- "columnpred"
column.ana <- "Moose_Count"
column.unitid <- "UnitID"
column.ana.formula <- "[UNKNOWN]"
strat <- "Stratum"
area <- "AreaMi"
column.lat <- "CentrLat"
column.lon <- "CentrLong"
sampled <- "Counted"
column.surveyid <- "surveyid"
```

```
Stdcalc.out <- geo.moosepop(column.ana = column.ana, strat =
      strat, data = data, sampled = sampled, area = area,
      column.pred = column.pred, column.lat=column.lat,
      column.lon=column.lon)
inpt.parms <- list(column.pred=column.pred,column.ana=
      column.ana,column.ana.formula=column.ana.formula,
      strat=strat,area=area,sampled=sampled)

##### Fill matrices with appropriate values during each iteration

PtEstStd[j,] <- Stdcalc.out$estimate.total
SEStd[j,] <- Stdcalc.out$estimate.standard.error
UnitsSamp[j,1] <- Stdcalc.out$sample.sizes[1,2]
UnitsSamp[j,2] <- Stdcalc.out$sample.sizes[2,2]
UnitsSamp[j,3] <- Stdcalc.out$sample.sizes[3,2]
TotalSamp[j,1] <- Stdcalc.out$total.samples[1,2]
TotalSamp[j,2] <- Stdcalc.out$total.samples[2,2]
TotalSamp[j,3] <- Stdcalc.out$total.samples[3,2]
MooseCount[j,1] <- Stdcalc.out$moose.counted[1,2]
MooseCount[j,2] <- Stdcalc.out$moose.counted[2,2]
MooseCount[j,3] <- Stdcalc.out$moose.counted[3,2]
ConfInt95Std[j,] <- Stdcalc.out$conf.int.95
ConfInt90Std[j,] <- Stdcalc.out$ci90
ConfInt80Std[j,] <- Stdcalc.out$ci80
SampAreaStd[j,1] <- Stdcalc.out$sampled.area[1,2]
SampAreaStd[j,2] <- Stdcalc.out$sampled.area[2,2]
SampAreaStd[j,3] <- Stdcalc.out$sampled.area[3,2]
TotalAreaStd[j,1] <- Stdcalc.out$total.area[1,2]
TotalAreaStd[j,2] <- Stdcalc.out$total.area[2,2]
TotalAreaStd[j,3] <- Stdcalc.out$total.area[3,2]
CIpropMean95[j,] <- Stdcalc.out$ci.prop.mean.95
CIpropMean90[j,] <- Stdcalc.out$ci.prop.mean.90
CIpropMean80[j,] <- Stdcalc.out$ci.prop.mean.80
StdSemVarHi[j,1] <- Stdcalc.out$parmest1[1,1]
StdSemVarHi[j,2] <- Stdcalc.out$parmest1[1,2]
StdSemVarHi[j,3] <- Stdcalc.out$parmest1[1,3]
StdSemVarLo[j,1] <- Stdcalc.out$parmest2[1,1]
StdSemVarLo[j,2] <- Stdcalc.out$parmest2[1,2]
StdSemVarLo[j,3] <- Stdcalc.out$parmest2[1,3]

##### Keep track of loop progress

print(paste("Loop #", j, " -- ", j/NumIter*100, "% complete"))
flush.console()

#*************************************************************
#=============================================================
# END j LOOP HERE
```

```
475 #=============================================================
476 #*************************************************************
477 }
478
479 #=============================================================
480 # Coverage
481 #=============================================================
482
483 ##### 95% #####
484
485 if(ANAREA == "ALL"){
486 CI95StdDF <- data.frame(ConfInt95Std)
487 colnames(CI95StdDF) <- c("Lower", "Upper")
488 CI95StdDF$in_interval <- 0
489 CI95Std <- as.matrix(CI95StdDF)
490 for (i in 1:NumIter){
491       if (CI95Std[i,1] < NumTotal & CI95Std[i,2] > NumTotal){
492             CI95Std[i,3] <- 1}
493       }
494 (Coverage95Std <- sum(CI95Std[,3]) / nrow(CI95Std))
495       }
496
497 if(ANAREA == "HIGH"){
498 CI95StdDF <- data.frame(ConfInt95Std)
499 colnames(CI95StdDF) <- c("Lower", "Upper")
500 CI95StdDF$in_interval <- 0
501 CI95Std <- as.matrix(CI95StdDF)
502 for (i in 1:NumIter){
503       if (CI95Std[i,1] < NumHigh & CI95Std[i,2] > NumHigh) {
504             CI95Std[i,3] <- 1}
505       }
506 (Coverage95Std <- sum(CI95Std[,3]) / nrow(CI95Std))
507       }
508
509 if(ANAREA == "LOW"){
510 CI95StdDF <- data.frame(ConfInt95Std)
511 colnames(CI95StdDF) <- c("Lower", "Upper")
512 CI95StdDF$in_interval <- 0
513 CI95Std <- as.matrix(CI95StdDF)
514 for (i in 1:NumIter){
515       if (CI95Std[i,1] < NumLow & CI95Std[i,2] > NumLow){
516             CI95Std[i,3] <- 1}
517       }
518 (Coverage95Std <- sum(CI95Std[,3]) / nrow(CI95Std))
519       }
520
521 ##### 90% #####
522
```

```
523 if(ANAREA == "ALL"){
524 CI90StdDF <- data.frame(ConfInt90Std)
525 colnames(CI90StdDF) <- c("Lower", "Upper")
526 CI90StdDF$in_interval <- 0
527 CI90Std <- as.matrix(CI90StdDF)
528 for (i in 1:NumIter){
529       if (CI90Std[i,1] < NumTotal & CI90Std[i,2] > NumTotal){
530             CI90Std[i,3] <- 1}
531       }
532 (Coverage90Std <- sum(CI90Std[,3]) / nrow(CI90Std))
533       }
534
535 if(ANAREA == "HIGH"){
536 CI90StdDF <- data.frame(ConfInt90Std)
537 colnames(CI90StdDF) <- c("Lower", "Upper")
538 CI90StdDF$in_interval <- 0
539 CI90Std <- as.matrix(CI90StdDF)
540 for (i in 1:NumIter){
541       if (CI90Std[i,1] < NumHigh & CI90Std[i,2] > NumHigh){
542             CI90Std[i,3] <- 1}
543       }
544 (Coverage90Std <- sum(CI90Std[,3]) / nrow(CI90Std))
545       }
546
547 if(ANAREA == "LOW"){
548 CI90StdDF <- data.frame(ConfInt90Std)
549 colnames(CI90StdDF) <- c("Lower", "Upper")
550 CI90StdDF$in_interval <- 0
551 CI90Std <- as.matrix(CI90StdDF)
552 for (i in 1:NumIter){
553       if (CI90Std[i,1] < NumLow & CI90Std[i,2] > NumLow){
554             CI90Std[i,3] <- 1}
555       }
556 (Coverage90Std <- sum(CI90Std[,3]) / nrow(CI90Std))
557       }
558
559 ##### 80% #####
560
561 if(ANAREA == "ALL"){
562 CI80StdDF <- data.frame(ConfInt80Std)
563 colnames(CI80StdDF) <- c("Lower", "Upper")
564 CI80StdDF$in_interval <- 0
565 CI80Std <- as.matrix(CI80StdDF)
566 for (i in 1:NumIter){
567       if (CI80Std[i,1] < NumTotal & CI80Std[i,2] > NumTotal){
568             CI80Std[i,3] <- 1}
569       }
570 (Coverage80Std <- sum(CI80Std[,3]) / nrow(CI80Std))
```

```
}

if(ANAREA == "HIGH"){
CI80StdDF <- data.frame(ConfInt80Std)
colnames(CI80StdDF) <- c("Lower", "Upper")
CI80StdDF$in_interval <- 0
CI80Std <- as.matrix(CI80StdDF)
for (i in 1:NumIter){
      if (CI80Std[i,1] < NumHigh & CI80Std[i,2] > NumHigh){
            CI80Std[i,3] <- 1}
      }
(Coverage80Std <- sum(CI80Std[,3]) / nrow(CI80Std))
}

if(ANAREA == "LOW"){
CI80StdDF <- data.frame(ConfInt80Std)
colnames(CI80StdDF) <- c("Lower", "Upper")
CI80StdDF$in_interval <- 0
CI80Std <- as.matrix(CI80StdDF)
for (i in 1:NumIter){
      if (CI80Std[i,1] < NumLow & CI80Std[i,2] > NumLow){
            CI80Std[i,3] <- 1}
      }
(Coverage80Std <- sum(CI80Std[,3]) / nrow(CI80Std))
}

#=============================================================
# Write results to text files
#=============================================================

workdir <- ""
setwd(workdir)
getwd()
dir()

##### File with raw results from each iteration

sink(paste("Togiak Cluster",Date,RunNo,ANAREA,"_",sizeHS,"H_",
      sizeLS,"L_",ClusterPropHigh,"Results.txt", sep=""))

cat("\n")
cat(paste("Togiak Cluster",Date,RunNo,ANAREA,"_",sizeHS,"H_",
      sizeLS,"L_",ClusterPropHigh,"Results", sep=""))

cat("\n\nNumber of iterations =")
print(NumIter)

cat("\nTrue number of low stratum moose =")
```

```
619 print(NumLow)
620
621 cat("\nTrue number of high stratum moose =")
622 print(NumHigh)
623
624 cat("\nTrue total number of moose =")
625 print(NumTotal)
626
627 cat("\nVector of Standard Grid Point Estimates =")
628 print(data.frame(PtEstStd))
629
630 cat("\nVector of Standard Grid SEs =")
631 print(data.frame(SEStd))
632
633 cat("\nStd 95% CI and 1/0 Pt Inclusion =")
634 print(CI95Std)
635
636 cat("\nStd 90% CI and 1/0 Pt Inclusion =")
637 print(CI90Std)
638
639 cat("\nStd 80% CI and 1/0 Pt Inclusion =")
640 print(CI80Std)
641
642 cat("\nMatrix of Standard Grid Sampled Areas (sq miles) = \n\n")
643 print(data.frame(SampAreaStd))
644
645 cat("\nMatrix of Standard Grid Total Areas (sq miles) = \n\n")
646 print(data.frame(TotalAreaStd))
647
648 cat("\nMatrix of Moose Counts = \n\n")
649 print(data.frame(MooseCount))
650
651 cat("\nVector of CIpropMean95 = \n\n")
652 print(data.frame(CIpropMean95))
653
654 cat("\nVector of CIpropMean90 = \n\n")
655 print(data.frame(CIpropMean90))
656
657 cat("\nVector of CIpropMean80 = \n\n")
658 print(data.frame(CIpropMean80))
659
660 cat("\nStandard Fitted Semi-variograms -- High =")
661 print(StdSemVarHi)
662 cat("\nStandard Fitted Semi-variograms -- Low =")
663 print(StdSemVarLo)
664
665 sink()
666
```

```
##### File with summarized results

sink(paste("Togiak Cluster",Date,RunNo,ANAREA,"_",sizeHS,"H_",
     sizeLS,"L_",ClusterPropHigh,"Summary.txt", sep=""))
cat("\n")
cat(paste("Togiak Cluster",Date,RunNo,ANAREA,"_",sizeHS,"H_",
     sizeLS,"L_",ClusterPropHigh,"Summary.txt", sep=""))

cat("\n\nNumber of iterations =")
print(NumIter)

cat("\nTrue number of low stratum moose =")
print(NumLow)

cat("\nTrue number of high stratum moose =")
print(NumHigh)

cat("\nTrue total number of moose =")
print(NumTotal)

cat("\nNumber of low stratum units sampled =")
print(sizeLS)

cat("\nNumber of high stratum units sampled =")
print(sizeHS)

cat("\nTotal number of units sampled =")
print(sizeLS + sizeHS)

cat("\nTrue total number of moose =")
print(NumTotal)

cat("\nMean(PtEstStd) =")
print(mean(PtEstStd))

cat("\nMean SE(SEStd) =")
print(mean(SEStd))

cat("\nSD of SE(SEStd) = ")
print(sd(SEStd))

cat("\n95% Wald CI for SE(SEStd) = \n")
cat(paste("Lower 95% CL = ",mean(SEStd) - 1.96*sd(SEStd),"\n"))
cat(paste("Upper 95% CL = ",mean(SEStd) + 1.96*sd(SEStd),"\n"))

cat("\n90% Wald CI for SE(SEStd) = \n")
cat(paste("Lower 90% CL = ",mean(SEStd) - 1.645*sd(SEStd),"\n"))
cat(paste("Upper 90% CL = ",mean(SEStd) + 1.645*sd(SEStd),"\n"))
```

```
715
716 cat("\n80% Wald CI for SE(SEStd) = \n")
717 cat(paste("Lower 80% CL = ",mean(SEStd) - 1.28*sd(SEStd),"\n"))
718 cat(paste("Upper 80% CL = ",mean(SEStd) + 1.28*sd(SEStd),"\n"))
719
720 cat("\nStd Coverage -- 95% CI =")
721 print(Coverage95Std)
722
723 cat("\nStd Coverage -- 90% CI =")
724 print(Coverage90Std)
725
726 cat("\nStd Coverage -- 80% CI =")
727 print(Coverage80Std)
728
729 if(ANAREA == "ALL"){
730 cat("\nStandard Unit Bias =")
731 print(mean(PtEstStd) - NumTotal)
732 }
733
734 if(ANAREA == "LOW"){
735 cat("\nStandard Unit Bias =")
736 print(mean(PtEstStd) - NumLow)
737 }
738
739 if(ANAREA == "HIGH"){
740 cat("\nStandard Unit Bias =")
741 print(mean(PtEstStd) - NumHigh)
742 }
743
744 cat("\nMean of Standard Grid Sampled Area (sq miles) = \n\n")
745 print(mean(SampAreaStd))
746
747 cat("\nMean of Standard Grid Total Area (sq miles) = \n\n")
748 print(mean(TotalAreaStd))
749
750 cat("\n% Area Sampled = \n\n")
751 print(mean(SampAreaStd) / mean(TotalAreaStd))
752
753 cat("\nRMSE = \n\n")
754 print(sqrt((sum((PtEstStd - NumTotal)^2))/NumIter))
755
756
757 sink()
758
759 #*************************************************************
760 #=============================================================
761 # END p LOOP HERE
762 #=============================================================
```

```
763 #*****************************************************************
764
765 }
766 ##### Restore old warnings setting
767 options(warn = oldw)
```

./Cluster_Simulations_Neat.R

Appendix 4: R functions for implementing the Geospatial Population Estimator

```
#########################################################
#####~~~ Functions used to implement GSPE  ~~~#####
#####~~~ Author: Jay Ver Hoef              ~~~~~#####
#########################################################

geo.moosepop <- function(column.ana, strat, data, sampled, area,
      column.pred, column.lat, column.lon){

      data <- cbind(data,
            LL.to.ARBUTM(mean(data[,column.lon]), data[,column.lat],
            data[,column.lon]) )
      data[,strat] <- as.factor(data[,strat])

      # ------------------- SUMMARY STATISTICS

      data.s <- data[!is.na(data[, sampled]) & data[, sampled] ==
          1, ]
      cds <- as.integer(data.s[,strat])
      cdu <- as.integer(data[,strat])
      lvs <- levels(data.s[,strat])
      nlvs <- max(cds)
      if(sum(lvs == "") > 0 | sum(is.na(lvs)) > 0)
            return(list(errstate = 1, errmessage = "Stratification
                has missing values",
            errextra = "" ))
      if(nlvs != 2)
            return(list(errstate = 1, errmessage = "Stratification
                must have exactly 2 levels",errextra = "" ))
      if(length(unique(cds)) != length(unique(cdu)))
            return(list(errstate = 1,
            errmessage = "Some strata have not been sampled",
            errextra = data.frame(sampled = unique(as.character(data
                .s[,strat])),all = levels(data[,strat]))))
      )
      means <- matrix(NA,nrow=nlvs,ncol=1)
      vars <- matrix(NA,nrow=nlvs,ncol=1)
      n <- matrix(NA,nrow=nlvs,ncol=1)
      N <- matrix(NA,nrow=nlvs,ncol=1)
      areas <- matrix(NA,nrow=nlvs,ncol=1)
      areato <- matrix(NA,nrow=nlvs,ncol=1)
      counted <- matrix(NA,nrow=nlvs,ncol=1)
      i <- 1
      for (i in 1:nlvs) {
```

```
40            ind.cds <- cds==i & data.s[,sampled]==1
41            ind.cdu <- cdu==i
42            counted[i] <- sum(data.s[ind.cds,column.ana],na.rm =
                  TRUE)
43            areas[i] <- sum(data.s[ind.cds,area])
44            means[i] <- counted[i]/areas[i]
45            n[i] <- length(data.s[ind.cds,column.ana])
46            N[i] <- length(data[ind.cdu,column.ana])
47            if(n[i] < 20 & N[i] != n[i])
48                  return(list(errstate = 1,
49                  errmessage = "Cannot estimate autocorrelation with
                        < 20 samples in a stratum",
50                  errextra = data.frame(stratum = as.character(lvs[i
                        ]), n = n[i], N = N[i]))
51            )
52            areato[i] <- sum(data[ind.cdu,area])
53      }
54      n.strat.df <- data.frame(Stratum=levels(data[,strat]),n = n)
55      n.strat.df <- rbind(n.strat.df,
56            data.frame(Stratum="TOTAL", n = sum(n.strat.df[,2])))
57      N.strat.df <- data.frame(Stratum=levels(data[,strat]), N = N)
58      N.strat.df <- rbind(N.strat.df,
59            data.frame(Stratum="TOTAL", N = sum(N.strat.df[,2])))
60      areas.strat.df <- data.frame(Stratum=levels(data[,strat]),
            Area=areas)
61      areas.strat.df <- rbind(areas.strat.df,
62            data.frame(Stratum="TOTAL", Area = sum(areas.strat.df
                  [,2])))
63      areato.strat.df <- data.frame(Stratum=levels(data[,strat]),
            Area=areato)
64      areato.strat.df <- rbind(areato.strat.df,
65            data.frame(Stratum="TOTAL", Area = sum(areato.strat.df
                  [,2])))
66      counted.strat.df <- data.frame(Stratum=levels(data[,strat]),
            Counted=counted)
67      counted.strat.df <- rbind(counted.strat.df,
68            data.frame(Stratum="TOTAL", Counted = sum(counted.strat.
                  df[,2])))
69
70      ind1 <- as.integer(data[,strat])==1 & data[,sampled]==1 & !is
            .na(data[,sampled])
71      ind2 <- as.integer(data[,strat])==2 & data[,sampled]==1 & !is
            .na(data[,sampled])
72      den1 <- data[ind1,column.ana]/data[ind1,area]
73      den2 <- data[ind2,column.ana]/data[ind2,area]
74      strat.1 <- data.frame(x = data[ind1, "x"], y = data[ind1, "y"
            ], var = den1)
```

```
75      strat.2 <- data.frame(x = data[ind2, "x"], y = data[ind2, "y"
            ], var = den2)
76
77      #---------------- EMPIRICAL SEMIVARIOGRAMS AND
            SEMICROSSVARIOGRAMS
78
79      emp.var1 <- empirical.semivariogram(data = strat.1, x = "x",
            y = "y", var = "var", nlag = 8,
80              maxlag = 50, directions = c(0), tolerance = 180,
81              nlagcutoff = 3)
82      emp.var2 <- empirical.semivariogram(data = strat.2, x = "x",
            y = "y", var = "var", nlag = 8,
83              maxlag = 50, directions = c(0), tolerance = 180,
84              nlagcutoff = 3)
85
86      #------------------------ FIT SEMIVARIOGRAMS
87
88      nugget1i <- mean(emp.var1[,"gamma"])/4
89      parsil1i <- mean(emp.var1[,"gamma"])
90      range1i <- mean(emp.var1[,"distance"])
91      theta <- c(nugget1i,parsil1i,range1i)
92      X1 <- matrix(1, nrow = length(den1), ncol = 1)
93      parmest1 <- optim(theta, m2LL, m2LLdata = strat.1, X=X1)$par
94
95      nugget2i <- mean(emp.var2[,"gamma"])/4
96      parsil2i <- mean(emp.var2[,"gamma"])
97      range2i <- mean(emp.var2[,"distance"])
98      theta <- c(nugget2i,parsil2i,range2i)
99      X2 <- matrix(1, nrow = length(den2), ncol = 1)
100     parmest2 <- optim(theta, m2LL, m2LLdata = strat.2, X=X2)$par
101
102     nugget1 <- parmest1[1]
103     parsil1 <- parmest1[2]
104     range1  <- parmest1[3]
105     nugget2 <- parmest2[1]
106     parsil2 <- parmest2[2]
107     range2  <- parmest2[3]
108     parmest1 <- data.frame(nugget = parmest1[1], parsil =
            parmest1[2], range = parmest1[3])
109     parmest2 <- data.frame(nugget = parmest2[1], parsil =
            parmest2[2], range = parmest2[3])
110
111     if(sum(den1) == 0) {
112          nugget1 <- 1e-6
113          parsil1 <- 0
114          range1 <- 1
115     }
116     if(sum(den2) == 0) {
```

```
117         nugget2 <- 1e-6
118         parsil2 <- 0
119         range2 <- 1
120       }
121
122       #------------------------------ BUILD MATRICES
123
124       SS <- SS.mat(data, nugget1 = nugget1, parsil1 = parsil1,
              range1 = range1,
125         nugget2 = nugget2, parsil2 = parsil2, range2 = range2,
126         sampled = sampled, strat = strat)
127       SU <- SU.mat(data, nugget1 = nugget1, parsil1 = parsil1,
              range1 = range1,
128         nugget2 = nugget2, parsil2 = parsil2, range2 = range2,
129         sampled = sampled, strat = strat)
130       UU <- UU.mat(data, nugget1 = nugget1, parsil1 = parsil1,
              range1 = range1,
131         nugget2 = nugget2, parsil2 = parsil2, range2 = range2,
132         sampled = sampled, strat = strat)
133
134       #------------------------------ FINITE POPULATION BLOCK KRIGING
135
136       ind.sa <- !is.na(data[, sampled] == 1) & data[, sampled] == 1
137       ns <- sum(ind.sa)
138       ind.un <- is.na(data[, sampled] == 1) | data[, sampled] != 1
139       nu <- sum(ind.un)
140       z <- matrix(data[ind.sa, column.ana], nrow = ns, ncol = 1)
141       area.s <- matrix(data[ind.sa, area], nrow = ns, ncol = 1)
142       z <- z/area.s
143       area.tot <- matrix(data[, area], nrow = nu+ns, ncol = 1)
144       B <- data[, column.pred]
145       B[is.na(B)] <- 0
146       Bs <- B[ind.sa]
147       Bu <- B[ind.un]
148       X1 <- as.numeric(as.integer(data[,strat])==1)
149       X2 <- as.numeric(as.integer(data[,strat])==2)
150       X <- cbind(rep(1,times=length(X1)),X1,X2)
151       Xs <- X[ind.sa,]
152       Xu <- X[ind.un,]
153       SSi <- solve(SS)
154
155       # ------------------------------ PREDICTIONS
156
157       part1 <- Xs %*% mginv(t(Xs) %*% SSi %*% Xs)
158       part2 <- t(Xu) - t(Xs) %*% SSi %*% SU
159       D <- SU + part1 %*% part2
160       FF <- SSi %*% D
161       Ao <- Bs + FF %*% Bu
```

```
162     y.est <- t(Ao) %*% z
163     y.est <- mean(area.tot)*y.est
164
165     # ------------------------ VARIANCE
166
167     part1 <- t(FF) %*% SS %*% FF
168     part2 <- t(FF) %*% SU
169     y.var <- t(Bu) %*% ( part1 - part2 - t(part2) + UU ) %*% Bu
170     y.se <- mean(area.tot)*sqrt(y.var)
171
172     # ------------------------ CONFIDENCE INTERVALS
173
174     ci80 <- c(y.est - y.se * qnorm(0.9), y.est + y.se *
175         qnorm(0.9))
176     cipm80 <- (y.se * qnorm(0.9))/y.est
177     ci90 <- c(y.est - y.se * qnorm(0.95), y.est + y.se *
178         qnorm(0.95))
179     cipm90 <- (y.se * qnorm(0.95))/y.est
180     ci95 <- c(y.est - y.se * qnorm(0.975), y.est + y.se *
181         qnorm(0.975))
182     cipm95 <- (y.se * qnorm(0.975))/y.est
183
184     outpt <- list(
185         estimate.total = as.numeric(y.est),
186         estimate.standard.error = as.numeric(y.se),
187         ci80 = ci80,
188         ci.prop.mean.80 = as.numeric(cipm80),
189         ci90 = ci90,
190         ci.prop.mean.90 = as.numeric(cipm90),
191         conf.int.95 = ci95,
192         ci.prop.mean.95 = as.numeric(cipm95),
193         sample.sizes = data.frame(n.strat.df, row.names = NULL),
194         total.samples = data.frame(N.strat.df, row.names = NULL)
                ,
195         moose.counted = data.frame(counted.strat.df, row.names =
                NULL),
196         sampled.area = data.frame(areas.strat.df, row.names =
                NULL),
197         total.area = data.frame(areato.strat.df, row.names =
                NULL),
198         strat.1.name = as.character(levels(data[,strat])[1]),
199         strat.2.name = as.character(levels(data[,strat])[2]),
200         empirical.semivariogram.strat1 = emp.var1[,1:3],
201         empirical.semivariogram.strat2 = emp.var2[,1:3],
202         parmest1 = parmest1,
203         parmest2 = parmest2
204     )
205     outpt
```

```
206 }
207
208 #(B)
209 # Latitude,Longitude to arbitrary UTM
210 LL.to.ARBUTM<-
211 function(cm,lat,lon)
212 # This function converts from Lat-Lon (decimal degrees) to the
213 # Universal Transverse Mercator Coordinates and returns the
214 # new coordinates in a 2-column matrix with x- and y- as
215 # columns.  In this program, the coordinates are calculated
216 # from a user supplied central meridian. # Coordinates are
217 # returned in kilometers from the western-most longitude
218 # and the southern-most latitude observed in the data set.
219
220 {
221 # initialize some variables
222       e2 <- 0.00676865799729
223       a <- 6378206.4
224       ep2 <- e2 / (1-e2)
225       drc <- pi / 180
226       sc <- 0.9996
227       fe <- 500000
228       ftm <- 0.30480371
229 #calculate some frequently used values
230       lar <- lat * drc
231       ls <- sin(lar)
232       ls2 <- ls^2
233       els2 <- ep2 * ls2
234       lc <- cos(lar)
235       lc2 <- lc^2
236       lc3 <- lc^3
237       lc5 <- lc^5
238       elc2 <- ep2 * lc2
239       lt2 <- tan(lar)^2
240       lt4 <- lt2^2
241 # do the transformation
242       v <- a/sqrt(1 - e2*ls2)
243       p <- drc*(cm - lon)
244       temp <- 5104.57388 - (lc2*(21.73607 - 0.11422*lc2))
245       r1 <- 6367399.689*(lar - ls*lc*0.000001*temp)
246       r2 <- (v*ls*lc*p^2)/2
247       temp <- 5 - lt2 + 9*elc2 + (2*elc2)^2
248       r3 <- (v*ls*lc3*p^4*temp)/24
249       r4 <- v*lc*p
250       temp <- 1 - lt2 + elc2
251       r5 <- (v*lc3*p^3*temp)/6
252       temp <- 61 - 58*lt2 + lt4 + 270*elc2 - 330*els2
253       ra6 <- (v*ls*lc5*p^6*temp)/720
```

```
254       temp <- 5 - 18*lt2 + lt4 + 14*elc2 - 58*els2
255       rb5 <- (v*lc5*p^5*temp)/120
256       northing <- sc*(r1 + r2 + r3 + ra6)
257       easting <- -sc*(r4 + r5 + rb5)
258       y<-(northing-min(northing))/1000
259       x<-(easting-min(easting))/1000
260       cbind(x,y)
261 }
262
263
264 # ------------------------------------------------------------------
265 #           GENERALIZED INVERSE OF A MATRIX
266 #------------------------------------------------------------------
267
268 #---------------------- GENERALIZED INVERSE OF A MATRIX
269
270 mginv <- function(X, tol = sqrt(.Machine$double.eps)) {
271       dnx <- dimnames(X)
272       if(is.null(dnx)) dnx <- vector("list", 2)
273       s <- svd(X)
274       nz <- s$d > tol * s$d[1]
275       structure(
276             if(any(nz)) s$v[, nz] %*% (t(s$u[, nz])/s$d[nz]) else X,
277             dimnames = dnx[2:1])
278 }
279
280 # ------------------------------------------------------------------
281 # EMPIRICAL SEMIVARIOGRAM AND SEMICROSSVARIOGRAM FUNCTIONS
282 #------------------------------------------------------------------
283
284 #---------------------- EMPIRICAL SEMIVARIOGRAM
285
286 empirical.semivariogram<-
287 function(data, x, y, var,
288       nlag = 20, directions = c(0,45,90,135),
289       tolerance = 22.5, inc = 0, maxlag = 1e32, nlagcutoff = 1)
290 # EMPIRICAL SEMIVARIOGRAM FUNCTION
291 # var1 is a matrix or data frame with x-coord in the first column
292 #                                     y-coord in the second column
293 #                                  z (response) in the third column
294 {
295       n1 <- length(data[,1])
296    # distance matrix among locations
297       distance <- sqrt( ( matrix(data[,x],nrow=n1,ncol=1) %*%
298             matrix(rep(1,times=n1),nrow=1,ncol=n1) -
299             matrix(rep(1,times=n1),nrow=n1,ncol=1) %*%
300             matrix(data[,x],nrow=1,ncol=n1) )^2 +
301             ( matrix(data[,y],nrow=n1,ncol=1) %*%
```

```
302           matrix(rep(1,times=n1),nrow=1,ncol=n1) -
303           matrix(rep(1,times=n1),nrow=n1,ncol=1) %*%
304           matrix(data[,y],nrow=1,ncol=n1) )^2 )
305      difx <- -(matrix(data[,y],nrow=n1,ncol=1) %*%
306           matrix(rep(1,times=n1),nrow=1,ncol=n1) -
307           matrix(rep(1,times=n1),nrow=n1,ncol=1) %*%
308           matrix(data[,y],nrow=1,ncol=n1))
309      signind <- -(matrix(data[,x],nrow=n1,ncol=1) %*%
310           matrix(rep(1,times=n1),nrow=1,ncol=n1) -
311           matrix(rep(1,times=n1),nrow=n1,ncol=1) %*%
312           matrix(data[,x],nrow=1,ncol=n1)) < 0
313      distance <- distance*1.0000000001
314      theta.deg<-acos(difx/distance)*180/pi
315    # matrix of degrees clockwise from north between locations
316      theta.deg[signind] <- 360-theta.deg[signind]
317      diff2 <- ( matrix(data[,var],nrow=n1,ncol=1) %*%
318           matrix(rep(1,times=n1),nrow=1,ncol=n1) -
319           matrix(rep(1,times=n1),nrow=n1,ncol=1) %*%
320           matrix(data[,var],nrow=1,ncol=n1) )^2
321 # convert to vectors
322      distance <- matrix(distance, ncol = 1)
323      theta.deg <- matrix(theta.deg, ncol = 1)
324      diff2 <- matrix(diff2, ncol = 1)
325 # trim off values greater than maxlag
326      indmax <- distance <= maxlag
327      distance <- distance[indmax,]
328      theta.deg <- theta.deg[indmax,]
329      diff2 <- diff2[indmax,]
330
331      maxd<-max(distance)
332      if( inc <= 0) inc <- maxd/nlag
333      ind <- distance==0
334      ndir <- length(directions)
335      store.results <- matrix(data = NA, ncol = 6,
336           dimnames = list(NULL, c("distance", "gamma", "np", "
                 azimuth", "hx", "hy")))
337      for (j in 1:ndir) {
338           for ( i in 1:nlag){
339                if( (directions[j]-tolerance)<0 && (directions[j]+
                     tolerance)>0 )
340                     ind1 <- theta.deg >= 360+directions[j]-
                          tolerance |
341                          theta.deg < directions[j]+tolerance
342                else if( (directions[j]+tolerance)>360 && (
                     directions[j]-tolerance)<360 )
343                     ind1 <- theta.deg < directions[j]+tolerance
                          -360 |
344                          theta.deg >= directions[j]-tolerance
```

```
345                 else
346                       ind1 <- theta.deg >= directions[j]-tolerance &
347                             theta.deg < directions[j]+tolerance
348                 ind<-distance>(i-1)*inc & distance<=i*inc &
349                       !is.na(theta.deg) & ind1
350                 nclass <- sum(ind)
351                 cv <- mean(diff2[ind])
352                 mean.dis <- mean(distance[ind])
353                 if(nclass > 0) store.results<-rbind(store.results,
354                       c(mean.dis,cv,nclass,directions[j],0,0))
355           }
356     }
357     store.results[,"hx"]<-store.results[,"distance"]*sin(store.
          results[,"azimuth"]*pi/180)
358     store.results[,"hy"]<-store.results[,"distance"]*cos(store.
          results[,"azimuth"]*pi/180)
359     store.results[,"gamma"]<-store.results[,"gamma"]/2
360     ind <- store.results[,"np"] >= nlagcutoff
361     store.results <- store.results[ind,]
362     ind <- !is.na(store.results[,"hx"])
363     store.results <- store.results[ind,]
364     as.data.frame(store.results)
365 }
366
367 #-------------------------------------------------------------------
368 #FUNCTIONS FOR FITTING THE SEMIVARIOGRAM MODEL TO VARIABLE 1
369 #-------------------------------------------------------------------
370
371 #---------- EXPONENTIAL VARIOGRAM MODEL
372
373 exponential.variogram.model<-
374 function(h, nugget = 0, parsil = 1, range = 1)
375 {
376     d <- sqrt(h[,1]^2 + h[,2]^2)
377     ind <- d == 0
378     v <- nugget + parsil*(1-exp(-d/range))
379     v[ind] <- 0
380     v
381 }
382
383 #---------- DATA COVARIANCE MATRIX BASED ON EXPONENTIAL VARIOGRAM
        MODEL
384
385 exp.vc.matrix <- function(vcmatdata, x = "x", y = "y",
386     nugget = nugget, parsil = parsil, range = range)
387 {
388     n <- length(vcmatdata[,1])
389     distance <- matrix(0, n, n)
```

```
390       distance[lower.tri(distance)] <- dist(as.matrix(vcmatdata[,c(
              x,y)]))
391       distance <- distance + t(distance)
392       distance <- parsil*exp(-distance/range) + diag(nugget, nrow =
              n, ncol = n)
393       distance
394 }
395
396 #------------- REML EQUATION TO MINIMIZE
397
398 m2LL <- function(theta, m2LLdata, X)
399 {
400       nugget <- theta[1]
401       parsil <- theta[2]
402       range <- theta[3]
403       z <- m2LLdata[,3]
404       if(nugget <= 0 || parsil <= 0 || range <= 0)
405             1e32
406       else {
407             n <- length(z)
408             p <- length(X[1,])
409             V <- exp.vc.matrix(vcmatdata = m2LLdata,
410                   nugget = nugget, parsil = parsil, range = range)
411             Vi <- solve(V)
412             b.hat <- mginv(t(X) %*% Vi %*% X) %*% t(X) %*% Vi %*% z
413             f1 <- sum(log(eigen(V)$values))
414             f2 <- t(z - X %*% b.hat) %*% Vi %*% (z - X %*% b.hat)
415             f3 <- sum(log(eigen(t(X) %*% Vi %*% X)$values))
416             f1 + f2 + f3 + (n - p) *log(2 * pi)
417       }
418 }
419
420
421 #-------------------------------------------------------------------
422 #              BUILD MATRICES
423 #-------------------------------------------------------------------
424
425 #
426 # ------- BUILDS THE SAMPLE VARIANCE-COVARIANCE MATRIX
427 SS.mat <-
428 function(data, nugget1, parsil1, range1,
429       nugget2, parsil2, range2, sampled, strat)
430 {
431       sampled.ind <- !is.na(data[, sampled] == 1) & data[, sampled]
              == 1
432       n <- sum(sampled.ind)
433       x <- matrix(data[sampled.ind, "x"], nrow = n, ncol =
434             1)
```

```
435       y <- matrix(data[sampled.ind, "y"], nrow = n, ncol =
436            1)
437       x.mat <- matrix(rep(x, times = n), nrow = n,ncol = n)
438       x.mat <- t(x.mat) - x.mat
439       y.mat <- matrix(rep(y, times = n), nrow = n, ncol = n)
440       y.mat <- t(y.mat) - y.mat
441       c.mat <- matrix(rep(as.integer(data[sampled.ind, strat]),
              times = n),nrow = n,ncol = n)
442       s11.ind <- matrix( c.mat == 1 & t(c.mat) == 1, nrow = n^2,
              ncol = 1)
443       s22.ind <- matrix( c.mat == 2 & t(c.mat) == 2, nrow = n^2,
              ncol = 1)
444       s12.ind <- matrix( c.mat == 1 & t(c.mat) == 2, nrow = n^2,
              ncol = 1)
445       s21.ind <- matrix( c.mat == 2 & t(c.mat) == 1, nrow = n^2,
              ncol = 1)
446       h <- cbind(matrix(x.mat, nrow = n^2, ncol = 1), matrix(y.mat,
               nrow = n^2, ncol = 1))
447       gamma11 <- nugget1 + parsil1 - exponential.variogram.model(h,
               nugget = nugget1, parsil = parsil1,
448            range = range1)
449       gamma22 <- nugget2 + parsil2 - exponential.variogram.model(h,
               nugget = nugget2, parsil = parsil2,
450            range = range2)
451       gamma <- matrix(NA, nrow = n^2, ncol = 1)
452       gamma[s11.ind] <- gamma11[s11.ind]
453       gamma[s22.ind] <- gamma22[s22.ind]
454       gamma[s12.ind] <- 0
455       gamma[s21.ind] <- 0
456       gamma <- matrix(gamma, nrow = n, ncol = n)
457       gamma
458  }
459
460  #
461  # ------- BUILDS VARIANCE-COVARIANCE MATRIX BETWEEN SAMPLED AND
         UNSAMPLED
462  SU.mat <-
463  function(data, nugget1, parsil1, range1,
464       nugget2, parsil2, range2, sampled, strat)
465  {
466       sampled.ind <- !is.na(data[, sampled] == 1) & data[, sampled]
              == 1
467       unsampled.ind <- is.na(data[, sampled] == 1) | data[, sampled
              ] != 1
468       ns <- sum(sampled.ind)
469       nu <- sum(unsampled.ind)
470       xs <- matrix(data[sampled.ind, "x"], nrow = ns, ncol = 1)
471       ys <- matrix(data[sampled.ind, "y"], nrow = ns, ncol = 1)
```

```
472       xu <- matrix(data[unsampled.ind, "x"], nrow = nu, ncol = 1)
473       yu <- matrix(data[unsampled.ind, "y"], nrow = nu, ncol = 1)
474       ones <- matrix(1, nrow = ns, ncol = 1)
475       oneu <- matrix(1, nrow = nu, ncol = 1)
476       x.mat <- -matrix(rep(xs, times = nu), nrow = ns, ncol = nu) +
477            t(matrix(rep(xu, times = ns), nrow = nu, ncol = ns))
478       y.mat <- -matrix(rep(ys, times = nu), nrow = ns, ncol = nu) +
479            t(matrix(rep(yu, times = ns), nrow = nu, ncol = ns))
480       c.mats<-matrix(rep(as.integer(data[sampled.ind, strat]),
              times = nu), nrow = ns, ncol = nu)
481       c.matu<-t(matrix(rep(as.integer(data[unsampled.ind, strat]),
              times = ns), nrow = nu, ncol = ns))
482       s11.ind <- matrix( c.mats==1 & c.matu==1, nrow=ns*nu, ncol=1)
483       s22.ind <- matrix( c.mats==2 & c.matu==2, nrow=ns*nu, ncol=1)
484       s12.ind <- matrix( c.mats==1 & c.matu==2, nrow=ns*nu, ncol=1)
485       s21.ind <- matrix( c.mats==2 & c.matu==1, nrow=ns*nu, ncol=1)
486       h<-cbind(matrix(x.mat,nrow=ns*nu,ncol=1),matrix(y.mat,nrow=ns
              *nu,ncol=1))
487       gamma11 <- nugget1 + parsil1 - exponential.variogram.model(h,
               nugget = nugget1, parsil = parsil1,
488               range = range1)
489       gamma22 <- nugget2 + parsil2 - exponential.variogram.model(h,
               nugget = nugget2, parsil = parsil2,
490               range = range2)
491       gamma<-matrix(NA, nrow = ns*nu, ncol = 1)
492       gamma[s11.ind] <- gamma11[s11.ind]
493       gamma[s22.ind] <- gamma22[s22.ind]
494       gamma[s12.ind] <- 0
495       gamma[s21.ind] <- 0
496       gamma<-matrix(gamma, nrow = ns, ncol = nu)
497       gamma
498  }
499
500  #
501  # ------- BUILDS THE UNSAMPLED VARIANCE-COVARIANCE MATRIX
502  UU.mat<-
503  function(data, nugget1, parsil1, range1,
504       nugget2, parsil2, range2, sampled, strat)
505  {
506       sampled.ind <- is.na(data[, sampled] == 1) | data[, sampled]
              != 1
507       n <- sum(sampled.ind)
508       x <- matrix(data[sampled.ind, "x"], nrow = n, ncol =
509            1)
510       y <- matrix(data[sampled.ind, "y"], nrow = n, ncol =
511            1)
512       x.mat<-matrix(rep(x,times=n),nrow=n,ncol=n)
513       x.mat<-t(x.mat)-x.mat
```

```
514       y.mat<-matrix(rep(y,times=n),nrow=n,ncol=n)
515       y.mat<-t(y.mat)-y.mat
516       c.mat<-matrix(rep(as.integer(data[sampled.ind,strat]),times=n
              ),nrow=n,ncol=n)
517       s11.ind <- matrix( c.mat==1 & t(c.mat)==1, nrow=n^2, ncol=1)
518       s22.ind <- matrix( c.mat==2 & t(c.mat)==2, nrow=n^2, ncol=1)
519       s12.ind <- matrix( c.mat==1 & t(c.mat)==2, nrow=n^2, ncol=1)
520       s21.ind <- matrix( c.mat==2 & t(c.mat)==1, nrow=n^2, ncol=1)
521       h<-cbind(matrix(x.mat,nrow=n^2,ncol=1),matrix(y.mat,nrow=n^2,
              ncol=1))
522       gamma11 <- nugget1 + parsil1 - exponential.variogram.model(h,
               nugget = nugget1, parsil = parsil1,
523                range = range1)
524       gamma22 <- nugget2 + parsil2 - exponential.variogram.model(h,
               nugget = nugget2, parsil = parsil2,
525                range = range2)
526       gamma<-matrix(NA,nrow=n^2,ncol=1)
527       gamma[s11.ind]<-gamma11[s11.ind]
528       gamma[s22.ind]<-gamma22[s22.ind]
529       gamma[s12.ind] <- 0
530       gamma[s21.ind] <- 0
531       gamma <- matrix(gamma, nrow = n,ncol = n)
532       gamma
533 }
534
535 # LIKELIHOOD FUNCTION FOR CROSS CORRELATION OF NUGGET EFFECT
536 m2LL.cross <- function(rho, theta1, theta2, m2LLdata, X)
537 {
538       nugget1 <- theta1[1]
539       parsil1 <- theta1[2]
540       range1 <- theta1[3]
541       nugget2 <- theta2[1]
542       parsil2 <- theta2[2]
543       range2 <- theta2[3]
544
545       z1 <- m2LLdata[,3]
546       z2 <- m2LLdata[,4]
547       z <- matrix(c(z1,z2), ncol = 1)
548
549       X <- diag(2) %x% X
550
551             n <- length(z1)
552             p <- length(X[1,])
553             V1 <- exp.vc.matrix(vcmatdata = m2LLdata,
554                   nugget = nugget1, parsil = parsil1, range = range1)
555             V2 <- exp.vc.matrix(vcmatdata = m2LLdata,
556                   nugget = nugget2, parsil = parsil2, range = range2)
557             V <- matrix(0, nrow = 2*n, ncol = 2*n)
```

```
558             V[1:n,1:n] <- exp.vc.matrix(vcmatdata = m2LLdata,
559                   nugget = nugget1, parsil = parsil1, range = range1)
560             V[(n+1):(2*n),(n+1):(2*n)] <- exp.vc.matrix(vcmatdata =
                    m2LLdata,
561                   nugget = nugget2, parsil = parsil2, range = range2)
562             Z <- rbind(diag(n), diag(n))
563             V[1:n,(n+1):(2*n)] <- rho*sqrt(nugget1*nugget2)*diag(n)
564             V[(n+1):(2*n),1:n] <- rho*sqrt(nugget1*nugget2)*diag(n)
565             Vi <- solve(V)
566             b.hat <- mginv(t(X) %*% Vi %*% X) %*% t(X) %*% Vi %*% z
567             f1 <- sum(log(eigen(V)$values))
568             f2 <- t(z - X %*% b.hat) %*% Vi %*% (z - X %*% b.hat)
569             f3 <- sum(log(eigen(t(X) %*% Vi %*% X)$values))
570             f1 + f2 + f3 + (n - p) *log(2 * pi)
571
572 }
```

./GSPE_Functions_Neat.R